Aalto University
School of Science and Technology

LULEÅ
UNIVERSITY
OF TECHNOLOGY

Sven Mikael Persson

Coordinated Control of Two Robotic Manipulators for Physical Interactions with Astronauts

Faculty of Electronics, Communications and Automation

Thesis submitted in partial fulfilment of the requirements for the degree of

Master of Science in Technology

Espoo, August 11, 2010

Instructor: M.Sc. Seppo Heikkilä
Aalto University
School of Science and Technology

Supervisors: Professor Aarne Halme
Aalto University
School of Science and Technology

Professor Kalevi Hyyppä
Luleå University of Technology

Acknowledgements

My kudos goes to the Department of Automation and System Technology at Aalto University for forging the casual and creative atmosphere which gave shape to my thesis. My special thanks go to my instructor, Seppo Heikkilä, for his support and guidance, and for that, I am deeply grateful. I am also thankful to my supervisors, Prof. Aarne Halme and Prof. Kalevi Hyyppä, for their valuable feedback and expert advise. Also, I would like to acknowledge the work of the staff who maintain the *WorkPartner* robot in working conditions.

As part of the Joint European Masters in Space Science and Technology, this thesis was made possible by the consorted effort of six universities and their dedicated staff of which I would like to thank Tomi Ylikorpi and Anja Hänninen from Aalto University, and Annette Snällfot-Brändström, Maria Winnebäck and Sven Molin from Luleå University of Technology. Furthermore, I am grateful to all my peers who have made this programme, a unique and unforgettable experience of cultural exchanges, inspiring encounters and sheer fun!

My Master's degree abroad, of which this thesis is a part, was made possible by scholarships from the European Commission via the Erasmus Mundus Programme and from the National Science and Engineering Research Council of Canada via their Post-Graduate Scholarship Programme. I am forever in their debt and have immense appreciation for their work in promoting and enabling academics at a graduate level.

Finally, I am deeply grateful to my parents for their unconditional love and support, for which I could never thank them enough.

Espoo, August 11, 2010

Sven Mikael Persson

Aalto University
School of Science and Technology Abstract of the Master's Thesis

Author:	Sven Mikael Persson
Title of the thesis:	Coordinated Control of Two Robotic Manipulators for Physical Interactions with Astronauts
Date:	August 11, 2010 **Number of pages:** 99
Faculty:	Faculty of Electronics, Communications and Automation
Department:	Automation and System Technology
Programme:	Master's Degree Programme in Space Science and Technology
Professorship:	Automation Technology (Aut-84)
Supervisors:	Professor Aarne Halme (Aalto)
	Professor Kalevi Hyyppä (LTU)
Instructor:	M.Sc. Seppo Heikkilä

The following thesis presents the complete development, simulation, and experimental work towards the coordinated control of a dual-manipulator system for physical human-robot interactions. The project aims at pushing robot autonomy towards new boundaries. In the context of space exploration, the potential gain of increased autonomy is enormous and has been a major area of research. Enabling cooperation between an astronaut and a robot has the potential to increase the overall productivity of extra-vehicular activities without the hazard and cost of an additional astronaut. The concept of physical human-robot interaction is motivated by a simple, reliable and robust language between humans and robots, *id est*, the physical world.

In this work, the scope of the problem is limited to the control of two robotic manipulators to manipulate a single object while allowing human input via external forces as part of mission objectives. The robot used here is the *WorkPartner*, a centaur-like robot whose upper body consists of a two degrees-of-freedom torso and two manipulators with five degrees-of-freedom each. This thesis presents revisited solutions to the underlying problems of dynamics compensation and external force estimation, integrated in a modern implementation using the virtual model control concept, kinetostatic transmission elements and a behaviour-based strategy.

Classic solutions are reviewed and a control strategy is adopted which incorporates widely applicable solutions, and thus, forms a modern basis for further developments. Some simple scenarios are validated for basic stability and performance through simulations, first with a simple pendulum system and then, with the *WorkPartner*'s upper body. Finally, experimental results further reinforces the validity and performance arguments for the proposed algorithms and control architecture.

Keywords: compliant control, dual-manipulator, physical human-robot interaction, virtual model control, kinetostatic transmission elements, behaviour-based, external force estimation, humanoid robotics, embodied intelligence

Contents

List of Tables

List of Figures

List of Algorithms

Symbols and Abbreviations

τ	Vector of Joint Torques
q	Vector of Joint Angles
$\dot{q}$	Vector of Joint Velocities
$\ddot{q}$	Vector of Joint Accelerations
$\vec{p}, \vec{v}, \vec{a}$	Position, velocity and acceleration vectors
$\vec{\omega}, \vec{\alpha}$	Angular velocity and angular acceleration vectors
i_m	Vector of Motor Currents
v_m	Vector of Motor Voltages
v_{emf}	Vector of Motor Back-EMF Voltages
$M(q)$	Inertia Matrix of the Manipulator
$C(q, \dot{q})$	Centripetal and Coriolis Force Terms, Christoffel Matrix
$g(q)$	Gravitational Generalized Forces
$\tau_f(\dot{q})$	Joint Friction Torques
G_m	Gear Ratio
I_m	Motor Inertia Matrix
K_m	Motor Torque Constant (Ideal BEMF Constant)
A^T	Matrix Transpose
A^{-1}	Matrix Inverse
A^{-T}	Matrix Transpose Inverse
$A^\sharp$	Matrix Generalized Inverse
$\mathit{\Gamma}$	Kinetostatic frame (2D or 3D)
$\mathit{\Theta}$	Kinetostatic generalized coordinate
$[\cdot]_i$	Quantity expressed in coordinates i
$[\vec{v}\times]$	Cross-product matrix of vector $\vec{v}$
$t_i, t_{cm,i}$	Link i twist vector, Centre-of-mass i twist vector
$\mu_{cm,i}$	Center-of-mass i spatial momentum vector
Q_i	Spatial rotation matrix
W_i	Spatial offset matrix

$[J_i]_j$	Jacobian of joint j on frame, link or center-of-mass i
r_{EE}	Some quantity r of the End-Effector (subscript "EE")
$J[:, i]$	In algorithms, column i of matrix J.
$\Gamma : \omega^T$	In algorithms, : signifies here "the ω member of Γ object or structure"

Aalto	Aalto University School of Science and Technology
AI	Artificial Intelligence
BBA	Behavior-Based Approach
CSA	Canadian Space Agency
DK	Direct Kinematics
DLR	Deutsche Luft- und Raumfahrt Zentrum
dof	degrees of freedom
DSP	Digital Signal Processing
ESA	European Space Agency
EVA	Extra-Vehicular Activities
FD	Forward Dynamics
ID	Inverse Dynamics
IK	Inverse Kinematics
ISS	International Space Station
KTE	Kinetostatic Transmission Element
LTU	Luleå University of Technology
MBD	Multibody Dynamics
MBS	Mobile Base System
MIT	Massachusetts Institute of Technology
MSS	Mobile Servicing System
NASA	National Aeronautics and Space Administration
PD	Proportional-Differential [controller]
PID	Proportional-Integral-Differential [controller]
RBF(NN)	Radial-Basis Function (Neural-Network)
VMC	Virtual Model Control

Chapter 1

Introduction

"I never believed in trying to do anything. Whatever I set out to do I found I had already accomplished."
- Johann Wolfgang von Goethe

While space exploration goes on to reach further into the solar system, to stay longer in space and to conduct more experiments of fundamental and technical importance, the time, hazard and costs of manned flights and extra-vehicular activities are now becoming a major limiting factor for the future (Singer and Akin, 2010). This problem has motivated several researchers, including the author, to pursue robot autonomy such that fewer astronauts are needed in space, especially for extra-vehicular activities. Robots can be mechanically designed to be resistant to radiation, as much or more dexterous than astronauts, and better suited for special purposes or operations such as heavy loads or large structures. However, robots, at the present, lack the required intelligence to be fully autonomous. It goes without saying that many components are required to match the level of autonomy of a human being, including sophisticated computer vision, intricate sensory-motor systems, natural communication skills, conceptualization and cognition, just to mention a few. Most of the required fields are being developed in parallel by a plethora of researchers around the world. In this respect, this thesis work belongs to the field of embodied intelligence.

Embodied intelligence heavily relies on the idea of intelligent behaviours emerging from the robot's interactions with the environment. Often summarized by the expression "to let the environment be the model and the interactions the

intelligence", this area of automation studies what it means to embody an intelligent system: how can a robot best utilize sensors and actuators; how does the environment affect a task or an objective; and how does the objective or the mission involve the environment. By studying a robot primarily as an agent of the real world, as an actor of the environment, one can appreciate how much intelligence, and thus, artificial intelligence is, in many instances, a result of physical interactions. Just as the human brain reaches out, via a tentacular nervous system, all the way to the tip of a toe, a robot's motion system and high-level software are heavily intertwined.

Humbly speaking, the goal of this thesis is to study how a robot can intelligently manipulate objects, towards an objective, in a human-friendly setting with the help of a physically interacting human operator. On a broader scale, the author hopes to contribute to both the expression of intelligent behaviours by a robot through its motion, and in turn, the recognition of human intentions by the robotic system, both of which are necessary for human-robot cooperation (Imai et al., 2005).

1.1 Context and Objectives

The work was conducted in the context of the *SpacePartner* project at Aalto University School of Science and Technology, at the Department of Automation and System Technology. With the support of the European Space Agency (ESA), this project is an extension to the *WorkPartner* project to study the applicability of this centaur-like robot for astronaut assistance in planetary exploration. The overall scope of the *SpacePartner* project is wide and the capabilities of the *WorkPartner* robot are also numerous with its roll-walking lower-body, hybrid energy system, laser scanning and computer vision systems, dual manipulators, and various speech recognition and common awareness control software.

On the upper-body of the *WorkPartner*, the latest work has involved various basic capabilities for manipulator control such as immersive tele-operation (Sheikh, 2008), a real-time simulation platform (Heiskanen, 2008) and a basic admittance controller (Zebenay, 2009). Lately, software and hardware upgrades have

allowed for more advanced control schemes. These upgrades include the implementation of new commercial off-the-shelf motor controllers, from *Elmo*, which allow force control and feedback where previously only position control and feedback were possible. Additionally, the transition from an older *QNX*-based software system towards the Machine Control Interface (MaCI) libraries, developed at Aalto University, also allows for more uniform and concerted efforts in the development of the control software.

This thesis work's objectives are to develop a flexible and simple control software to run control algorithms which will allow the *WorkPartner* to manipulate objects in a human environment with the help of a physically-interacting operator. As this project is the first to involve the newly upgraded hardware and software systems of the *WorkPartner*'s upper-body, a significant amount of work, in addition to that presented here, has been performed by the author, his instructor and many other technicians to develop, implement, test and validate these new systems.

1.2 Problem Definition

One could formulate the core subject of this thesis by the following problem statement:

> *To develop a dual manipulator control software which includes low-level compliant control as well as high-level cognition capable of manipulating objects towards practical goals while coping with and making use of physical human-robot interactions in a manner which is safe for a human operator in the robot's workspace.*

The above statement requires a few additional notes. First, the problem is tackled primarily from a control system's point of view, as it is the author's field of expertise. At the high-level layer of the software lies cognition, but note that this thesis does not concentrate of artificial intelligence algorithms *per se*, but rather hopes offer some basic case-specific solutions. Second, the manipulation of objects towards practical goals is generally limited to the manipulation itself, and thus, does not include pick and place operations which are considered as

problems of a different nature, quite independent from the subject of this thesis and whose solutions available now in the scientific literature are usually sufficient for most applications.

In summary, the sub-tasks that are tackled here include external force estimation, compliant control, physical human-robot interactions solely through force feedback, object manipulation by two robotic manipulators, and behaviour-based robotics and thus, some artificial intelligence. At the other end, the sub-tasks that are circumvented include, for example, computer vision, speech commands or natural language interactions, and autonomous mission planning. Although these aforementioned topics would be very interesting additions to this work, the scope was intentionally limited to clearly defined boundaries in order to better focus on the effort and allow for a timely delivery of the results.

1.2.1 Functional Breakdown

The design of a control system for two robotic arm manipulation of objects is a difficult task because it involves challenges on many levels, from hardware to high-level software. As Figure 1.1 shows, the functionalities required to achieve the task are highly coupled and sensitive. The first challenge is to achieve hybrid control of the manipulators to be able to cope with each other and a dynamic environment (Lewis et al., 2004). Here, hybrid is meant as position and force control which requires the control system to take different forms or behaviours in different directions of motion of the end-effector, see Section 2.2. A second challenge, now at the hardware level, is to obtain useful force feedback from the system, see Section 2.3. The high speed switching regulators most often found in electrical drives cause significant noise in the current sensor's output, making force estimation a daunting task. Furthermore, when human interactions are involved, the external forces are of most interest but add a level of difficulty in the estimation task. This brings about the final main challenge which is to develop a comprehensive high-level software which is capable of synthesizing a proper reaction to the detected human inputs, see Section 2.4. In doing so, the human interaction layer must also bring the mission objectives of the robot in harmony with the operator's physical inputs, this implies a certain level of learning or adaptation in the software as well as a framework for flexible definitions of mission objectives. Lastly, significant software development is

necessary to bring the control laws, estimation laws and mission planning into concrete algorithms, running in a flexible software environment, see Chapter 3.

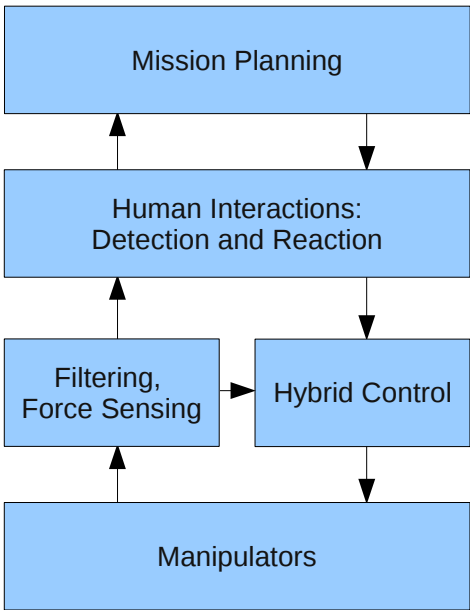


Figure 1.1: Functional breakdown of the manipulation task.

1.3 Outline

Chapter 2 first presents a throughout literature review of the subject. This review is a critical assessment of the various existing hardware and control strategies used in the past for similar problems such as manipulator control in general, dual-manipulator systems, external force estimation, and some basic notions of physical human-robot interaction, common awareness and cognition.

Chapter 3 then demonstrates the implementation details, methods chosen, and innovative efforts of this thesis. An overview of the core software platform elements are presented, followed by selected topics in the mathematical treatment of the proposed control algorithms, and, finally, exposing the implementation of the control architecture of the *WorkPartner*'s upper-body.

Chapter 4 goes on to present some of the early evidence of the suitability of the proposed method using simulated multi-body dynamics. First a simple example of the control of a pendulum is explored for sake of demonstration and

validation of the control architecture and simulation results. Then, simulation results are presented for the *WorkPartner*'s upper-body dynamics and control.

Chapter 5 moves into the actual hardware; presenting the methodology, milestone tests chosen, and ultimately the results obtained when applying the control methods and algorithms to the *WorkPartner*. The experiments presented go from simple dynamics compensation to object manipulation and physical human-robot interaction.

Chapter 6 finally concludes this thesis, presents the challenges ahead in this field and future directions that could be taken for the *WorkPartner* robot and the *SpacePartner* project.

Chapter 2

Literature Review

"Being a philosopher, I have a problem with every solution."
- Robert Zend

Latest trends in space robotics have a strong emphasis on achieving greater autonomy and developing robots that can help humans in a productive manner. One objective at the frontier of human-robot cooperation is the use of robots to manipulate objects and assemble them with the help of other robots or humans. In this literature review, the problem of manipulation of objects with two robotic manipulators physically interacting with humans is studied through the global and specific challenges it poses and the solutions proposed by the latest scientific contributors in the field.

First, a selected set of robots are presented in Section 2.1 with emphasis on actuator, sensor, control software, and achieved performance. These exemplary robots were chosen with preference towards space applications, but some earth-bound system are also presented. Then, the formal requirements of Chapter 1 are assessed with various existing solutions and possible variations upon them. The areas breakdown to: the manipulator control strategies which are able to achieve position tracking, force compliance, and dual-arm manipulation, in Section 2.2; the estimation of external forces via motor current measurements which can be used reliably for full or partial force feedback, in Section 2.3; and, finally, the possibilities for high-level software architecture and algorithms that can cope with the physical interaction of a human in an intelligent manner, in Section 2.4.

2.1 State-of-the-art

Recent directions in space robotics have targeted human-robot cooperation at all levels. Universities, research institutes and space agencies around the world have striven to increase the level of autonomy in robotics systems and the ease of interactions with human operators or astronauts. The inherent dangers of Extra-Vehicular Activities (EVA) in space have motivated the development of space robotics since the CanadArm to the latest Space Station Remote Manipulator System (SSRMS) and Mobile Base System (MBS) modules (Mukherji et al., 2001). NASA has been developing robotics systems for planetary exploration and is now actively pursuing multi-robot systems and human-robot cooperative systems (Rojas, 2009), notably at the Jet Propulsion Laboratory (JPL) and NASA Johnson Space Center (JSC). The Japanese Aerospace Exploration Agency (JAXA) and the European Space Agency (ESA) are amongst the other main contributors to space robotics. The common trends are for developing both robotics systems capable of performing semi- or fully-autonomous tasks outside a space station as well as on the surface of the Moon or Mars, and developing schemes that allow a robot to collaborate with an astronaut which reduces the need for EVA while keeping the operation times and performance up to an advantageous level (Rojas, 2009; Singer and Akin, 2010).

The efforts of the aerospace community mirrors that of the robotics community which has also striven to achieve these technical feats. Leading the way is the Massachusetts Institute of Technology (MIT) which has actively tackled many problems in robot autonomy and artificial intelligence, including bipedal walking, cognition, visual perception, human interactions, machine learning, and robust manipulation. Other institutes and universities around the world are also breaking the barriers of machine autonomy and human interaction in the aim of one day solving this enormous jigsaw puzzle. For earthlings, the applications of machine autonomy are vast, ranging from more flexible manufacturing technology to robots more capable of serving humans on a daily basis. The problem of manipulation of objects with two robotic arms is classic and has started to resound in the field since the 1970s.

In the past few decades, many systems have been developed in the pursuit of collaborative manipulation of objects with two robotic arms. In the earlier days, those systems were regarded as a special case of parallel platforms or closed

kinematic chains and were thus controlled by a centralized controller whose purpose was to resolve the kinematics and dynamics of the closed kinematic chain formed by the two or more manipulators and the object. Later, alternative strategies arose that considered master-slave approaches as well as decentralized approaches (Rojas, 2009).

2.1.1 Special Purpose Dexterous Manipulator - DEXTRE

When it comes to hardware, one leading and used system is the Special Purpose Dexterous Manipulator (SPDM) developed by MacDonald Dettwiler Space and Advanced Robotics (MDR) and the Canadian Space Agency (CSA). The system, shown in Figure 2.1, is a dual arm manipulator where each 7 degree-of-freedom (dof) arm is approximately 3.3 m long and is mounted on a single dof body joint (Mukherji et al., 2001). The unit is part of CSA's Mobile Servicing System along with the *CanadArm2* and the MBS. The SPDM, or *DEXTRE*, is a teleoperated unit which is capable of performing assembly and maintenance operations on the International Space Station (ISS). It is equipped with several interfaces for tooling, docking, and grasping, that are compatible with the ISS' Mobile Base System (MBS), *CanadArm2* and Orbital Replacement Units (ORU), to name a few (Coleshill et al., 2009). Its level of autonomy is essentially nil as it is fully teleoperated either by the ISS crew or the ground operators. As crew time has become one of the most critical factors in achieving the goals of the ISS project, NASA and CSA are working towards delegating more tasks to ground operators (Coleshill et al., 2009). No significant work is known, to the author, to have been done toward autonomous tasks, but it could certainly contribute to a reduction of the burden of the ISS crew members.

2.1.2 The Robonaut

The latest efforts of NASA in humanoid robotics is the so-called *Robonaut*. This human-sized humanoid robot is built as a dexterous torso equipped with two 5 dof arms, each with a dexterous hand (12 dof) and wrist (2 dof) (Ambrose et al., 2000). In addition, this robot is fully capable of operating in space due to its full-featured thermal and radiation shielding. Its sensory system include highly

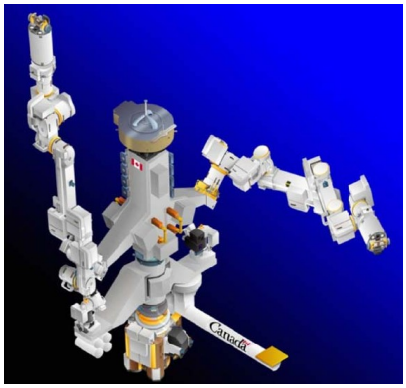


Figure 2.1: Special Purpose Dexterous Manipulator (SPDM) or *DEXTRE*, from Doetsch (2005).

sensitive force sensors that allow haptics feedback to the teleoperator and has thus shown to be capable of performing precise operations making it at least as capable as an astronaut in an EVA suit (Ambrose et al., 2000). *Robonaut* has been augmented to include new sensors and software resulting in increased skills that allow for more shared control with the teleoperator, and ever increasing levels of autonomy (Diftler et al., 2003).

One such autonomy transfer is compliance control at the low-level of the arm controllers which allow smoother and more reliable teleoperation, in the presence of misalignments for example. A set of autonomous grasping motions are also built-in to simplify the control of the 12 dof hands of *Robonaut* (Diftler et al., 2003). Furthermore, autonomous operations are gradually introduced to *Robonaut* via a mesh structure of primitive control nodes, including reflexive grab, haptics exploration, visual perception, *et cetera* (Diftler et al., 2003). The Johnson Space Center is also working with a Cooperative Manipulation Test-bed (CMT) facility which enables them to test both homogeneous and heterogeneous operations with two symmetric manipulators and a third, larger manipulator and all matching tooling and end-effectors. Finally, after a decade of development of the *Robonaut*, it is now being upgraded to *Robonaut 2 (R2)* in cooperation with General Motors inc. to provide a more technologically advanced version of the *Robonaut* (enhanced sensors, controls and drives). The information on both *Robonaut* and *R2* is very limited for public access, so no more can be said here.

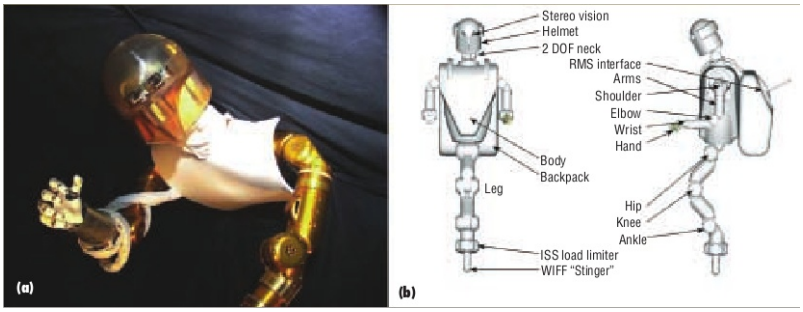


Figure 2.2: NASA's *Robonaut* (a) upper-body and (b) full anatomy, from Ambrose et al. (2000).

2.1.3 Domo and Cog

Recently, at the Massachusetts Institute of Technology (MIT), the *Domo* robot was developed to study artificial intelligence (AI) strategies for robotic manipulation in human environments (Edsinger and Weber, 2004). The Computer Science and Artificial Intelligence Laboratory (CSAIL) has developed, within the Humanoid Robotics Group, a robotic torso equipped with two manipulators. This system combines 29 actuated joints to provide two arm-like manipulators (6 dof each), two hand-like grippers (4 dof each), a pan-tilt neck (2 dof), and a 7 dof active vision head, (Edsinger and Weber, 2004). The subject of this ongoing study includes the vision, grasping, and manipulation tasks. The latter being of most relevance to this literature review. Although it seems little work has been done for manipulation of large objects with both manipulators, the strategies developed for the single manipulator control are very interesting and applicable for two-arm manipulation. *Domo* incorporates force-feedback through Serial Elastic Actuators (SEA) which additionally provide natural passive compliance at high frequencies as well as shock tolerance, amongst other advantages (Edsinger and Weber, 2004). The force control is provided through a typical motor, gear-train and winch arrangement, which for all practical purposes is equivalent to *WorkPartner*'s manipulators. The main innovation in the *Domo* system is on the software side however.

A behaviour-based control system was applied to the manipulation tasks by Edsinger (2004). The architecture is based on the subsumption model, originally introduced by Brooks (1986) with the intended application to mobile robots which he later applied to the *Cog* robot (Brooks et al., 1999). Through the concurrence of various simple behaviours, Edsinger was able to develop complex

yet natural and robust behaviours (Edsinger, 2007) which were found applicable for basic human-robot cooperation (give-and-take style of operations) (Edsinger and Kemp, 2007b,a). Also worth mentioning is the achieved discrimination between external (exo) and internal (ego) forces from the sensor outputs at the actuator joints (Edsinger, 2005), as well as a comprehensive stability analysis which bridges the dreaded gap between classical control system theory and artificial intelligence theory (Pratt, 1995).

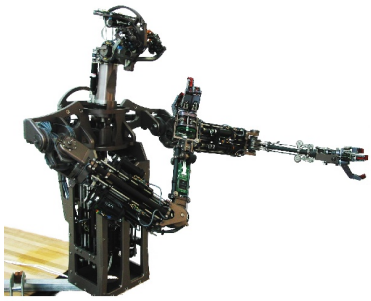


Figure 2.3: MIT's *Domo* robot, from Edsinger (2004).

2.1.4 WorkPartner

Last but certainly not least, the *WorkPartner* robot is a centaur-like robot developed by Aalto University School of Science and Technology (Aalto) at the Department of Automation and System Technology, now in collaboration with ESA for the next stage of the project: *SpacePartner*. This robot is equipped with a hybrid rolling-walking lower body and a humanoid torso. The torso has two symmetrical 5 dof manipulators with single degree of freedom grippers. The actuation and sensor system of the dual manipulators is a classic motor, gear-train, encoder, and electronic controller on each joint. The commercial *Elmo* controllers are capable of controlling position, velocity, or current, and also relay the position feedback of the encoder along with the current sensing output; all on a single CAN Open interface. So far, dual arm manipulation of an object has not been studied, but previous work has involved advanced teleoperation (Sheikh, 2008) and compliant control of one manipulator for human-robot interaction (Zebenay, 2009), however, the time frame of the thesis have limited the extent of the experiments conducted to assess the performance of the proposed control algorithms. At the present, the *Elmo* controllers are still in the process of integration to the hardware, while the control software has been upgraded as

well as the simulator for the *WorkPartner*, called *SimPartner*. The robot is shown in Figure 2.4.

Figure 2.4: Aalto's *WorkPartner* or *SpacePartner* robot.

2.2 Manipulator Control Strategies

As mentioned before, the compliant control of the *WorkPartner* robot has been the subject of a previous thesis at Aalto, and the subject was examined throughout (Zebenay, 2009). However, in the light of the new challenges that dual manipulation poses, it is relevant to revisit some of these approaches. The control of the manipulators for use in a coordinated manner poses the following challenges, mainly compiled from the works of Bonitz and Hsia (1996); Rojas (2009).

- A controller shall be able to achieve position control and force control objectives, *exempli gratia*, pushing an object on a wall or snap-on at a precise position.
- The dynamics of the handled object need to be incorporated in the control scheme, yet the off-line definition of the object's dynamics should be minimal, *exempli gratia*, defining the handled object as free or unconstrained but without needing to specify its mass or dimension.
- The computational burden of the control should be minimal to allow a tighter control loop, with gain in reactivity and stability in the compliance or admittance sense.

- The controller should not rely on accurate and complete force feedback since it is expected that such measurements are inherently difficult to get, *id est*, it should be robust to unreliable force feedback.
- The modelling of the manipulator and object dynamics should require as little *à priori* knowledge as possible and should be robust to uncertain or dynamic environments, which include the manipulators, objects, and environments.
- The controller should be as fault tolerant as possible, *exempli gratia*, in case of joint failure or collisions, the controller should remain stable and safe.

The above puts a clear burden on the design of an appropriate controller. As with many other projects, the incorporation of modularity and flexibility in the adopted control scheme will allow better development and smoothen the path through the design iterations that will certainly be a characteristic of such a design endeavour. From the author's research, two distinct categories of control strategies have been found: Impedance-Based Control and Behaviour-Based Control. Thus, the following two sub-sections will summarize the published work in both of those categories. Similarities and differences will also be drawn with relation to the aforementioned design challenges or objectives.

2.2.1 Impedance Control for Position and Force

The concept of impedance control enjoys a wide body of research and many variants have been developed and tested to the point that it becomes difficult to narrow down to the essential concepts of impedance control. The predominant original development of impedance control is found in (Hogan, 1985a,b,c). The fundamental theoretical finding of Hogan (1985a) was the distinction of admittance and impedance in multi-body dynamics. These two relations define the causality between velocity and force. An admittance is one where an applied force will uniquely cause a motion (e.g. a free mass), while an impedance is one where a motion will uniquely cause a force (e.g. a spring or a damper). As an analogy on which the terms admittance and impedance are based is in electrical systems, where an impedance is characterised by a electromotive force

(voltage) induced by a flow or accumulation of charge such as in a resistance or a capacitor, respectively. While an admittance is charactised by a voltage which admits or causes a flow of charge, typically called an inductance. Hence, the relationships between currents and voltages are analogous to those between velocities and forces.

In Hogan (1985a), it was also clearly stated that a manipulator is, in all controlled degrees of freedom, an admittance because actuation forces will cause motion. So, an impedance controller turns the dof of the manipulators to impedances by closing the feedback loop from position sensor to force actuators, *exempli gratia*, a PID controller is one such impedance controller. It is evident by going back to its primal definition in (Hogan, 1985a) that the concept of impedance control is incredibly general. The most useful idea of it is to realize the duality of impedance and admittance, *id est*, to control an admitting environment (manipulating objects), an impedance controller is needed and their match is what characterizes the performance or behaviour of the system. The novel idea of impedance control is to tune the match and allow hybrid behaviours in orthogonal subspaces of the degrees of freedom of the manipulator. The real question is: How?

