# A Bond Graph Approach for Modelling Systems of Rigid Bodies in Spatial Motion

Børge Rokseth

Marine Technology
Submission date: June 2014
Supervisor: Eilif Pedersen, IMT
Co-supervisor: Asgeir Sørensen, IMT

Norwegian University of Science and Technology
Department of Marine Technology

NTNU – Trondheim
Norwegian University of
Science and Technology

# A Bond Graph Approach for Modelling Systems of Rigid Bodies in Spatial Motion

**Børge Rokseth**

|  |  |
|---|---|
| Submission date: | June 2014 |
| Supervisor: | Associate Professor Eilif Pedersen |
| Co-Supervisor: | Professor Asgeir J. Sørensen |

Norwegian University of Science and Technology
Department of Marine Technology

**MASTER THESIS IN MARINE CYBERNETICS**
**SPRING 2014**
**FOR**
**STUD.TECH.BØRGE ROKSETH**

**A BOND GRAPH APPROACH FOR MODELLING SYSTEMS OF RIGID BODIES IN SPATIAL MOTION**

**Problem description:**

*Dynamic modelling of systems such as marine surface vessels, underwater vehicles, robotic manipulators and more, have become increasingly important the last decades due to, among other things, the increased use of model based motion control. These systems can however be complicated to model, especially when several systems are interconnected, such an underwater vehicle with robotic manipulators. Furthermore, the dynamics of these systems are typically subject to various external loads such as waves, wind and actuator systems. It is as such important to develop a modelling approach for the dynamics of such systems which allows for both effective production of the models, and flexibility in interfacing other systems and external loads to the model.*

*In this thesis a bond graph frame work based on Lagrangian mechanics is developed for modelling systems of rigid bodies in spatial motion, such as robotic manipulators and underwater vehicles. Applications of the framework is presented trough two case studies. In the first case study, a simulator for a seven degrees of freedom robotic manipulator is developed, and in the second case study, an advance model of a remotely operated vehicle with a robotic manipulator is developed.*

***Scope of work***

1. Review literature on Lagrangian mechanics.
2. Review literature on bond graph modelling.
3. Develop a bond graph method for modelling systems of rigid bodies in spatial motion.
4. Apply the modelling framework in order to design a simulation model for the Titan 4 manipulator.
5. Design and interface various sub models to the Titan 4 model, most importantly, the hydraulic actuator system of the Titan 4 manipulator.

6. Design a controller for the Titan 4 servo valves such that the manipulator joint angles can track a reference signal.
7. Use the modelling framework in order to develop a dynamic model for a remotely operated vehicle with a manipulator.
8. Design and interface various sub models such as vehicle thrusters and hydrodynamic damping to the model.

The report shall be written in English and edited as a research report including literature survey, description of mathematical models, description of control algorithms, simulations results, discussion and conclusion including a proposal for further work. Source code developed shall be provided on a CD with code listing enclosed in appendix.

It is supposed that Department of Marine Technology, NTNU, can use the results freely in its research work by referring to the students work.

The thesis should be submitted in three copies within 10th June 2014

| | |
|---|---|
| **Supervisor:** | Associate Professor Eilif Pedersen |
| **Co-Supervisor:** | Professor Asgeir J. Sørensen |

# Preface

This master's thesis is submitted to the Norwegian University of Science and Technology (NTNU) in fulfilment of the requirements for the degree Master of Science in Marine Technology. The research presented aims to develop efficient manners in which to produce and implement dynamic models of mechanical systems of rigid bodies. Two papers are written in conjunction with the thesis, and these can be found in the appendices of this thesis.

The research have been conducted at the Department of Marine Technology, under the supervision of Associate Professor Eilif Pedersen, to whom I would like to extend a special thank for allot of good advice and guidance during the project. I have greatly appreciated the freedom he has given me, letting me explore ideas and various paths along which to conduct the research. In addition, I would like to thank my fellow student Stian Skjong for countless interesting discussions regarding the subject of this thesis. My co-supervisor Professor Asgeir J. Sørensen also deserve a thank for taking of his time to help me.

Finally I would like to thank my fiancée for being such a great moral support during the work.

Børge Rokseth
Trondheim, Thursday 5th June, 2014

# Abstract

The research presented in this thesis seeks to improve the current state and technique for dynamic modelling of systems of rigid bodies in spatial motion. The research have been focused around modelling of underwater vehicles and robotic manipulators. Dynamic modelling of such systems are of growing importance as the demand for precise motion control with an increasing degree of autonomy continue to grow.

This thesis presents a modelling framework which enables model developers to create a basic bond graph template for a system of rigid bodies in spatial motion. This template can be used as a base for further model development, on which the developer can utilize the modularity virtue of the bond graph by interfacing sub models of various systems. An effective method for developing and implementing such a basic template in bond graph software is presented. Then the virtues of this approach is demonstrated through two case studies. In the first case study a simulator for a seven degrees of freedom manipulator is created, onto which a high fidelity hydraulic actuator system is developed and connected seamlessly. Then a control system, taking input from a joystick is developed for the manipulator. The second case study presents a dynamic model of a remotely operated vehicle equipped with a robotic manipulator. First a basic template for this system is developed, according to the method presented in this thesis. The basic template is then extended by integrating an advanced thruster system and a motion control system to the vehicle. Hydrodynamic added mass and damping is also included to both the manipulator and the vehicle, and the manipulator is equipped with a control system. Finally, an interface to the external world through the manipulator end effector is developed. This external interface is used in order to create a simulation where the vehicle and manipulator system lifts an object, relocates it and places it down at the new location. Simulation results from this is provided.

# Contents

# List of Figures

Chapter 1

# Introduction

## 1.1 Background and Motivation

The work presented in this thesis seeks to improve the current state and technique for dynamic modelling of systems of interconnected bodies in spatial motion. In particular, a modelling framework that allows for effective development and implementation of such models is presented. The main research focus is modelling of underwater vehicles and robotic manipulators. Of particular interest is the interconnected dynamics of these systems when robotic manipulators are attached to an underwater vehicle. The motivation behind this is that manipulators mounted on vehicles in some cases may have a severe impact on the dynamic behaviour of the vehicle, or conversely, the vehicle may have a severe impact on the dynamics of the manipulator. As dynamic models play a vital role in, among other things, motion control of both manipulators and the vehicles, it is important to be able to capture the effects that the two systems have upon each others dynamical behaviour. Furthermore, such interconnected models can potentially be an important tool in the planning of marine operations by providing simulations of critical aspects of the operations and training personnel on simulators. Also note that even though the main focus of this research is underwater vehicles and manipulators, the modelling frame work developed is applicable to any other system that can be described as a system of rigid bodies in spatial motion. Two typical examples of such systems is shown in figure 1.1. Here both an underwater vehicle with manipulators, and a surface vessel with various lifting and handling equipment is shown, both of which can be described as systems of rigid bodies in spatial motion.

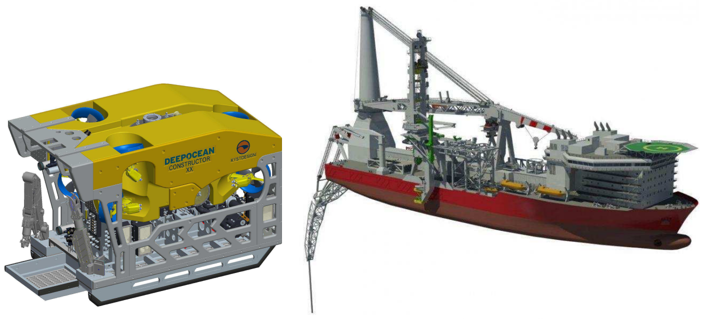


Figure 1.1: Examples of systems of bodies in spatial motion.

In both systems shown in the figure, the marine vehicle constitutes one body, and the manipulators or handling equipment constitutes several bodies, somehow linked to the vehicle and to each other. A vehicle, moving in six degrees of freedom can by it self represent a complicated system to model. This is also true for the equipment attached to the vehicle, especially when consisting of a great number of bodies linked together. There are several modelling methods able to handle such tasks. Among the most noteworthy are the Lagrangian method and the Newton-Euler method. The modelling framework presented in this thesis is based on the Lagrangian method and bond graph modelling.

The bond graph approach is a graphical modelling tool, based on identifying the energetic structure in systems. The greatest virtue of this approach, in the authors opinion, is the flexibility and modularity it offers. Given a bond graph of an interconnected system of bodies such as an underwater vehicle with manipulators, we can connect any type of system to the vehicle and the manipulators without having to consider the model equations directly. As the bond graph language handles interdisciplinary systems seamlessly and intuitively, it is very convenient in connecting various sub systems such as actuators, environmental loads, hydrodynamic effects, hydraulic systems, electrical systems and external object interfaces. However, there are some problems connected to using bond graphs for modelling of complicated systems of bodies in spatial motion. Firstly, the graphical representation may loose its inherent transparency in such cases, simply because of the size of the problem, and the number of connections that must be made. Secondly, causality issues, i.e., issues regarding the relation between cause and effect, tends to arise when systems of interconnected rigid bodies are modelled. Such causality issues are discussed briefly in section 2.2.

The motivation behind the work presented in this thesis is to combine the power of the bond graph language and Lagrangian mechanics in order to make a framework for modelling of systems of interconnected bodies in spatial motion, maintaining the properties of the bond graph methods. This work results in a method by which the modeller efficiently can build a bond graph template for a certain system, and then expand the model to include interdisciplinary sub systems such as actuators and hydrodynamic loads, as required.

## 1.2 Exposition of the Problem

The research focus of this thesis is to make use of the virtues of both the bond graph language and Lagrangian mechanics in order to develop a new framework for dynamic modelling of interconnected systems of rigid bodies, such as underwater vehicles and manipulators. The main idea is to utilize the IC-field element of the bond graph language in order to implement what we shall call *the basic dynamics* of the system, by which is meant the dynamic behaviour associated to the inertia forces, the Coriolis and centrifugal forces, and the restoring forces. A typical argument against such an implementation is that the basic dynamics then are hidden away in the IC-field element, and as such, that the modularity and flexibility virtue of the bond graph is lost. However, when considering this, modelling of basic dynamics, such as inertia forces and Coriolis and centrifugal forces, is a precise science. As such, if the basic dynamics are modelled correctly in the first place, there are no value in altering them. By this argument, we are not interested in any direct flexibility with regard to the basic dynamics. On the other hand, any dynamics that are typically non-conserving with regards to energy, such as friction forces, or other systems that interface the system of rigid bodies, are typically subject to model simplifications. As an example, an electric motor interfacing the system of rigid bodies, can in the simplest case be modelled as a torque input to the system. On the other hand, the voltage over the motor can be taken as the input to the system, such that the dynamics of the motor is described in more or less detail. As such, the flexibility and modularity of the bond graph can still be of great value if a good framework for interfacing the dynamics modelled in the IC-field is provided.

Thus, the main challenges of this research is first to develop a framework for modelling of the basic dynamics, described by the Lagrangian equations of motion, in the bond graph language. This framework should provide an effective manner in which to both develop the model, and implement it. Secondly, it is to establish manners in which to create interfaces for sub systems to interact with the system. Given such a framework, a model developer can create a basic template for a system, and use bond graphs in order to develop sub models of actuators, friction models, various hydrodynamical effects, and any other system which is desirable to interface to the basic template.

After presenting a solution to the above mentioned challenges, applications of the method presented here are demonstrated through two case studies. First, a simulator for the seven degrees of freedom Titan 4 robotic manipulator is developed. Some of the virtues of this modelling frame work is then demonstrated by developing and connecting a hydraulic actuator system to the manipulator. The case study also demonstrates how a joystick can be connected to the model through the bond graph software 20-sim, and how 20-sim can be used in order to develop a 3D visualization of the manipulator response to joystick input. The simulator is also equipped with an advanced control system. In the second case study, the interconnected dynamics of a remotely operated vehicle with a manipulator is studied. In this case study, the virtues of this modelling framework is demonstrated further by connecting a variety of subsystems to the vehicle and manipulator. Furthermore, a simulation scenario where the vehicle and manipulator lifts and relocates an external object is developed and presented.

## 1.3 Related Work

This section presents the most important references used in this thesis.

Dynamic modelling, and in particular Lagrangian mechanics, is of high relevance for the work presented in this thesis. The main references used regarding these subjects are Meirovitch (2003) and Ginsberg (1995). Both these references provides a thorough understanding of rigid body dynamics in general, discussing anything from fundamental theory to applications. Both references also discuss motion relative to moving reference frames, and kinematics in general. This subject is treated especially thoroughly in the latter reference. The former reference treat the concept of quasi-coordinates in Lagrangian mechanics. This discussion have been vital in modelling of the marine vehicle in the second case study.

The two main references used for bond graph theory are Karnopp et al. (2006) and Pedersen and Engja (2008). These references provides an understanding of the fundamental concepts of bond graphs, as well as a great deal of discussions on modelling of systems within various modelling disciplines, such as hydraulic, mechanical and electric systems. The latter reference also provides a short introduction to the bond graph software 20-sim used in this thesis. An other reference which have proven to be useful under the work done in this thesis is Karnopp (1969). This reference discusses the concept of power conserving transformations in physical systems in the bond graph context. General theory regarding bond graph fields, and especially the link between the IC-field and the Lagrangian mechanics, are provided both in

Karnopp et al. (2006) and Pedersen and Engja (2008). Pedersen (2012) present applications of this theory. In particular, a marine vehicle is modelled using bond graph and fields, something which have been highly relevant for this work.

The reference Sciavicco and Siciliano (2000) presents a wide range of topics of interest regarding dynamic modelling of robotic manipulators. The textbook presents methods for, among other things, development the kinematic relations of manipulators, developing dynamic models for manipulator, using both the Lagrangian and the Newton-Euler approach, as well as methods for manipulator control design. Berger et al. (1990) and Vaz et al. (2003), both treat the subject of modelling robotic manipulators using bond graphs, whereas the former, demonstrates the great advantage of using the bond graph by connecting an electrical motor to a manipulator joint. The latter of these references introduce modulated transformer fields in order to do the necessary transformations between manipulator links. The approaches presented in these references are however limited in that causality issues arises in the bond graphs, forcing the introduction of compliance to the manipulator links.

In the control design of hydraulically actuated manipulators, a variety of references have been used. The main reference for control design have been Khalil (2002), where both stability analysis of non-linear system, and various control system design techniques are discussed at at fundamental level. More specific references for control of hydraulic systems are Angue-Mintsa et al. (2011), where an adaptive feedback linearizing controller for controlling servo valves is proposed, and (Zeng and Sepehri, 2006), where an adaptive backstepping controller for hydraulic manipulators is proposed. Both these papers have been useful both in deciding upon which controller design method to apply for the controller design, and in researching possible model simplifications for the controller design.

Pedersen (2012) and Fossen (2011), both treat the modelling of marine vehicles. As already mentioned, the former of these references implement the Lagrangian equations for a marine vehicle in bond graph, using an IC-field. The latter reference treats both modelling and control of marine vehicles in detail. Both these references are used in order to develop the second case study, where a remotely operated vehicle and a manipulator is modelled. Furthermore, Fossen (2011) and Faltinsen (1993) have been useful in connecting various hydrodynamic effects such as hydrodynamic damping and hydrodynamic added mass. The hydrodynamic damping of the manipulator is based on a method proposed in McLain and Rock (1998). In this case study, the vehicle is equipped with a thruster system. This system is based on a model proposed in Healey et al. (1995), though in the work presented here, this model is implemented in bond graph. Important references regarding the interconnected dynamics of the vehicle and the manipulator are Soylu et al. (2010) and Kim et al. (2003). Although the focus of these papers are control of the interconnected systems, both address the issue of modelling the systems.

## 1.4 Structure of the Thesis

This section presents the structure of this thesis. This is done by stating briefly the main content of each of the following chapters in the report. The purpose of the appended are papers are also stated.

**Chapter 2** provides brief introductions to Lagrangian mechanics, bond graph modelling, and kinematic analysis systems. Applications of the theory presented is shown through an example which we follow through the whole chapter.

**Chapter 3** presents the basic idea which is the foundation for the work of this thesis, namely a bond graph approach based on Lagrangian mechanics, for modelling the dynamics of systems of rigid bodies in spatial motion. A method for effective implementation of this in the bond graph software 20-sim is proposed. The example from chapter 2 is resumed in this chapter.

**Chapter 4** gives examples on applications of the theory presented in chapter 3 through two case studies. In the first case study, a simulation model of a robotic manipulator with hydraulic actuators is developed, and in the second case study, a simulation model of a remotely operated vehicle with a robotic manipulator is developed.

**Chapter 5** concludes the work and suggests further research on the subject.

In **Appendix A.1**, the paper *Bond Graph Modelling of Marine Vehicle with Manipulator Equipment* is attached. This paper presents the theory behind case study two, i.e., the specifics on how the method presented in this thesis can be used in order to model marine vehicles with equipment such as manipulators.

In **Appendix A.2**, the paper *Backstepping Controller for Manipulator with Hydraulic Actuators* is attached. Here a controller design for a hydraulic manipulator, based on the backstepping technique is presented. The necessity for doing research on this field arose during the development of the simulator in case study two. This paper also investigates the goodness of typical model simplifications used in the literature for hydraulic system control design.

The remaining appendices contains relevant code and tables of parameters for the simulations performed in the case studies.

# Chapter 2

# Background Material

Much of the work presented in this thesis is based on Lagrangian mechanics, bond graph modelling and kinematic analysis of systems of rigid bodies. This chapter provides introductions to these subjects, starting with Lagrangian mechanics.

## 2.1 Lagrangian Mechanics

In this section, Lagrangian equations of motion for a system of rigid bodies in spatial motion are presented. These are derived from expressions for the kinetic and potential energy of the system. The potential energy can be expressed in terms of a set of generalized coordinates, $\boldsymbol{q}$, while the kinetic energy can be expressed in terms of the generalized coordinates, and their time derivatives, $\dot{\boldsymbol{q}}$. In the standard Lagrangian formulation, as opposed to the quasi-coordinate formulation, it is in fact required that the energy functions of the system are expressed only in terms of the general coordinates and their time rates. We therefore start the introduction to Lagrangian mechanics with the generalized coordinates and a short discussion on quasi-coordinates, before proceeding to find expressions for the potential and kinetic energy of a system of rigid bodies in spatial motion. Next, the concept of generalized forces is addressed, before finally, the Lagrangian equations of motion are presented, both for the standard formulation and the quasi-coordinate formulation. For a more comprehensive discussion on the topic, the reader is advised to consult Ginsberg (1995) and Meirovitch (2003).

### 2.1.1 Generalized Coordinates

The generalized coordinates of a system are a set of geometric parameters that uniquely defines the position and orientation of the system, relative to some frame of reference. In order to uniquely describe the position and orientation of a system, one need at least a number of generalized coordinates equal to the number of degrees of freedom for the system. If the number of generalized coordinates, $n_{gc}$, is greater than the number of degrees of freedoms, $n_{dof}$, a set of $n_{ce}$ constraint equations need to be defined such that $n_{gc} - n_{dof} = n_{ce}$. The constraint equations are relations between the generalized coordinates, placing restrictions on the motion of the system.

A set of generalized coordinates are not a unique set, meaning that for a particular system, there are generally several possible manners in which to define the set of generalized coordinates. As such it is advisable to consider the choice of generalized coordinates well. Consider as an example the simple case of a particle moving in a plane. Then the Cartesian coordinates $\boldsymbol{q} = [x, y]^T$ are a possible set of generalized coordinates. Other equally good choices are the polar coordinates $\boldsymbol{q} = [r, \theta]^T$, or mixtures between the polar and Cartesian coordinates, e.g., $\boldsymbol{q} = [x, \theta]^T$, where $r$ is the length of a line from the origin of the reference frame to the particle, and $\theta$ is the angle between the line and one of the principal axis of the plane, i.e., the reference frame.