Lewis et al. (2004) go into great depth in describing different control schemes including: hybrid position/force control, impedance control, stiffness or compliance control, and computed-torque control. Zebenay (2009) summarized all of these in the process of selecting the best scheme for manipulator control with physical human robot interaction. However, as the previous paragraph hints at, these schemes are not fundamentally different (Pratt, 1995). As a start, the so-called "computed-torque controller" is really not a controller by itself but a scheme very well known in classical control theory as "feedback linearisation. It is used as an inner loop that uses the sensed states of the manipulator along with an accurate dynamic model to cancel out the undesirable non-linear terms to ease the development of the outer control loop. This scheme violates many of the design challenges in the introduction such as robustness to uncertainty, computational burden, and fault tolerance, and hence, it is usually taught in control theory, even at beginner level, not to use feedback linearisation whenever avoidable.

One early approach was the so-called "hybrid position/force controller", first

introduced by Raibert and Craig (1981). This concept introduces two distinct but complementary control loops. The task-space is divided into two orthogonal spaces, one whose environment is admitting and the other is impeding motion. The admitting space can be controlled to desired position via an impedance controller while the impeding space (e.g. physical constraints) can be controlled to desired applied force via an admittance or compliance controller. As noted by Bonitz and Hsia (1996), problems with tuning Cartesian position control gains and the compliance of the force control loop has effectively turned this scheme into an impedance control scheme. With respect to the aforementioned design challenges, this scheme mainly suffers from the high level of environment modelling required to implement the controller, which was acceptable to its original applications in industrial robotics for highly controlled environments and is thus used extensively (Rojas, 2009).

A later proposal, the admittance control, built on the same idea as the hybrid controller but in this case, the position controller is used for the entire workspace with the addition of an external force control loop. The force loop, as an admittance controller, would compute desired positions, in the subspace where compliance is needed, that will cause the position controller to apply the desired forces. This was applied successfully on the WorkPartner (Zebenay, 2009). The main advantage of this scheme is to overcome the problem of harmonizing position and force control outputs to avoid saturation of actuators or other undesirable effects, since a single low-level controller is driving the manipulator. However, a major draw-back, as expressed by Bonitz and Hsia (1996), due to small-gain theorem, the gain of the admittance function of the force control loop is limited by the gain of the position controller to guarantee stability.

Active Stiffness control, first proposed by Salisbury (1980), builds on the concept that stiffness (or conversely compliance) is the measure of how accurate the positioning can be or of how strongly the manipulator enforces a desired position of the end-effector. Starting with a desired stiffness matrix in Cartesian coordinates at the end-effector, where certain directions have lower stiffness (force control) than others (position control). This Cartesian stiffness matrix can be mapped to the joint-space by the Jacobian of the manipulator, thus imposing a stiffness value for each joint controller, but also cross-coupled to other joints, *id est*, position/velocity error in one joint generally affect the actuation of all other joints. Note that it uses a similarity mapping and thus, does not require

the computation of the inverse of the Jacobian (Salisbury, 1980). This scheme also includes the necessary damping term, for stability, as well as the feedback linearisation terms characteristic of all "computed-torque controller" and an optional force-feedback term with a compensation gain matrix. As noted by Salisbury (1980), several issues of digital control and instability resulting from frequency aliasing arose in the application and significant efforts in digital signal processing (DSP) were required. As noted by Yang et al. (1993), the active stiffness control presents a very restrictive trade-off between the range of applicable stiffnesses and the robustness of the controller due to the controller gains. They proposed the use of sliding mode control. In this scheme, the controller gain is computed from the desired stiffness in such a way that the states and control inputs of the system are brought, robustly, to a control surface where the desired performance is prescribed and achievable (Yang et al., 1993). As for the design objectives, this scheme is an improvement with respect to computational burden, robustness and fault-tolerance, however, it still suffers from lack of modularity and flexibility as well as a high level of knowledge of the environment's dynamics.

Internal-Force Impedance Control is the last method of interest presented here. This scheme, presented by Bonitz and Hsia (1996), tackles the problem of coordinated manipulation of an object with two or more manipulators (homo- or heterogeneous). The measured or estimated forces, at the end-effector, are decomposed, based on the Jacobian of the manipulators, into internal forces and external forces. In this case, the internal forces are meant as internal to the manipulated object. By injecting a force control element into the impedance control law, the internal forces can be controlled to a desired value, *exempli gratia*, not to crush the object nor to let it go. The advantages of this method are: only one control loop is necessary (no position and force control loops); the object's dynamics don't contribute to tracking errors in positioning the object; finally, the internal forces are decomposed based on the sensed forces and manipulator kinematics, and thus, do not require a dynamics model of the object (Bonitz and Hsia, 1996). However, the issue of human interaction or compliance to the environment is not addressed. It also lacks flexibility and is fairly heavy, computationally, requiring large matrix multiplications and general inversions.

2.2.2 Behaviour-Based, Virtual Model Control

The most concrete foundations of Behaviour-Based Approach (BBA) was first laid out by Rodney Brooks 1986 via his subsumption architecture. The idea draws from the occurrences in nature of complex behaviours which emerge from a simple set of primitive behaviours, such as an ant colony. Initially developed with the aim of autonomously controlling mobile robots such that they could move in a dynamic environment and robustly perform complex tasks. The real power of this approach was the intuitive prescription of simple desired behaviours which could concur into more complex emergent behaviours.

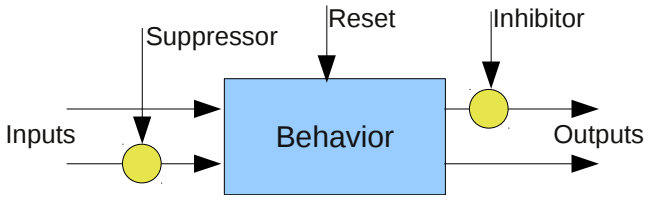


Figure 2.5: Simple behaviour block in the Subsumption Architecture (Brooks, 1986).

Since then, Brooks has extended the framework to manipulators such as *Cog* in Brooks et al. (1999), and others have followed. Still mainly within different work groups at MIT, BBA was successfully applied to two planar biped robots: the *Spring Turkey*, developed by Peter Dilworth and Jerry Pratt in 1994 (Pratt, 1995); and later, the *Spring Flamingo*, developed by Jerry Pratt in 1996 (Pratt et al., 2001), which essentially differs from the *Spring Turkey* by its serial elastic actuator joints and additional degrees of freedom. The true breakthrough initiated by these projects was the introduction by Pratt (1995) of the Virtual Model Control (VMC) principle. Akin in nature to other techniques such as Virtual Reality and Haptics Interfaces where virtual dynamics models are used to generate haptics (or force) feedback to a human operating in a virtual environment, VMC uses a concurrence of simple virtual dynamics elements (springs, dampers, masses, *et cetera*) to prescribe or entice the robot into performing the desired motion or exhibiting the desired behaviour. The idea was radical, but otherly intuitive. Later, Aaron Edsinger would apply this very same concept to the control of the *Domo* robot as mentioned in the introduction, proving that this approach's power reaches beyond dynamic walking, to manipulation tasks and possibly to two arm manipulation as well. Pratt formulates the main

characteristics of his novel approach in his master's thesis (Pratt, 1995), and so he expresses it:

- Virtual model control is an intuitive language for describing complex motion control tasks.
- Virtual model control allows for the use of generalized variables and generalized force functions.
- Virtual model control can be implemented on any serial link or set of serial links on the robot and not just between a base and an end-effector.
- Virtual model control can be implemented on serial or parallel, redundant or under-actuated, fixed or free-flying robotic systems.
- Virtual model control allows one to specify mechanical constraints, such as unactuated joints, or design constraints, such as force equalization.
- Adaptive and learning techniques can be implemented with virtual models, thereby creating adaptive or learned virtual components.

The implementation of Virtual Model Control includes distinct parts. First a model of the robot is developed for the purpose of direct kinematics, and in some cases, if necessary, inverse dynamics. Then, in very much the same way, virtual dynamics elements are developed as a "toolbox" for implementing the simple behaviours. Finally, at a higher level in the software (cognitive level), the simple behaviours are constructed, reconfigured and manipulated in real-time. The principle of VMC also extends naturally to machine learning and adaptation. Hu et al. (1998) showed that additional robustness could be achieved with adaptation laws on a bipedal walking robot and it was proven that stability is achievable in certain directions only (forward walking direction), but this was a result of the inherent under-actuation of a bipedal robot with free ankles (Hu et al., 1998). All these parts are simple and intuitive to build, given the proper framework.

The C++ programming language, now in standard use, offers great possibilities for object-oriented programming as well as template meta-programming (Stroustrup, 1997), it is a candidate of choice to form the basis of any programming framework. Since virtual model control is so strikingly close to multi-body

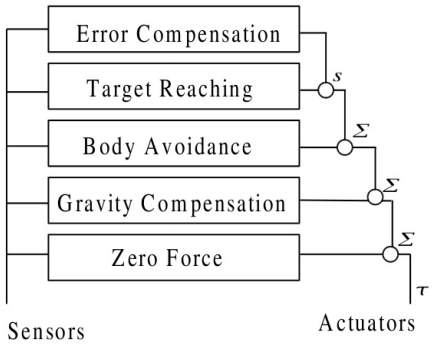


Figure 2.6: Example of Virtual Model Control for a manipulator, from Edsinger (2004).

dynamics simulation, it makes perfect sense to explore the types of frameworks used in this area.

Object-Oriented Programming and Multi-body Dynamics

One of the milestones in multi-body dynamics was the creation of ADAMS, now proprietary of MSC. Orlandea (1987) proposed a complete framework for multi-body dynamics simulation which summarized, in a well structured and rigorous way, the issues and underlying structure of such simulators. The continued success of this software platform is a testament to its qualities. Since then, much progress has been done mainly propelled by the adoption of object-oriented programming strategies. A myriad of software platforms exist from the crudest to the most clever.

Many have taken approaches which group the elements of MBD simulation into class hierarchies which reflect classical mechanical engineering analyses (Heiskanen, 2008), (Persson, 2007), (Jet Propulsion Laboratory et al., 2008). It is classic and intuitive to group classes under a hierarchy which reflects the human intellect. In these cases, the usual MBD simulation libraries include familiar constructs such as frames, rigid-bodies, actuators, sensors, *et cetera*. Although it seems conceptually sound to do so, Sutter and Alexandrescu (2004) repeatedly point out the dangers of creating class hierarchies based on conceptual grouping rather then through classifications oriented towards functionality, as in the coined terms: "programming by contract" and "classification by responsibility". Furthermore, a good example can be very simply drawn from the acronym of the CLARAty software: Coupled-Layer Architecture for Robotic Autonomy.

As Sutter and Alexandrescu constantly remark on (Sutter and Alexandrescu, 2004), the main recipe for a good software design is independence, *id est* decoupling the layers of a software always favours modularity, flexibility, portability, and a myriad of other desirables. Most of these classical realization of MBD simulation have suffered tremendously from these problems, with little gain.

In the last decade, alternative forms of classifications of MBD simulation software has given rise to far more productive and flexible frameworks. As introduced in (Kecskeméthy et al., 2001) and additionally exemplified in (Romano, 2003), a novel structure was presented which utilizes so-called *Kinetostatic Transmission Elements* (KTE) to generalize all dynamics elements. This is a true example of "generic programming" which relies on extracting the fundamental functions of an element as an interface and abstract away the rest, unleashing the power of object-oriented programming. Unfortunately, it's impossible to go into great depth in describing this framework in this literature review, but the idea behind KTEs can be summarized as a generalization of all dynamics elements as elements which map generalized coordinates and their derivatives (*kineto-*) as well as the forces (*-static*) from one smooth manifold to another, *id est* kinematics are mapped forward and dynamics backward. Romano (2003) gives a nice and complete example of how this framework generalizes beyond multi-body dynamics and can be used to simulate virtually any analogous system (e.g. electric circuit, control system, *et cetera*). As a final thought, I remark upon the fact that these kinetostatic transmission elements are exactly analogous to the virtual model control's building blocks presented by Pratt (1995) and Edsinger (2004).

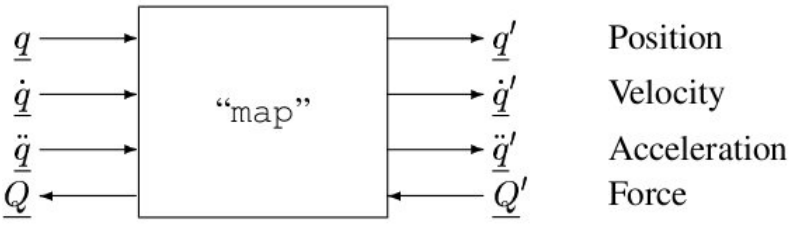


Figure 2.7: Simple Kinetostatic Transmission Element, from Kecskeméthy et al. (2001).

2.3 External Force Estimation

A major issue for both compliant manipulation and human interaction is the sensing of forces. As noted in Edsinger (2005), there are several distinct types of torques experienced at the joints:

- τ_{exo}, the torques resulting from the environment or human interactions.
- τ_{ego}, the torques resulting from one manipulator or the robot's body on the other manipulator.
- $\tau_{dyn} = M(q)\ddot{q}+C(q,\dot{q})\dot{q}+\tau_f(\dot{q})$, the torques induced by acceleration effects, e.g., centripetal, Coriolis, and d'Alembert.
- $\tau_{grav} = G(q)$, the torques resulting from gravity.
- τ_{mot}, the torques applied by the motors and sensed via current sensing.
- $\tau_{ext} = \tau_{exo} + \tau_{ego}$, the external torques coming from the environment, the human, or self-induced.

The challenge here is to estimate τ_{exo} and τ_{ego} from only the measurements of τ_{mot} and the state variables q and $\dot{q}$. In Edsinger (2005), the problem is much simplified with the use of SEAs which measure the total force at a joint, including all of the above, and the problem is merely to extract τ_{exo} and τ_{ego} from those measurements. In addition to the obvious mathematical challenge, the physical reality adds noise, instability, and uncertainty to the sensed motor currents. All in all, the requirements for external force estimation can be summarized as follows:

- The estimation shall optimally use the current sensing and other feedback and feed-forward terms to achieve best and most reliable estimation of the motor, total, and external torques.
- The estimation shall be able to converge to reliable estimates at run-time and shall settle quickly enough to allow a reasonable reaction time.
- The estimation should be as accurate as possible to allow for good control performance.

- The estimation should be deterministic and repeatable such that both manipulators have well defined and repeatable behaviours.
- The estimation shall compensate for static friction or stiction, a classic problem in force estimation based on motor current sensing.

Aksman et al. (2007) just recently tackled this exact problem and achieved, as they put it, a first attempt at estimating the external forces on a manipulator using current sensing. The following treatment of the subject will be largely based on their findings. They demonstrated the use of an adaptive learning algorithm to obtain a good estimate of the dynamics model of the manipulator, undisturbed, and then using this model to eliminate all internal forces, including stiction via a RBF neural network. As an alternative, the results of Chen et al. (2000); Korayem and Haghighi (2008); Nikoobin and Haghighi (2009) also have achieved good performance that show that it is possible to observe the external forces, or disturbances, from a non-linear observer law which uses only the joint position and velocity measurements. In this text, the following equations will be used as a basis for the dynamics of the manipulator:

$$(M(q) + G_m^2 I_m)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_f(\dot{q}) + g(q) = \tau_{mot} + \tau_{ext} \tag{2.1}$$

$$\tau_{dyn} + \tau_{grav} = \tau_{mot} + \tau_{exo} + \tau_{ego} \tag{2.2}$$

where $M(q)$ is the generalized inertia matrix of the manipulator, G_m is a diagonal matrix of gear ratios, I_m is a diagonal matrix of motor inertias, $C(q,\dot{q})$ is the Christoffel matrix of centripetal and Coriolis effects, $\tau_f(\dot{q})$ is the friction torques, $g(q)$ is the gravity load, and others are as defined previously.

2.3.1 Adaptive Robust Control Approach

After a throughout examination of literature, Aksman et al. (2007) have devised a method for estimation of external forces applied to a manipulator based on motor current sensing. The *Elmo* controllers used on the *WorkPartner* sense current in the motor through a low-side shunt resistance which are a reliable and cost-effective means of sensing the current with the main drawbacks of noisy measurements and inherent bias coming from the leaking current in the voltage measuring amplifier (Lepkowski, 2003). As a consequence, the use of low-pass

filtering is necessary as the first DSP stage of the force estimation, this can easily be applied directly at the output of the current sensing, *exempli gratia*, Aksman et al. (2007) used a 20Hz cut-off frequency for a 3kHz sampling rate, typical figures in this field. Once a "clean" measurement of the motor current is obtained, it can be transformed to a motor torque via the gear-ratio and torque constant, given that they are accurate, through the following relation:

$$\hat{\tau}_{mot} = G_m K_m i_m \tag{2.3}$$

where G_m is the gear-ratio, K_m is the torque constant, and i_m is the motor current.

The central idea for estimating the external torques is to use an internal model of the manipulator, precisely obtained through an adaptation law, fed with the current estimated state of the manipulator (position, velocity and acceleration) to estimate the left-hand side of Equation 2.1, and finally, subtracting the sensed motor torques to be left with an estimate of the external torques. As shown in Figure 2.8, the area outlined as force estimation does exactly that. The main issue with this principle is the reliance on a precise model of the manipulator's dynamics, stiction being especially difficult to model. To cope with this, Aksman et al. (2007) have developed a two-stage approach where in the training mode, the manipulator is left free and runs a number of prescribed trajectories to "learn" its dynamics model within its workspace via a robust adaptive control law. Then, the mode is switched to estimation mode where the dynamics model is frozen, after about 10 minutes of training in (Aksman et al., 2007), and used in the external force estimation procedure described above and shown graphically in Figure 2.8. The modelling of the stiction can be achieved via a Radial-Basis Function (RBF) Neural Network, which has the proven capability of modelling any general function to an accuracy dependent on the number of hidden-layer nodes. The details of this adaptive law will be skipped for the sake of conciseness but the reader is referred to (Aksman et al., 2007) for further details. In this control architecture, Aksman et al. (2007) have also demonstrated the concept with the use of a compliance controller (or admittance controller) along with feedback linearisation with the help of the adaptive dynamics model. It should be noted, however, that the use of any control scheme is equivalent since the robustness and quality of the external force estimation does not rely on a particular controller.

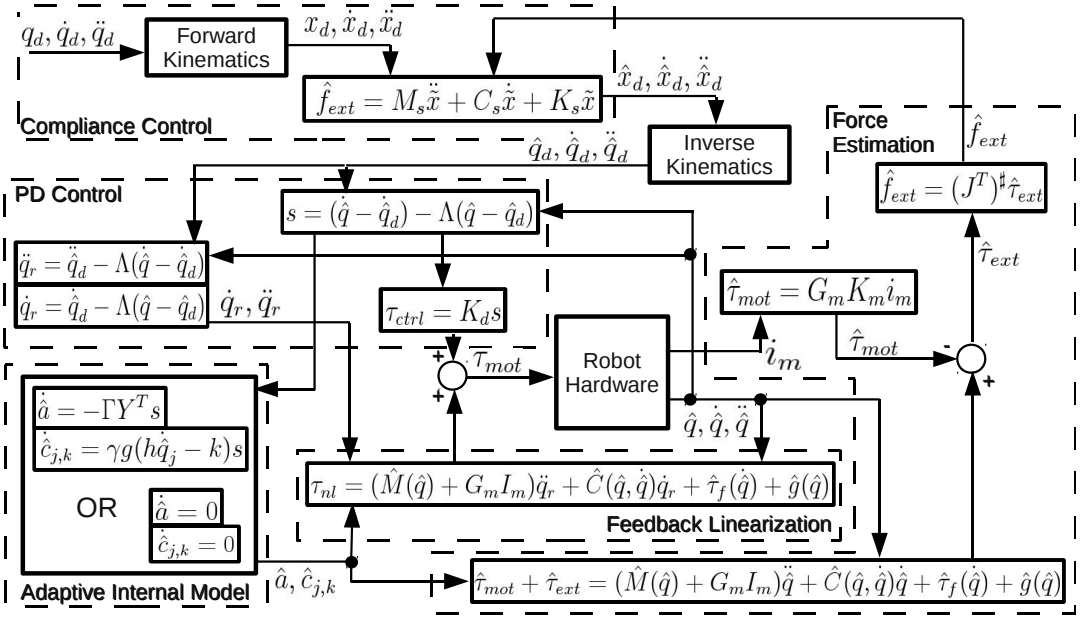


Figure 2.8: Block-diagram of adaptive model for external force estimation with compliance control (Aksman et al., 2007).

This estimation scheme essentially meet all requirements expressed in the introduction of this section. However, it does present a certain number of problems. First and foremost, the reliance on acceleration estimates of the joint variables can cause severe problems of noise amplification during finite differencing and suffers from frequency aliasing effects when applied to discrete encoder increments. Aksman et al. (2007) resolve the issue with low-pass filtering of the differential signals, but this is not a perfect solution and introduces problems of its own such as phase lagging. The two mode approach is also a problem since, in theory, training sessions are required on a fairly regular basis.

2.3.2 Non-linear Disturbance Observer

Non-linear Disturbance Observer design is a topic that already enjoys a wide body of literature for robust manipulator control. The developments in this field have started at the turn of the 21st century and has boomed in this last decade. Techniques for both linear and non-linear observers have been devised using rigorous methods of robust control theory such as H_∞ optimal control, μ-synthesis, and Lyapunov's direct method. Not to overly burden the reader with cumbersome mathematics, a single such method will be put forth which is

both simple yet representative of the main ideas behind disturbance observers. The method, initiated by Chen et al. (2000) and extended by Korayem and Haghighi (2008) and later by Nikoobin and Haghighi (2009), uses Lyapunov's direct method to achieve an observer law which is asymptotically stable to estimate the external disturbances to the manipulator. Starting from equation 2.1, the error in the estimate of the external torques is used to correct the estimated disturbances using the following relation:

$$\dot{\hat{\tau}}_{ext} = -L(q,\dot{q})\hat{\tau}_{ext} + L(q,\dot{q})(M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) - \tau_{mot}) \tag{2.4}$$

where the only new term, $L(q,\dot{q})$, is the observer gain forming the basis of the observer law. The above implies that $L(q,\dot{q})$ needs to be chosen such that it stabilizes the error to zero. One problem remains, as expressed in the previous subsection, it is undesirable to use an estimate of the joint accelerations in the estimation because of noise amplification and frequency aliasing. To eliminate this need for $\ddot{q}$, Chen et al. (2000) proposed a clever trick with the following change of variables:

$$\hat{\tau}_{ext} = \psi + p(\dot{q}) \tag{2.5}$$

$$\dot{\hat{\tau}}_{ext} = \dot{\psi} + \frac{\partial p}{\partial \dot{q}}\ddot{q} \tag{2.6}$$

$$\frac{\partial p}{\partial \dot{q}} = L(q,\dot{q})M(q) \tag{2.7}$$

With this new set of equations, the estimation of the external torque vector is now achieved through the estimation of the variable ψ. The new estimation equation becomes:

$$\dot{\psi} = -L(q,\dot{q})\psi + L(q,\dot{q})(C(q,\dot{q})\dot{q} + g(q) - \tau_{mot} - p(\dot{q})) \tag{2.8}$$

Note the disappearance of the acceleration term from Equations 2.4 to 2.8. The new function $p(\dot{q})$ can be chosen by Lyapunov's direct method such that it guarantees asymptotic stability. Nikoobin and Haghighi (2009) proposed such a function for an N-revolute-joint manipulator and simulation shows drastic improvement on the tracking performance of a computed-torque position controller for a 3-link manipulator. This on-line estimation method is robust and reliable, does not require the estimation of joint acceleration, but it does require the inversion of the generalized inertia matrix of the manipulator, a reasonably good dynamics model, and some matrix multiplications. It should be noted, however, that this method does not exclude the previously mentioned method by Aksman et al. (2007), it can be substituted directly in place of the force estimation algorithm, preserving the advantages of the adaptive internal model.

2.4 Physical Human-Robot Interactions: Detection and Reaction

The issue of detection and reaction to physical human-robot interactions has been a very important topic in the last decade as more and more robotics systems aim to work in human environments. The paramount issue is the safety of human operators who share the workspace of the robot. The PHRIENDS work group (PHRIENDS et al., 2008) has dedicated several years of research on establishing milestones and quantitative measures of the impact forces and possible harm a human-robot collision can cause. They have also studied various strategies, mainly involving the methods presented in Section 2.2, to reduce the impacts and favour safe responses by the robot. Haddadin et al. (2008) presented the latest and most comprehensive study at DLR using the Lightweight Robot III (LWRIII). Through extensive experimentations with various control schemes and various methodologies, their conclusions can be summarized in the following points:

- External torque estimation through either schemes presented in Section 2.3 provide a suitable and fast enough sensory input to detect collisions.
- The use of high-pass filtering of the estimated external forces can contribute to faster reactivity especially in compliant control schemes.
- The reaction strategy that gives the most natural feel to the interacting human is to stop all position tracking controls and leave simple gravity compensation control to make the robot "weightless".
- The reaction strategy that reduced the impacts the most were those which "flee" the collision or disturbance, such as an admittance controller with incentives in the opposite direction from the collision or a simple reversal of the motion of the robot, *id est*, reversing the desired trajectory or back-tracking.

All in all, these conclusions agree with what one would expect from intuition alone, but Haddadin et al. (2008) proved experimentally that those guidelines are safe and sound. Again, it is important to remark how the above guidelines are fully compatible with all that was presented in Sections 2.2 and 2.3. Given

that the above can guarantee safety, it is time to ask how can humans and robots collaborate?

2.4.1 Physical Interactions vs. Cognitive Interactions

As noted by Guo and Sharlin (2008), the use of our innate skills, as humans, to interact with physical objects can be used to greatly reduce the new skills an operator needs to acquire when collaborating with a robot. Since (Guo and Sharlin, 2008) is simply a presentation of future research developments, it has yet to produce results. However, several key concepts are presented for remote control of robots through tangible user interfaces. In the context of physical human-robot interface, the problem is simpler because the human operator has direct access to the robot's workspace and full common situation awareness, for the robot, it's another situation altogether.

As reported in Imai et al. (2005), the human-human cooperation, communications, and collaborations are driven by a "mind-reading" behaviour, *id est*, the humans will infer the others' intentions as a mean of getting immersed in the common situation. The proposal in (Imai et al., 2005) is to increase the willingness of humans to interact with robots through incentives that make humans infer the intentions of the robot which they are generally refractory to do because of the artificial nature of a robot. Although this is not the case for physical human-robot interactions as it is meant in this literature review because the willingness of a human operator to interact with the robot is the given starting point. However, it does hold the basic necessary human behaviour that the robot needs to emulate in order to get immersed in the physical interaction that the human operator offers, that is, inferring the intentions of the human. For the robot to understand the intentions of the human, one can use the so-called "action-based approach for mind-reading" which uses the actions of the human in addition to the common goal or target to deduced the intentions (Imai et al., 2005). For example, say the task is to nail a piece of wood on a wall, the robot holds the piece of wood while the human drives in the nail. The robot's goal is to keep the piece of wood on the wall. If the human pushes on the piece of wood to change its placement on the wall, the robot can easily infer the human's intention to apply a correction to the placement. Furthermore, if the human pulls the piece of wood away from the wall, it is

clear that his intentions are to suspend the task, say to clean the wall's surface before nailing the piece of wood. This idea of inferred intentions combines very naturally to the subsumption architecture where passive behaviours can observe the human's interactions and be triggered into an active state and suppress or inhibit other behaviours.

As exemplified by Arai et al. (2000), the realization of collaborative manipulation of objects can be achieved through the application of additional constraints on the motion. In their paper, Arai et al. showed how the application of a non-holonomic constraint on the compliant motion of the robotic manipulator can be used to achieve safer and more reliable manipulation. Again, this goes to strengthen the idea that a behaviour-based approach to controlling the manipulator is a general framework in which physical human-robot interaction and collaboration can be naturally implemented.

Chapter 3

Implementation

"Tout arrive à point à ceux qui savent attendre."
- French Saying

"Tout arrive plus vite à qui court après."
- Québécois Saying

This chapter presents the details of the implementation of the control software developed and tested in this thesis work. At first, some important and innovative elements of the software framework are presented, namely the implementation of the Subsumption Architecture into a massively multi-threaded application, and the development of a modular multi-body dynamics modelling via kinetostatic transmission elements. Then, the fine details of the construction of a complete control software is exemplified by a simple pendulum position-controller. And finally, the outline of the construction of the control software for the *WorkPartner* is presented.

It should be noted, as a starting point, that the implementation of the control software was entirely written and tested using the C++ programming language. This choice was motivated by the many factors including the availability of standard libraries, the use of the Machine Control Interface (MaCI) library developed locally at Aalto which provides a robust hardware interface, the author's expert knowledge and experience with C++, among many other more classic arguments in favour of this low-level object-oriented programming language. Also, it should be said that the following chapter can only give a broad

overview of the actual implementation of several tens of thousands of lines of code for which only the source code documentation can give a complete and detailed account of, and thus, if the reader is looking for such detail, he shall find them in the latter document accompanying the thesis submission.

The literature review of the previous chapter has shed light on many issues of collaborative physical human interaction with a dual-manipulator robotic system. The solutions that were presented on the three major sub-topics show that existing technology, control schemes, and artificial intelligence algorithms are suitable to solve the overall problem.

Arguments in support of the impedance control strategies mainly involve the massive body of research already available on the subject which provides a strong theoretical basis on which to develop the control software. As for the behaviour-based approach, the harmony that exists between the cognitive-layer and the control-layer of the software architecture is the strong incentive that led to the choice of this strategy for the implementation, as seen in Figure 3.1. Additionally, BBA offers the modularity and flexibility in implementation that most impedance controllers lack.

In light of all things, a behaviour-based approach through a virtual model controller was chosen. The motivation for VMC is through the realization by Pratt (1995) that impedance control can be implemented as a VMC, and thus, such a control scheme can still benefit from the body of research in this field and eventually bridge the gap between classical robust control theory and artificial intelligence approaches.