#### Quasi-coordinates

When using the standard Lagrangian method to produce the equations of motion, the equations will be functions of the generalized coordinates and their time rates. For some systems however, it might be of interest to produce equations of motion that depend on the generalized coordinates and linear combinations of their time rates. A good example of this might be the orientation of a body in free spatial motion. In this case the angular velocity of the body cannot be integrated directly to obtain a set of meaningful generalized coordinates (Meirovitch, 2003). In such cases, it is useful to define a set of quasi-coordinates, related to the generalized coordinates as(Meirovitch, 2003)

$$\boldsymbol{\omega} = \boldsymbol{\alpha}^T \dot{\boldsymbol{q}} \tag{2.1}$$

where $\boldsymbol{\omega}$ are the quasi-coordinates, and $\boldsymbol{\alpha}^T$ is an invertible matrix. In the case of the orientation of the body in free spatial motion, the quasi-coordinates can then represent the angular velocity of the body. Doing so makes it possible to obtain set of equations of motion dependent on the angular velocity and a set of meaningful generalized coordinates. The necessary expressions, and the equations of motion in the case of quasi-coordinates are presented in parallel to the standard formulations in the following sections.

### 2.1.2 Potential and Kinetic Energy of a System of Rigid Bodies in Spatial Motion

We now proceed to find general expressions for the potential and kinetic energy for a system of $m$ rigid bodies in spatial motion, within a gravity field.

In order to find an expression for the potential energy, we start by defining the position of the center of gravity of body $i$, relative to the origin of an inertial reference frame as $\boldsymbol{r}_{cg_i}$. Note that this vector always can be expressed as a function of the generalized coordinates. With the mass $m_i$ of the body, and the acceleration of gravity $\boldsymbol{g}$, the potential energy can be found as

$$V_i(\boldsymbol{q}) = -m_i \boldsymbol{g}^T \boldsymbol{r}_{cg_i}(\boldsymbol{q}) + f_{ki}(\boldsymbol{q}) \tag{2.2}$$

where the first term accounts for the potential energy due to the gravity field, and the second term accounts for all other manners in which the body can store potential energy, such as by tightening springs. The total potential energy, $V(\boldsymbol{q})$, of the system is the sum of the contribution from each of the $m$ bodies, i.e.,

$$V(\boldsymbol{q}) = \sum_{i=1}^{m} V_i(\boldsymbol{q}) \tag{2.3}$$

For body $i$, the kinetic energy is a function of the linear velocity of the center of gravity $\boldsymbol{v}_{cg_i}$ of the body, and the angular velocity $\boldsymbol{\omega}_i$ of the body. Both these velocities can be expressed as functions of the set of $n$ generalized coordinates and their time derivatives. In fact, we propose the following.

**Proposition 2.1.** *The velocity $\boldsymbol{\nu}$ at any point on a system of rigid bodies, described by $n$ generalized coordinates $\boldsymbol{q}$, can be expressed in the form $\boldsymbol{\nu} = \boldsymbol{\mathcal{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}$ where the dimensions of $\boldsymbol{\nu}$ is $k \times 1$, and $\boldsymbol{\mathcal{J}}(\boldsymbol{q})$ is a $k \times n$ matrix.*

*Proof.* By the definition of generalized coordinates, the position and orientation $\boldsymbol{r}$ at any point on the system can be expressed in the form $\boldsymbol{r} = \boldsymbol{f}(\boldsymbol{q})$. It then follows that the velocity

$$\begin{aligned} \dot{\boldsymbol{r}} &= \frac{d}{dt}\left(\boldsymbol{f}(\boldsymbol{q})\right) = \frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} \\ &\triangleq \boldsymbol{\mathcal{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} \end{aligned} \tag{2.4}$$

□

Using the linear velocity, $\boldsymbol{v}_{cg_i}$, of the center of gravity of the $i$-th body, and the angular velocity, $\boldsymbol{\omega}_i$, of the $i$-th body, the kinetic energy of body $i$ can be expressed as

$$T_i(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2} m_i \left(\boldsymbol{v}_{cg_i}\right)^T \boldsymbol{v}_{cg_i} + \frac{1}{2}\boldsymbol{\omega}_i^T \boldsymbol{I}_{bi} \boldsymbol{\omega}_i \tag{2.5}$$

where $\boldsymbol{I}_{bi}$ is the inertia tensor of the body. The total kinetic energy $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$ of the system is the sum of all contributions from individual bodies, such that

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_{i=1}^{m} T_i(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.6}$$

### Kinetic energy in terms of quasi-coordinates

In the case where we want to end up with equations of motion in terms of quasi-coordinates as opposed to the time rates of the generalized coordinates, the kinetic energy must be expressed in terms of the generalized coordinates and the quasi-coordinates. From (2.1), we find that the time rates of the generalized coordinates can be expressed as

$$\dot{\boldsymbol{q}} = \boldsymbol{\beta}\boldsymbol{\omega} \tag{2.7}$$

where $\boldsymbol{\beta} = (\boldsymbol{\alpha}^T)^{-1}$. Substituting this into the expression for the kinetic energy we get the form

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = T(\boldsymbol{q}, \boldsymbol{\beta}\boldsymbol{\omega}) = \bar{T}(\boldsymbol{q}, \boldsymbol{\omega}) \tag{2.8}$$

This expression will be specified further in chapter 3. Note that the potential energy is not a function of the time rate of the generalized coordinates, and is therefore not affected by the introduction of quasi-coordinates.

## 2.1.3 Generalized Forces

With expressions for the kinetic and potential energy of the system, we can keep track of the internal energy flow of system. We can however not keep track of the energy entering or exiting the system through non-conservative forces by monitoring the mechanical energy alone. In order to account for these non-conservative forces, the generalized forces are introduces. Consider the non-conservative force $\boldsymbol{F}_j$ acting on a rigid body. Let this force result in the virtual displacement $\delta\boldsymbol{r}_j$. The virtual work done by the force can then be expressed as (Ginsberg, 1995)

$$\delta W_j = \boldsymbol{F}_j^T \delta\boldsymbol{r}_j \tag{2.9}$$

The virtual displacement due to the $j$-th non-conservative force can be expressed in terms of the virtual displacement of the $n$ generalized coordinates as (Ginsberg, 1995)

$$\delta\boldsymbol{r}_j = \sum_{i=1}^{n} \frac{\partial \boldsymbol{r}_j}{\partial q_i} \delta q_i \tag{2.10}$$

We can also rewrite the virtual work done by the force $\boldsymbol{F}_j$ in terms of the generalized forces, $\boldsymbol{\tau}_j$, and the virtual displacement of the generalized coordinates as

$$\delta W_j = \sum_{i=1}^{n} \tau_{ji} \delta q_i \tag{2.11}$$

By substituting $\delta \boldsymbol{r}_j$ for (2.10) in (2.9), and combining this with (2.11), we get

$$\boldsymbol{F}_j^T \sum_{i=1}^{n} \frac{\partial \boldsymbol{r}_j}{\partial q_i} \delta q_i = \sum_{i=1}^{n} \tau_{ji} \delta q_i \tag{2.12}$$

We can then compare the $i$-th term on each side of the equation to get the $i$-th generalized force due to the non-conservative force $\boldsymbol{F}_j$ as

$$\tau_{ij} = \boldsymbol{F}_j^T \frac{\partial \boldsymbol{r}}{\partial q_i} \tag{2.13}$$

The total generalized force associated to the $i$-th generalized coordinate is then the sum of $\tau_{ij}$ over all $j$.

### 2.1.4 Equations of Motion

Lagrangian equations of motion can now be developed, using the theory presented in the preceding sections. For a system described by $n$ generalized coordinates, these equations appear as a set of $n$ second order differential equations.

Given expressions for the potential and kinetic energy of a system, $V(\boldsymbol{q})$, $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$, and a set of generalized forces, $\boldsymbol{\tau}$, the Lagrangian equations of motion are

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}}\right) - \left(\frac{\partial T}{\partial \boldsymbol{q}} - \frac{\partial V}{\partial \boldsymbol{q}}\right) = \boldsymbol{\tau} \tag{2.14}$$

Throughout this thesis, a scalar differentiated with respect to a column vector, results in a column vector, such that

$$\frac{\partial T}{\partial \dot{\boldsymbol{q}}} = \begin{bmatrix} \frac{\partial T}{\partial \dot{q}_1} \\ \vdots \\ \frac{\partial T}{\partial \dot{q}_n} \end{bmatrix}, \quad \frac{\partial T}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial T}{\partial q_1} \\ \vdots \\ \frac{\partial T}{\partial q_n} \end{bmatrix}, \quad \frac{\partial V}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial V}{\partial q_1} \\ \vdots \\ \frac{\partial V}{\partial q_n} \end{bmatrix} \tag{2.15}$$

As an example, we shall develop equations of motion for a simple system with two degrees of freedom.

**Example 2.2.** In this example we derive equations of motion using the Lagrangian method, for the system shown in figure 2.1, where an inverted pendulum is fixed to a moving wagon. The wagon with mass $m_1$ is connected to a stationary wall through a spring with linear stiffness coefficient $k_1$, and a damper with a linear damping coefficient $d_1$. It can move along a plane surface on frictionless wheels, and is in its equilibrium position, i.e., no tension in the spring, when the wagon center of gravity is a distance $L_0$ from the wall in the horizontal direction. The pendulum is attached to the wagon through a pin joint on top of the wagon at the height $h$, and a rotational spring with linear stiffness coefficient $k_2$ seeks to

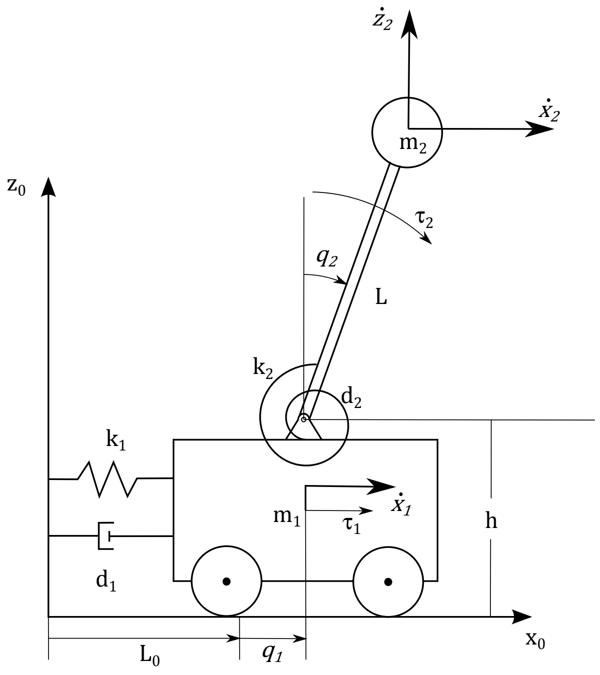


Figure 2.1: Inverted pendulum on moving wagon.

keep the pendulum pointing straight upwards. The joint have a linear friction coefficient $d_2$. The mass $m_2$ of the pendulum is concentrated at the length $L$ from the joint.

The system have two degrees of freedom. A reasonable choice for the two generalized coordinates $q_1$, and $q_2$ are the horizontal displacement of the wagon center of gravity, and the angular displacement of the pendulum as illustrated in the figure. The system is excited by the force $\tau_1$, and the torque $\tau_2$ as seen from the figure. Notice that $\tau_1$ and $\tau_2$ acts directly on the generalized coordinates, such that they are in fact already expressed as generalized forces. Notice also that the linear damping forces can be expressed as $f_{d1}(q_1) = d_1\dot{q}_1$ and $f_{d2} = d_2\dot{q}_2$. These forces are non-conservative, meaning that there are neither storing of kinetic nor potential energy associated to them, and as such, they must be included in the generalized forces. Also these forces act upon the system in the same manner as the generalized forces, only in the opposite direction, such that they can be included directly as generalized forces.

In order to find the potential energy of the system we find the position of the two centres of gravity and the displacements of the springs. The positions are given by

$$\boldsymbol{r}_{cg1/0} = \begin{bmatrix} L_0 + q_1 \\ h/2 \end{bmatrix}, \quad \boldsymbol{r}_{cg2/0} = \begin{bmatrix} L_0 + q_1 + L\sin(q_2) \\ h + L\cos(q_2) \end{bmatrix} \tag{2.16}$$

where the subscript indicate that we consider the positions of the center of gravity for the first and second body, i.e., the wagon and the pendulum respectively, relative to the origin of the reference frame $x_0 z_0$. The displacements of the two springs out of their equilibrium positions are $q_1$ for the first spring, and $q_2$ for the second. Thus we find the potential energy due to the springs as

$$f_{k1}(q_1) = \frac{1}{2}k_1 q_1^2, \quad f_{k2}(q_2) = \frac{1}{2}k_2 q_2^2 \tag{2.17}$$

Inserting (2.16) and (2.17) into (2.2), and defining the acceleration of gravity as $\boldsymbol{g} = [0, -g]^T$, pointing straight downwards along the $z_0$ axis, we find

$$\begin{aligned} V_1(\boldsymbol{q}) &= -m_1 \boldsymbol{g}^T \boldsymbol{r}_{cg1/0} + f_{k1}(q_1) = \frac{h}{2}gm_1 + \frac{1}{2}k_1 q_1^2 \\ V_2(\boldsymbol{q}) &= -m_2 \boldsymbol{g}^T \boldsymbol{r}_{cg2/0} + f_{k2}(q_2) = m_2 g(h + L\cos(q_2)) + \frac{1}{2}k_2 q_2^2 \end{aligned} \tag{2.18}$$

The linear velocities of the centres of gravity for the two bodies are denoted $\boldsymbol{v}_{cg1/0}$ and $\boldsymbol{v}_{cg2/0}$. These are expressed in terms of the generalized coordinates as

$$\boldsymbol{v}_{cg1/0} = \begin{bmatrix} \dot{x}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ 0 \end{bmatrix}, \quad \boldsymbol{v}_{cg2/0} = \begin{bmatrix} \dot{x}_2 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 + L\cos(q_2)\dot{q}_2 \\ -L\sin(q_2)\dot{q}_2 \end{bmatrix} \tag{2.19}$$

where the meaning of $\dot{x}_1$, $\dot{x}_2$, and $\dot{z}_2$ can be seen fro figure 2.1. Using (2.5), we see that the kinetic energy of the bodies are

$$\begin{aligned} T_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \frac{1}{2}m_1 \dot{q}_1^2 \\ T_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \frac{1}{2}m_2 \left(L^2 \dot{q}_2^2 + 2L\cos(q_2)\dot{q}_1\dot{q}_2 + \dot{q}_1^2\right) \end{aligned} \tag{2.20}$$

Summing up the two contributions to the potential energy and the two contributions to the kinetic energy, we find the total energy expressions as

$$\begin{aligned} V(\boldsymbol{q}) &= \frac{h}{2}gm_1 + \frac{1}{2}k_1 q_1^2 + m_2 g(h + L\cos(q_2)) + \frac{1}{2}k_2 q_2^2 \\ T(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \frac{1}{2}m_1 \dot{q}_1^2 + \frac{1}{2}m_2 \left(L^2 \dot{q}_2^2 + 2L\cos(q_2)\dot{q}_1\dot{q}_2 + \dot{q}_1^2\right) \end{aligned} \tag{2.21}$$

In order to find the equations of motion for the system, we find each of the terms stated in (2.14). Starting with the first, we find

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}}\right) &= \frac{d}{dt}\left(\begin{bmatrix} m_1\dot{q}_1 + m_2(\dot{q}_1 + L\cos(q_2)\dot{q}_2) \\ m_2 L(L\dot{q}_2 + \cos(q_2)\dot{q}_1 \end{bmatrix}\right) \\ &= \begin{bmatrix} (m_1 + m_2)\ddot{q}_1 + Lm_2\cos(q_2)\ddot{q}_2 - Lm_2\sin(q_2)\dot{q}_2^2 \\ m_2 L\cos(q_2)\ddot{q}_1 + m_2 L^2 \ddot{q}_2 - m_2 L\sin(q_2)\dot{q}_1\dot{q}_2 \end{bmatrix} \end{aligned} \tag{2.22}$$

Next we find the kinetic energy differentiated with respect to the generalized coordinates as

$$\frac{\partial T}{\partial \boldsymbol{q}} = \begin{bmatrix} 0 \\ -Lm_2 \sin(q_2)\dot{q}_1\dot{q}_2 \end{bmatrix} \tag{2.23}$$

and the potential energy differentiated with respect to the generalized coordinates as

$$\frac{\partial V}{\partial \boldsymbol{q}} = \begin{bmatrix} k_1 q_1 \\ -Lm_2 g \sin(q_2) + k_2 q_2 \end{bmatrix} \tag{2.24}$$

Substituting (2.22), (2.23), and (2.24) into (2.14), and including the generalized forces and the damping forces, we find the equations of motion as the set of $n = 2$ second order differential equations

$$\begin{bmatrix} (m_1 + m_2)\ddot{q}_1 + Lm_2 \cos(q_2)\ddot{q}_2 - Lm_2 \sin(q_2)\dot{q}_2^2 + k_1 q_1 \\ Lm_2 \cos(q_2)\ddot{q}_1 + L^2 m_2 \ddot{q}_2 - Lm_2 g \sin(q_2) + k_2 q_2 \end{bmatrix} = \begin{bmatrix} \tau_1 - d_1\dot{q}_1 \\ \tau_2 - d_2\dot{q}_2 \end{bmatrix} \tag{2.25}$$

which concludes the example.

## Equations of motion in quasi-coordinates

When using quasi-coordinates, the equations of motion takes on a slightly different form than the one presented in (2.14) for generalized coordinates. In Meirovitch (2003), the equations of motion for quasi-coordinates are derived without considering the potential energy of the system. In this case, the resulting equations of motion are given as

$$\boldsymbol{\alpha}\frac{d}{dt}\left(\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}}\right) + \boldsymbol{\gamma}\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} - \frac{\partial \bar{T}}{\partial \boldsymbol{q}} = \boldsymbol{\tau} \tag{2.26}$$

where $\bar{T}$ was presented in (2.8), and the $n \times n$ matrix $\boldsymbol{\gamma}$ is given as

$$\boldsymbol{\gamma} = \begin{bmatrix} \xi_{11} & \cdots & \xi_{1n} \\ \vdots & \ddots & \vdots \\ \xi_{n1} & \cdots & \xi_{nn} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\omega}^T \boldsymbol{\beta}^T \frac{\partial \boldsymbol{\alpha}}{\partial q_1} \\ \vdots \\ \boldsymbol{\omega}^T \boldsymbol{\beta}^T \frac{\partial \boldsymbol{\alpha}}{\partial q_n} \end{bmatrix} \tag{2.27}$$

and

$$\xi_{ij} = \boldsymbol{\omega}^T \boldsymbol{\beta}^T \frac{\partial \alpha_{ij}}{\partial \boldsymbol{q}} \tag{2.28}$$

Note that $\partial\boldsymbol{\alpha}/\partial q_i$ is a square matrix, in which each element $\alpha_{ij}$ are differentiated with respect to $q_i$, whereas $\partial \alpha_{ij}/\partial \boldsymbol{q}$ is a column vector in which the element $\alpha_{ij}$ is differentiated with respect to each of the generalized coordinates.