3.1 Subsumption Architecture

The following section presents a description of the implementation of the underlying software framework used to materialize the behaviour-based approach. Many researchers have criticized the Subsumption Architecture as a basis for the entire control software of a robotic system. Their critiques mainly involved the so-called intermediate-layer problem which arise as a tight reactive control loop, required for safety and stability purposes, is combined with a planning or cognitive layer typically of much slower time scale and larger space resolution,

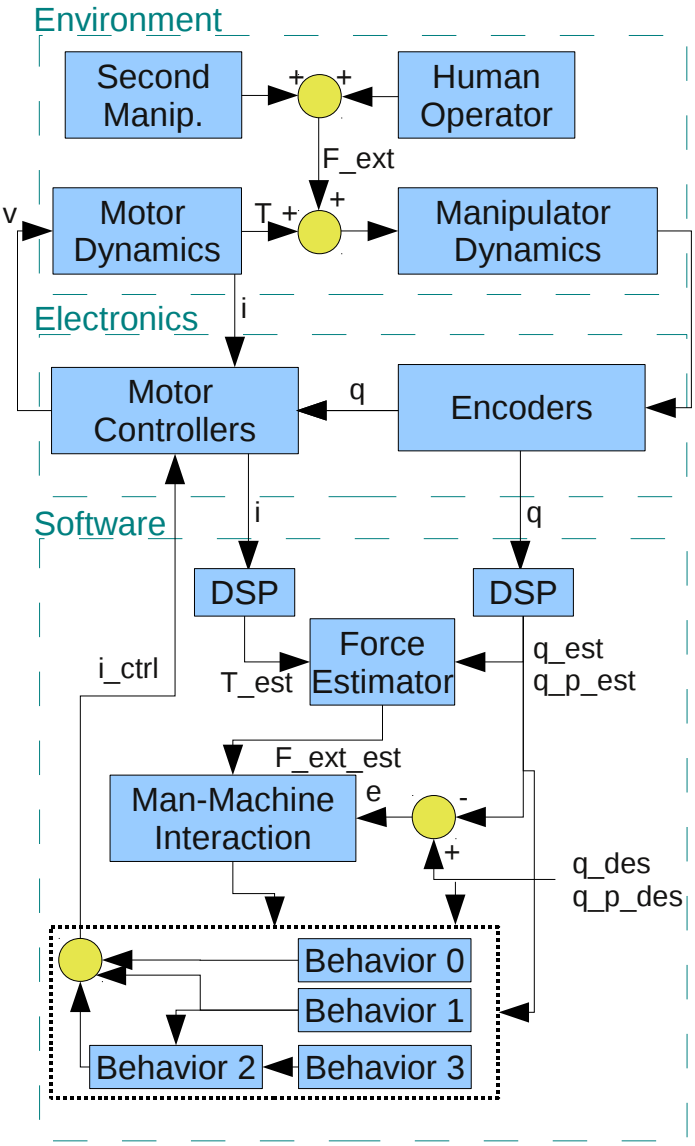


Figure 3.1: Overall system for a behaviour-based approach.

exempli gratia, a reactive wall-avoidance control loop and a global path-planner, in the case of a mobile robot. However, the widespread use of multi-threading in modern software provides a simple solution to this problem, and thus, the task of implementing the Subsumption Architecture will reduce to proper use of multi-threading and synchronization techniques.

3.1.1 Asynchronous Signals and Systems

The topic of asynchronous signals and systems is very well known in the software engineering field and many specific platforms offer these capabilities while

countless more use them. The underlying idea is the use of independent systems, *id est*, atomic units of the software which run in a loop, on individual threads, and process certain input signals and produce output signals. The signals, in turn, are simple synchronization interfaces which hold the written values which could be read after each write operation (synchronous or discrete-time signals) or at any given time (asynchronous or continuous-time zero-order held signals). One can, for example, use such an architecture to simulate an electronics system where there are several units which perform different tasks and some are clocked, some are triggered, and some are purely analogue. In the robot control software context, the inherent qualities of this architecture are utilized for analogous reasons:

- The ability to synchronize the systems through discrete signals allows for a sequence of control computations to be triggered by the measurement loop and close the chain of subsystems towards the control hardware.
- The synchronization guarantees that only the minimum amount of necessary calculations are done with respect to the capabilities of hardware, *id est*, no more than one loop will be performed with the same feedback information.
- Asynchronous signals for their part allows certain subsystems to take longer to produce outputs without affecting the speed of the underlying control loop, *exempli gratia*, the external force estimation subsystem can take a longer estimation settling-time to produce intelligent reactions to the human input while another subsystem can watch via a high-pass filter for an impact force on a much tighter reactive control loop.
- The independence of the different systems allows for dynamic reconfiguration of the control software, which is the essential idea behind the inhibitors and suppressors of the classic Subsumption Architecture by Brooks. This is a capability akin to shifting gears on an auto-mobile, but much more powerful as it allows shifts in behaviour.

As for the actual implementation, it is quite straight forward and many libraries exist to ease the way. In this case, the Boost.Thread library was used for threading and synchronization, because of it's quality, widespread use and

maintenance by the Boost peer-review committee. No other dependencies were needed and it was decided to implement the architecture for signals and systems without the help of any third party open-source software for which neither the quality nor the suitability can be guaranteed.

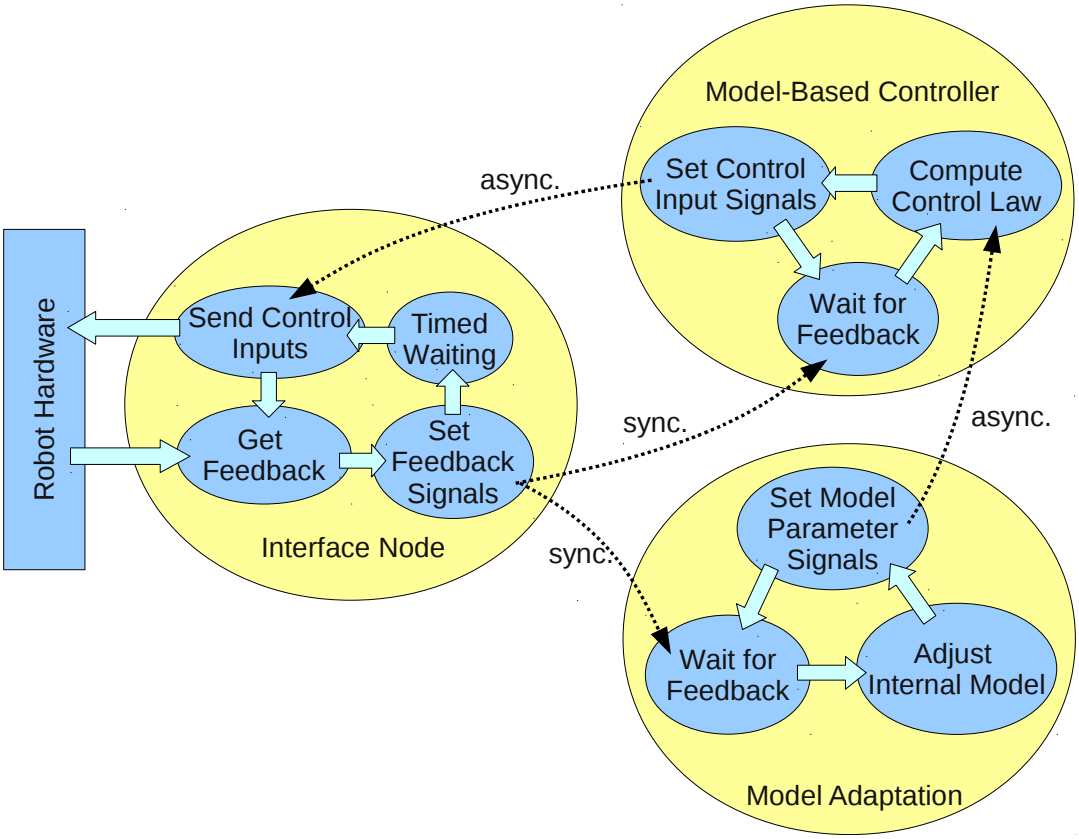


Figure 3.2: Simple Control Loop using Signals and Systems Concept.

Figure 3.2 shows an example of a simple adaptive controller implemented using the signals and systems concept. Here one can clearly see how each system runs its own loop (and thus, thread of execution) to provide a certain functionality. On the left, the interface node is dedicated to hardware communication on a timed loop to provide a fast and reliable sampling rate for the controller. The interface node takes the latest control inputs at a time instant, sends them, acquires feedback information, and sets its output signals with the updated feedback information. Other nodes, *id est*, the controller and model adaptation, wait for a new set of feedback information and thus, synchronize to the interface node. On one side, the controller node takes the feedback information as well as the current model parameters, asynchronously, to compute the controller input required for the particular task. On the other side, the model adaptation can

also use the feedback information to adjust the current model parameters which it can asynchronously update in the controller.

One can clearly see from the simple example of Figure 3.2 that this framework is very flexible. To perform an adaptation run, one could easily substitute the controller node for a predefined training trajectory. Or, the model adaptation node can be inhibited or simply removed altogether when the model parameters are considered settled and that adaptation calculations are unnecessary. Furthermore, additional outer control loops can be realized with the introduction of a control input addition system. In other words, insertion, deletion, addition and suppression are some of the many structural operations that can be realized by this signals and systems architecture.

3.1.2 Nodes, Inhibitors and Suppressors

Above is essentially the description of the backbone on which the Subsumption Architecture is constructed. In the classic sense, the Subsumption Architecture is formed of nodes with input and output signals which may or may not be inhibited or suppressed. In this implementation, the `node` class is the interface which forms the basis of the control architecture. This interface has a collection of a few methods to connect to any desired signal and to launch and run its own thread of execution. When one creates a new class derived from `node`, one is required to implement the "loop" function (or `process` method) and two standard methods, `connectTo` and `getOutputSignal`, which establish a connection to a given node by populating the input signals and delivering the requested output signals, respectively. This framework allows for maximum flexibility as nodes can be connected to any other nodes with the only requirement that the signals follow some user-defined name policy. Furthermore, the implementation required by the author of a derived node is at a minimum, allowing more effective development.

The classic elements of the Subsumption Architecture were implemented as well. These are the inhibitor and suppressors. Since those elements take inputs and deliver an output, it was logical to implement them as derived from the `node` interface. Consequently, the implementation of both the `inhibitor` and `suppressor` class were very similar. Both take an input signal, which is syn-

chronously carried through the output signal, under normal circumstances. Additionally, they both take a boolean signal which marks a inhibited or suppressed state. Finally, an override signal is also added as input which synchronously replaces the input signal when the boolean signal is `true`. This very simple collection of classes builds the foundation of the Subsumption Architecture used in this thesis work.

3.2 Kinetostatic Transmission Elements

As mentioned before, so-called *Kinetostatic Transmission Elements* generalize all types of multi-body dynamics elements. In other words, the task of building a dynamics model of a manipulator or any other kinematic chain can be regarded as the task of putting KTE building blocks together in a group which in turn is also a KTE and thus can be linked to other KTEs or groups of KTEs. This flexible and modular approach allows for any type of building blocks and for the substitution of one KTE for another, *exempli gratia*, substituting a crude dynamics model for a more sophisticated one as the development or modelling phase progresses.

In the presented implementation, the foundation class of the KTE framework for dynamics modelling is the so-called `kte_map` class. This class has no data members (left for the derived classes) and thus the definition of the kinetostatic manifolds that are mapped by the KTE are defined in a specific derived class. Three types of kinetostatic manifolds were created: `gen_coord`, `frame_2D`, and `frame_3D` which hold values of zeroth, first and second derivatives of position (and orientation) as well as forces (and torques) for a single degree-of-freedom generalized coordinate, a two-dimensional and three-dimensional kinematic frame, respectively. Thus, the `kte_map` base class really only needs to have three pure virtual functions:

- `kte_map::doMotion` which performs a kinematics calculation pass. In other words, it maps the input kinematics of the KTE to its output kinematics (if any).
- `kte_map::clearForce` which sets all the forces to zero before making any dynamics calculation. This is needed because a kinetostatic manifold can

serve as the kinematics input to many KTEs and thus requires a sum of the applied forces which needs to start from zero, obviously.

- `kte_map::doForce` which computes the forces to be applied to the input kinetostatic manifolds based on the already computed kinematics and the forces that flow through the output kinetostatic manifolds (if any).

The most important characteristics of the KTE mappings is that all the kinematics need to be computed in a forward sequence and the forces can be computed in the exact reverse order. This is simply because all the input kinematics need to be calculated before a KTE does its motion calculation and vice versa for the forces. This is a very useful characteristic because it allows KTEs to be simply serialized in a list which is traversed one way for the `doMotion` calls and traversed in reverse for the `doForce` calls. The `clearForce` calls come between the other two calls and can be performed in any order. It thus leads to a second foundation class of the KTE framework and that is the `kte_map_chain` which is a simple container of a list of `kte_map` objects. Since the `kte_map_chain` class has the same characteristic as the `kte_map` class it is itself derived from `kte_map` which allows KTE chains to be contained in other KTE chains.

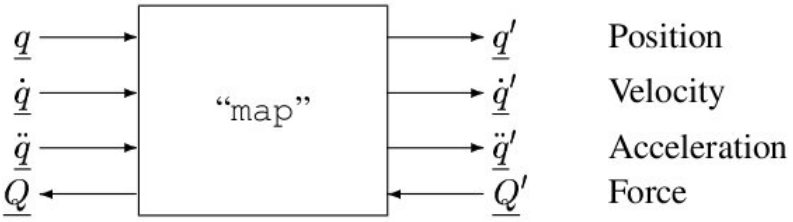


Figure 3.3: Simple Kinetostatic Transmission Element, from Kecskeméthy et al. (2001).

3.2.1 Building Blocks

Now that the foundation classes are defined, several classes derived from `kte_map` can be defined as the simple building blocks for multi-body dynamics modelling and subsequently used for control purposes. Although Figure 3.3 suggests all KTEs map a set of generalized coordinates (or frames) to another, it does not have to be the case, many KTEs are terminal, *id est*, take kinematics input and produce a force. In fact, one can put most simple KTEs into two categories:

those who perform kinematic transformations and those who produce forces. Intuitively one can immediately see that KTEs representing joints or rigid links are of the first kind while elements such as inertia or spring / damper models are of the second.

Kinematic Mappings

First let us consider the elements which map the kinematics from one manifold to another. For instance, a revolute joint can be seen as taking as input the kinematics of a generalized coordinate Θ and of a base frame Γ_b, then applying the necessary rotations about its defined joint-axis to produce an end frame Γ_e, and similarly for prismatic joints. The forces, on the other end, are mapped backwards via simple static force transformation from Γ_e to Γ_b and simple projection of the forces onto the two orthogonal spaces formed by the joint-axis' one-dimensional space and its null-space onto Θ and Γ_b, respectively, with special attention needed for the reaction forces of an actuated joint. The expressions used in the implementation are standard formulae taken from any undergraduate multi-body dynamics textbook such as (Angeles, 2007) or (Hibbeler, 2003). Notably, the equations 3.1 and 3.2 that follow are the so-called "rotating frame" formulae (shown for sake of completeness), to obtain the global-referenced kinematics quantities of Γ_e from its expression relative to Γ_b. Where, $\vec{p}_e$, $\vec{v}_e$ and $\vec{a}_e$ are the position, velocity and acceleration vectors of the end-frame with respect to the "global" frame (that to which the base-frame is expressed in, but not necessarily an inertial frame). Similarly, Q_e, $[\vec{\omega}_e]_e$ and $[\vec{\alpha}_e]_e$ are the rotation matrix, angular velocity and angular acceleration of the end-frame with respect to the "global" frame, expressed in the coordinate system of the end-frame. Additionally, a subscript such as in $\vec{\omega}_{e,b}$ or $Q_{e,b}$ denotes a rotation of the end-frame with respect to the base-frame in this case and the subscripts following the brackets denote the coordinate system in which the vector components are expressed. Finally, expressions such as $[\vec{\omega}\times]$ represent the cross-product matrix, often casually referred to as the skew-matrix, of $\vec{\omega}$ in this case.

$$
\begin{aligned}
\vec{p}_e &= \vec{p}_b + Q_b\,[\vec{p}_e]_b \\
\dot{\vec{p}}_e &= \vec{v}_e = \vec{v}_b + Q_b\,[\vec{\omega}_b]_b \times [\vec{p}_e]_b + Q_b\,[\vec{v}_e]_b \\
\ddot{\vec{p}}_e &= \vec{a}_e = \vec{a}_b + Q_b\,[\vec{\omega}_b]_b \times [\vec{\omega}_b]_b \times [\vec{p}_e]_b \\
&\quad +2Q_b\,[\vec{\omega}_b]_b \times [\vec{v}_e]_b + Q_b\,[\vec{\alpha}_b]_b \times [\vec{p}_e]_b + Q_b\,[\vec{a}_e]_b \qquad (3.1) \\
Q_e &= Q_b Q_{e,b} \\
\dot{Q}_e &= [\vec{\omega}_e\times]\,Q_e = Q_e\,[[\vec{\omega}_e]_e\times] \\
[\vec{\omega}_e]_e &= Q_{e,b}^T\,[\vec{\omega}_b]_b + [\vec{\omega}_{e,b}]_e \\
\left[\dot{\vec{\omega}}_e\right]_e &= [\vec{\alpha}_e]_e = Q_{e,b}^T\,[\vec{\alpha}_b]_b - [\vec{\omega}_{e,b}]_e \times Q_{e,b}^T\,[\vec{\omega}_b]_b + [\vec{\alpha}_{e,b}]_e \qquad (3.2)
\end{aligned}
$$

Rigid links for their part are also very straight forward, they map some base frame Γ_b to some end frame Γ_e via some fixed position and orientation offset. Again standard textbook formulas are implemented directly to map the kinematics forward, computing the required centripetal acceleration term and others, and the forces backward via standard rigid transmission of forces from Γ_e to Γ_b. Any other type of kinematics mapping can be implemented by a variation on Equations 3.1 and 3.2 by taking out some null terms or simplifying vectors of constant magnitude or direction. At this point, a simple massless kinematic chain can be formed by a chain of joints and rigid links as shown in Figure 3.4. With a minimal amount of programming, serial kinematic chains of all most common kinds can be built with ease and its forward kinematics and inverse statics can be computed. Using the same model for inverse kinematics requires some iterative methods, but as it will be shown, IK is not necessary for control purposes using the Virtual Model Control principle even for position control of the end-effector.

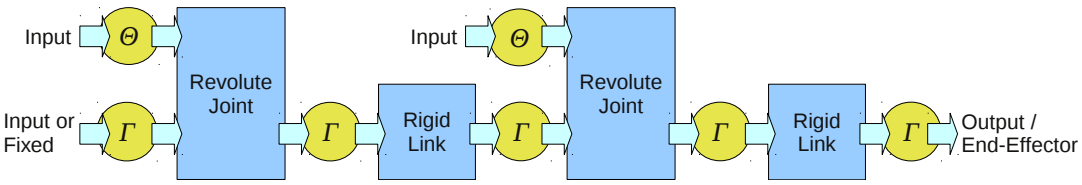


Figure 3.4: Massless two dof kinematic chain using KTE models.

Dynamic Elements

Now for the second type of KTE, there are those elements which produce forces. Generally speaking, one can think of this group as energy absorbers, dissipaters, or generators (some external power supply). The first obvious energy absorber is inertia which can be regarded as taking input mechanical work to build kinetic energy. However, in this context it is more convenient to regard inertia as giving a resistance to losing momentum through acceleration. Thus, in the d'Alembert sense, the inertia KTE simply produces a force on a kinetostatic manifold which is the subtraction of the acceleration on the manifold times the inertia (held as a parameter of the KTE). With this definition, it is simple to picture an inertia KTE as a terminal KTE which does no kinematic mapping and simply produces a d'Alembert force.

Two other simple types of dynamics elements are the spring and damper models. Again, these are elements which perform no kinematic mapping but rather compute a force which resists to the motion between two frames, either of which could be fixed. Those two frames, referred to as anchors, come to the spring or damper KTE with computed kinematics and the KTE simply computes a force (tension or compression) that is a function of the relative position (spring) or velocity (damper) between them. Again, the models for those KTEs are simple linear ideal spring and damper but can of course be extended to any special-purpose model. Figure 3.5 below shows an example of a double pendulum system modelled using only the five basic models mentioned so far. Needless to say, more advanced models are needed to better reflect reality, such as dry friction at the joints, backlash of the gears, and even impacts, but they are all implemented in similar ways to those five basic elements.

Special Models

In addition to the aforementioned simple models such as mass elements, springs, rigid links, and joints, model-based control schemes require a deeper level of modelling rectitude. Consequently, special models were also implemented to better reflect reality. First, on a practical level, joints which are actuated will produce reaction forces on the preceding link, and thus, a simple `force_actuator` model is used for each actuated joint to simply apply the reaction

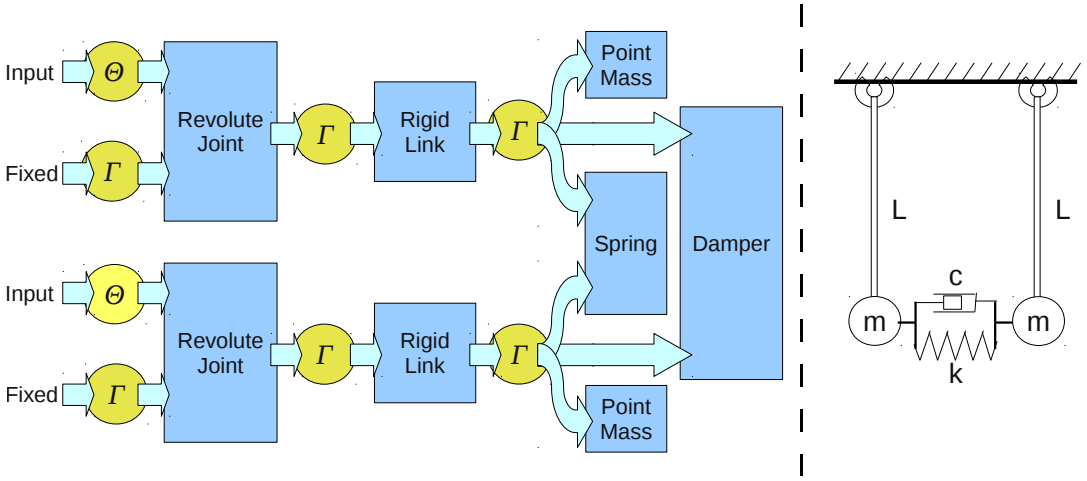


Figure 3.5: Double pendulum with spring and damper using KTE models.

force of the actuator to a joint. This required the addition of an interface class named `reacting_kte` which has a simple method to apply a reaction force. This interface is implemented by both types of joints, prismatic and revolute, and used by the `force_actuator` class to apply the joint torque or force.

Second, the joint friction has a significant effect on the dynamics of any manipulator or other type of kinematic chain. Models for both dry friction and viscous friction were implemented and can, of course, be combined. Viscous friction is simply implemented with a viscosity factor multiplying the velocity of the joint, in the reverse direction. Dry friction is much more difficult to model. As a simple first model, the micro-slip approach was taken in which the friction force is a function of the joint micro-velocity and the normal force, but not of the local contact-surface deformation (as in real Coulomb friction). For the purpose of this thesis work, it was found to be a sufficient model, however, special frictional joint models could be made to include more realistic micro-strain calculations, or even, as in Aksman et al. (2007), using Radial-Basis Function Artificial Neural Networks.

Third, the backlash effect on the gear-train was implemented as a KTE model. However, it was clear, from simulations that special modifications to the simulation and control algorithms would be needed to obtain valid results. Those modifications would have involved certain iterative methods, *exempli gratia*, non-linear root-finding, in order to find the solution to each simulation or control step. It was deemed that such complications would be avoided, suffering

the possible loss of performance or precision in the final implementation.

Finally, in order to model the manipulated object, a model for a flexible beam was implemented. The idea builds on the concepts elaborated by Bonitz and Hsia (1996) in which the internal forces through the manipulated object are part of the control objectives, *id est*, maintain the internal forces to some desired values such as all zero or a slight axial compression between the two end-effectors. The object is modelled as a classic Euler-Bernoulli beam with compression and bending resistances. It is modelled as a beam and not a general body because there are always two points between which deformation can occur; other attachment points, if any, are rigid for the control purposes of this thesis.

3.2.2 Multi-body Dynamics Algorithms

Now that the KTE framework has been defined, algorithms can be formulated to use them. It has already been mentioned that the forward kinematics and inverse dynamics can be computed via a forward sequence of `doMotion` calls followed by an inverse sequence of `doForce` calls. Considering the case of a multiple degree-of-freedom kinematic chain such as a robotic manipulator, one can formulate a few useful algorithms for simulation a system, handling virtual models, controlling a system, computing Jacobians and the mass matrix in closed-form, and estimating external forces.

Simulation

In order to simulate a model one first defines the degrees of freedom for the motion. Typically, these are the joint coordinates but could also be coordinate frames (which only require a few additional consideration in the numerical integration of rotational motion). The general equation of motion for a multiple degree-of-freedom mechanical system is as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau_{mot} + \tau_{endo}(q,\dot{q}) + \tau_{ext} \tag{3.3}$$

Where $M(q)$ is the generalized inertia matrix, $C(q,\dot{q})$ is the Christoffel matrix of Coriolis and centripetal effects, τ_{mot} are the generalized forces applied at the joints, τ_{ext} are those resulting from externally applied forces, and $\tau_{endo}(q,\dot{q})$ are

any generalized forces resulting from miscellaneous endogenous forces (friction, gravity, etc.). In the perspective of a KTE, the applied forces at the joints are not, strictly speaking, part of the KTE computation, but everything else is. In order to perform the numerical integration, the unknown quantity is the acceleration vector $\ddot{q}$ while all the other elements are encapsulated in the KTE model of the mechanical system. In order to obtain the generalized inertia matrix, one can use the fact that the acceleration vector only affects the system via the inertia matrix, and hence, by setting it to zero, the sum of all other terms of equation 3.3 can be obtained (except for the known applied torques). By sequentially setting each component of the acceleration vector to one, each column of the inertia matrix can be obtained. Finally inverting the mass matrix and multiplying it to the sum of forces gives the acceleration to be numerically integrated. This gives Algorithm 3.1 in which $KTE(q, \dot{q}, \ddot{q})$ is the forward kinematics and inverse dynamics calculations that produce a generalized force vector for the given joint kinematics or manipulator configuration. Also, in Algorithm 3.1, e_i refers to a vector of dimension n whose i^{th} component is equal to one while all others are zero and $M[:, i]$ refers to the i^{th} column of the generalized inertia matrix. Finally, $\tau_{nonlin} = \tau_{mot} + \tau_{ext} + \tau_{endo}(q, \dot{q}) - C(q, \dot{q})\dot{q}$ as the collection of all forces resulting at the joint that could cause acceleration, while $\tau_{nonlin-in}$ is a temporary variable that also includes a column of the mass matrix, subtracted from τ_{nonlin}.

for $t = 0$ **to** *endtime* **do**
  $\tau_{nonlin} \leftarrow KTE(q, \dot{q}, 0)$ $\{= \tau_{mot} + \tau_{ext} + \tau_{endo}(q, \dot{q}) - C(q, \dot{q})\dot{q}\}$
  for $i = 0$ **to** $n - 1$ **do**
    $\tau_{nonlin-in} \leftarrow KTE(q, \dot{q}, e_i)$ $\{= \tau_{mot} + \tau_{ext} + \tau_{endo}(q, \dot{q}) - C(q, \dot{q})\dot{q} - M(q)e_i\}$
    $M[:, i] \leftarrow \tau_{nonlin} - \tau_{nonlin-in}$
  end for
  $\ddot{q} \leftarrow M^{-1}\tau_{nonlin}$
  $q, \dot{q} \leftarrow integrate(q, \dot{q}, \ddot{q})$
  $t \leftarrow t + \delta t$
end for

Algorithm 3.1: Multi-body dynamics simulation using KTE models.

Virtual Model Interface

Now moving towards the Virtual Model Control (VMC) scheme, the concept of a virtual part to the dynamics model needs some attention. It is important to differentiate reality and virtual reality. When using a KTE model for a manipulator which maps a set of joint coordinates to an end-effector frame, the model can be used to compute, for the current state of the manipulator, the torques at the joints in order to cancel them in the final commanded joint-torques, this is classic feedback linearisation or internal model control. In the VMC approach, if one simply appends to the end-effector frame another chain of KTEs to form a compound system of internal and virtual models, the forces contributed by the virtual model will be lumped to the internal model and will be treated as real forces that should be cancelled. This, of course, has the disastrous effect of applying forces in the opposite direction from what is intended. For instance, putting a virtual spring from the end-effector to some object to be grasped should simulate a tension force towards the object, but if treated as real, the resulting control torque will act to cancel this non-existing spring and will thus pull the end-effector away from the target. Fortunately, there is a very simple answer to this problem: action / reaction. All that is required to make the bridge between a virtual model and a real model is a KTE block which inverts the signs of the forces and torques.

But the story does not end here. Some attention is needed with regards to what is virtual and what is not. If two manipulators handle the same object, should the object be part of the internal model or part of the virtual model? There is no simple answer to this question since it is merely a reformulation of the, now ageing, debate over centralized, decentralized, or master-slave control architectures of multiple-manipulator coordination. One view is to consider the object as a rigid link between the two grippers, this closes the kinematic chain and requires one of a plethora of special, used techniques, possibly leading to a master-slave approach if the closed kinematic chain is cut at one of the grippers who then becomes the slave.

Considering the modelling of the manipulated object as a flexible beam, mentioned earlier, one first notices that the kinematic chain is no longer closed since the kinematics constraints posed by the manipulated object are transformed to compliance forces on the two end-effectors. Another important observation is

that the stiffness of the object should be high in order to avoid large control errors that could crush or loosen the grip on the object. Consequently, it is essential that the sampling period and manipulator velocities be kept within reasonable boundaries during object manipulation, in other words, the distance travelled by either end-effectors in one sampling period should not in any way come close to the maximum allowed deformation on the manipulated object, but this is a classic requirement in the control of robotic manipulators.

Since the flexible beam approach produces compliance forces, it is desirable to follow those forces with the manipulator and not counteract them. So just as with the spring example above, the flexible beam model should be enclosed within virtual model interfaces, Figure 3.6 illustrates this idea. It should be noted, however, that if mass is added to the manipulated object, then this constitutes a model of the real mass of the object and thus, by applying a second virtual model interface before the application of inertial forces, those forces can be compensated for.