As the potential energy is not dependent on the rate of the generalized coordinates, the dynamics resulting from the potential energy, i.e., the term $\partial V/\partial \boldsymbol{q}$, can be added to the system as

$$\boldsymbol{\alpha}\frac{d}{dt}\left(\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}}\right)+\boldsymbol{\gamma}\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}}-\frac{\partial \bar{T}}{\partial \boldsymbol{q}}+\frac{\partial V}{\partial \boldsymbol{q}}=\boldsymbol{\tau} \tag{2.29}$$

Using that $\boldsymbol{\alpha}^{-1}=\boldsymbol{\beta}^T$, (2.29) can be rewritten to

$$\frac{d}{dt}\left(\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}}\right)+\boldsymbol{\beta}^T\boldsymbol{\gamma}\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}}-\boldsymbol{\beta}^T\frac{\partial \bar{T}}{\partial \boldsymbol{q}}+\boldsymbol{\beta}^T\frac{\partial V}{\partial \boldsymbol{q}}=\boldsymbol{\beta}^T\boldsymbol{\tau} \tag{2.30}$$

From this equation, we see that the dependency on $\dot{\boldsymbol{q}}$ is replaced by a dependency on the quasi-coordinates $\boldsymbol{\omega}$, which was the purpose of using quasi-coordinates.

## 2.2 Bond Graph Modelling

The purpose of this section is to introduce some of the basic concepts of bond graph modelling. For a comprehensive study of this field, the reader is advised to consult Karnopp et al. (2006) or Pedersen and Engja (2008).

### 2.2.1 The Basics of Bond Graph Modelling

The bond graph is a graphical method for dynamic modelling, based on identifying the energetic structure in the system. The dynamic behaviour of systems are governed by the flow, storage and interchange of energy between various sub systems, or elements of the system. As such, a dynamic system can be divided into a set of basic elements, either storing, dissipating, transforming or supplying energy. When making a dynamic model of a system in the bond graph language, these basic elements, and the interaction between them are mapped graphically. The bond graph language provides such basic elements for storing kinetic energy and potential energy, for dissipating energy, and for transforming energy. Each such element have one or more power ports through which energy can flow. Power is transmitted between such ports over power bonds.

The power $P(t)$ transmitted over a power bond is defined by two power variables, namely the effort $e(t)$ and the flow $f(t)$, related as

$$P(t)=e(t)f(t) \tag{2.31}$$

As an example, for a mechanical system an effort can represent a force or a torque, while the flow can represent a linear or an angular velocity. For a hydraulic system, an effort can represent a hydraulic pressure and a flow can be the corresponding volumetric displacement rate. Figure 2.2 illustrates how power is transmitted between

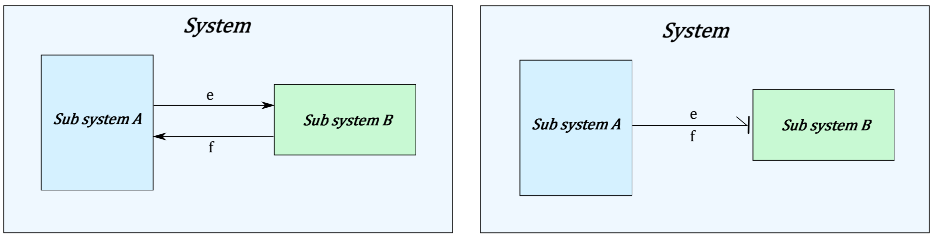


Figure 2.2: Illustration of power bond transmitting power between two sub systems.

two sub systems in terms of effort and flow. The direction of the half arrow of the power bond defines the positive power direction. In figure 2.2, the power is positive when energy flow from sub system A to sub system B, and negative otherwise. The casual stroke, which can be seen as the vertical line between the power bond and sub system B, defines to which of the sub systems the effort is input, and to which system the flow is input. The effort is always input on the side where the casual stroke is found, as can be seen by comparing the left and right system in the figure.

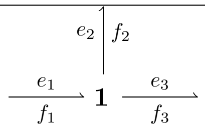
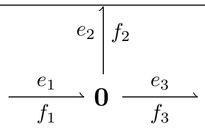
Two other important variables in the bond graph language, besides the effort and the flow, are the momentum $p(t)$ and the displacement $q(t)$. These are related to the power as

$$
\begin{aligned}
p(t) &= \int_{\tau=0}^{t} e(t)d\tau + p_0 \\
q(t) &= \int_{\tau=0}^{t} f(t)d\tau + q_0
\end{aligned}
\tag{2.32}
$$

where $p_0$ and $q_0$ are the initial momentum and displacement. One of the advantages of using bond graph is that the effort, the flow, the momentum and the displacement, are the only four variables necessary in order to describe the dynamics of a system. Furthermore, these variables can be identified in a wide variety of dynamic systems of different disciplines, such as mechanical systems, hydraulic systems, electrical systems, and so forth. This, together with the graphical interface, makes it convenient to combine different disciplines into one system.

Previously, a set of basic elements which could either store, dissipate, transform or supply energy were mentioned. In addition to these, there are two types of junction elements defined within the formalism of bond graphs. These elements are used in order to route power, and does as such nothing but transmit it. One of these elements is called a common effort junction, or a 0-junction, and is characterized by the property that all adjoining bonds have a common effort. The other junction element is the common flow junction, or the 1-junction, which is characterized by the property that all adjoining bonds have a common flow. Table 2.1 summarize the most common bond graph elements used in this thesis, and state the elements constitutive relations, i.e., the relations between the effort and flow.

Table 2.1: Some common basic bond graph elements.

| **Symbol** | **Relation** |
|---|---|
| $S_e \longrightarrow\!\mid$ | $e = e(t)$ |
| $S_f \vdash\!\!\longrightarrow$ | $f = f(t)$ |
| $\vdash\!\!\longrightarrow \mathrm{R}$ <br> $\longrightarrow\!\mid \mathrm{R}$ | $e = \Phi_R(f)$ <br> $f = \Phi_R^{-1}(e)$ |
| $\vdash\!\!\longrightarrow \mathrm{C}$ <br> $\longrightarrow\!\mid \mathrm{C}$ | $e = \Phi_C^{-1}(\int_0^t f dt)$ <br> $f = \frac{d}{dt}[\Phi_C(e)]$ |
| $\longrightarrow\!\mid \mathrm{I}$ <br> $\vdash\!\!\longrightarrow \mathrm{I}$ | $f = \Phi_I^{-1}(\int_0^t e dt)$ <br> $e = \frac{d}{dt}[\Phi_I(f)]$ |
| $\vdash\!\!\xrightarrow{\;1\;} TF \vdash\!\!\xrightarrow{\;2\;}$ <br> $\xrightarrow{\;1\;}\!\mid TF \xrightarrow{\;2\;}\!\mid$ | $e_1 = me_2$ <br> $f_2 = mf_1$ <br> $e_2 = \frac{1}{m}e_1$ <br> $f_1 = \frac{1}{m}f_2$ |
|  | $e_1 - e_2 - e_3 = 0$ <br> $f_1 = f_2 = f_3$ |
|  | $e_1 = e_2 = e_3$ <br> $f_1 - f_2 - f_3 = 0$ |

From the constitutive relations given in this table, it can be seen that the effort source, i.e., the S$_e$-element, is an element which represents an energy source, inputting an effort to the system. A physical phenomena which could be represented by an effort source is a force input to a mechanical system. Similarly, the flow source, or the S$_f$-element, represents an energy source inputting a flow to the system. For a mechanical system, this represents a forced velocity. Both the S$_e$-element and the S$_f$-element can be extended to take in a signal input. It is then customary to denote them as MS$_e$

and MS$_f$-elements respectively. The dissipation element, or the R-element, removes energy from the system, and can be used in order to model e.g. friction in a mechanical system. The C-element stores energy as potential energy, whereas the I-element stores kinetic energy. Finally, the transformer element, or the TF-element, transforms energy from one form to another. Notice from the constitutive relation of the TF-element that the power $P_1$ into the element equals the power $P_2$ out of the element such that $P_1 = e_1 f_1 = e_2 f_2 = P_2$. The transformer can also be extended to accept a signal as an input, making the element a modulated transformer, or a MTF-element.

Causality, or the relation between cause and effect, is an important aspect in bond graph modelling. Recall that the causal stroke of a power bond defines to which basic element the effort is input, and to which element the flow is input. This explicit assignment of causality associated to the bond graph, may provide the modeller with a deeper understanding of the dynamics of the system. As an example, the casual stroke on the power bond of the S$_e$-element in table 2.1, tells us that the effort source sets the effort to the sub system to which it is connected. The S$_f$-element on the other hand, sets a flow to the sub system to which it is connected. For the R-element, the causality is arbitrary, meaning that either a flow or an effort can be input to this element. The C-element and the I-element can also take either the effort or the flow as inputs. However, the causality is not arbitrary in these cases. From table 2.1, we see that for a C-element with flow as input, the constitutive relation is

$$e = \Phi_C^{-1}\left(\int_0^t f dt\right) = \Phi_C^{-1}(q) \tag{2.33}$$

Due to the time integration of the flow variable, the causality resulting from flow input to the C-element is called integral causality. An I-element with the effort as input have the constitutive relation

$$f = \Phi_I^{-1}(\int_0^t edt) \tag{2.34}$$

Due to the time integral of the effort variable, the causality which yields the effort as input is named integral causality. When a flow is imposed on an I-element, the effort is the derivative of a function of the flow, and this causality is therefore called differential causality. Similarly, if an effort is imposed on a C-element, the flow is the derivative of a function of the effort. Due to this, the causality which yields a effort as input to a C-element is called differential causality. Due to both physical and mathematical reasons, both the C-element and the I-element prefer integral causality, and causality issues arise when differential causality is forced upon either the C-element or the I-element. For a more detailed discussion on this, the reader is advised to see Pedersen and Engja (2008). The causality issues mentioned in the introduction of this thesis, related to bond graph modelling of systems of rigid bodies in spatial motion, arise because some of the I-elements is the system is forced to have differential causality. An demonstration of this can be seen in the example 2.3, which follows shortly.

### 2.2.2 Field and Vector Notation in the Bond Graph Language

In addition to basic elements, the concept of fields are relevant for this thesis. Fields are multi port generalizations of the basic scalar elements, and can be used in order to model complex multi dimensional systems (Pedersen and Engja, 2008). As an example, when modelling systems of bodies in spatial motion, it might be convenient to express forces, velocities, momenta and displacement in terms of three dimensional vectors. Then it is convenient to both have power bonds able to carry the effort and flow in terms of vectors, and to have basic elements which can interface such bonds, i.e., basic elements with multiple ports. In order to demonstrate this, we consider a C-field and an I-field. In addition, we shall see how the C-field and I-field can be combined into the IC-field, which is highly relevant for this thesis. Finally, some comments on the vector version of the TF-element is given.