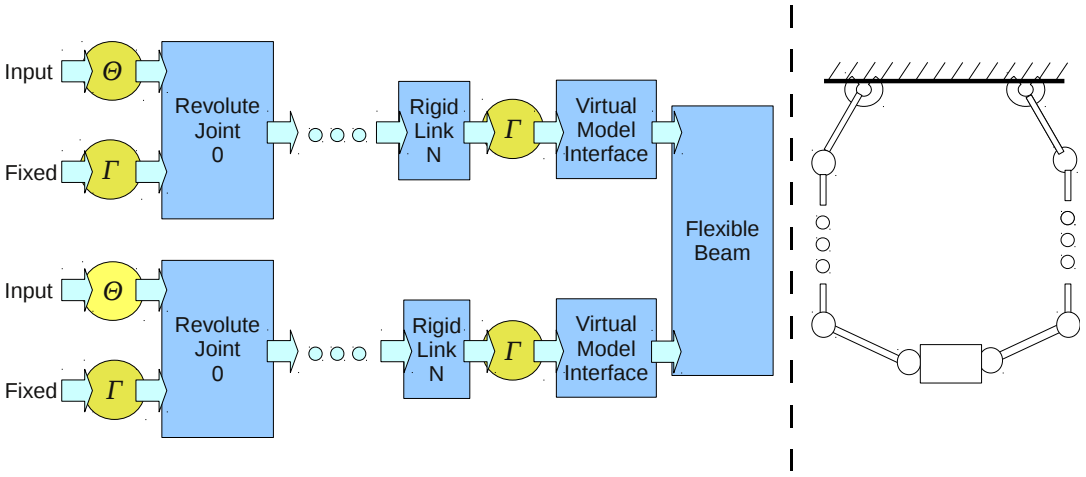


Figure 3.6: Dual manipulation of an object using KTE models and a flexible beam.

Control Algorithms

In the context of VMC, the control algorithm can be very simple indeed. The computation merely requires the forward kinematics and inverse dynamics of the internal model, augmented by the virtual model, to be calculated. The forces

(or joint torques) which result from the inverse dynamics represent the combination of non-linear dynamics forces to be compensated for (internal model) and desired behaviour of the manipulator (virtual model). Therefore, all that remains is to apply the inverse of those torques to the joints. Algorithm 3.2 is thus formulated and one can observe its simplicity. Of course, the underlying definition of the virtual model is where the complex behaviours are encoded and hence, where the complexity of the control system design resides. Also, in relation to Section 3.1, it should be noted that, in practice, this simple control algorithm is distributed on a few nodes of the Subsumption Architecture to allow quick substitution or inhibition of certain virtual models.

while $t \leq endtime$ **do**
  $q, \dot{q}, t \leftarrow measurements$
  $\tau \leftarrow KTE\_with\_VM(q, \dot{q}, 0)$
  $sendTorques(-\tau)$
  $wait(\delta t); t \leftarrow t + \delta t$
end while

Algorithm 3.2: Simple Virtual Model Control using KTE models.

Jacobian Computation

It was previously noted that KTE models realize abstraction to a great level by hiding their implementations, however, there is some useful global information that can be extracted directly from the models. One such quantity is the Jacobian matrix of a manipulator or more generally a multiple degree-of-freedom kinematic chain. The following are the well-known kinematics relations for a manipulator:

$$\begin{bmatrix} \vec{\omega}_{EE} \\ \vec{v}_{EE} \end{bmatrix} = J(q)\dot{q} \tag{3.4}$$

$$\begin{bmatrix} \vec{\alpha}_{EE} \\ \vec{a}_{EE} \end{bmatrix} = \dot{J}(q, \dot{q})\dot{q} + J(q)\ddot{q} \tag{3.5}$$

It is clear from the above that two matrices should be extracted, the Jacobian $J(q)$ and it's time derivative $\dot{J}(q, \dot{q})$. The first being a velocity quantity and the latter being an acceleration quantity, it is clear that these matrices can be stored as a kinematic frame per column. Therefore, the natural way to extract

them from the KTE is to associate to each joint coordinate a Jacobian kinematic frame which can be easily mapped to the end-effector or any other frame after all kinematics calculations for the KTE chain has been performed. For example, a revolute joint KTE, which maps a base frame Γ_b and joint coordinate Θ to an end frame Γ_e, will also compute a Jacobian frame Γ_J whose velocities, linear and angular, correspond to a column of a Jacobian which maps the joint velocity to the change in velocity from the base frame Γ_b to the end frame Γ_e, and similarly for the accelerations. After all kinematics are computed, each Jacobian frame can be transformed to any desire frame, such as the end-effector frame, with standard frame transformation formulas and be stacked together column-by-column to form the $J(q)$ and $\dot{J}(q, \dot{q})$ matrices. Algorithm 3.3 thus follows, where $\Gamma_J[:]$ is the list of Jacobian frames extracted from the KTEs, $\Gamma_{J,i}$ is the Jacobian frame from the joint i to the end-effector, and $\oplus$ stands for the frame transformation formulas ("rotating frame" formulas, see Equations 3.1 and 3.2).

for $t = 0$ **to** *endtime* **do**
  $\Gamma_J[:] \leftarrow KTE\_with\_jacobian(q, \dot{q}, 0)$
  for $i = 0$ **to** $n - 1$ **do**
    $\Gamma_{J,i} \leftarrow \Gamma_J[i] \oplus \Gamma_{EE}$
    $J[:, i] \leftarrow [\Gamma_{J,i} : \omega^T \ \Gamma_{J,i} : v^T]^T$
    $\dot{J}[:, i] \leftarrow [\Gamma_{J,i} : \alpha^T \ \Gamma_{J,i} : a^T]^T$
  end for
  $t \leftarrow t + \delta t$
end for

Algorithm 3.3: Jacobian extraction from KTE models.

Single-Pass Mass Matrix Computation

As of the classic algorithms for dynamics modelling using kinetostatic transmission elements, the mass matrix of the system is calculated column by column by sequential application of unit joint accelerations. This method is not ideal and in fact is not entirely valid, depending on the intricacies of the KTE building blocks. The first obvious issue is the computational burden of computing a forward kinematics and inverse dynamics pass for every degree of freedom. The

second, not so obvious issue is the propagation of the inertial forces through the joints. In a Lagrangian approach using the virtual work principle, the premise is that the inertia affecting one joint is obtained with the assumption that all other joints are held fix, and thus, rigidly transmit the inertial loads. However, the joint models are set up to only transmit the forces in the joint's axis null-space which leaves out one component (either a torque about a revolute joint or a force along a prismatic joint). This would, in theory, lead to an asymmetric system mass matrix, in blatant disagreement with all multi-body dynamics literature, and indeed, it was found while testing Algorithm 3.1 that the system's mass matrix turned out to be asymmetric. Consequently, a novel formulation was developed to compute the mass matrix in a single pass through the KTE model, but first, let us expose the required mathematics, most of which can be found in Angeles (2007).

First, the following equations can be used to describe the twist, momentum and wrench of each inertial element stacked as one vector for all inertial elements:

$$t_{cm}(q,\dot{q}) = T_{cm}(q)\dot{q} = \begin{bmatrix} [J_{cm,1}(q)]_0 & \cdots & [J_{cm,1}(q)]_{n-1} \\ \vdots & \ddots & \vdots \\ [J_{cm,m}(q)]_0 & \cdots & [J_{cm,m}(q)]_{n-1} \end{bmatrix} \begin{bmatrix} \dot{q}_0 \\ \vdots \\ \dot{q}_{n-1} \end{bmatrix} \tag{3.6}$$

$$\mu_{cm}(q,\dot{q}) = M_{cm}t_{cm}(q,\dot{q}) = \begin{bmatrix} M_{cm,1} & 0 & \cdots & 0 \\ 0 & M_{cm,2} & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & M_{cm,m} \end{bmatrix} \begin{bmatrix} t_{cm,1} \\ t_{cm,2} \\ \vdots \\ t_{cm,m} \end{bmatrix} \tag{3.7}$$

$$w_{cm} = \dot{\mu}_{cm}(q,\dot{q}) = M_{cm}\dot{t}_{cm}(q,\dot{q}) + \Omega_{cm}M_{cm}t_{cm}(q,\dot{q}) \tag{3.8}$$

Where the M_{cm} is the block-diagonal matrix of all mass elements (at centre of masses and expressed in principal axis coordinates); t_{cm}, μ_{cm} and w_{cm} are the stacked vectors of twist, momentum and wrench, respectively, for all mass elements; $[J_{cm,j}]_i$ is the Jacobian matrix mapping joint coordinate i to the centre-of-mass twist of mass element j, and all combined gives $T_{cm}(q)$, the so-called twist-shaping matrix; and finally, Ω_{cm} is the angular velocity cross-product matrix, stacked for all mass elements. By formulating the total kinetic energy of the system, as in Chapter 7 of Angeles (2007), the system's mass matrix $M(q)$ can be expressed as follows:

$$M(q) = T_{cm}^T(q)M_{cm}T_{cm}(q) \tag{3.9}$$

$$\dot{M}(q) = \dot{T}_{cm}^T(q)M_{cm}T_{cm}(q) + T_{cm}^T(q)M_{cm}\dot{T}_{cm}(q) \tag{3.10}$$

Also note from the above that the only time dependency is in the twist-shaping matrix, given that the time-derivative is not required to be taken in an inertial frame of reference (which would be the case if it was used to formulate the equations of motion, but here it isn't so). Moreover, the time-derivative of the twist-shaping matrix can easily be formulated from its definition in Equation 3.6:

$$\dot{T}_{cm}(q) = \begin{bmatrix} \left[\dot{J}_{cm,1}(q)\right]_0 & \cdots & \left[\dot{J}_{cm,1}(q)\right]_{n-1} \\ \vdots & \ddots & \vdots \\ \left[\dot{J}_{cm,m}(q)\right]_0 & \cdots & \left[\dot{J}_{cm,m}(q)\right]_{n-1} \end{bmatrix} \tag{3.11}$$

Finally, given Algorithm 3.3, the twist-shaping matrix and its time-derivative can easily be obtained by extracting joint Jacobians and their time-derivatives, subsequently transforming them to the centre-of-mass frame of reference of each inertial element and finally stacking them to form the $T_{cm}(q)$ and $\dot{T}_{cm}(q)$ matrices. Using $T_{cm}(q)$ to perform a similarity transformation of M_{cm} will lead to the system mass matrix $M(q)$. Furthermore, Equation 3.10 leads to the computation of the time-derivative of the system's mass matrix which will prove to be useful in the next section. The only overhead is the maintenance a mapping to relate joint Jacobians to the inertial elements they affect (or vice-versa), but this is rather low-cost in terms memory and house-keeping. One benefit is trading the costly KTE pass for every degree-of-freedom with a minimal amount of frame transformations and a few large-dimension, but manageable matrix multiplications. But of course, the main advantages are the correctness of the result without any possibly-circumvented *ad hoc* solution to the issue mentioned in the introduction of this section as well as a low-cost "bonus" in that it also produces the time-derivative of the system's mass matrix.

Applying this method to the *WorkPartner*'s upper-body, the trial runs have revealed a 2-norm difference between the classic formulation of Algorithm 3.1 and this novel, single-pass formulation of about 6%. By examining samples of the mass matrices from both methods, at various configurations, it was found that the classic formulation, as implemented in this thesis work, failed to produce the correct, analytically determined system mass matrix, and in fact, failed to produce a symmetric matrix. On the other hand, the novel computation of the system mass matrix using twist-shaping matrices produced the correct results, validating both the equations presented in this section, but also the correctness of its implementation in the context of this thesis.

3.2.3 External Force Estimation

One crucial piece of the puzzle is the external force estimation algorithm. In this context, the method of non-linear disturbance observer of (Nikoobin and Haghighi, 2009) is followed and reformulated. The basic estimation equations are as follows:

$$\tau_{ext} = M(q)\ddot{q} + C(q,\dot{q})\dot{q} - \tau_{endo}(q,\dot{q}) - \tau_{mot} \quad (3.12)$$

$$\dot{\hat{\tau}}_{ext} = -L(q,\dot{q})\hat{\tau}_{ext} + L(q,\dot{q})(M(q)\ddot{q} + C(q,\dot{q})\dot{q} - \tau_{endo}(q,\dot{q}) - \tau_{mot}) \quad (3.13)$$

Where $\hat{\tau}_{ext}$ is the estimate of joint torques that are a result of the external forces applied on the system and $L(q,\dot{q})$ is a correction gain which can be determined to stabilize the dynamics of the error. The following change of variable is made to facilitate the estimation (with new variable ψ):

$$\hat{\tau}_{ext} = \psi + p(q,\dot{q}) \quad (3.14)$$

$$\dot{\hat{\tau}}_{ext} = \dot{\psi} + \frac{\partial p}{\partial \dot{q}}\ddot{q} + \frac{\partial p}{\partial q}\dot{q} \quad (3.15)$$

By comparing Equations 3.15 and 3.13, the following relation is established that will eliminate the need for the acceleration term in the estimation equation:

$$\frac{\partial p}{\partial \dot{q}} = L(q,\dot{q})M(q) \quad (3.16)$$

And thus the following estimation equation is obtained:

$$\dot{\psi} = -L(q,\dot{q})\psi + L(q,\dot{q})(C(q,\dot{q})\dot{q} - \tau_{endo}(q,\dot{q}) - \tau_{mot} - p(q,\dot{q})) - \frac{\partial p}{\partial q}\dot{q} \quad (3.17)$$

The function $p(q,\dot{q})$ can be chosen to be:

$$p(q,\dot{q}) = \gamma M(q)\dot{q} \quad (3.18)$$

$$\dot{p} = \gamma M(q)\ddot{q} + \gamma \dot{M}(q)\dot{q} \quad (3.19)$$

$$L(q,\dot{q}) = L = \gamma > 0 \quad (3.20)$$

Given that $M(q)$ is a symmetric, positive-definite matrix, it makes the following function a candidate Lyapunov function:

$$V \equiv \frac{1}{2}\epsilon^T M(q)\epsilon \quad (3.21)$$

By taking the time derivative and the error dynamics relation $\dot{\epsilon} = -L(q,\dot{q})\epsilon$ (which can be obtained directly from Equation 3.13 by assuming pseudo-constant

external forces), we can obtain the time derivative of the Lyapunov function and use the expanded definition of the mass matrix of the previous section to formulate the following analysis:

$$\begin{aligned}
\dot{V} &= \frac{1}{2}\left(\dot{\epsilon}^T M(q)\epsilon + \epsilon^T \dot{M}(q)\epsilon + \epsilon^T M(q)\dot{\epsilon}\right) \\
&= \frac{1}{2}\left(-\epsilon^T L^T(q,\dot{q})M(q)\epsilon - \epsilon^T M(q)L(q,\dot{q})\epsilon + \epsilon^T \dot{M}(q)\epsilon\right) \\
&= \frac{1}{2}\left(-\epsilon^T (2\gamma M(q) - \dot{M}(q))\epsilon\right) \\
&= \frac{1}{2}\left(-\epsilon^T ((\gamma T_{cm}^T(q) - \dot{T}_{cm}^T(q))M_{cm}T_{cm}(q) + T_{cm}^T(q)M_{cm}(\gamma T_{cm} - \dot{T}_{cm}(q)))\epsilon\right) \\
&= - < (\gamma T_{cm}(q) - \dot{T}_{cm}(q))\epsilon, M_{cm}T_{cm}(q)\epsilon > \qquad (3.22)
\end{aligned}$$

As one would expect, making γ very large would make $\gamma T_{cm}(q) - \dot{T}_{cm}(q) \approx \gamma T_{cm}(q)$ which, in turn, would make Equation 3.22 negative definite and hence, asymptotic stability would be achieved. However, due to noise and uncertainty amplification by a large value of γ, it is always desirable to obtain the lowest stability bound on γ such that robustness margins can be assessed for a system.

A lengthy mathematical derivation, presented in Appendix B, leads to the formulation of a simple lower-bound for the estimation gain which makes the candidate Lyapunov function of Equation 3.21 asymptotically stable. It is assumed that the manipulator is serial or tree-structured, and only has revolute joints. Thus a fairly conservative but very simple lower-bound for the estimation gain γ is:

$$\gamma \geq \frac{\lambda_{max}(M(q))}{\lambda_{min}(M(q))} \sum_{l=1}^{n-1} |\dot{q}_l|_{max} \qquad (3.23)$$

Where $\lambda_{max}(\cdot)$ and $\lambda_{min}(\cdot)$ denote the maximum and minimum eigenvalue, and $|\dot{q}_l|_{max}$ denote the maximum speed of joint l. Alternatively, when including motor inertias $G^2 I_m$, a modified derivation leads to the following lower-bound:

$$\tau_{ext} = (M(q) + G^2 I_m)\ddot{q} + C(q,\dot{q})\dot{q} - \tau_{ego}(q,\dot{q}) - \tau_{mot} \qquad (3.24)$$

$$\gamma \geq \frac{\lambda_{max}(M(q))}{\min_i(G_i^2 I_{m,i})} \sum_{l=1}^{n-1} |\dot{q}_l|_{max} \qquad (3.25)$$

In this thesis, the observer was applied to the *WorkPartner* robot which has dominant motor inertias, like most actuated manipulators, on a tree-structured but symmetric upper-body, so the latter lower bound on γ was used and calculated to be about 100 (see Table 3.1 for the values used in the calculations where

the parameters refer to those used in Appendix B). To be more precise, using Equation 3.25 a lower-bound of 92.925 is found. Alternatively, a less conservative value can be obtained from an equation appearing earlier in the complete derivation (again see Appendix B, for equation (48)), this was calculated to 16.182. In other words, the successive applications of upper-bounds corroborates to the calculated values and it is also observed that the application of a stricter but more complex expression for the lower-bound of the estimator gain can allow for a significantly lower value if required. In this particular calculation, this large difference was due to the fact that links closer to the base were more massive while moving considerably slower (for the same reason). This is quite common for manipulators and it is thus preferable to use a less conservative but still asymptotically stable lower-bound on γ in order to maximize performance and robustness of the observer.

Table 3.1: *WorkPartner*'s upper-body joint inertial parameters.

i	$\|a_{i+1}\|_2$ [m]	$\|a_{cm,i+1}\|_2$ [m]	$\|\dot{q}_i\|_{max}$ $[\frac{rad}{s}]$	$G_i^2 I_{m,i}$ [kgm^2]	m_{i+1} [kg]
0	0	0	0.4	12.65	0
1	0.34	0.3	0.1	71.26	24
2	0.1	0.1	0.4	1.838	1
3	0.4	0.4	0.4	12.65	2
4	0.4	0.4	0.4	12.65	0.5
5	0	0	0.4	1.838	0
6	0.1	0.1	0.4	1.838	0.5

3.3 Position Control of a Pendulum

To best illustrate the construction of a control software, a simple pendulum example is used. The simplicity of the example will allow for the complete code to be displayed as well as a detailed explanation. In the next section on the *WorkPartner* control software, such detailed explanation and code is impossible within a reasonable amount of writing. Figure 3.7 represents the simple pendulum model that will be used for this example.

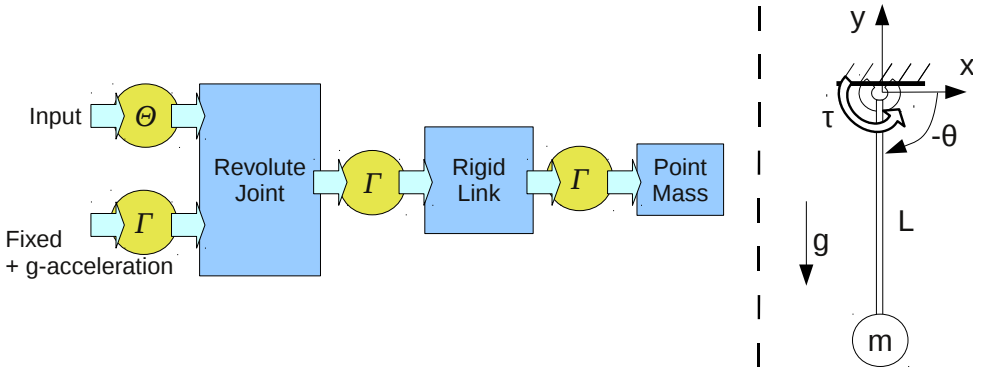


Figure 3.7: Simple pendulum model with point-mass and gravity using KTE models.

3.3.1 Pendulum Model

Using the *Kinetostatic Transmission Elements* framework presented in Section 3.2, the model for a simple pendulum as illustrated in Figure 3.7 can be constructed with the C++ code presented in Algorithm 3.4. Note that the latter is an actual extract of a program and hence, this is actual executable code and represents the typical sequence of constructs required by the user to build a dynamics model using this KTE framework.

In order to make the steps clear, they will be explained here. First, the necessary kinetostatic frames and generalized coordinates are created, in this case, one joint coordinate and a base, joint and end frame. Then, the various KTE building blocks can be created, taking those intermediate manifolds and other values as parameters. In this case: a revolute joint is created between the base frame and the joint frame, driven by the joint coordinate; a rigid link is created between the joint frame and the end frame with an offset of 0.5m and no rotation transformation; finally, a mass element is attached to the end frame with a mass of 1kg and no moment of inertia (point-mass). Finally, an instance of `kte_map_chain`, called "pendulum", is created and the three new KTE building blocks are stacked onto it. At this point, the model of the pendulum is complete and ready to be used. In this code example, the model has also been saved to an XML archive such that another program can simply load the model alongside the input (joint coordinate) and output (end frame). Of course the above example is especially verbose for sake of example, in practice the necessary code can be largely compressed down to its essentials. For sake of completeness, the code used for the loading of the model is presented in Algorithm 3.5 and the

```
//create base and intermediate kinetostatic frames
shared_ptr<frame_2D<double> > base_frame =
  rk_dynamic_ptr_cast< frame_2D<double> >(frame_2D<double>::Create());
shared_ptr<frame_2D<double> > joint_frame =
  rk_dynamic_ptr_cast< frame_2D<double> >(frame_2D<double>::Create());
shared_ptr<frame_2D<double> > end_frame =
  rk_dynamic_ptr_cast< frame_2D<double> >(frame_2D<double>::Create());
shared_ptr<gen_coord<double> > joint_coord =
  rk_dynamic_ptr_cast< gen_coord<double> >(gen_coord<double>::Create());

//add gravitational acceleration to base frame
base_frame->Acceleration = vect<double,2>(0,9.81); //add gravity

//create revolute joint
shared_ptr<revolute_joint_2D> rev_joint(
  new revolute_joint_2D("joint1", joint_coord, base_frame, joint_frame));

//create link of 0.5 meter in length
shared_ptr<rigid_link_2D> link1(
  new rigid_link_2D("link1", joint_frame, end_frame,
                    pose_2D<double>(weak_ptr<pose_2D<double> >(),
                                    vect<double,2>(0.5,0.0),
                                    rot_mat_2D<double>(0.0))));

//create end mass of 1.0 kg (point mass only)
shared_ptr<inertia_2D> mass1(new inertia_2D("mass1", end_frame, 1.0, 0.0));

//create a kte chain named "pendulum"
kte_map_chain pendulum("pendulum");
pendulum << rev_joint << link1 << mass1;

{
  //save the pendulum model to XML archive file "pendulum.xml"
  xml_oarchive pendulum_arc("pendulum.xml");
  oarchive& arc_ref = pendulum_arc;
  //save the pendulum model as well as the joint coordinate and end_frame
  arc_ref << joint_coord << pendulum << end_frame;
};
```

Algorithm 3.4: Pendulum model construction using KTE building blocks

numerical simulation of the model, corresponding to Algorithm 3.1, is presented in actual C++ code in Algorithm 3.6. They are both self-explanatory.

3.3.2 Control Software

In order to demonstrate the basic use of Virtual Model Control (VMC), a simple controller was applied to the pendulum to hold it up-right, *id est*, in the unstable

```
//create pointers to store joint coordinates and end frame
shared_ptr< gen_coord<double> > joint_coord;
shared_ptr<frame_2D<double> > end_frame;
//create a kte chain to hold the pendulum model
kte_map_chain pendulum;

{
  //open the XML archive file "pendulum.xml"
  xml_iarchive pendulum_arc("pendulum.xml");
  iarchive& arc_ref = pendulum_arc;
  //load the pendulum model
  arc_ref >> joint_coord >> pendulum >> end_frame;
};
```

Algorithm 3.5: Loading the pendulum model from XML archive

equilibrium point. Note that any other point could be used as well. The control architecture, in terms of signals and systems, is illustrated in Figure 3.8 for the simulated pendulum model. On the left, a typical simulation node is shown which takes the control inputs, computes the dynamics model, performs an integration step and set the feedback signals to the new "measured" values. On the left, the same model-based controller as in Figure 3.2 is used where the feedback is acquired synchronously, the control law is computed and the control inputs are set asynchronously.

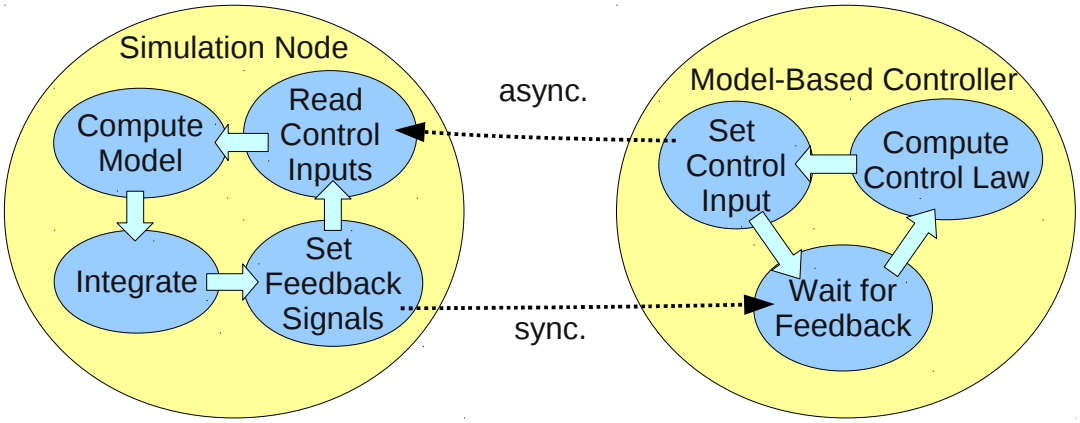


Figure 3.8: Simple simulation and control nodes for the pendulum example.

As a first demonstration of a trivial controller for the pendulum model, a proportional-differential (PD) controller was implemented in the outer-loop of a model-based controller. Algorithm 3.7 shows the implementation of the `process` methods or loop function which produces the controller's action. This is a typical model of a control node of the presented software framework. First, the node

```
//Open Space-Separated-Values data file for the results
ssv_recorder output_rec("pendulum_results.ssvdat");
output_rec << "time" << "q" << "qd" << "qdd" << "f" << end_name_row;
//start of simulation loop.
for (double sim_time = 0.0;sim_time < 50.0;sim_time += 0.001) {
  joint_coord->q_ddot = 0.0; //set acc. to 0
  pendulum.doMotion(); pendulum.clearForce(); pendulum.doForce();
  double f_nl = joint_coord->f; //= -C(q,qd)*qd + f_ext

  joint_coord->q_ddot = 1.0; //set acc. to 1
  pendulum.doMotion(); pendulum.clearForce(); pendulum.doForce();
  double f_nl_in = joint_coord->f; //= -m - C(q,qd)*qd + f_ext

  //compute joint acceleration, a = f / m
  joint_coord->q_ddot = f_nl / (f_nl - f_nl_in);

  //record the values to the data file
  output_rec << sim_time << joint_coord->q << joint_coord->q_dot
             << joint_coord->q_ddot << f_nl << end_value_row;

  //perform Euler integration step of 0.001 seconds
  joint_coord->q += joint_coord->q_dot * 0.001;
  joint_coord->q_dot += joint_coord->q_ddot * 0.001;
}; //end of simulation loop.

output_rec << recorder::data_recorder::close; //close data file.
```

Algorithm 3.6: Numerical simulation of the pendulum model

is registered on the waiting list for the next value of the input signals which will be reset to zero whenever a new value is set, and then, the node waits for both inputs before updating its internal storage of the feedback information. Note also that a typical implementation will include time tags on the input data such that real-time simulation is not necessary, but in this simple example it poses no problem to simulate the system in real-time.

Second, the node computes the PD control law as a desired acceleration on the pendulum's joint. The proportional error is calculated here from the sine of the difference between desired and measured angles, a classic approach. The differential error is simply the negative of the joint velocity, making the controller a regulator. Finally, the desired acceleration is fed to the internal model of the pendulum and the compensation torque at the joint is calculated and sent as an output signal which will be ultimately read by the simulation node. With the applied PD gains of 16 and 8 on a unit-mass system (prescribed acceleration), the system is expected to show a behaviour characterized by a natural

frequency of 4 rad/s, a damping ratio of 1 and a settling time around 1 second, linearised about the desired angle. Chapter 4 presents those simulation results, amongst many other results.

```
//Register to the waiting list for the input signals and wait
const uint& w_a = input_angle->waitingListForNext();
const uint& w_v = input_velocity->waitingListForNext();
while((w_a) && (w_v) && (!is_done)) this_thread::yield();
joint_coord->q = input_angle->getLast();          //Read joint angle
joint_coord->q_dot = input_velocity->getLast(); //Read joint velocity

//Compute the desired acceleration (PD control law)
double prop_err = sin(set_angle)*cos(joint_coord->q)
                - cos(set_angle)*cos(joint_coord->q);
joint_coord->q_ddot = prop_err * 16.0 - joint_coord->q_dot * 8.0;

pendulum.doMotion(); pendulum.clearForce(); pendulum.doForce();

//set the force required to produce the desired acceleration
output_force->setValue(-joint_coord->f);
```

Algorithm 3.7: Proportional-Differential model-based control of a pendulum

Now to exemplify the Virtual Model Control implementation, the pendulum control system was constructed by using a virtual spring from the end of the pendulum to a fixed anchor at 0.51m above the pendulum's joint. Algorithm 3.8 shows extracts of the `pendulum_vmc_node` class which implements the VMC law. First, the constructor is shown in which the pendulum model is loaded, the fixed anchor created and the spring / damper KTE chain is created with negative stiffness and damping to produce the desired behaviour (note that the virtual model interface could have been used also). Then, the `process` method is shown in which the same control procedure corresponding the Figure 3.8 is implemented. The feedback signals are acquired, then the internal and virtual models are computed for the resulting joint torque, and finally, the compensation torque is sent as output signal.

3.4 Control of *WorkPartner*'s Manipulators

Using algorithms such as those of the previous section, the models and control systems for the *WorkPartner*'s upper-body were constructed and tested. This

```
//Constructor
pendulum_vmc_node::pendulum_vmc_node(double aSetAngle, const bool& aIsDone) :
          pendulum_control_node(aSetAngle,aIsDone), end_frame(),
          spring_damper_system("spring_damper_system") {
  { ... }; //Load pendulum model first

  //Create fixed anchor for the VMC
  shared_ptr<frame_2D<double> > fixed_anchor =
    rk_dynamic_ptr_cast< frame_2D<double> >(frame_2D<double>::Create());
  fixed_anchor->Position = vect<double,2>(0.0,0.51);

  //Create Virtual Model (negative forces, i.e. -16 stiffness and -8 damping)
  spring_damper_system
    << shared_ptr< spring_2D>(new spring_2D("vmc_spring", end_frame,
                                            fixed_anchor, 0.01, -16.0))
    << shared_ptr< damper_2D>(new damper_2D("vmc_damper", end_frame,
                                            fixed_anchor, -8.0));
};
//Loop Function
void pendulum_vmc_node::process() {
  {...}; //Wait for and Read input values

  pendulum.doMotion();    spring_damper_system.doMotion();
  pendulum.clearForce(); spring_damper_system.clearForce();
  spring_damper_system.doForce(); pendulum.doForce();

  //set the force prescribed by the virtual model.
  output_force->setValue(-joint_coord->f);
};
```

Algorithm 3.8: Virtual Model Control of a pendulum

section will briefly outline the construction and characteristics of those dynamics models and control systems.