Let some effort vector be defined as $\boldsymbol{e} = [e_1, e_2, ..., e_n]^T$, and a corresponding flow be defined as $\boldsymbol{f} = [f_1, f_2, ..., f_n]^T$. Consider now a C-field, to which a power bond carrying these efforts and flows are connected. Assuming integral causality, the constitutive relation is given as

$$\boldsymbol{e} = \boldsymbol{\Phi}_C^{-1}\left(\int_0^t \boldsymbol{f} dt\right) = \boldsymbol{\Phi}_C^{-1}(\boldsymbol{q}) \tag{2.35}$$

where $\boldsymbol{\Phi}_C : R^n \to R^n$, such that

$$\begin{aligned} e_1 &= \Phi_{C1}^{-1}(q_1, q_2, ..., q_n) \\ e_2 &= \Phi_{C2}^{-1}(q_1, q_2, ..., q_n) \\ &\vdots \\ e_n &= \Phi_{Cn}^{-1}(q_1, q_2, ..., q_n) \end{aligned} \tag{2.36}$$

The constitution relations for an I-field, assuming integral causality, becomes

$$\boldsymbol{f} = \boldsymbol{\Phi}_I^{-1}\left(\int_0^t \boldsymbol{e} dt\right) = \boldsymbol{\Phi}_I^{-1}(\boldsymbol{p}) \tag{2.37}$$

where $\boldsymbol{\Phi}_I : R^n \to R^n$, such that

$$\begin{aligned} f_1 &= \Phi_{I1}^{-1}(p_1, p_2, ..., p_n) \\ f_2 &= \Phi_{I2}^{-1}(p_1, p_2, ..., p_n) \\ &\vdots \\ f_n &= \Phi_{In}^{-1}(p_1, p_2, ..., p_n) \end{aligned} \tag{2.38}$$

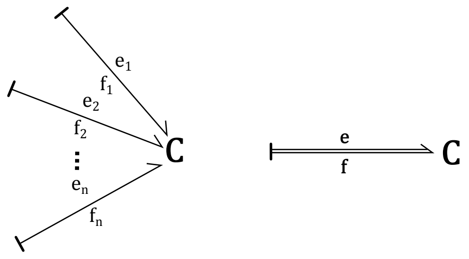


Figure 2.3: C-field with scalar power bonds to the left and vector bond to the right.

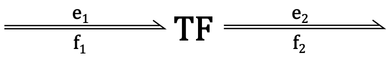


Figure 2.4: Vector version of transformer element.

Figure 2.3 show a C-field with n ports. To the left, $n$ scalar power bonds are used, while to the right, the power bond vector notation used throughout this thesis is presented.

Consider now a case when it is convenient, or even necessary, to express the relations of an effort and flow as

$$\begin{aligned} \boldsymbol{e} &= \boldsymbol{f}(\boldsymbol{p}, \boldsymbol{q}) \\ \boldsymbol{f} &= \boldsymbol{g}(\boldsymbol{p}, \boldsymbol{q}) \end{aligned} \tag{2.39}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are some vector valued functions. In such a case it is convenient to be able to combine a C-field and an I-field. This can be done with the IC-field, which can use the equations (2.39) as its constitutive relation.

Finally, the vector version of the transformer is discussed. To this effect we define the effort and flow vectors $\boldsymbol{e_1} = [e_{1,1}, e_{1,2}, ..., e_{1,n}]^T$ and $\boldsymbol{f_1} = [f_{1,1}, f_{1,2}, ..., f_{1,n}]^T$, as well as $\boldsymbol{e_2} = [e_{2,1}, e_{2,2}, ..., e_{2,m}]^T$ and $\boldsymbol{f_2} = [f_{2,1}, f_{2,2}, ..., f_{2,m}]^T$. Consider now the bond graph shown in figure 2.4, where the causality may be defined in either of the two manners shown for the TF-element in table 2.1. Then, using the $n \times m$ modulus matrix $\boldsymbol{M}$, the effort transformation can be written as (Karnopp, 1969)

$$\boldsymbol{M}\boldsymbol{e}_1 = \boldsymbol{e}_2 \tag{2.40}$$

The flow transformation can now be found by using the power conserving property of the transformer element by noting that the power input $P_1$ and the power output $P_2$ are related by

$$P_1 = \boldsymbol{f}_1^T \boldsymbol{e}_1 = \boldsymbol{f}_2^T \boldsymbol{e}_2 = P_2 \tag{2.41}$$

Then, by inserting the effort transformation for $\boldsymbol{e}_2$, we find

$$\boldsymbol{f}_1^T \boldsymbol{e}_1 = \boldsymbol{f}_2^T \boldsymbol{M} \boldsymbol{e}_1 \tag{2.42}$$

Solving this for $\boldsymbol{f}_1$ yields the flow transformation

$$\boldsymbol{f}_1 = \boldsymbol{M}^T \boldsymbol{f}_2 \tag{2.43}$$

As a final remark, before concluding this section with the example 2.2 in bond graph, we introduce the *powermuxer*. This is an element that takes in $n$ power bonds, and outputs the a single vectorial power bond with a dimension corresponding to the total number of input power, i.e., dimension $n$ if all inputs are scalar. Figure 2.5 show the graphical icon for this.

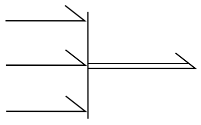


Figure 2.5: Power muxer illustration.

**Example 2.3.** Recall from example 2.2 the inverted pendulum on a moving wagon shown in figure 2.1. The equations of motion for the system was found using the Lagrangian method. In this example, we model the system using bond graph. Consider first the horizontal velocity, $\dot{x}_1$ of the wagon. In figure 2.6 we see that this velocity is represented by the 1-junction denoted $\dot{q}_1$ because $\dot{x}_1 = \dot{q}_1$. This velocity is associated to the generalized force $\tau_1$, represented by an effort source, the damper with damping coefficient $d_1$, represented by a R-element, the inertia $m_1$, represented by an I-element, and finally the spring with coefficient $k_1$, modelled with a C-element.

Next, recall that the linear velocity of the concentrated mass at the end of the pendulum was found as $\boldsymbol{v}_{cg2/0} = [\dot{x}_2,\, \dot{z}_2]^T$ where

$$\begin{aligned} \dot{x}_2 &= \dot{q}_1 + L\cos(q_2)\dot{q}_2 \\ \dot{z}_2 &= -L\sin(q_2)\dot{q}_2 \end{aligned} \tag{2.44}$$

and $q_2$ is the angular displacement of the pendulum. These velocity components of $\boldsymbol{v}_{cg2/0}$ are represented by the two 1-junctions denoted $\dot{x}_2$ and $\dot{z}_2$. The component $\dot{x}_2$ consists of the contribution $\dot{q}_1$ and the contribution $L\cos(q_2)\dot{q}_2$ found in the topmost MTF-element in figure 2.6. These contributions are summed together in the 0-junction. The component $\dot{z}_2$ of $\boldsymbol{v}_{cg2/0}$ is found in the lower MTF-element.

To $\dot{x}_2$, we associate only the inertia $m_2$. The velocity $\dot{z}_2$ is associated to the inertia

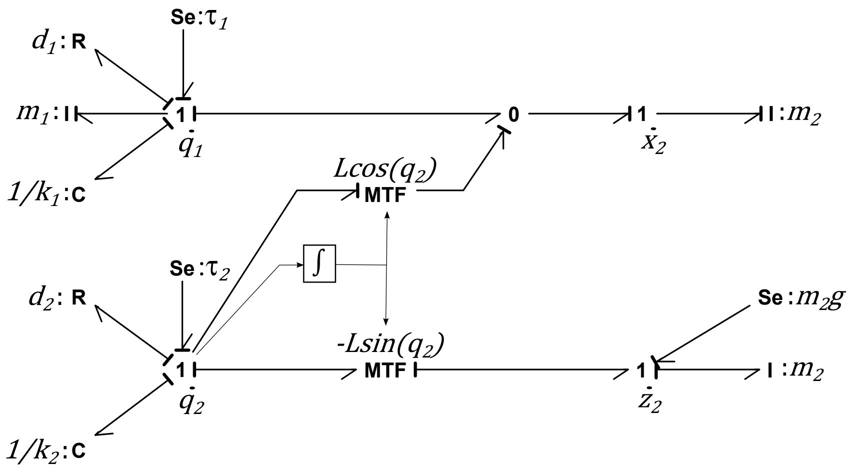


Figure 2.6: Bond graph of inverted pendulum on moving wagon.

$m_2$ and to the weight $m_2g$, represented by an effort source. The generalized force $\tau_2$, the friction force and the spring force on the pendulum are all associated to the velocity $\dot{q}_2$, as shown in figure 2.6. Notice from the figure that the causality issues mentioned in the introduction of this thesis does occur in this example. In this case, the result is that the I-element connected to the vertical velocity of the pendulum mass have differential causality, rather than integral causality.

## 2.3 Kinematics for Systems of Linked Rigid Bodies

When studying the kinematics of a system of rigid bodies in spatial motion, we are concerned with finding the positions and velocities of all relevant points on the system. In the context of Lagrange mechanics, we are in particular concerned with expressing the positions of all joints, i.e., points where bodies are linked together, and the bodies center of gravity as functions of the generalized coordinates. Furthermore, we seek to find expressions for the linear and angular velocities of each of the bodies center of gravity as functions of the generalized coordinates and their time derivatives.