3.4.1 Dynamics Modelling

The first step towards controlling a robotic system such as the upper-body of the *WorkPartner* robot is to obtain a dynamics model of the system. In this thesis, the modelling is done using the KTE framework presented in Section 3.2. The upper-body of the *WorkPartner* is a tree-structured serial kinematic chain of revolute joints. Most of the joints were modelled in correspondence to the KTE model depicted in Figure 3.9. One can observe that each joint encompasses motor inertia (appropriately scaled by the gear-ratio), dry-friction, a force actuator model of reaction force and a lumped-mass at the end of the

following link. Some joints do not require a mass because it is negligible and some links have a centre-of-mass and a full inertia matrix representation.

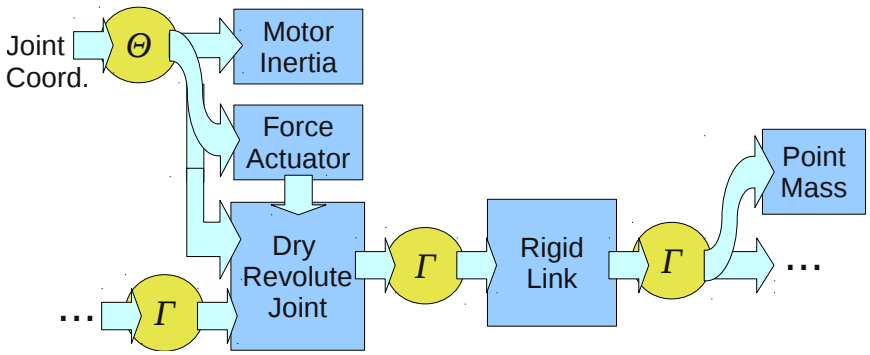


Figure 3.9: KTE modelling of a typical joint of *WorkPartner*'s upper-body.

The *WorkPartner*'s upper-body includes twelve joints (excluding grippers): two in the torso and five in each manipulator. A letter coding is used to identify the various joints, as shown in Figure 3.10. Since the manipulators are symmetrical, the letter coding is also the same for both. It is worth noting that the joint orientations also differ slightly from one manipulator to the other depending on the configuration of the motors and gear-trains.

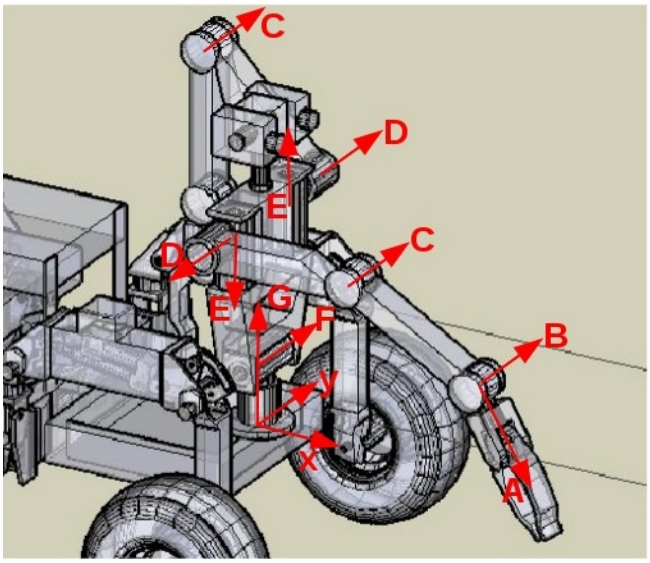


Figure 3.10: Joint axes of *WorkPartner*'s upper-body.

Table 3.2 refers to Figure 3.10 and presents the joint parameters used to construct the nominal model of the upper-body of the *WorkPartner*. The first three columns with numerical values are the Denavit-Hartenberg parameters of the classical notation of the same name, as described in Angeles (2007). Although,

it should be noted that the dynamics model does not directly use the DH parameters and that the reference position (zero) of the joints correspond to a configuration of the robot in which the torso is straight and facing directly forward with both manipulators fully stretched forward (parallel to the ground). The following last three columns show, respectively, the gear-ratio, the motor inertia (rotor) and the lumped-mass at the joint (attached at the end of the previous joint assembly as of Figure 3.9).

Table 3.2: *WorkPartner*'s upper-body joint parameters.

Joint Name	a_{i-1} [m]	b_i [m]	α_{i-1} [°]	G	I_m [kgm^2]	m_{i-1} [kg]
Torso Joint G	0	0.8	0	957.6	1.38e-5	∞
Torso Joint F	0	$\pm$0.15	90	2272.4	1.38e-5	0
Manip. Joint E	0	0.4	0	1323	1.05e-6	12
Manip. Joint D	0	0.1	90	957.6	1.38e-5	1
Manip. Joint C	0.4	0	0	957.6	1.38e-5	2
Manip. Joint B	0.4	0	0	1323	1.05e-6	0.5
Manip. Joint A	0	0.1	90	1323	1.05e-6	0.3

3.4.2 Interface Nodes

The *WorkPartner* control software is built on "interface nodes" which act as the entry point from software to hardware or simulated-hardware. The interface nodes serve the same set of output data (state feedback) and receive the same set of input data (control inputs), with an expected "similar" mapping from the control inputs to the system's state, real or simulated. For the *WorkPartner*'s upper-body, the state feedback information is composed of each joint's position and velocity as well as the world-referenced position and orientation of the base of the torso (assumed fixed for the purpose of this thesis). The control input signals are the forces applied on the joint motors. The nodes' responsibilities are limited to those aforementioned signals, and thus, do not extend to information which does not correspond to a direct measurement on the hardware, such as, for example, the end-effector position.

MaCI Interface

The hardware interfacing is implemented through the Machine Control Interface (MaCI). MaCI is a collection of libraries and hardware interfaces developed in the Department of Automation and System Technology of Aalto University, School of Science and Technology. MaCI implements a standard set of interface classes structured in a client / server architecture across a network using TCP/IP packets. The MaCI class used in this implementation for the *WorkPartner*'s upper-body is the Joint Group Control Client / Server. This class provides optional feedback and control of position, velocity and motor force for each joint in a group. The relevant part of the *WorkPartner*'s control programs were updated, by the author, to implement the MaCI libraries, replacing an older QNX-based system.

For the *WorkPartner*'s upper-body, three joint group control servers are initiated by the *WorkPartner*'s computer: the `Torso` for the two joints to rotate and incline the torso; and the `LeftManipulator` and `RightManipulator` for the five joints of each manipulators. The MaCI interface node will instantiate three joint group control clients to connect to those three joint groups. At a regular time interval during the operation of the control software, the interface node polls the network for the feedback information, associates it with a time-tag, and sends them through the signals and systems implementation, as described in Section 3.1. At the same moment as it polls for feedback, the node asynchronously reads the control inputs (motor torques) and sends them via the MaCI network communication channels which are usually made local or private for better performance.

It is worth noting that the MaCI interface was also implemented, by the author, as a modification to the *SimPartner* software developed by Heiskanen (2008) in a previous Master's thesis. Consequently, this real-time simulator can be used to test the control algorithms for the *WorkPartner* via the same network interface.

Simulation Interface

Given the KTE framework presented earlier in this chapter, it was quite straightforward to implement a simulator which can calculate more precise dynamics models without the burden of real-time computation. Therefore, the simulation node was implemented with the same input and output signals as the MaCI interface node and is thus interchangeable with it. Furthermore, the off-line simulation of the control software is made possible by the time-tags which are associated to the feedback information. Naturally, certain "real-life" issues are not reproduced by either simulation environments, such as the time delays, real-time computational burden on the control side and network bottleneck which limits the sampling period for feedback and control signals. To some extent, the *SimPartner* software can reproduce those difficulties encountered on real hardware. It should be said, however, that the focus of this thesis is not on simulations but on experimental work and development, and thus, the purpose of the simulations are limited.

3.4.3 Inner-Loop Controllers

Certain basic functionalities are required to achieve the higher-level control over the *WorkPartner*. Those correspond to certain control nodes that compensate for gravity and non-linear dynamics. These basic control nodes are used for the initial tests, assessing the natural controllability via virtual model control and mitigating certain artefacts of the system's hardware and motor drives. The following Figure 3.11 shows the basic architecture for the inner-loops of the controller configurations used for most of the test cases.

Dynamics Compensation

Dynamics compensation or inner-loop control node is responsible for counteracting the forces that are inflicted on the joints as a result of gravitational, centripetal, Coriolis, and frictional effects. In Virtual Model Control terms, say for a single manipulator, the inner-loop control node uses the feedback information and a dynamics model of the *WorkPartner* to compute the end-effector

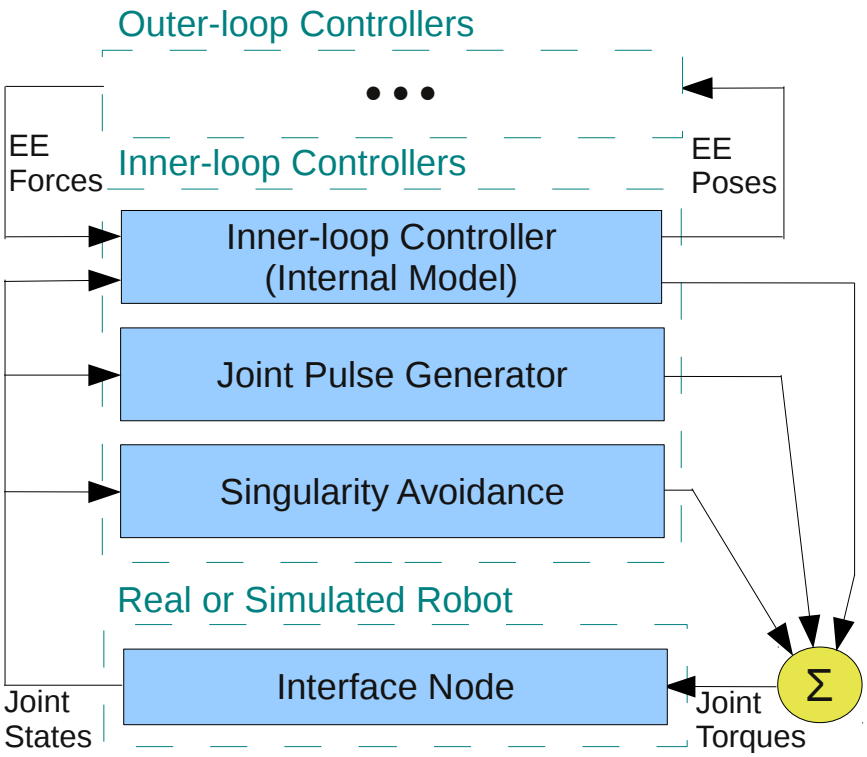


Figure 3.11: Inner-loop controllers on the *WorkPartner*'s upper-body.

kinematics, exposing them for input to any virtual model attached to the end-effector, and finally, uses the spatial forces resulting from the virtual model to compute the joint torques to be applied. Note that the virtual models could easily be attached to other intermediate kinetostatic frames, for example to attach a virtual repulsive spring between the elbows and body to avoid self-collision, but this was not implemented in this thesis.

Mitigation of Stiction Friction

Given the motor drives of the *WorkPartner*'s upper-body, it was clear that with those very high gear ratios (between 957 and 2200) and backlash the controllers would have difficulty in setting joints in motion in a consistent and predictable manner. To mitigate the problems with stiction friction and to some extent backlash, two strategies were employed: first a joint hysteresis was applied, with little success, then better performance was obtained with a pulsing signal added to the control loop. The idea of joint hysteresis is to apply a discontinuity during zero-velocity crossings, *id est*, if the control torque to be applied on a stationary joint is positive, an additional positive torque would be applied to

break the stiction bound. On the other end, The idea with joint pulses is to break the stiction friction using a high-frequency force signal (symmetric square waveform at the Nyquist frequency of the controller) to be added to the other control inputs. This controller works by simply checking if the speed measurement of each joint is below a small threshold value in which case it is assumed static and the pulsing signal is applied, inverting at each discrete control cycle. If the pulses are high enough, they will help to take the controller-applied torques above the stiction friction threshold and allow the static joints to be set in motion. On the other hand, if the main controller's action is not meant to cause any active motion, as in the case of the simple dynamics compensation discussed above, the symmetric nature of the pulse signals will cancel over time, in fact largely filtered out by the motor's inductance, and not cause any visible motion. See Section 5.1 for experimental results showing the positive impact of applying pulse signals on the joints.

Avoiding Singularities

Another difficult issue with manipulator control, especially when prescribing control actions on the end-effector, is avoiding singular configurations. The problem is well-known and occurs when the robotic joints are in or in the vicinity of a configuration which does not allow certain end-effector motions to occur. This translates into the end-effector forces obtained from the outer-loop controllers (VMC in this case) not resulting in any joint torques. Two types of singular configurations exist: boundary singularities which occur when a manipulator is fully extended to its workspace boundary; and internal singularities which occur when the motion is inside the workspace, or towards the inside of the workspace, but an ill-conditioned configuration makes this motion unrealisable. In the control architecture presented here, with the force propagation through the internal model of the mechanical system, the boundary singularities are unavoidable, obviously since it is mechanically impossible for the robot to reach beyond its workspace, but the internal singularities can be somewhat avoided with an additional behaviour.

Internal singularities are typically handled with techniques such as damped least-square solutions to the end-effector force to joint torque transformation which consists of relaxing the strict accuracy of exact methods to avoid moving

close to singularities. In the same spirit, the singularity avoidance behaviour was developed such that joint torques are added to some joints as they get close to singular positions: most notably, the elbow joints which are made to avoid the fully extended arm configuration. In addition, other undesirable configurations are also avoided: notably, an incentive is given to keep the torso straight up and facing forward. Again, experimental results of Section 5.1 validate this strategy.

External Force Estimation

The external force estimation is achieved using a node which listens to the state feedback of the joints as well as the force applied to them by the controller(s). These inputs are processed in a simulated system along with the time-integration of the external joint torque estimator to implement the estimation equation 3.17. The output of this node are the current estimated external torques. It is thus meant that an additional controller node, such as a collision reaction controller, listens to these outputs and uses them to achieve the desired behaviour. However, due to time limitations, such high-level controllers were not implemented.

3.4.4 Outer-Loop Controllers

The following section outlines the controllers which are used on the outer-loop, as shown in Figure 3.11. These are end-effector controllers that use the Virtual Model Control principle and are, thus, generally composed of a virtual dynamic model and possible intermediate signals in conjunction with subsystems.

End-Effector Attractor

The first basic test cases involved the control of the position of the end-effector for either tracking a trajectory or reaching a particular point in the workspace. To achieve this, the end-effector attractor controller was developed which applies a virtual spring-damper model between a set anchor in the workspace and either end-effectors (left or right). The spring and damper constants essentially have

to be set either by inspection or through analysis. Given the high degree-of-freedom count and complex model, the trial-and-error method was found to be much quicker to narrow down to suitable spring constants (between 40 and 100 newtons per metre) and damping coefficients (between 100 and 150 newton-seconds per metre). In summary, the controller synchronously reads the end-effector pose, asynchronously reads the anchor pose which could be constant or not, depending on another control node which provides the anchor, then computes the virtual model dynamics, and finally writes the force and torque vectors to be applied to the end-effector.

Object Manipulation

The object manipulation node is responsible for controlling the force exerted by the two end-effectors of the *WorkPartner* on the manipulated object as well as the net force exerted on it. These functionalities are implemented using a KTE model of a flexible beam as seen from Figure 3.6. By obtaining the end-effectors' kinematics from the dynamics compensation node, the position and orientation deformation of the object can be calculated and thus, the restitution forces necessary to control the internal forces. In addition, the average motion is used in concert with desired motion or driving virtual models to obtain the net force to be applied to the object's bulk which is ultimately propagated to each end-effector. Note that the object's mass is also compensated for.

In other words, the object manipulation node receives the end-effector kinematics and computes object kinematics. Then, the object kinematics can be used by another virtual model node to drive the object as a semi-free mass. Finally, the net object forces are read and transmitted to the each end-effector along with restitution virtual forces from the flexible beam model.

In the absence of a controller to drive the object, this controller behaves as a fully compliant object manipulator and thus can be used to passively allow physical human interaction. In this case, the human operator can displace and manipulated the object in a relatively weightless condition, but not massless or frictionless in the sense that inertia and stiction friction still has an effect on the required physical effort by the operator.

Hang-a-Picture

A so-called "hang-a-picture" controller was developed which holds the manipulated object along a given plane, assumed to be a wall, but could be a ceiling or any other planar surface. This controller works in concert with the aforementioned object manipulation controller to provide an additional virtual model that keeps the object sliding along a plane. Here, a special kinematics KTE mapping is used to compute the point on the plane which is the closest to the centre of the object (minimum distance formula). Then, a spring-damper model is applied between the point on the plane and the object's centre of mass which produces the net force that the object manipulation controller will propagate to the end-effectors.

In addition, for this test case, the base of the *WorkPartner* was set in motion to extend the workspace of the manipulators because the intersection space between the given plane and the non-singular workspace would be fairly limited otherwise. Finally, as with the object manipulation controller, the free direction of motion, that is, along the given plane, is also passively compliant to physical human interaction and thus, could be used in a scenario where the robot supports an object while the human operator adjusts its position along the wall, or any similar scenarios that could occur in assembly or measurement tasks on interplanetary missions.

Chapter 4

Simulations

"There are many methods for predicting the future. For example, you can read horoscopes, tea leaves, tarot cards, or crystal balls. Collectively, these methods are known as "nutty methods". Or you can put well-researched facts into sophisticated computer models, more commonly referred to as "a complete waste of time"."

- Scott Adams

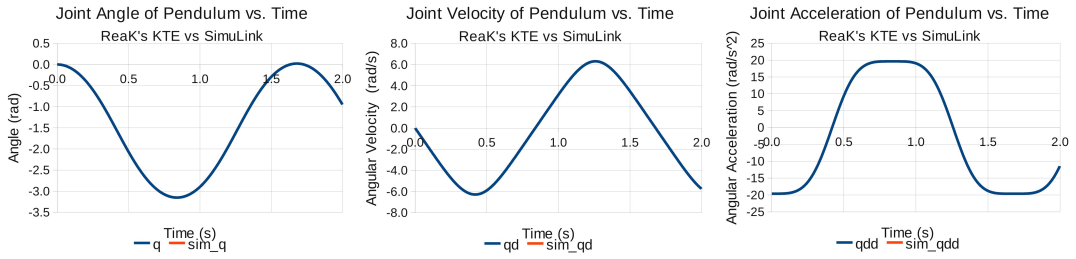
In this chapter, simulation results are presented that serve to validate the algorithms used in advanced simulations and in Virtual Model Control calculations. Given the substantial amount of code produced for this project, it is impossible to present evidence of the validity of each and every individual parts of the implementation. However, let it be known that this significant amount of programming would not have been successful nor so rapid without rigorous programming rules and careful testing of every individual implementation unit (unit-testing) including the base classes, memory management, custom run-time type identification system, serialization libraries, data buffers, geometric computation libraries, matrix numerical methods, signals and systems, and KTE models. For the reader's reference, the electronic submission of the source code which accompanies this thesis contains all relevant unit test programs, *doxygen*-generated documentation and the programming guidelines that were followed.

First, the results of the pendulum control simulations are presented which correspond to the controllers and dynamics models presented in Section 3.3. Those results are validated against theoretical predictions and MatLab / SimuLink results. Second, results are presented for the basic controllers developed for the

WorkPartner robot such as dynamics compensation (or feedback linearisation), object grasping and dual-arm manipulation. Finally, some simulated results are presented for the external force estimator applied on the *WorkPartner*'s upper-body, with mitigated outcome.

4.1 Validation Results for the Pendulum Model

This section briefly presents the results of simulations done on a simple pendulum system. The numerical integrations were performed using a simple Euler integrator with a fixed time-step of 1 millisecond. First, to validate the KTE-based model of the pendulum and its underlying implementation, the pendulum system was constructed in SimuLink as well, for comparison. Figure 4.1 shows the results for one period of oscillation of the pendulum. As it can been seen, the results are indistinguishable from each other as expected from the use of the same numerical integrator and thus, the same numerical accuracy. Based on those results, it can be said that the simulation of the pendulum model in KTE form is valid, in theory and implementation, and exactly equivalent to the analytically derived equation of motion obtained from a Newtonian approach and simulated in SimuLink.



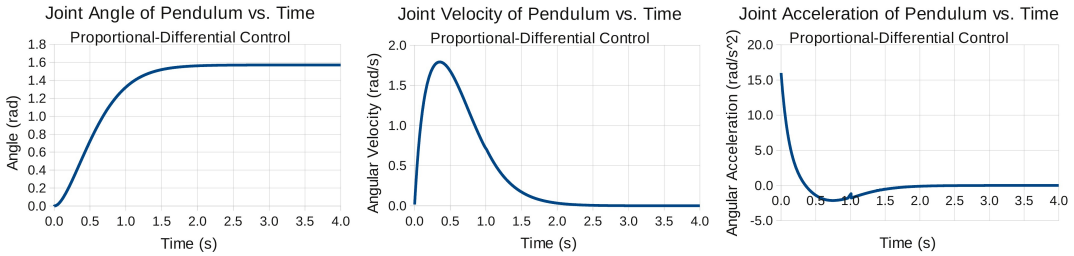
(a) (b) (c)

Figure 4.1: Results of dynamics simulation of the unactuated pendulum, a) angle, b) angular velocity and c) angular acceleration.

4.1.1 Proportional-Differential Control Performance

Given that the pendulum simulation is accurate, the Proportional-Differential controller presented in Section 3.3 can be tested. As mentioned in the description of the controller, the expected behaviour of the system should produce a

critically damped system with natural frequency of 4 rad/s which will lead to a 2% settling time of roughly 1 second. Using the controller, as implemented in Algorithm 3.7, Figure 4.2 was obtained and presents the expected behaviour.

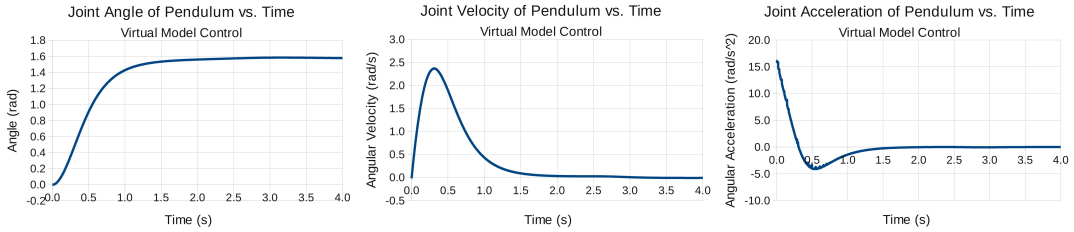


(a) (b) (c)

Figure 4.2: Simulation results for the Proportional-Differential control of a pendulum, a) angle, b) angular velocity and c) angular acceleration.

4.1.2 Virtual Model Control Performance

As a starting point in validating the Virtual Model Control implementation, a controller was constructed and tested in simulation. The implementation of the VMC controller was described in Section 3.3 and it was designed to match the resulting behaviour of the Proportional-Differential controller, as it can be seen from the stiffness and damping constants used in Algorithm 3.8. Again, the results produced by the simulations, shown in Figure 4.3, proves the practical equivalence of the VMC to the classic PD controller.



(a) (b) (c)

Figure 4.3: Simulation results for the Virtual Model Control of a pendulum, a) angle, b) angular velocity and c) angular acceleration.

4.2 *WorkPartner*: Position Control

This section presents simulation results that were obtained from the simulation of the *WorkPartner*'s KTE model presented in Section 3.4. The numerical integration algorithm used here is a simple Euler integration with a fixed time-step of 1ms to 0.1ms, unless otherwise noted.

4.2.1 Object Grasping

The "object grasping" control was tested in simulation and the results are presented here. By "object grasping", it is meant that the end-effectors of the *WorkPartner*'s upper-body were each moved to a separate location. In other words, this scenario is essentially position control of the end-effectors via virtual spring-damper systems attached to them. The gripper motion is not part of this scenario since it is not directly relevant to the tasks of this thesis.

Figure 4.4 shows the traces of both end-effector positions subject to virtual linear spring-damper systems attached at the shown target positions. The control inputs were simulated at a discrete sampling period of 0.1s, corresponding to the typical sampling rate that could be obtained on the actual hardware at the time the simulation was performed. The starting configuration is the home position of the *WorkPartner*'s upper-body, *id est*, straight-forward, up-right torso with both manipulators fully extended forward and straight. As seen from the graph, for a stiffness of 70N/m and a damping of 100Ns/m, there is a certain amount of elliptic overshoot (mainly due to poor sampling rate) and a steady state error around 5cm off-target for both manipulators. The latter is due to the addition of joint friction discrepancies between the simulation model and the controller's model including stiction friction that is not compensated by the controller and slightly increased dry-slipping friction which both reflect to some extent the experimental situation. Nonetheless, the results are in accordance with expectations and show the success of the implementation of the controller and its virtual models.

After experimental tests, the parameters of the control for reaching a target were tuned to better accommodate the limitations of the real hardware. Thus the simulated results presented in Figure 4.5 show the target reaching capability of

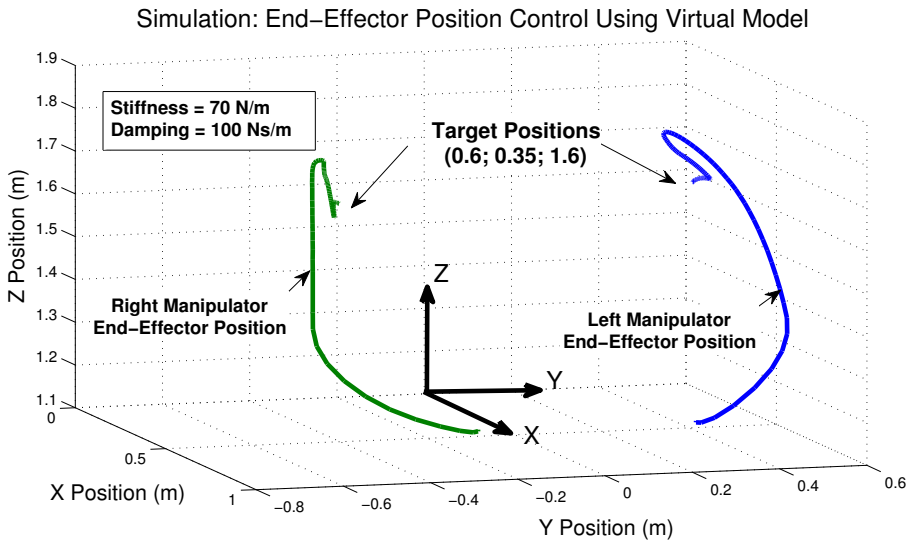


Figure 4.4: Simulation results for position control of two end-effectors of the *WorkPartner* robot, displayed in a 3D plot.

the right manipulators with parameters of 40N/m stiffness of the virtual spring, 8N saturation of the virtual spring's force, and 120Ns/m damping. Again, the results show steady-state errors due to stiction friction, but overall the controller behaves with a critically damped response, as desired. It will be seen in Section 5.1 that the experimental results are in general agreement with the simulated results for Figure 4.5.

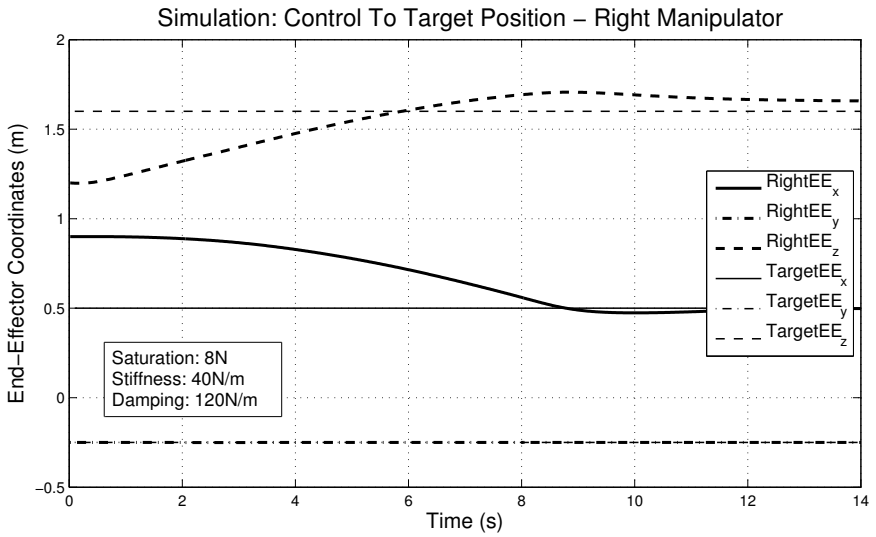


Figure 4.5: Simulation results for position control of the right end-effector of the *WorkPartner* robot, displayed in a time plot.

4.2.2 Trajectory Tracking

The next natural step in position control is trajectory tracking. In this study, the trajectory to be followed is the undamped mass-spring system, *id est*, a harmonic oscillation between the starting point of the manipulator end-effectors and a point 0.4m above and 0.4m closer to the torso. The virtual model parameters were selected during experimental tests and were thus set to 80N/m spring-stiffness, 10N spring-saturation, and 120Ns/m or 150Ns/m damping. In Figure 4.6, the results correspond to a joint hysteresis compensation of stiction friction as described in Section 3.4. One can clearly observe certain jerk behaviour as the manipulator overshoots or undershoots the desired trajectory, comes to a stop, and is prohibited to move by stiction. This is an early evidence that even in the deterministic environment of a numerical simulation, stiction friction is a problematic issue for the performance and accuracy of the trajectory tracking.