The manipulators we shall study in this thesis are characterized by a structure corresponding to an open chain of linked rigid bodies with lower pair joints, i.e., joints with one degree of freedom (Sciavicco and Siciliano, 2000). Therefore, in section 2.3.2, a general method for finding the necessary kinematic relations for such systems are outlined. First however, we discuss the concept of rotation transformations, which is a vital tool in kinematic analysis of systems of bodies in spatial motion. The theory presented in the following section is found in Ginsberg (1995, chapter 3) and

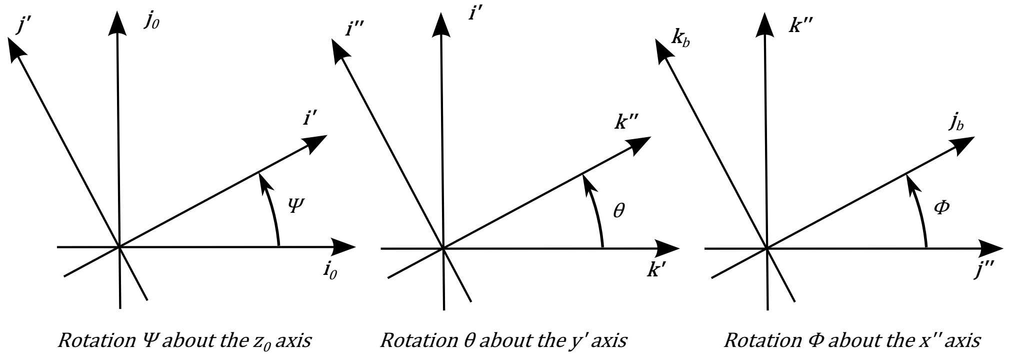
Meirovitch (2003, chapter 3), as well as Sciavicco and Siciliano (2000).

### 2.3.1 Rotation Transformations

When developing equations of motion for a system of rigid bodies, we frequently have to express vectors, such as position and velocities, in terms of a common reference frame. Recall from section 2.1.2 that we summed the potential energy for each body in a system, in order to find the potential energy for the whole system, and likewise, that we summed all individual contributions to the kinetic energy in order to find the total kinetic energy of the system. In order to do this, we must express all velocities and coordinates in terms of the same reference frame, i.e., we must define the position, and observe velocities from a common reference frame, or rather, from reference frames with the same orientation. In order to achieve this, we sometimes have to change the direction in which the components of a vector is defined. More specifically, if the components of a vector are defined in the principal directions of the moving reference frame $x_b y_b z_b$, with the unit normal vector $\boldsymbol{i}_b$, $\boldsymbol{j}_b$, and $\boldsymbol{k}_b$, we might wish to define the components of the vector instead in the principal directions $\boldsymbol{i}_0$, $\boldsymbol{j}_0$, and $\boldsymbol{k}_0$ of some inertial reference frame $x_0 y_0 z_0$. In order to achieve this, we first need a means in which to define the orientation of the moving reference frame relative to the inertial reference frame. To this effect, consider now a moving body on which the reference frame $x_b y_b z_b$ is attached. If a reference frame, initially with the same orientation as the inertial reference frame, is rotated an angle $\psi$ about its z-axis, the intermediate reference frame $x^{'} y^{'} z^{'}$ results from the rotation. If we now rotate this reference frame an angle $\theta$ about the $y^{'}$-axis, this results in the new intermediate reference frame $x^{''} y^{''} z^{''}$. If a third rotation $\phi$ is performed about the $x^{''}$-axis, and the resulting reference frame have the same orientation as the reference frame attached to the body, $x_b y_b z_b$, then $\boldsymbol{\Theta} = [\phi,\, \theta,\, \psi]^T$ are the Euler angles describing the orientation of the body relative to the inertial reference frame. The sequence of rotations described above is illustrated in figure 2.7. Note that this is one manner in which to define the Euler angles. Other sequences of principal rotations would produce an other set of Euler angles. Throughout this thesis, the definition stated above is used.

For each of the principal rotations described above, we find expressions for the unit normal vectors before the principal rotation takes place, in terms of the unit normal vectors of the reference frame resulting from the rotation. For the rotation $\psi$ about the $z_0$-axis we have

$$\begin{aligned} \boldsymbol{i}_0 &= \boldsymbol{i}^{'} \cos(\psi) - \boldsymbol{j}^{'} \sin(\psi) \\ \boldsymbol{j}_0 &= \boldsymbol{i}^{'} \sin(\psi) + \boldsymbol{j}^{'} \cos(\psi) \\ \boldsymbol{k}_0 &= \boldsymbol{k}^{'} \end{aligned} \tag{2.45}$$



j'
j0
i'
Ψ
i0
i''
i'
k''
θ
k'
kb
k''
jb
Φ
j''
Rotation Ψ about the z0 axis
Rotation θ about the y' axis
Rotation Φ about the x'' axis

Figure 2.7: Illustration of a sequence of the three principal rotations defining the Euler angles.

Likewise, for the rotation $\theta$ about the $y^{'}$-axis, we get

$$\begin{aligned} \boldsymbol{i}^{'} &= \boldsymbol{i}^{''}\cos(\theta) - \boldsymbol{k}^{''}\sin(\theta) \\ \boldsymbol{j}^{'} &= \boldsymbol{j}^{''} \\ \boldsymbol{k}^{'} &= -\boldsymbol{i}^{''}\sin(\theta) + \boldsymbol{k}^{''}\cos(\theta) \end{aligned} \tag{2.46}$$

Finally, for the rotation $\phi$ about the $x^{''}$-axis we find

$$\begin{aligned} \boldsymbol{i}^{''} &= \boldsymbol{i}_b \\ \boldsymbol{j}^{''} &= \boldsymbol{j}_b\cos(\phi) - \boldsymbol{k}_b\sin(\phi) \\ \boldsymbol{k}^{''} &= \boldsymbol{j}_b\sin(\phi) + \boldsymbol{k}_b\cos(\phi) \end{aligned} \tag{2.47}$$

By substituting (2.47) for the right hand expressions in (2.46), and then substituting this for the right hand side expressions in (2.45), we see that the unit vectors of the reference frame parallel to the inertial reference frame can be expressed in terms of the unit vectors of the reference frame attached to the moving body. Consider now a vector $\boldsymbol{r}^0 = a\boldsymbol{i}_0 + b\boldsymbol{j}_0 + c\boldsymbol{k}_0$, where the superscript 0 indicate that the vector is expressed in terms of the inertial reference frame $x_0y_0z_0$. By substituting the unit vectors as outlined above, the vector $\boldsymbol{r}^0$ is transformed such that its components are defined in the directions of the moving reference frame $x_by_bz_b$. After this substitution, we denote the vector $\boldsymbol{r}^b$ in order to specify that it is expressed in terms of the moving reference frame $x_by_bz_b$. In order to do this transformation in a compact manner, we introduce the principal rotation transformation matrices

$$\begin{aligned} \boldsymbol{R}_z(\psi) &= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \boldsymbol{R}_y(\theta) = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \\ \boldsymbol{R}_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \end{aligned} \tag{2.48}$$

where $c_x = \cos(x)$, $s_x = \sin(x)$. Doing so allows us to write

$$\boldsymbol{r}^0 = \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi)\boldsymbol{r}^b = \boldsymbol{R}_b^0(\boldsymbol{\Theta})\boldsymbol{r}^b \tag{2.49}$$

where the orthogonal $3 \times 3$ matrix $\boldsymbol{R}_b^0(\boldsymbol{\Theta})$ is the rotation transformation matrix from 0 to $b$. The fact that any rotation transformation matrix $\boldsymbol{R}$ is orthogonal, allows us to write $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_{3\times3}$(Meirovitch, 2003). This implies that $\boldsymbol{R}^{\boldsymbol{T}} = \boldsymbol{R}^{-1}$, and we can define the inverse of (2.49) as

$$\boldsymbol{r}^b = (\boldsymbol{R}_b^0(\Theta))^T\boldsymbol{r}^0 = \boldsymbol{R}_0^b(\Theta)\boldsymbol{r}^0 \tag{2.50}$$

We can for any set of two reference frames apply a sequence of three, or fewer principal rotations which will align one with the other, and thus relate any two reference frames using compositions of the three principal rotation matrices defined in (2.48). Using this, and given two bodies moving relative to each other, each with a reference frame attached, we can take any vector defined in the directions of one of the reference frames, and express it in terms of the other. As we shall see shortly, this is useful when analysing the kinematics of a system of rigid bodies in spatial motion.

## 2.3.2 Open Chains of Linked Rigid Bodies

In this section we outline a general procedure for finding the relevant coordinates and velocities of an open chain of linked rigid bodies. We are interested in this kind of structure because equipment such as deck cranes, robotic manipulators and other manipulator equipment often can be described as open chains of linked rigid bodies, although there are of course numerous exceptions to this.

The contents presented in this section is based on Sciavicco and Siciliano (2000), as well as the discussion provided in the previous section. The kinematics discussions presented in Sciavicco and Siciliano (2000) is based on the Denavit-Heartenberg method. Although this method provide a systematic and efficient manner in which to analyse the kinematics of a system, it also tends to discourage physical understanding of the system in question by hiding the various geometric relations in a set

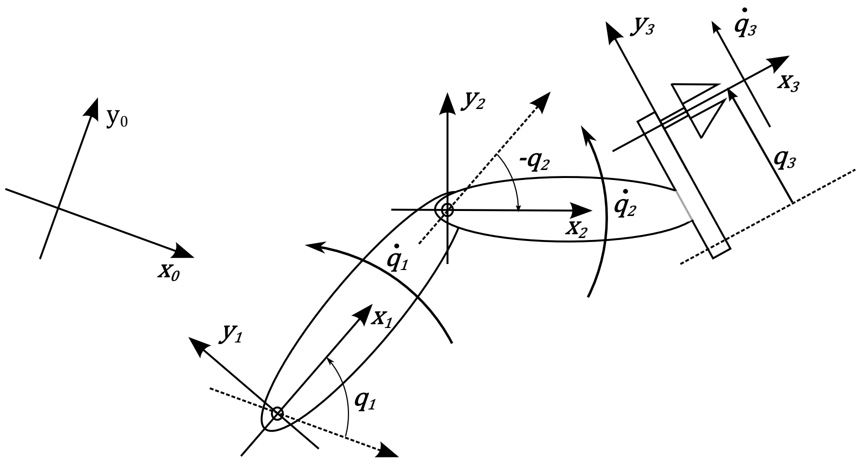


Figure 2.8: Schematic representation of an manipulator constituting an open chain of linked bodies with lower pair joints.

of algorithm-like procedures, at least in the opinion of the author. Therefore the material in this section is presented outside the context of the Denavit-Heartenberg method, but still with the same result and basic ideas.