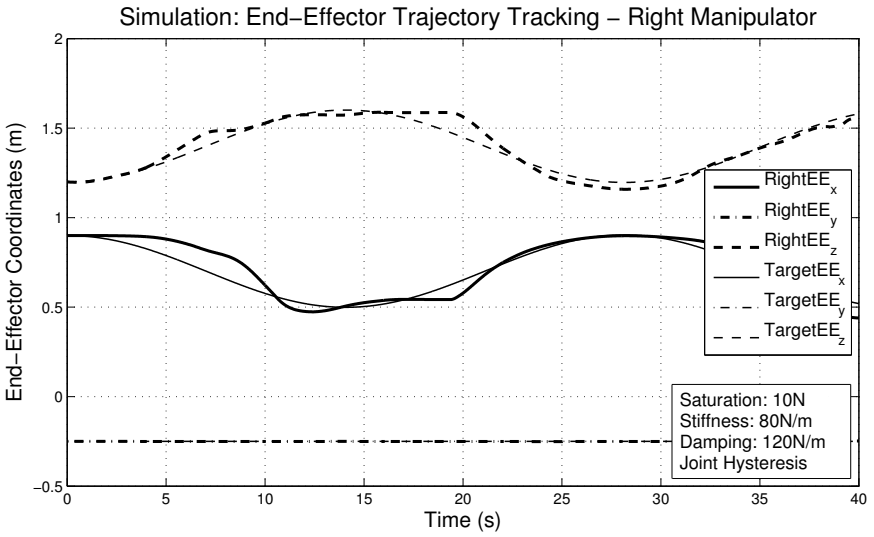


Figure 4.6: Simulation results for trajectory tracking of the right end-effector of the *WorkPartner*, with joint hysteresis.

In a further effort to mitigate the stiction problematic, a pulsing signal was added to the control inputs at the joints in the hopes of breaking stiction friction. Figure 4.7 shows the same tests as the previous one for the same manipulator, with slightly increased damping constant and joint pulses. One can observe that the tracking performance is evidently improved and that stiction stages are not apparent anymore. Of course, the unmodelled inductance of the

electrical drives are a factor that could render the joint pulse strategy less effective during experiments. Finally, it should be noticed that the presence of stiction friction modelled with the micro-slip model and the addition of rapidly changing joint torques rendered the numerical simulation highly unstable and several trials were made with different integrators. First, fixed-step explicit numerical integrators such as Euler, midpoint or Runge-Kutta were completely ineffective. Second, the error-control effort required by variable-step explicit integrators such as Dormand-Prince and Fehlberg methods were also prohibitive and would require unreasonable simulation times (calculated to more than 20 days for 100 seconds). Finally, a fixed-step iterative predictor-corrector Hamming method was used to produce the results of Figure 4.7: although this integrator is unconditionally stable (but lacking error-control), the numerical integration still went unstable after about 35 seconds, as one can observe. This remains an open issue at this time.

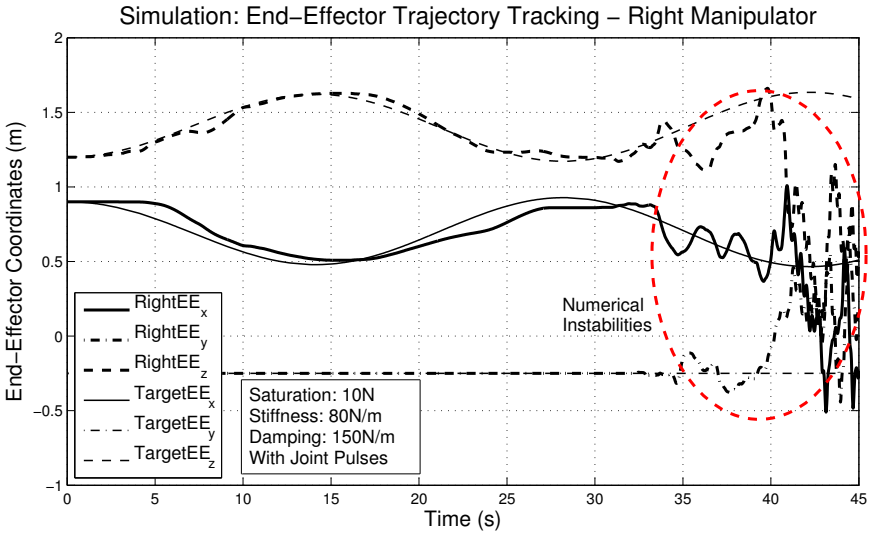


Figure 4.7: Simulation results for trajectory tracking of the right end-effector of the *WorkPartner*, with joint pulses.

4.3 *WorkPartner*: External Force Estimation

A final key element to be tested in simulation was the external force estimator. In this simple case, a weight of 2kg was fixed to both end-effectors at five seconds into the simulation up until the tenth second. The only controller that was applied was the dynamics compensation which would hold the manipulator

steady for the initial segment and then would not be able to support the additional mass, letting the manipulators fall down. Figure 4.8 shows the result of that test in which the sampling rate was set to 50Hz, representing the average sampling rate achievable on the actual hardware. Of course, one can observe from the results that this fairly low sampling rate increases the fluttering of the estimated torque. It is worth noting that similar simulations at high sampling rate exhibited much smoother estimates. On the graph, the estimated torques of the left shoulder inclination and elbow as well as the torso inclination need to be regarded as additive towards the total external force estimated. From the estimates, the values are considered to be very accurate once compensated for the moment-arm of the additional weight and the additional, unmodelled frictional effects. Moreover, a rise-time of less than 0.5 seconds of the estimates demonstrate the dynamic response of the estimator, originally designed with a quasi-steady external force in mind. These results constitute positive evidence of the performance and stability of the external force estimator, experimental results of Section 5.3 will further confirm this observation.

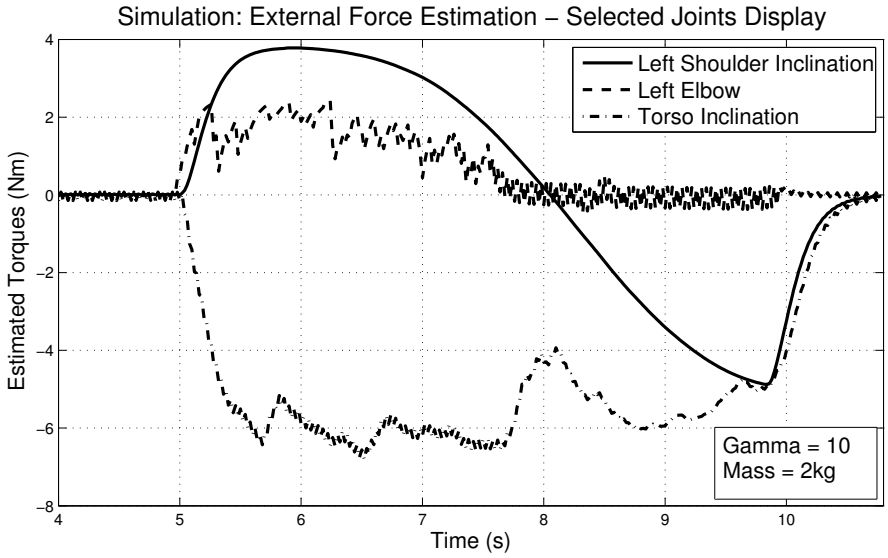


Figure 4.8: Simulated results for external force estimation with 2kg end-mass on each end-effector.

Chapter 5

Experiments

"As far as the laws of mathematics refer to reality, they are not certain, and as far as they are certain, they do not refer to reality."
- Albert Einstein

5.1 Position Control

The following section shows the performance of various basic controllers on the *WorkPartner* robot. The experiments were conducted in June and July 2010 in the laboratories of the Department of Automation and System Technology of Aalto University. Experiments presented here involve the three basic tasks of classical model-based position control for robotic manipulators, these are: dynamics compensation, *id est*, producing a "weightless" manipulator; object grasping, *id est*, approaching an object or designated locations in space using one or both manipulators; and trajectory tracking.

5.1.1 Dynamics Compensation

The first step in the validation of a model-based controller is to estimate the performance of the feedback-linearisation. Ideally, a fully linearised model would make the manipulators completely weightless in the sense that the inertia is

retained (not compensated) but gravity, friction and non-linear dynamics effects are taken into account such that the resulting behaviour is a manipulator that has some inertia but when released should keep a constant speed. In other words, setting the manipulator in motion is similar to pushing a mass into motion, but once motion is acquired it will be conserved until the next human-interaction.

The first test sessions involved manually fine-tuning friction and mass parameters to achieve the aforementioned dynamic behaviour. Then, once a satisfactory state was obtained, the results with small "release periods" in shown in Figure 5.1. It can be seen that the results are not perfect which is expected on real hardware without a model-adaptation training period. Nevertheless, the results are an encouraging start.

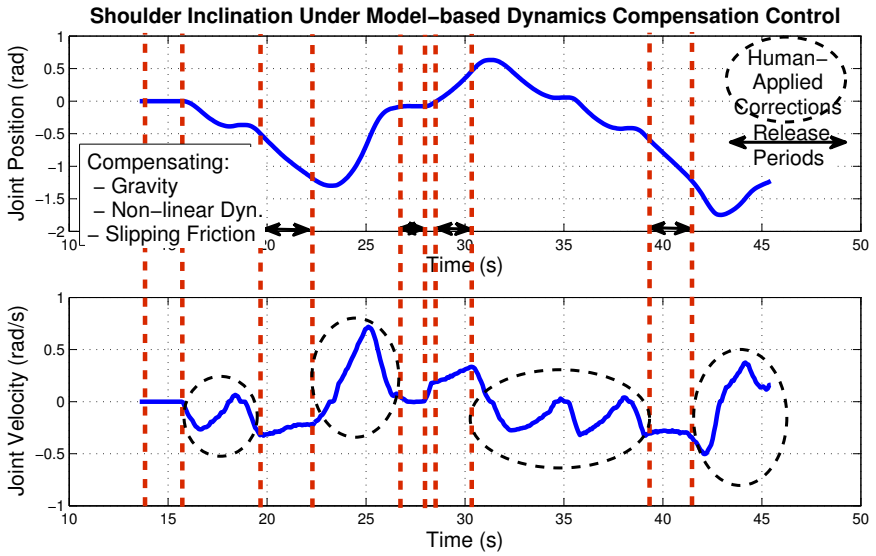


Figure 5.1: Experimental results for dynamics compensation of the *WorkPartner* left manipulator, here the shoulder inclination is shown.

5.1.2 Object Grasping

The previous section has shown reasonable performance of the dynamics compensation, it is now possible to test the object grasping or position control. In this section, "object grasping" is reduced in scope, just as in Section 4.2, to simply reaching given positions with the end-effectors. Similar to the simulated case, the first test was performed by attaching virtual spring-damper systems to two fixed points at coordinate (0.5m; ±0.25m; 1.6m), *id est*, 0.4m above the

initial position of the arms and 0.4m closer to the body. Figure 5.2 shows the results with a spring constant of 40N/m, a saturation value on the spring of 8N and a damping constant of 120Ns/m. In this case, stiction friction was compensated by a hysteresis applied at the joints directly. Note that a hysteresis was also tentatively put on the virtual spring but the resulting oscillations made it impractical and this strategy was abandoned.

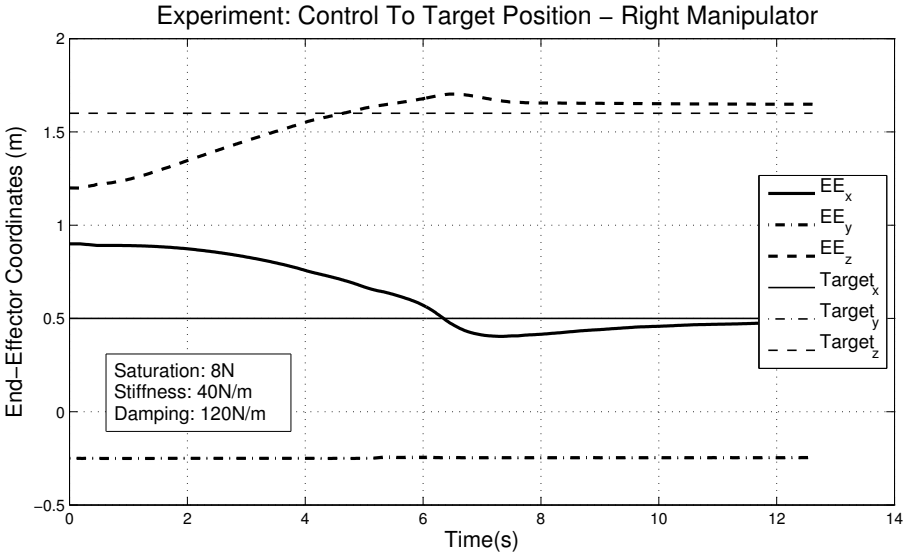


Figure 5.2: Experimental results for target reaching control of the *WorkPartner* right manipulator with a virtual spring-damper model.

Results of Figure 5.2 show a steady-state error, as expected from simulation results, when the end-effectors are close enough to the targets so that control actions are too weak to break stiction of the joints. It can also be observed that the settling time of the controller is roughly 20 seconds with the aforementioned control parameters which were deemed to give the strongest action possible while keeping the joint efforts within their hard limits in speed and torque.

5.1.3 Trajectory Tracking

Next, trajectory tracking naturally followed in the planned test sequence. In this case, the control action could be strengthen as the error between actual end-effector positions and their targets were assumed to remain smaller, the planned trajectories being easier to follow and starting from the initial positions

of the end-effectors. Figure 5.3 shows the results obtained from a virtual spring constant of 80N/m (saturated at 10N) with damping of 120Ns/m. The target trajectories, corresponding to the notably smoother lines in Figure 5.3, were set as harmonic oscillations of low frequency (0.035Hz) from the initial end-effector positions to the same targets as in the previous test case.

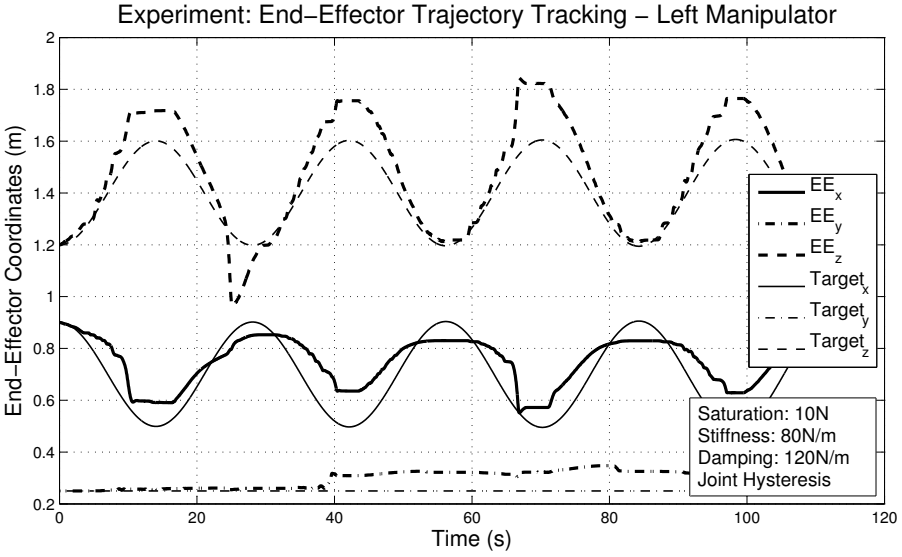


Figure 5.3: Experimental results for trajectory tracking control of the *Work-Partner* left manipulator with a virtual spring-damper model.

It can be seen from the trajectory tracking results that the stiction on the joints creates jerk phenomena. This problem results from the inability of the control action to break stiction until the gap between a target and end-effector is large enough. At this point, the control action is already too strong once the joints are in motion and causes overshoot, which in turn reverses the control action, effectively stopping the motion and creating a new stiction state.

A final strategy was develop to try to finally mitigate stiction problems in the control of the manipulators. Instead of a hysteresis on the joints or on the virtual model controlling the manipulators, pulsing signals were added to the commanded joint torques. These pulse signals were applied at maximum frequency (the Nyquist frequency, about 25Hz in this case) by adding a torque to each joint, corresponding to more than enough to break stiction friction, and adding the opposite torque at the next sampling time of the controller. This method does not effectively cause any visible or significant motion while

always keeping the joints in motion (one could say "micro-motion"). Figure 5.4 shows the improvement of this strategy over the joint hysteresis used in the previous case. One can observe a much smoother tracking although issues remain with the accuracy and reachability for which the singular configurations of the manipulators play a greater role.

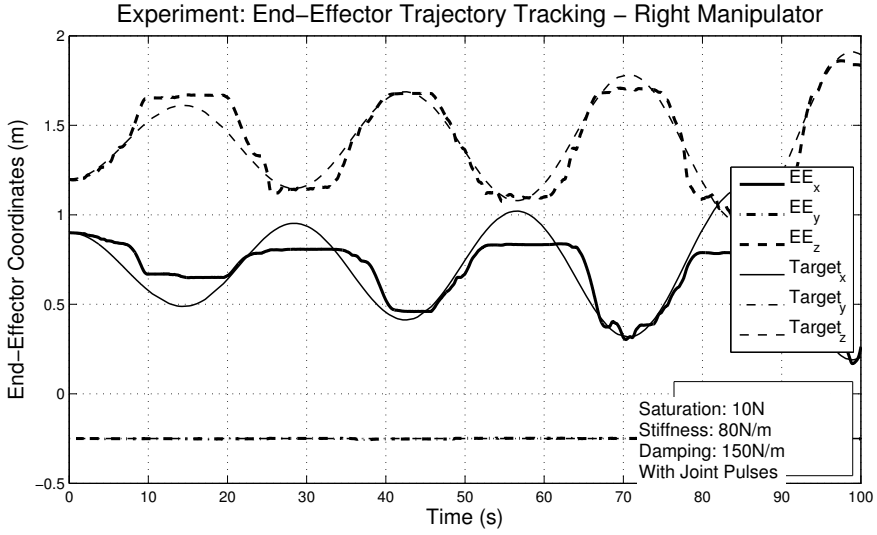


Figure 5.4: Experimental results for trajectory tracking control of the *Work-Partner* right manipulator with a virtual spring-damper model and the application of joint pulses.

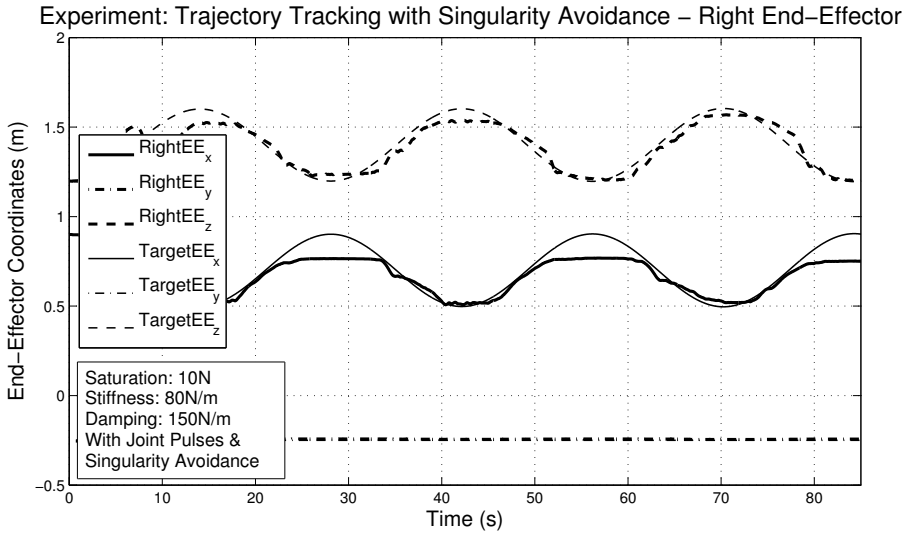


Figure 5.5: Experimental results for trajectory tracking control of the *Work-Partner* right manipulator with a virtual spring-damper model, with joint pulses and singularity avoidance.

Finally, the application of the singularity avoidance behaviour in addition to the trajectory tracking and joint pulses has produced the results shown in Figure 5.5. It can be observed that the quality of the tracking highly increased by staying away from a singular stretched elbow configuration. One can observe as well that the tracking cannot reach the far x-coordinates because these can only be acheived with fully stretched elbows. In order to further compensate this problem, higher gain were applied, as seen in Figure 5.6, where the gains and saturation are doubled for increased performance. One can see there that high trajectory tracking performance can be acheived, finally.

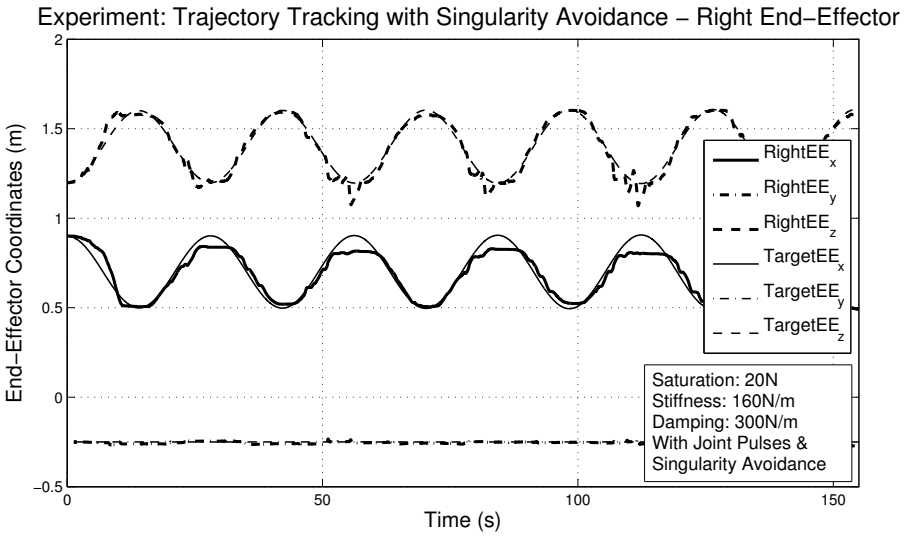


Figure 5.6: Experimental results for trajectory tracking control of the *Work-Partner* right manipulator with a virtual high-gain spring-damper model, with joint pulses and singularity avoidance.

5.2 Object Manipulation with Basic Human Interaction

In this section, the experimental results for object manipulation using both end-effectors as a grasping device are presented in support of the controllers presented in Section 3.4. In all cases, the object is modelled through a virtual flexible-beam model as well as a real inertia model representing the weight of the object such that its gravity pull is compensated for. First, a basic test of the handling of the object is presented, followed by the object manipulation along a

vertical wall, the so-called "hang-a-picture" human-robot cooperative scenario.

5.2.1 Dual-Arm Object Manipulation

The next stage in this project was to test the object manipulation with both manipulators of the *WorkPartner*'s upper-body. In this test case, a box made of soft cardboard of about 35cm width was inserted between the two end-effectors and the controller press-handled it between the end-effectors. The virtual model used was a flexible beam model with a nominal width of 18cm in addition to a 5cm offset to both end-effector centres and a structural stiffness of 80N/m. At first, the compressive action of the virtual model would secure the hold on the box between the end-effectors. Then, the control actions had the effect of keeping the box steady between the end-effectors by compensating for its mass (about 0.35kg in this test). Finally, the test involved a human operator who would move the box to different locations by exerting force on the box only and with slight help from the dynamically compensated manipulators. Figure 5.7 shows the paths of the two end-effectors throughout the five minutes of testing. It must be said that the rapid changes and oscillations are due to the human's interactions and not to control instabilities. Attached to the thesis is also a video of a similar test which shows how steadily the controller handles the box.

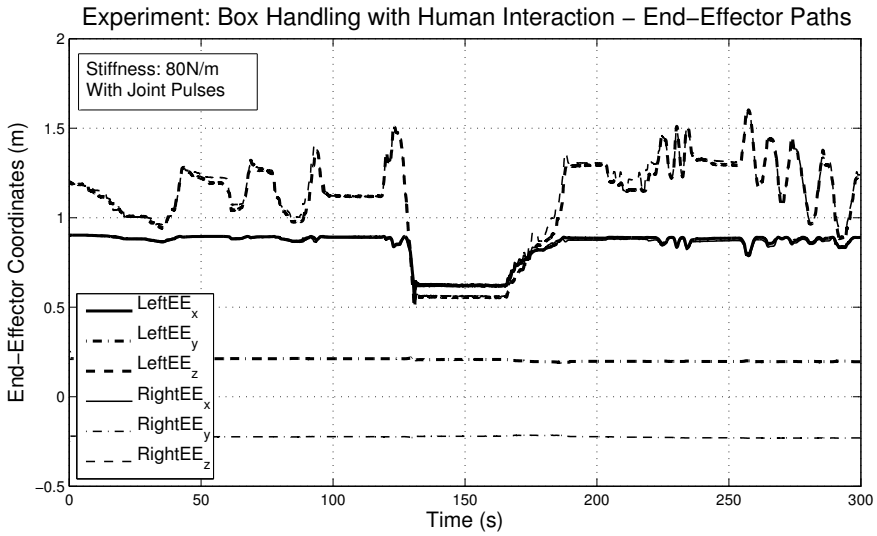


Figure 5.7: Experimental end-effector paths for dual-arm manipulation (regulation), under human interaction, using a virtual flexible beam model and joint pulses.

To further demonstrate the performance of the dual-arm object manipulation in the simple box handling test, Figure 5.8 shows the differences in positions of the end-effectors. Also shown in this figure is the magnitude (Euclidean norm) of the difference vector between end-effectors. One can observe a mean value of 43cm separation which corresponds to a gap of 33cm between the two contact points on the box which was, evidently, compressed by 2cm along its width. Most impressive, however, is a standard deviation of less than a centimetre and a maximum measured deviation of 2.7cm.

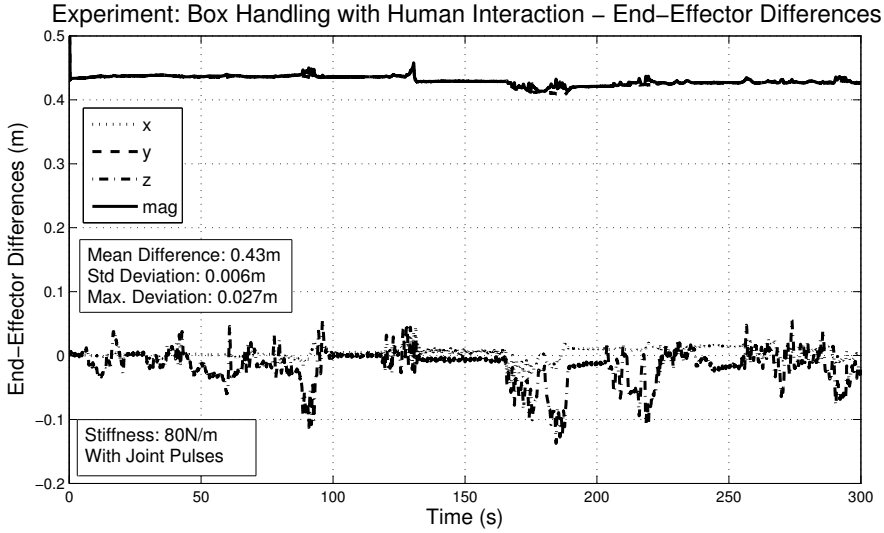


Figure 5.8: Experimental end-effector position differences for dual-arm manipulation (regulation), under human interaction, using a virtual flexible beam model and joint pulses.

5.2.2 Hang-a-Picture Scenario

One of the main objectives of this thesis was to demonstrate a basic cooperative scenario between a human operator and the *WorkPartner* robot, physically interacting. Although it was anticipated that high-level control would be required as a human-intention recognition by the robot, it was found during tests that simple dynamics compensation by the robot was sufficiently compliant to human interactions. And thus, the mission objectives could be encoded as a control incentive in one direction of motion and complete compliance in the others. These observations have enabled this "hang-a-picture" scenario to be formulated as a virtual model controller pulling the manipulated object to-

wards a vertical wall (with 80N/m stiffness) and allowing human interactions easily displace the object or box in any direction along to the wall. To give additional workspace to the robot, the wheels were also used to move the robot forward or backward as a result of the virtual model control computation of the base's reaction force and torque vectors which would be projected to their 2D components and multiplied by a small gain (0.05) to produce driving speed and angular speed control signals.

The results presented in Figure 5.9 show both the capacity of the controller to keep the object on the wall within an reasonable error margin, in time and space, and its ability to cope with the lateral displacements induced by the human operator. After the various human-induced displacements, the *Work-Partner* is capable of keeping the static height of the object while driving it back against the wall. One can easily imagine scenarios related to the assembly of experimental devices or structural components during planetary exploration that could benefit from even such a simple control strategy, or simply use the *WorkPartner* to help hang a picture on a wall, although that would probably be slightly unrealistic.

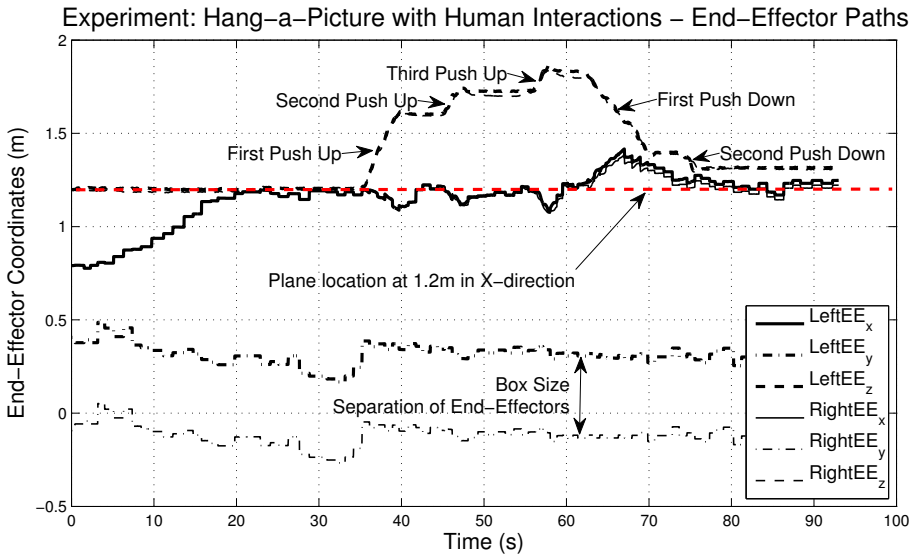


Figure 5.9: Experimental end-effector positions for dual-arm manipulation along a wall, under human interaction, using a virtual flexible beam model, joint pulses and planar constraints.

5.3 External Force Estimation

Although much could be achieved without force feedback, it was of great interest to test the performance of the external force estimation algorithm developed in Section 3.2.3. Because of time constraints, extensive testing was not possible, but the results presented here show promising results of this class of non-linear disturbance observer that was never tested on real hardware in the previous published work (Chen et al., 2000; Korayem and Haghighi, 2008; Nikoobin and Haghighi, 2009).

5.3.1 Estimation under Overweight End-Effector

As first test case, measured weights were hung on the left end-effector of the *WorkPartner* and the resulting external torque estimates were obtained. In those cases, only the feedback linearisation (or dynamics compensation) controller was applied to keep the end-effector stationary at the start of the test and dynamically compensated throughout the rest of the tests. Also, since the stiction friction on joints is generally high, only the shoulder inclination joint would move under the hung weight. And thus, results of Figure 5.10 and 5.11 show the estimated torque on the left shoulder inclination joint (noting that the joint direction means that a positive torque equals to a downward force on the manipulator). First, Figure 5.10 shows the result for a weight of 2kg. One can observe that the estimate fairly rapidly moves up to a value of 10Nm, which eventually reaches zero as the left manipulator reaches a fully dropped position and thus reducing the moment-arm of the weight to none. Subsequently, of course, a rapid and opposite torque is observed which corresponds to the impact, as the manipulator hits the lower-body and stops. An obvious remark to make is that 10Nm does not correspond to a 2kg weight with a 0.9m moment-arm, which should be 17.7Nm. However, the additional, and thus unmodelled, friction caused by the increased normal force on the shoulder joint was estimated, with the best known friction parameters, to account for at least 4Nm of friction torque that reduces the observable external torque to about 14Nm. Moreover, the additional unmodelled or poorly estimated system parameters can easily cause the remaining 4Nm error, not to mention the d'Alembert force reduction due to the acceleration of the end-mass and the rapid decrease of the

external torque by the reduction of the moment-arm does not leave enough time of the estimator to reach a steady value.

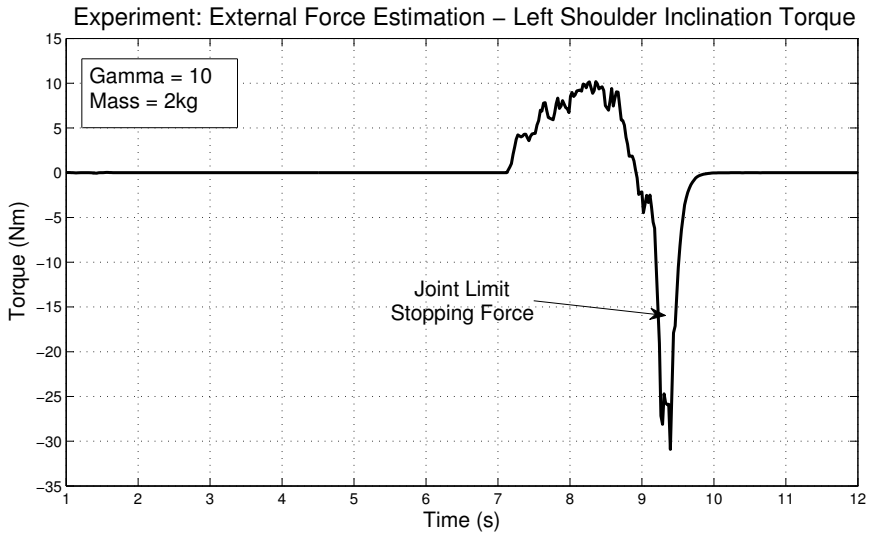


Figure 5.10: Experimental estimation of the external torque on the left shoulder inclination joint due to a 2kg weight, with an observer gain of 10.

Two of the most important aspects of an estimator of a physical quantity, such as external forces, are the bias and the stability. The latter is difficult to assess in the limited tests that were possible before the end of the thesis work. Furthermore, the aforementioned model-accuracy problems constitute many causes for the observed discrepancies and it is thus difficult to verify the biasing of the estimator without improving the model's accuracy via, for example, an adaptive parameter identification method that would automatically minimize discrepancies in the dynamics model. As for stability, even a solid theoretical proof of asymptotic stability is often not guarantee that the estimator can be used safely on the real hardware. Nevertheless, a good indication of stability was provided by a small implementation issue which caused the first estimation step to occur before all hardware drivers were on-line. Once on-line, after 15 to 30 seconds, the second estimation step was subject to a very large integration time-step and, consequently, the first estimates of the external forces were exploding into completely unsound ranges, but yet, after less than a second, the estimates would stabilize right back to expected values, with minimal residual oscillations. It goes without saying further investigation of stability and bias would deepen the knowledge of the behaviour of this estimator.

In the hopes of determining the sensibility of the external force estimation, the

Figure 5.11: Experimental estimation of the external torque on the left shoulder inclination joint due to a 0.75kg weight, with an observer gain of 10.

hung weight was reduced to the bare minimal force which would be enough to set the manipulator in a falling motion: that weight was determined to be 0.75kg. Figure 5.11 shows the result of that experiment. Again, the pattern shows an estimate of the external torque until the impact to the lower-body (in this case, actually, the motion was caught manually). More interestingly, initial small impacts were given to the end-effector to set the manipulator in motion because of the relative weakness of the external force in presence of stiction, and those impacts were observed by the estimator. This gives good indication that, although this observer is designed with a quasi-steady external force assumption, it seems to be capable of also detecting rapidly changing external forces. Finally, the order of magnitude is analogous to the previous test. The theoretical external torque is 6.6Nm from the full moment-arm, and once compensated with the best known friction parameters, the observable torque is about 5.1Nm. The observed torque is around 3.5Nm, from Figure 5.11, and thus there is an error remaining of about 1.6Nm, which is correct proportion with the 2kg case and another test with 1.25kg is also similar in proportion.

5.3.2 Estimation under Trajectory Tracking

As a next test, it was important to verify the behaviour of the external force estimator under full control action. In this case, the trajectory tracking controller,

as used in test cases of Section 5.1.3, was employed. Figure 5.12 shows the results obtained under the sole action of the trajectory tracking virtual model controller, *id est*, without singularity avoidance or joint pulses. An extra weight of 1.25kg was hung from the left end-effector for the last two oscillation periods of the test. One can observe a shift of about 3Nm in the DC-value or constant component of the signal for the last two periods. Additionally, the oscillations observed correspond to the unmodelled dry-friction and uncompensated stiction which seem to amount to about 4Nm peak-to-peak before the additional weight and 6Nm peak-to-peak subsequently. Although, there are obvious problems with the observation of external forces with unmodelled dynamics, these are promising results.

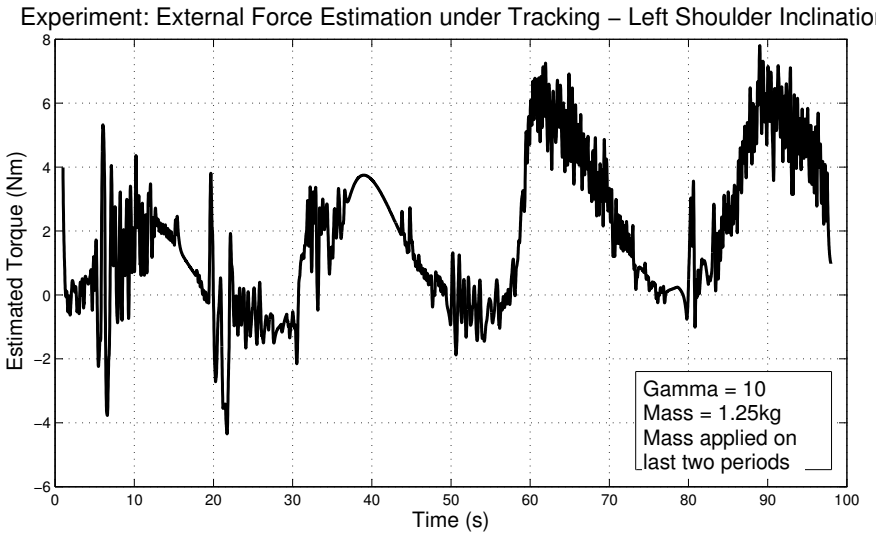


Figure 5.12: Experimental estimation of the external torque on the left shoulder inclination joint due to a 1.25kg weight during last two periods of trajectory tracking control, with an observer gain of 10.

To further the investigation of the behaviour of the external force estimator, a test was performed with not only the trajectory tracking controller, but also the singularity avoidance and the joint pulses. Figure 5.13 shows the results of that test. Again, a weight of 1.25kg was added for the last two periods of the trajectory. It was expected that the applied joint pulses would increase the flutter of the estimator, which could already be observed in the previous test case. After applying a low-pass filter *à posteriori*, the results still show a similar behaviour as the previous case. This time, due to singularity avoidance, the left elbow goes through more motion and thus, most external force effects are observed on that joint. One can, again, observe an oscillation of the stiction

friction estimation which grows larger in amplitude when the additional weight is applied accompanied with a shift of about 4Nm in the DC-value or constant component.

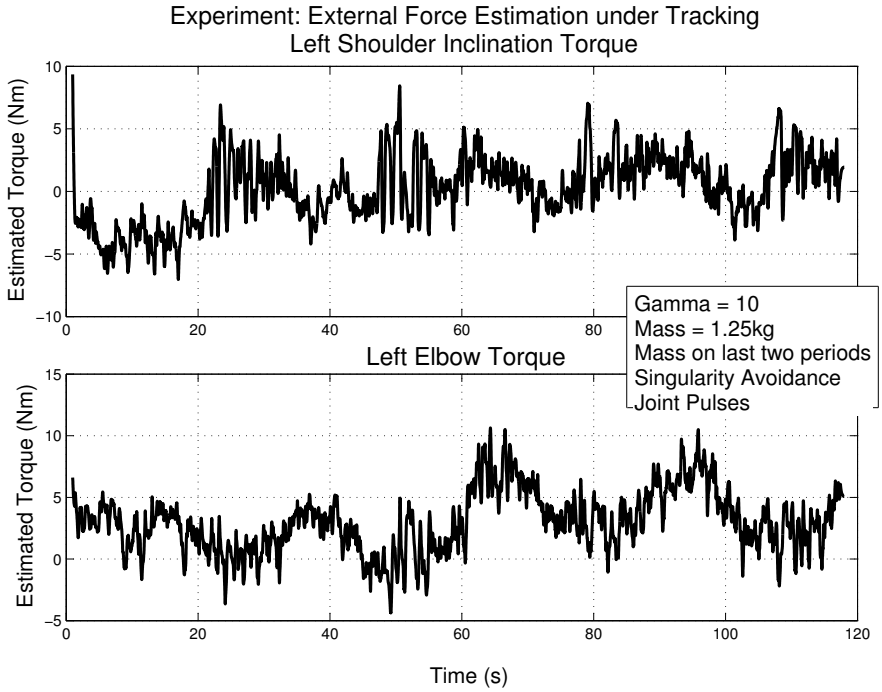


Figure 5.13: Experimental estimation of the external torque due to a 1.25kg weight during last two periods of trajectory tracking control with singularity avoidance and joint pulses, with an observer gain of 10.

5.3.3 Collision Detection

The final test case that time could permit was collision detection. In this case, only dynamics compensation is used to keep the manipulators compliant and the human operator is repeatedly hitting and shoving the manipulators. The collision detection, as previously mentioned, is implemented using thresholds on the external joint torques. In this case, the thresholds of the left and right shoulder inclination joints are of 15Nm. Figure 5.14 shows the results of the external force estimation as well as the collision detector output, whose events are encircled in the graph. Most importantly, the repeated impacts are easily distinguishable on the resulting graph which shows the applicability of this non-linear disturbance observer as the basis of a collision detection scheme.

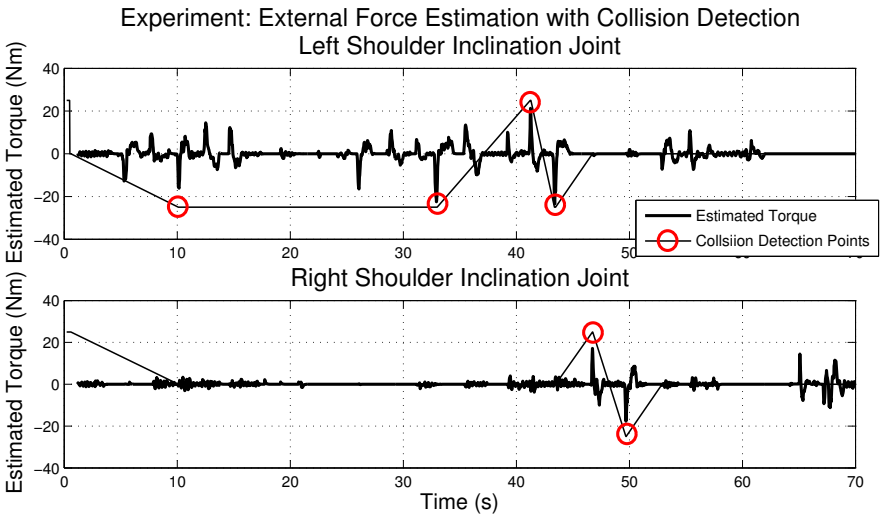


Figure 5.14: Experimental estimation of the external torque on the left and right shoulder inclination joint with the output of collision detection via joint external torque thresholds.

Chapter 6

Summary and Conclusions

"The best computer is a man, and it's the only one that can be mass-produced by unskilled labor."

- Wernher Magnus Maximilian von Braun

This thesis has presented an overview of the work done over the span of seven months on the *SpacePartner* project. The original goals were highly ambitious and have thus lead to great achievements. The work presented has started from incomplete hardware, outdated software and a little amount of serious work on the control architecture for the *WorkPartner*'s upper-body. After these past months, the project is left in much better shape than before, it now has, on a technical point-of-view:

- a full-featured, flexible software platform for developing and running controllers;
- an improved and functional set of motor controllers;
- new MaCI interfaces for the upper-body's drivers;
- a proper multi-body dynamics simulator for the *WorkPartner*'s upper-body based on a modern modelling tool, *id est*, Kinetostatic Transmission Elements;
- a Virtual Model Control architecture that allows intuitive development of behaviour-based control software;

- a set of controllers which perform basic manipulator control tasks, dual-manipulator control and allow physical human-robot interactions.

More importantly, scientific contributions to the field of manipulator dynamics and control, as well as embodied intelligence are numerous in this thesis work. First, several new developments to the Kinetostatic Transmission Elements modelling technique were presented, including Jacobian matrix extraction, single-pass mass matrix computation and a simple framework for building virtual model extensions to real models. Moreover, it was shown that this modelling technique can be used for model-based control as a flexible tool for developing, constructing and computing a kinematic chain model.

Second, contributions were made to the concept of Virtual Model Control. In addition to simply incrementing the body of experimental work and literature on the subject by showing its implementation on a new application, some new behaviours were tested such as dual-arm object manipulation capable of controlling the internal forces through the object, stiction friction compensation to some extent, and singularity avoidance. Furthermore, the newly established harmony between VMC and KTE modelling is a powerful tool for control purposes as well as a very convenient link to multi-body dynamics simulation.

Third, a theoretical contribution to the problem of external force estimation was demonstrated through a novel mathematical formulation for a non-linear disturbance observer. Simulated and experimental results show that the technique is as applicable as it theoretically is demonstrated to be, although performance is not perfect, it showed great promise for future applications.

Finally, experimental results were obtained that demonstrated the performance of the presented controllers as well as showing the sufficiency of passive methods for physical human-robot interactions. Here, passive is meant as control strategies that are not actively interpreting human intentions and reacting to them but rather being completely compliant to the human's interactions. This conclusion agrees with a large body of literature on physical human-robot interactions where it is generally observed that dynamics and gravity compensation alone gives the robot a very natural and compliant feel to the human touch.

6.1 Future Work

Although much has been achieved in this thesis, there is still a large amount of interesting avenues to be taken for future research with the *WorkPartner* robot or other similar robots. As part of the initial plans of this thesis that could not be achieved due to time constraints, there are several further studies to be made, notably, active human intention interpretation, requiring better external force estimation, as well as intelligent reactions and robot intention synthesis. On low-level control, adaptivity should be investigated to increase performance of both the control laws and the external force estimation. Finally, dynamic reconfigurability of the behaviours in the control architecture has not been investigated in terms of synchronization issues and potential for more complex or emergent behaviours.

On the *WorkPartner*'s hardware and software, there are several improvements to be made. First, hardware reliability is a never-ending issue with any custom system, and the *WorkPartner* is no exception. One issue in particular is the maintenance of the motors and gear-trains which has impeded some of the work in this thesis, especially the mechanical safety brakes which could benefit from an update towards motor-mounted disk-brakes as opposed to solenoid-driven gear-hooks. Direct drives would also increase performance by lower gear ratios and reduced backlash and friction.

Second, graphical user interfaces for the software platform presented in this thesis could facilitate its use by third parties. Such a feature was contemplated in this thesis but was of low-priority and thus, not realized, although it would be neat for both the simulation and control software.

In this thesis, the mobility of the robot was used to extend the workspace of the manipulators via an *ad hoc* control of the driving speed as proportional to the VMC-computed base force. However, the development of force-control for the base could allow better integration either through an open-architecture of the lower-body controllers or through emulation of force-control via the development of an admittance controller.

References

Aksman, L.M., Carignan, C.R., and Akin, D.L. (2007). *Force Estimation Based Compliance Control of Harmonically Driven Manipulators.* In *Proceedings of the 2007 IEEE International Conference on Robotics & Automation (ICRA)*, pages 4208–4213.

Ambrose, R., Aldridge, H., Askew, R., Burridge, R., Bluethmann, W., Diftler, M., Lovchik, C., Magruder, D., and Rehnmark, F. (2000). *Robonaut: NASA's Space Humanoid. Intelligent Systems and their Applications, IEEE*, 15(4):57–63. ISSN 1094-7167.

Angeles, J. (2007). *Fundamentals of Robotics Mechanical Systems - Theory, Methods, and Algorithms.* Mechanical Engineering Series. Springer Science+Business Media, 3rd edition.

Arai, H., Takubo, T., Hayashibara, Y., and Tanie, K. (2000). *Human-Robot Cooperative Manipulation Using a Virtual Nonholonomic Constraint.* In *Proceedings of the 2000 IEEE International Conference on Robotics & Automation (ICRA)*, pages 4063–4069.

Bonitz, R.G. and Hsia, T.C. (1996). *Internal Force-Based Impedance Control for Cooperating Manipulators.* In *IEEE Transactions on Robotics and Automation*, volume 12.

Brooks, R. (1986). *A robust layered control system for a mobile robot. Robotics and Automation, IEEE Journal of*, 2(1):14–23. ISSN 0882-4967.

Brooks, R.A., Breazeal, C., Marjanovic, M., Scassellati, B., and Williamson, M.M. (1999). *The Cog project: Building a humanoid robot.*, volume 1562, pages 52–87. Springer Berlin / Heidelberg. ISBN 978-3-540-65959-4.

Chen, W., Balance, D., Gawthrop, P., and O'Reilly, J. (2000). *A nonlinear disturbance observer for robotic manipulators.* In *IEEE Transactions on Industrial Electronics*, volume 47, pages 932–938.

Coleshill, E., Oshinowo, L., Rembala, R., Bina, B., Rey, D., and Sindelar, S. (2009). *Dextre: Improving maintenance operations on the International Space Station. Acta Astronautica*, 64:869–874.

Diftler, M.A., R. Platt, J., Culbert, C.J., Ambrose, R.O., and Bluethmann, W.J. (2003). *Evolution of the NASA/DARPA Robonaut Control System.* In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation (ICRA)*, pages 2543–2548.

Doetsch, K. (2005). *Canada's role on space station. Acta Astronautica*, 57:661–675.

Edsinger, A. (2004). *A Behavior Based Approach to Humanoid Robot Manipulation.* Technical report, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Research Qualifying Exam, Cambridge, MA.

Edsinger, A. (2005). *Developmentally Guided Ego-Exo Force Discrimination for a Humanoid Robot.* In *Proceedings of the Fifth International Workshop on Epigenetic Robotics.*

Edsinger, A. (2007). *Robot Manipulation in Human Environments.* Ph.D. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

Edsinger, A. and Kemp, C. (2007a). *Human-Robot Interaction for Cooperative Manipulation: Handing Objects to One Another.* In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication (ROMAN).*

Edsinger, A. and Kemp, C. (2007b). *Two Arms are Better than One: Designing Robots that Assist People in Everyday Manual Tasks.* In *Proceedings of the IEEE International Conference on Advanced Robotics (ICAR).*

Edsinger, A. and Weber, J. (2004). *Domo: A Force Sensing Humanoid Robot for Manipulation Research.* In *Proceedings of the IEEE/RSJ International Conference on Humanoid Robotics.*

Guo, C. and Sharlin, E. (2008). *Utilizing Physical Objects and Metaphors for Human Robot Interaction.* In *Proceedings of Artificial Intelligence and Simulation of Behavior (AISB '08)*. AISB Press.

Haddadin, S., Albu-Schäffer, A., Luca, A.D., and Hirzinger, G. (2008). *Collision Detection and Reaction: A Contribution to Safe Physical Human-Robot Interaction.* In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363.

Heiskanen, P. (2008). *Development of a Dynamic Simulator of a Mobile Robot for Astronaut Assistance.* Master's thesis, Luleå University of Technology and Helsinki University of Technology.

Hibbeler, R.C. (2003). *Engineering Mechanics: Statics & Dynamics.* Prentice Hall, 10th edition. ISBN 0131046241.

Hogan, N. (1985a). *Impedance Control: An Approach to Manipulation: Part I - Theory. Journal of Dynamic Systems, Measurement, and Control*, 107:1–7.

Hogan, N. (1985b). *Impedance Control: An Approach to Manipulation: Part II - Implementation. Journal of Dynamic Systems, Measurement, and Control*, 107:8–16.

Hogan, N. (1985c). *Impedance Control: An Approach to Manipulation: Part III - Applications. Journal of Dynamic Systems, Measurement, and Control*, 107:17–24.

Hu, J., Pratt, J., Chew, C.M., Herr, H., and Pratt, G. (1998). *Adaptive Virtual Model Control of a Bipedal Walking Robot. Intelligence and Systems, IEEE International Joint Symposia on*, page 245. doi:http://doi.ieeecomputersociety.org/10.1109/IJSIS.1998.685453.

Imai, M., Ohsawa, H., and Narumi, M. (2005). *Designing Human-Robot Interaction via Physical World Objects.* In *Proceedings of the 2005 IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 155–159.

Jet Propulsion Laboratory, NASA Ames Research Center, Carnegie-Mellon, and University of Minesota (2008). *CLARAty*

Reusable Robotic Software (http://claraty.jpl.nasa.gov).
URL: *http://claraty.jpl.nasa.gov*

Kecskeméthy, A., Lange, C., and Grabner, G. (2001). *Object-Oriented Modeling of Multibody Dynamics Including Impacts.* In *European Conference on Computational Mechanics (ECCM).*

Korayem, M.H. and Haghighi, R. (2008). *Nonlinear disturnbance observer for robot manipulators in 3D space*, volume 5314 of *Lecture Notes in Computer Science*, pages 14–23. Springer Berlin / Heidelberg.

Lepkowski, J. (2003). *Motor Control Sensor Feedback Circuits.* Technical report, Microchip Technology Inc.

Lewis, F.L., Dawson, D.M., and Abdallah, C.T. (2004). *Robot Manipulator Control: Theory and Practice.* Control Engineering Series. Marcel Dekker, Inc., 2nd edition. ISBN 0-8247-4072-6.

Mukherji, R., Rey, D.A., Stieber, M., and Lymer, J. (2001). *Special Purpose Dexterous Manipulator (SPDM) Advanced Control Features and Development Test Results.* In *Proceeding of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space.*

Nikoobin, A. and Haghighi, R. (2009). *Lyapunov-Based Nonlinear Disturbance Observer for Serial n-Link Robot Manipulators. Journal of Intelligent Robotic Systems*, 55:135–153.

Orlandea, N. (1987). *ADAMS Theory and Application*, volume 16, pages 121–166. Taylor & Francis Group.

Persson, S.M. (2007). *A General Implementation of a Closed-Form Solution to Contact Dynamics.* Undergraduate Honours Thesis, McGill University, Department of Mechanical Engineering.

PHRIENDS, Center E. Piaggio, Institute of Robotics & Mechatronics (DLR), KUKA Robotics, Laboratoire d'Analyse et d'Architecture des Systèmes (Centre National de la Recherche Scientifique), DIS Robotics Laboratory (Università di Roma), and PRISMA Lab (Università degli Studi di

Napoli Frederico II) (2008). *Physical Human-Robot Interaction: Dependability and Safety (www.phriends.org).*
URL: *www.phriends.org*

Pratt, J. (1995). *Virtual Model Control of Biped Walking Robot.* Master's thesis, Massachusetts Institute of Technology.

Pratt, J., Chew, C.M., Torres, A., Dilworth, P., and Pratt, G. (2001). *Virtual Model Control: An Intuitive Approach for Bipedal Locomotion. The International Journal of Robotics Research*, 20(2):129–143.

Raibert, M. and Craig, J. (1981). *Hybrid position/force control of manipulators. In the Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, 103:126–133.

Rojas, J.L. (2009). *Autonomous Cooperative Assembly by Force Feedback Using a Control Basis Approach.* Ph.D. thesis, Graduate School of Vanderbilt University.

Romano, R.A. (2003). *Real-Time Multi-Body Vehicle Dynamics Using A Modular Modeling Methodology. SAE Technical Paper Series.*

Salisbury, J.K. (1980). *Active stiffness control of a manipulator in Cartesian coordinates.* In *Proceedings of the 19th IEEE Conference on Decision and Control.*

Sheikh, F.I. (2008). *Real-Time Human Arm Motion Translation for the WorkPartner Robot.* Master's thesis, Luleå University of Technology and Helsinki University of Technology.

Singer, S.M. and Akin, D.L. (2010). *Scheduling robot task performance for a cooperative human and robotic team. Acta Astronautica*, 66:102–116.

Stroustrup, B. (1997). *The C++ Programming Language.* Addison-Wesley Professional, 3rd edition.

Sutter, H. and Alexandrescu, A. (2004). *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices.* Bjarne Stroustrup's C++ In Depth. Addison-Wesley Professional.

YANG, Z., POO, A., AND HONG, G. (1993). *A new method for implementing active stiffness control of robot manipulator by using sliding mode.* In *Proceedings of the 1993 IEEE*, pages 165–170. IEEE.

ZEBENAY, M.M. (2009). *Manipulator Control for Physical Astronaut-Robot Interaction.* Master's thesis, Luleå University of Technology and Helsinki University of Technology.

Appendix A

Summary List of Control and Simulation Nodes

The following is a list of nodes, or systems as of the signals and systems implementation, that were implemented for the purpose of this thesis.

Name:	**MaCI Joint Group Control Client Interface Node**
Inputs:	Joint torque (async.), for each joint.
Outputs:	Joint position and velocity, for each joint with a time tag for the measurement.
Function:	Acquires the measurements of the joint state variables and asynchronously sets the control torques for the joints, all via the Machine Control Interface (MaCI).

Name:	**Simulation Interface Node**
Inputs:	Joint torque (async.), for each joint.
Outputs:	Joint position and velocity, for each joint with a time tag for the measurement.
Function:	Simulates the dynamics model of the manipulators, extracts the simulated joint states at some fixed sampling rate and asynchronously uses the control torques to drive the simulation.

Name: **Feedback Linearisation Control Node**
Inputs: Joint positions and velocities (sync.).
Outputs: Joint torques.
Function: Uses the measured (or simulated) joint states to compute the internal model dynamics and applies the inverse torques to cancel the non-linear dynamics.

Name: **Inner-loop Control Node**
Inputs: Joint states (sync.), base frame position (async.), and end-effectors force and torque vectors.
Outputs: Joint torques, base's linear and angular speed, and end-effector frames.
Function: Uses the measured (or simulated) joint states and base position to compute the internal model dynamics, serves the calculated end-effector frames to sub-sequent nodes, and uses the end-effector force and torque vectors to inversely propagate forces through the internal model to compute the joint torques and desired base frame speeds.

Name: **MaCI Base Interface Node**
Inputs: Desired base-frame speeds (linear and angular).
Outputs: Measured base-frame position.
Function: Sets up a MaCI SpeedCtrl client as well as a MaCI Position client to communicate with the *WorkPartner*'s base controllers to obtain the SLAM-estimated pose as well as controlling the linear and angular speed via the wheel driver.

Name: **Elmo Joint Interface Node**
Inputs: Joint torque (async.), for each joint.
Outputs: Joint position and velocity, for each joint with a time tag for the measurement.
Function: Acquires the measurements of the joint state variables and asynchronously sets the control torques for the joints, using direct Elmo motor control drivers (by-passing MaCI for better performance).

Name: **Joint Pulses Node**
Inputs: Joint velocities (sync.).
Outputs: Joint torques.
Function: Generates a fixed-amplitude square signal on all static joints whose speed is below a given threshold. The frequency of the signal corresponds exactly to that of the interface node providing the joint velocity measurements.

Name: **Singularity Avoidance Node**
Inputs: Joint states (sync.), only those of concern for singularities.
Outputs: Joint torques.
Function: Generates PD control torques on certain joints (left-C, right-C, torso-G and torso-F) to avoid undesirable singular configurations, such as bent elbows, or maintain a desired configuration, such as straight-up and forward torso.

Name: **Attract End-Effector Node**
Inputs: Either end-effector frame measurements (sync.) and some anchor frame measurements (async.).
Outputs: Force and torque vectors to be applied to the end-effector.
Function: Keeps a virtual model of a linear spring-damper system with saturation that is linked between the given end-effector frame and some anchor, also given as input. The anchor is thus the desired position or trajectory-tracker. The node output control forces and torques to the given end-effector.

Name: **Constant Attract Point Node**
Inputs: None.
Outputs: Fixed anchor frame to serve the attract end-effector node.
Function: This node simply periodically resets an anchor signal to a given fixed value.

Name: **Oscillatory Attract Point Node**
Inputs: Feedback synchronizer signal (sync.).
Outputs: Oscillating anchor frame to serve the attract end-effector node.
Function: This node synchronizes to the interface node (with a special synchronizer signal that bares only a time-stamp and no data) and performs a time-integration of a harmonic oscillator from a given starting point and about a given central point in space.

Name: **Object-Handling Node**
Inputs: Both end-effector frame measurements (sync.) and net force on the handled object (async.).
Outputs: Force and torque vectors for each end-effector, and the updated object's centre-of-mass.
Function: This node holds a virtual model of a flexible beam between both end-effectors. At every new feedback signal, the control force and torque vectors are applied to the virtual model of the object, along with its inertial loads, to compute the combined effect of net forces and restitutive forces on both end-effectors.

Name: **Object-to-Plane Node**
Inputs: Object's centre-of-mass (sync.).
Outputs: Object's force vector.
Function: This node reads the estimated object frame and applies a linear, saturated spring-damper virtual model between the object and an anchor which is sliding on a given virtual plane.

Appendix B

External Force Estimation Article

Formulation of a Lyapunov-Stable Design of a Non-Linear Disturbance Observer for N-Revolute Joint, Serial Manipulator

Sven Mikael Persson

August 1, 2010

1 Non-linear Disturbance Observer Law

The following presents the design of a non-linear disturbance observer for estimating external forces on a N-revolute joint serial or tree-structured manipulator. The work is inspired by the work of Chen et al. (2000), Korayem and Haghighi (2008) and Nikoobin and Haghighi (2009) who designed non-linear disturbance observers to be Lyapunov-stable with respect to the generalized mass matrix in estimating the external or unmodelled joint torques. By digging deeper in the structure of the generalized mass matrix for a serial manipulator with only revolute joints, the stability criteria are reformulated in terms of induced norms and manipulator parameters. The basic estimation equations for any kinematic chain are as follows:

$$\tau_{ext} = M(q)\ddot{q} + C(q,\dot{q})\dot{q} - \tau_{ego}(q,\dot{q}) - \tau_{mot} \tag{1}$$

$$\dot{\hat{\tau}}_{ext} = -L(q,\dot{q})\hat{\tau}_{ext} + L(q,\dot{q})(M(q)\ddot{q} + C(q,\dot{q})\dot{q} - \tau_{ego}(q,\dot{q}) - \tau_{mot}) \tag{2}$$

Where $\hat{\tau}_{ext}$ is the estimate of joint torques that are a result of the external forces applied on the system and $L(q,\dot{q})$ is a correction gain which can be determined to stabilize the error dynamics. The following substitution is made to facilitate the estimation:

$$\hat{\tau}_{ext} = \psi + p(q,\dot{q}) \tag{3}$$

$$\dot{\hat{\tau}}_{ext} = \dot{\psi} + \frac{\partial p}{\partial \dot{q}}\ddot{q} + \frac{\partial p}{\partial q}\dot{q} \tag{4}$$

By substitution of equation 4 into equation 2 the following relation is established that will eliminate the need for the acceleration term in the estimation equation:

$$\frac{\partial p}{\partial \dot{q}} = L(q,\dot{q})M(q) \tag{5}$$

And thus the following estimation equation is obtained:

$$\dot{\psi} = -L(q,\dot{q})\psi + L(q,\dot{q})(C(q,\dot{q})\dot{q} - \tau_{ego}(q,\dot{q}) - \tau_{mot} - p(q,\dot{q})) - \frac{\partial p}{\partial q}\dot{q} \tag{6}$$

Noting that the error dynamics remain the same after the change of variable and thus:

$$\dot{e} = -L(q,\dot{q})e \tag{7}$$

From the dynamics of a serial kinematic chain, we have the following relations:

$$\mu_{cm}(q,\dot{q}) = M_{cm}t_{cm}(q,\dot{q}) = \begin{bmatrix} M_{cm,1} & 0 & \cdots & 0 \\ 0 & M_{cm,2} & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & M_{cm,m} \end{bmatrix} \begin{bmatrix} t_{cm,1} \\ t_{cm,2} \\ \vdots \\ t_{cm,m} \end{bmatrix} \tag{8}$$

$$t_{cm}(q,\dot{q}) = T_{cm}(q)\dot{q} = \begin{bmatrix} [J_{cm,1}(q)]_0 & \cdots & [J_{cm,1}(q)]_{n-1} \\ \vdots & \ddots & \vdots \\ [J_{cm,m}(q)]_0 & \cdots & [J_{cm,m}(q)]_{n-1} \end{bmatrix} \begin{bmatrix} \dot{q}_0 \\ \vdots \\ \dot{q}_{n-1} \end{bmatrix} \tag{9}$$

$$w_{cm} = \dot{\mu}_{cm}(q,\dot{q}) = M_{cm}\dot{t}_{cm}(q,\dot{q}) + \Omega_{cm}M_{cm}t_{cm}(q,\dot{q}) \tag{10}$$

Where the M_{cm} is the block-diagonal matrix of all mass elements (at centre of masses and expressed in principal axis coordinates); t_{cm}, μ_{cm} and w_{cm} are the stacked vectors of twist, momentum and wrench, respectively, for all mass elements; $[J_{cm,j}]_i$ is the Jacobian matrix mapping joint coordinate i to the centre-of-mass twist of mass element j; and finally, Ω_{cm} is the angular velocity cross-product matrix, stacked for all mass elements. These equations will not explicitly be needed in the non-linear disturbance observer, but it is useful to remark that the mass matrix of all mass elements is, component-wise, constant and thus all time-derivatives of the generalized system mass matrix can be expressed via the time-derivatives of the twist-shaping matrix T_{cm}. As seen in chapter 7 of Angeles (2007), the generalized mass matrix $M(q)$ can be expressed as follows:

$$M(q) = T_{cm}^T(q)M_{cm}T_{cm}(q) \tag{11}$$

$$\dot{M}(q) = \dot{T}_{cm}^T(q)M_{cm}T_{cm}(q) + T_{cm}^T(q)M_{cm}\dot{T}_{cm}(q) \tag{12}$$

Coming back to the non-linear observer law, the function $p(q,\dot{q})$ can be chosen to be:

$$p(q,\dot{q}) \equiv \gamma M(q)\dot{q} = \gamma T_{cm}^T(q)M_{cm}T_{cm}(q)\dot{q} \tag{13}$$

$$\dot{p} = \gamma M(q)\ddot{q} + \gamma\dot{M}(q)\dot{q} \tag{14}$$

$$L(q,\dot{q}) \equiv L = \gamma > 0 \tag{15}$$

2 Lyapunov Stability

Given that $M(q)$ is a symmetric, positive-definite matrix, it makes the following function a candidate Lyapunov function:

$$V \equiv \frac{1}{2}e^T M(q)e \tag{16}$$

By taking the time-derivative and the error dynamics equation 7, we get:

$$\begin{aligned} \dot{V} &= \frac{1}{2}\left(\dot{e}^T M(q)e + e^T\dot{M}(q)e + e^T M(q)\dot{e}\right) \\ &= \frac{1}{2}\left(-e^T L^T(q,\dot{q})M(q)e - e^T M(q)L(q,\dot{q})e + e^T\dot{M}(q)e\right) \\ &= \frac{1}{2}\left(-e^T(2\gamma M(q) - \dot{M}(q))e\right) \\ &= \frac{1}{2}\left(-e^T((\gamma T_{cm}^T(q) - \dot{T}_{cm}^T(q))M_{cm}T_{cm}(q) + T_{cm}^T(q)M_{cm}(\gamma T_{cm} - \dot{T}_{cm}(q)))e\right) \\ &= -<(\gamma T_{cm}(q) - \dot{T}_{cm}(q))e, M_{cm}T_{cm}(q)e> \end{aligned} \tag{17}$$

As one would expect, making γ very large would make $\gamma T_{cm}(q) - \dot{T}_{cm}(q) \approx \gamma T_{cm}(q)$ which, in turn, would make equation 17 negative definite and hence, asymptotic stability would be achieved. However, due to noise and uncertainty amplification caused by a large value of γ, it is always desirable to know to lower stability bound on γ such that robustness trade-offs can be assessed for the system given its inherent uncertainty. Using linearity of the inner-product $< \cdot, \cdot >$ and its upper-bound, we get:

$$\dot{V} = -\gamma e^T T_{cm}^T(q) M_{cm} T_{cm}(q) e + e^T \dot{T}_{cm}^T(q) M_{cm} T_{cm}(q) e \tag{18}$$

$$\dot{V} \leq (-(\gamma \lambda_{min}(M(q)) + \lambda_{max}(\dot{T}_{cm}^T(q) M_{cm} T_{cm}(q))) e^T e \tag{19}$$

The above leads to the following lower bound on γ for asymptotic stability of the Lyapunov function:

$$\begin{aligned} \gamma &\geq \frac{\lambda_{max}\left(\dot{T}_{cm}^T(q) M_{cm} T_{cm}(q)\right)}{\lambda_{min}(M(q))} \\ &\geq \frac{\rho\left(\dot{T}_{cm}^T(q) M_{cm} T_{cm}(q)\right)}{\lambda_{min}(M(q))} \end{aligned} \tag{20}$$

3 Mass Matrix of a Serial Manipulator

In equation 20, both elements of the fraction are required for the lower-bound on γ. The denominator, the lowest eigenvalue of the mass matrix, is generally easily obtained for any kinematic chain and is always positive because of the positive-definite property of this matrix. To obtain it, one can either analytically derive it or run some numerical optimization method to find the minimum eigenvalue, or yet again, assess the minimum-inertia configuration with respect to every joint. An even easier but more conservative approximation, in the case of an actuated manipulator, is to simply use the constant term in the definition of the mass matrix, corresponding to the motor inertias which diagonally represent lower bounds to the lowest eigenvalues. The numerator in equation 20, on the other hand, does not depend on the motor inertias which are constant and thus require the inspection of the overall kinematics. Therefore, we start by a basic definition of the matrix $\dot{T}_{cm}$:

$$\dot{T}_{cm}(q) = \begin{bmatrix} \left[\dot{J}_{cm,1}(q)\right]_0 & \cdots & \left[\dot{J}_{cm,1}(q)\right]_{n-1} \\ \vdots & \ddots & \vdots \\ \left[\dot{J}_{cm,m}(q)\right]_0 & \cdots & \left[\dot{J}_{cm,m}(q)\right]_{n-1} \end{bmatrix} \tag{21}$$

Equation 21 does not bare much insight into the structure and induced norm of $\dot{T}_{cm}$ and hence, further derivations are needed. Up to this point, no assumptions have been made about the structure of the kinematic chain and thus, equation 20 can be applied, as a starting point, to derive general relations for almost any type of mechanical system. The two assumptions about the kinematic chain that will be made in the derivation that follows are that:

- the system is a serial kinematic chain (also generalizes to a tree-structured chain);
- and all the joints are actuated revolute joints (as is the case for the majority of applications in practice, including the *WorkPartner* robot used in this work).

By ignoring the motor inertias excluded from $M(q)$ because they are nilpotent after the time-derivative is taken, and by assuming that each rigid-link between two joints has only one mass element associated with it, we can formulate the $T_{cm}(q)$ matrix, assuming proper ordering as follows (and similarly for $\dot{T}_{cm}(q)$):

$$T_{cm}(q) = \begin{bmatrix} [J_{cm,1}(q)]_0 & 0_{6\times1} & \cdots & 0_{6\times1} \\ [J_{cm,2}(q)]_0 & [J_{cm,2}(q)]_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots \\ [J_{cm,n}(q)]_0 & [J_{cm,n}(q)]_1 & \cdots & [J_{cm,n}(q)]_{n-1} \end{bmatrix} \tag{22}$$

For a serial, revolute-joint kinematic chain, we can expand the definition of the Jacobians $[J_{cm,j}]_i$ which compose the block-matrix $T_{cm}(q)$ and represent the twist at centre-of-mass j resulting from unit joint velocity i as follows:

$$[J_{cm,j}]_i(q) = W_{cm,j} Q_{j-1}^T(q_{j-1}) W_{j-1} \cdots Q_{i+1}^T(q_{i+1}) W_{i+1} [J_{i+1}]_i \tag{23}$$

where,

$$W_k = \begin{bmatrix} [Q_k]_{k-1}^T & 0_{3\times3} \\ 0_{3\times3} & [Q_k]_{k-1}^T \end{bmatrix} \begin{bmatrix} I_{3\times3} & 0_{3\times3} \\ -[[\vec{a}_k]_{k-1}\times] & I_{3\times3} \end{bmatrix} \tag{24}$$

$$Q_k^T(q_k) = \begin{bmatrix} [Q_k]_k^T(q_k) & 0_{3\times3} \\ 0_{3\times3} & [Q_k]_k^T(q_k) \end{bmatrix} \tag{25}$$

with the following terms:

$[J_i]_{i-1}$ is the Jacobian of the i^{th} link by the $(i-1)^{th}$ joint;
$[Q_i]_{i-1}$ is the i^{th} link angular offset in $(i-1)^{th}$ coordinates;
$[Q_{cm,i}]_{i-1}$ is the i^{th} link's principal axes in $(i-1)^{th}$ coordinates;
$[\vec{a}_i]_{i-1}$ is the i^{th} link offset in $(i-1)^{th}$ coordinates;
$[\vec{a}_{cm,i}]_{i-1}$ is the i^{th} link centre-of-mass offset in $(i-1)^{th}$ coordinates;
$[Q_i]_i(q_i)$ is the i^{th} joint rotation matrix in i^{th} coordinates.

In other words, the W_i matrix represents the spatial link offset which can take a twist vector located and expressed in the coordinate system directly after the previous joint transformation to the coordinate system directly before the next joint on the link or, terminally, at the link's centre-of-mass. While the $Q_i^T(q_i)$ matrix is the spatial transformation of the joint i, transposed or inverted; thus, it takes a twist vector expressed in the coordinate system just before joint transformation i to the coordinate system just after it. Therefore, a simpler way to mathematically state the above relationship is by the following recursive relations between the link twists:

$$[t_i]_i = W_i(Q_{i-1}^T(q_{i-1})[t_{i-1}]_{i-1} + [J_i]_{i-1}\dot{q}_{i-1}) \tag{26}$$

$$[t_{cm,i}]_i = W_{cm,i}(Q_{i-1}^T(q_{i-1})[t_{i-1}]_{i-1} + [J_i]_{i-1}\dot{q}_{i-1}) \tag{27}$$

With the definition of the $T_{cm}(q)$ matrix of equation 22 and it's relation to the vector of stacked centre-of-mass twists of equation 9, it becomes natural to call it a twist-shaping matrix or more exactly, the link centre-of-mass twist-shaping matrix.

Moving towards the time-derivation of the $T_{cm}(q)$ matrix, it is clear from equation 23 that the only time-dependencies reside in the $Q_k^T(q_k)$ matrices. Classic *SO3* equations can be used

here and are repeated for quick reference:

$$\begin{aligned}
\dot{Q}_k^T(q_k) &= \begin{bmatrix} \left[\dot{Q}_k\right]_k^T(q_k) & 0_{3\times3} \\ 0_{3\times3} & \left[\dot{Q}_k\right]_k^T(q_k) \end{bmatrix} \\
&= \begin{bmatrix} [Q_k]_k^T(q_k)\,[-\vec{e}_k\times] & 0_{3\times3} \\ 0_{3\times3} & [Q_k]_k^T(q_k)\,[-\vec{e}_k\times] \end{bmatrix} \dot{q}_k \\
&= \dot{q}_k Q_k^T(q_k) \begin{bmatrix} [-\vec{e}_k\times] & 0_{3\times3} \\ 0_{3\times3} & [-\vec{e}_k\times] \end{bmatrix} \equiv \dot{q}_k Q_k^T(q_k) E_k^T \qquad (28)
\end{aligned}$$

It follows that the time-derivative of the individual Jacobians $[J_{cm,j}]_i(q)$ can be obtained, as in the equation 29, and will thus complete the definition of $\dot{T}_{cm}(q)$ which is not shown explicitly due to its length.

$$\begin{aligned}
\left[\dot{J}_{cm,j}\right]_i(q,\dot{q}) &= \sum_{k=i+1}^{j-1} \dot{q}_k W_{cm,j} Q_{j-1}^T(q_{j-1}) W_{j-1} \cdots Q_k^T(q_k) E_k^T W_k \cdots \\
&\quad Q_{i+1}^T(q_{i+1}) W_{i+1} [J_{i+1}]_i \qquad (29)
\end{aligned}$$

4 Derivation of Matrix Induced Norms

We could obtain the induced norm of $\left[\dot{J}_{cm,j}\right]_i(q,\dot{q})$ by the following sub-multiplicative law:

$$\begin{aligned}
\| \left[\dot{J}_{cm,j}\right]_i \| &\leq \sum_{k=i+1}^{j-1} |\dot{q}_k| \|W_{cm,j}\| \|Q_{j-1}^T(q_{j-1})\| \|W_{j-1}\| \cdots \\
&\quad \|Q_k^T(q_k)\| \|E_k^T\| \|W_k\| \cdots \\
&\quad \|Q_{i+1}^T(q_{i+1})\| \|W_{i+1}\| \| [J_{i+1}]_i \| \qquad (30)
\end{aligned}$$

However, it is less restrictive to pair the matrices together and observe the following induced 2-norms:

$$\begin{aligned}
W_i [J_i]_{i-1} &= \begin{bmatrix} [Q_i]_{i-1}^T & 0_{3\times3} \\ -[Q_i]_{i-1}^T [[\vec{a}_i]_{i-1}\times] & [Q_i]_{i-1}^T \end{bmatrix} \begin{bmatrix} \vec{e}_{i-1} \\ \vec{0} \end{bmatrix} \\
&= \begin{bmatrix} [Q_i]_{i-1}^T \vec{e}_{i-1} \\ -[Q_i]_{i-1}^T [[\vec{a}_i]_{i-1}\times] \vec{e}_{i-1} \end{bmatrix} \qquad (31) \\
\|W_i [J_i]_{i-1}\|_2 &= \sqrt{\| [Q_i]_{i-1}^T \vec{e}_{i-1}\|_2^2 + \| [Q_i]_{i-1}^T [\vec{a}_i]_{i-1} \times \vec{e}_{i-1}\|_2^2} \qquad (32) \\
&\leq \sqrt{1 + \| [\vec{a}_i]_{i-1} \times \vec{e}_{i-1}\|_2^2} \qquad (33)
\end{aligned}$$

and

$$\begin{aligned}
\|W_i Q_{i-1}^T(q_{i-1})\|_2 &= \|W_i\|_2 = \sqrt{1 + \|\vec{a}_i\|_2^2} \qquad (34) \\
\|W_i \dot{Q}_{i-1}^T(q_{i-1},\dot{q}_{i-1})\|_2 &= |\dot{q}_{i-1}| \|W_i Q_{i-1}^T(q_{i-1}) E_{i-1}^T\|_2 \qquad (35) \\
&\leq |\dot{q}_{i-1}|_{max} \sqrt{1 + \|\vec{a}_i\|_2^2} \qquad (36)
\end{aligned}$$

Consequently, we can formulate the induced 2-norm of the Jacobian matrices as follows:

$$\begin{aligned}
\| [J_{cm,j}]_i(q)\|_2 &\leq \|W_{cm,j} Q_{j-1}^T(q_{j-1})\|_2 \cdots \|W_{i+2} Q_{i+1}^T(q_{i+1})\|_2 \|W_{i+1} [J_{i+1}]_i\|_2 \\
&\leq \sqrt{1 + \|\vec{a}_{cm,j}\|_2^2} \cdots \\
&\quad \sqrt{1 + \|\vec{a}_{i+2}\|_2^2} \sqrt{1 + \| [\vec{a}_{i+1}]_i \times \vec{e}_i\|_2^2} \qquad (37)
\end{aligned}$$

and of their time-derivatives:

$$\begin{aligned}
\| \left[\dot{J}_{cm,j}\right]_i (q,\dot{q})\|_2 &\leq \sum_{k=i+1}^{j-1} \|W_{cm,j} Q_{j-1}^T(q_{j-1})\|_2 \cdots \\
&\quad \|W_{k+1}\dot{Q}_k^T(q_k)\|_2 \cdots \\
&\quad \|W_{i+2}Q_{i+1}^T(q_{i+1})\|_2 \|W_{i+1}\left[J_{i+1}\right]_i\|_2 \qquad (38)\\
&\leq \sum_{k=i+1}^{j-1} |\dot{q}_k|_{max}\sqrt{1+\|\vec{a}_{cm,j}\|_2^2}\cdots \\
&\quad \sqrt{1+\|\vec{a}_{i+2}\|_2^2}\sqrt{1+\|\left[\vec{a}_{i+1}\right]_i \times \vec{e}_i\|_2^2} \qquad (39)\\
&\leq \|\left[J_{cm,j}\right]_i(q)\|_{2,max} \sum_{k=i+1}^{j-1} |\dot{q}_k|_{max} \qquad (40)
\end{aligned}$$

In the process of finally bounding the spectral radius of $\dot{T}_{cm}^T(q)M_{cm}T_{cm}(q)$, it is useful first to observe the core matrix elements that will inevitably be found after expansion. Notably, the $W_{cm,i}^T M_{cm,i} W_{cm,i}$ matrices:

$$W_{cm,i}^T M_{cm,i} W_{cm,i} = W_{cm,i}^T \begin{bmatrix} I_i & 0_{3\times3} \\ 0_{3\times3} & m_i I_{3\times3} \end{bmatrix} W_{cm,i} =$$

$$\begin{bmatrix} \left[Q_{cm,i}\right]_{i-1} I_i \left[Q_{cm,i}\right]_{i-1}^T - m_i \left[\vec{a}_{cm,i}\right]_{i-1} \times \left[\vec{a}_{cm,i}\right]_{i-1} \times & m_i \left[\vec{a}_{cm,i}\right]_{i-1} \times \\ -m_i \left[\vec{a}_{cm,i}\right]_{i-1} \times & m_i I_{3\times3} \end{bmatrix} \qquad (41)$$

Where I_i and m_i are, respectively, the inertia tensor and mass of the i^th link. Furthermore, the spectral norm of these cores can also be obtained with the following result:

$$\begin{aligned}
\|W_{cm,i}^T M_{cm,i} W_{cm,i}\|_2 &= \max(\sqrt{(\|I_i\|_2 + m_i\|\vec{a}_{cm,i}\|_2^2)^2 + m_i^2\|\vec{a}_{cm,i}\|_2^2}, \\
&\quad m_i\sqrt{\|\vec{a}_{cm,i}\|_2^2+1}) \qquad (42)\\
&= \|Q_{i-1}(q_{i-1})W_{cm,i}^T M_{cm,i} W_{cm,i} Q_{i-1}^T(q_{i-1})\|_2 \qquad (43)
\end{aligned}$$

We now have all the tools to bring forth the expansion of the $\dot{T}_{cm}^T(q)M_{cm}T_{cm}(q)$ matrix:

$$\begin{aligned}
&\dot{T}_{cm}^T(q)M_{cm}T_{cm}(q) \\
&= \begin{bmatrix} \dot{J}_{1,0}^T & \cdots & \dot{J}_{n,0}^T \\ 0_{1\times6} & \ddots & \vdots \\ \vdots & \ddots & \dot{J}_{n,n-1}^T \end{bmatrix} \begin{bmatrix} M_1 & 0_{6\times6} & \cdots \\ 0_{6\times6} & \ddots & \ddots \\ \vdots & \ddots & M_n \end{bmatrix} \begin{bmatrix} J_{1,0} & 0_{6\times1} & \cdots \\ \vdots & \ddots & \ddots \\ J_{n,0} & \cdots & J_{n,n-1} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^n \dot{J}_{i,0}^T M_i J_{i,0} & \sum_{i=2}^n \dot{J}_{i,0}^T M_i J_{i,1} & \cdots & \dot{J}_{n,0}^T M_n J_{n,n-1} \\ \sum_{i=2}^n \dot{J}_{i,1}^T M_i J_{i,0} & \sum_{i=2}^n \dot{J}_{i,1}^T M_i J_{i,1} & \ddots & \dot{J}_{n,1}^T M_n J_{n,n-1} \\ \vdots & \ddots & \ddots & \vdots \\ \dot{J}_{n,n-1}^T M_n J_{n,0} & \dot{J}_{n,n-1}^T M_n J_{n,1} & \cdots & \dot{J}_{n,n-1}^T M_n J_{n,n-1} \end{bmatrix} \qquad (44)
\end{aligned}$$

where $J_{i,j}$ is short for $\left[J_{cm,i}\right]_j$ and similarly for $\dot{J}_{i,j}$. Now, by formulating the supremum of each element of $\dot{T}_{cm}^T(q)M_{cm}T_{cm}(q)$ in terms of their spectral norms, we can formulate the induced ∞-norm which will be used as upper-bound to the spectral radius.

$$\begin{aligned}
&\|\dot{T}_{cm}^T(q)M_{cm}T_{cm}(q)\|_\infty \leq \\
&\quad \max_{0\leq j\leq n-1} \sum_{k=0}^{n-1} \left(\sum_{i=j+1}^{n} \left(\|J_{i,j}\|_{2,max}\|M_i\|_2\|J_{i,k}\|_{2,max} \sum_{l=j+1}^{i-1} |\dot{q}_l|_{max} \right) \right) \qquad (45)
\end{aligned}$$

Now observing the expressions for the spectral norms of $J_{i,j}$ or $[J_{cm,i}]_j$ from equation 37, the following statement becomes self-evident:

$$\|J_{i,j}\|_{2,max} \geq \|J_{i,j+k}\|_{2,max} \quad \text{for } k > 0 \tag{46}$$

The above naturally comes from the composition of upper-bounds and the serial nature of the kinematic chain. The serial structure makes it so that any point on the manipulator is affected the strongest by the first joint because it is in general the furthest, and so on for the next joints. The generality of the latter statement is a result of the upper-bounds taken every step, and is thus not always true *per se*. One can picture a manipulator whose end-effector is folded back to near the base joint, making the base joint Jacobian to the end-effector relatively small. However, again, this is a result of upper-bounding, so in reality, the maximum motion that the base joint can induce to the end-effector, in its most favourable configuration, is greater then any other joint down-stream, in its respective most favourable configuration. Understood this way, it is clear that equation 46 holds. Furthermore, the relation extends to its time-derivative:

$$\begin{aligned} \|\dot{J}_{i,j}\|_{2,max} &= \|J_{i,j}\|_{2,max} \sum_{l=j+1}^{i-1} |\dot{q}_l|_{max} \\ &\geq \|J_{i,j+k}\|_{2,max} \sum_{l=j+k+1}^{i-1} |\dot{q}_l|_{max} = \|\dot{J}_{i,j+k}\|_{2,max} \quad \text{for } k > 0 \end{aligned} \tag{47}$$

Equation 46 greatly simplifies the calculation of the induced ∞-norm since it guarantees that the first row-sum will be the largest and thus we have:

$$\|\dot{T}_{cm}^T(q) M_{cm} T_{cm}(q)\|_\infty \leq \sum_{k=0}^{n-1} \left(\sum_{i=1}^{n} \left(\|J_{i,0}\|_{2,max} \|M_i\|_2 \|J_{i,k}\|_{2,max} \sum_{l=1}^{i-1} |\dot{q}_l|_{max} \right) \right) \tag{48}$$

$$\rho\left(\dot{T}_{cm}^T(q) M_{cm} T_{cm}(q)\right) \leq \|\dot{T}_{cm}^T(q) M_{cm} T_{cm}(q)\|_\infty \tag{49}$$

The above equation can be used along with equations 33, 34, 37 and 42 to calculate the upper-bound of the spectral radius of $\dot{T}_{cm}^T(q) M_{cm} T_{cm}(q)$, needed for the lower-bound of γ of equation 20. However, from the above, we can yet further expand the upper-bound by grossing the sum of maximum joint speeds and taking it out of the summations:

$$\rho\left(\dot{T}_{cm}^T(q) M_{cm} T_{cm}(q)\right) \leq \sum_{l=1}^{n-1} |\dot{q}_l|_{max} \sum_{k=0}^{n-1} \left(\sum_{i=1}^{n} (\|J_{i,0}\|_{2,max} \|M_i\|_2 \|J_{i,k}\|_{2,max}) \right) \tag{50}$$

$$\leq \sum_{l=1}^{n-1} |\dot{q}_l|_{max} \|T_{cm}^T(q) M_{cm} T_{cm}(q)\|_\infty \tag{51}$$

Making use of the fact that $T_{cm}^T(q) M_{cm} T_{cm}(q) = M(q)$ by definition and that $M(q)$ is a symmetric, positive-definite matrix, and thus, $\|M(q)\|_\infty = \|M(q)\|_2 = \rho(M(q)) = \lambda_{max}(M(q))$, we can formulate a fairly conservative but very simple lower-bound for the estimation gain γ:

$$\gamma \geq \frac{\lambda_{max}(M(q))}{\lambda_{min}(M(q))} \sum_{l=1}^{n-1} |\dot{q}_l|_{max} \tag{52}$$

Or, when considering motor inertias $G^2 I_m$, where G and I_m are, respectively, diagonal matrices of gear-ratios and motor inertia, a slightly modified derivation to the one presented above would lead to the following equation of motion and lower-bound on γ:

$$\tau_{ext} = (M(q) + G^2 I_m)\ddot{q} + C(q,\dot{q})\dot{q} - \tau_{ego}(q,\dot{q}) - \tau_{mot} \quad (53)$$

$$\gamma \geq \frac{\lambda_{max}(M(q))}{\min_i(G_i^2 I_{m,i})} \sum_{l=1}^{n-1} |\dot{q}_l|_{max} \quad (54)$$

This concludes the demonstration of this section and proves the asymptotic stability of the non-linear disturbance observer under either lower-bounds on estimation gain between equation 20, 52 or 54, depending on the application. It should be noted that in summary, the estimator requires neither acceleration measurements nor any matrix inversion, and a stable estimation gain can be obtained from manipulator parameters in a simple matrix formulation or even by inspection of the highest and lowest inertia possible for each joint, not even requiring an analytical expression of the generalized mass matrix. The latter constitutes a major improvement in design effort in comparison to the expressions presented by Nikoobin and Haghighi (2009) which not only required inversion of the mass matrix in the estimation calculations and an analytical expression of the mass matrix in the design process, but also significant trigonometric manipulations to arrive at the required form to calculate the lower-bound to the observer gain.

References

*References

Angeles, J. (2007). *Fundamentals of Robotics Mechanical Systems - Theory, Methods, and Algorithms.* Mechanical Engineering Series. Springer Science+Business Media, 3rd edition.

Chen, W., Balance, D., Gawthrop, P., and O'Reilly, J. (2000). *A nonlinear disturbance observer for robotic manipulators.* In *IEEE Transactions on Industrial Electronics*, volume 47, pages 932–938.

Korayem, M.H. and Haghighi, R. (2008). *Nonlinear disturnbance observer for robot manipulators in 3D space*, volume 5314 of *Lecture Notes in Computer Science*, pages 14–23. Springer Berlin / Heidelberg.

Nikoobin, A. and Haghighi, R. (2009). *Lyapunov-Based Nonlinear Disturbance Observer for Serial n-Link Robot Manipulators. Journal of Intelligent Robotic Systems*, 55:135–153.