The relevant coordinates and velocities we seek to find are those necessary in order to find the potential and kinetic energy of the system. For the potential energy expression given in (2.2), we see that it is at least necessary to find the coordinates of each of the bodies center of gravity relative to some common reference point. Furthermore, if there are springs connected to the system, the elongation of the springs must be found in terms of the generalized coordinates. In order to find these coordinates, it is necessary to also find the coordinates of each of the joints of the system relative to the common reference point. In order to find the expression for the kinetic energy, we see from (2.5), that it is necessary to find the linear velocity of the center of gravity for each body, as well as the angular velocity of each body.

We start by finding the coordinates of the center of gravity for the $i$-th body in the system, relative to the origin of an inertial reference frame. Figure 2.8 show an open chain of linked rigid bodies. Notice from the figure that a generalized coordinate is defined in each joint as the angular or linear displacement relative to the preceding body. We see that an inertial reference frame denoted 0 is defined to the left. In addition, a reference frame is defined in each of the joints of the system. Reference frame $i$ is defined in joint $i$, and is attached to the $i$-th body. The coordinate $\boldsymbol{r}^{i}_{i+1/i}$, where the superscript indicate the coordinate is expressed in terms of reference frame $i$, is that of the origin of reference frame $(i+1)$, relative to reference frame $i$.

The coordinate of the center of gravity of body $i$ relative to the origin of reference frame $i$, expressed in terms of reference frame $i$ is denoted $\boldsymbol{r}^i_{cm_i/i}$. The unit normal vector about or along which joint $i$ displace is denoted $\boldsymbol{e}_i$. These coordinates can be expressed in terms of the inertial reference frame, or more precisely, the components of these vectors can be defined in the principal directions of the inertial reference frame, by applying rotation transformations. To see how this can be done, consider an arbitrary vector $\boldsymbol{r}^i$, the components of which defined in the directions of the reference frame $i$. Identifying the sequence of principal rotation necessary to apply to the reference frame $(i-1)$ in order to align it with reference frame $i$, we can define the rotation transformation matrix $\boldsymbol{R}^{i-1}_i(q_i)$, as in the previous section. We then have that

$$\boldsymbol{r}^{i-1} = \boldsymbol{R}^{i-1}_i(q_i)\boldsymbol{r}^i \tag{2.51}$$

Given such rotation transformation matrices for all $i \in [1, n]$, we can write

$$\begin{aligned}\boldsymbol{r}^0 &= \boldsymbol{R}^0_1(q_1)\boldsymbol{r}^1 = \boldsymbol{R}^0_1(q_1)\boldsymbol{R}^1_2(q_2)\boldsymbol{r}^2 \\ &= \boldsymbol{R}^0_1(q_1)\boldsymbol{R}^1_2(q_2)\cdots\boldsymbol{R}^{i-2}_{i-1}(q_{i-1})\boldsymbol{R}^{i-1}_i(q_i)\boldsymbol{r}^i \\ &= \boldsymbol{R}^0_i(\boldsymbol{q})\boldsymbol{r}^i\end{aligned} \tag{2.52}$$

where $\boldsymbol{R}^0_i(\boldsymbol{q}) = \boldsymbol{R}^0_1(q_1)\boldsymbol{R}^1_2(q_2)\cdots\boldsymbol{R}^{i-2}_{i-1}(q_{i-1})\boldsymbol{R}^{i-1}_i(q_i)$. Thus, we are able to express any vector given in terms of reference frame $i$ in terms of the inertial reference frame. Then the position of the center of gravity of body $i$, relative to the origin of the inertial reference frame, expressed in terms of the inertial reference frame, is

$$\boldsymbol{r}^0_{cm_i/0} = \boldsymbol{r}^0_{1/0} + \boldsymbol{r}^0_{2/1} + \cdots + \boldsymbol{r}^0_{i/i-1} + \boldsymbol{r}^0_{cm_i/i} \tag{2.53}$$

We note that the vector $\boldsymbol{r}^{i-1}_{i/i-1}$ is constant in the case where joint $i$ is revolute. However, if joint $i$ is prismatic, meaning that it can translate along the axis $\boldsymbol{e}_i$, this vector is defined by

$$\boldsymbol{r}^{i-1}_{i/i-1} = \boldsymbol{r}^{i-1}_{io/i-1} + \boldsymbol{e}^{i-1}_i q_i \tag{2.54}$$

where $\boldsymbol{r}^{i-1}_{io/i-1}$ is the position of the origin of reference frame $i$ relative to the origin of reference frame $(i-1)$ given that $q_i = 0$.

We now proceed to find the linear velocity of the center of gravity, as well as the angular velocity of body $i$. We have already seen from proposition 2.1, that the velocity at some arbitrary point $p$ on the system can be found through some geometric Jacobian matrix $\boldsymbol{\mathcal{J}}(\boldsymbol{q})$ as

$$\boldsymbol{\nu}_p = \boldsymbol{\mathcal{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.55}$$

where $\boldsymbol{\nu}_p$ denotes the velocity at $p$. We now proceed to find expressions for the geometric Jacobian matrices for the relevant velocities for the manipulator. In order to do this, we need only determine what contribution each of the generalized coordinate rates have to the velocity in question. We start by finding the contributions to the linear velocity of the center of gravity of body $i$ from the generalized coordinate $k$. To this effect we introduce the notation

$$\boldsymbol{v}_{cg_i/0}^{(\dot{q}_k)} \triangleq \boldsymbol{\mathcal{J}}_{(\dot{q}_k)}^{\boldsymbol{v}_{cgi}} \dot{q}_k \tag{2.56}$$

where the index $(\dot{q}_k)$ indicate from where the contribution comes, and the index $\boldsymbol{v}_{cg_i}$ indicates to what velocity the contribution goes, namely the center of gravity of body $i$ relative to the inertial reference frame. Both $\boldsymbol{v}_{cg_i/0}^{(\dot{q}_k)}$ and $\boldsymbol{\mathcal{J}}_{(\dot{q}_k)}^{\boldsymbol{v}_{cg_i}}$ are $3 \times 1$ vectors. If $k > i$, then joint $k$ have no contribution to the linear velocity of body $i$. On the other hand, if $k \leq i$, the contribution depends on whether joint $k$ is revolute or prismatic. Specifically we find

$$\boldsymbol{v}_{cg_i}^{(\dot{q}_k)} = \begin{cases} \dot{q}_k \boldsymbol{e}_k^0 \times \boldsymbol{r}_{cg_i/k}^0, & \text{if k is revolute} \\ \dot{q}_k \boldsymbol{e}_k^0, & \text{if k is prismatic} \end{cases} \tag{2.57}$$

where $\boldsymbol{e}_k^0$ is the unit vector, expressed in terms of the inertial reference frame, about or along which joint $k$ displace. Combining (2.56) and (2.57), we find that

$$\boldsymbol{\mathcal{J}}_{(\dot{q}_k)}^{\boldsymbol{v}_{cgi}} = \begin{cases} \boldsymbol{e}_k^0 \times \boldsymbol{r}_{cg_i/k}^0, & \text{if k is revolute} \\ \boldsymbol{e}_k^0, & \text{if k is prismatic} \end{cases} \tag{2.58}$$

Contributions to the angular velocity of body $i$ from joint $k$ is defined by

$$\boldsymbol{\omega}_{i/0}^{(\dot{q}_k)} \triangleq \boldsymbol{\mathcal{J}}_{(\dot{q}_k)}^{\boldsymbol{\omega}_i} \dot{q}_k \tag{2.59}$$

where the indices have the same meaning as for the linear velocities. If $k > i$, there are no contribution, while if $k \leq i$, the contribution is

$$\boldsymbol{\omega}_{i/0}^{(\dot{q}_k)} = \begin{cases} \dot{q}_k \boldsymbol{e}_k^0, & \text{if k is revolute} \\ 0, & \text{if k is prismatic} \end{cases} \tag{2.60}$$

By combining (2.59) and (2.60), we find

$$\boldsymbol{\mathcal{J}}_{(\dot{q}_k)}^{\boldsymbol{\omega}_i} \begin{cases} \boldsymbol{e}_k^0, & \text{if k is revolute} \\ 0, & \text{if k is prismatic} \end{cases} \tag{2.61}$$

We can now sum the contributions to the linear velocity of the center of gravity of body $i$ as

$$\begin{aligned} \boldsymbol{v}_{cg_i/0}^0 &= \boldsymbol{\mathcal{J}}_{(\dot{q}_1)}^{\boldsymbol{v}_{cgi}} \dot{q}_1 + \boldsymbol{\mathcal{J}}_{(\dot{q}_2)}^{\boldsymbol{v}_{cgi}} \dot{q}_2 + \cdots + \boldsymbol{\mathcal{J}}_{(\dot{q}_i)}^{\boldsymbol{v}_{cgi}} \dot{q}_i + \mathbf{0}_{3\times1}\dot{q}_{i+1} + \cdots + \mathbf{0}_{3\times1}\dot{q}_n \\ &= \begin{bmatrix} \boldsymbol{\mathcal{J}}_{(\dot{q}_1)}^{\boldsymbol{v}_{cgi}} & \boldsymbol{\mathcal{J}}_{(\dot{q}_2)}^{\boldsymbol{v}_{cgi}} & \cdots & \boldsymbol{\mathcal{J}}_{(\dot{q}_i)}^{\boldsymbol{v}_{cgi}} & \mathbf{0}_{3\times(n-i)} \end{bmatrix} \dot{\boldsymbol{q}} \\ &\triangleq \boldsymbol{J}_{cg_i}^{v}(\boldsymbol{q})\dot{\boldsymbol{q}} \end{aligned} \tag{2.62}$$

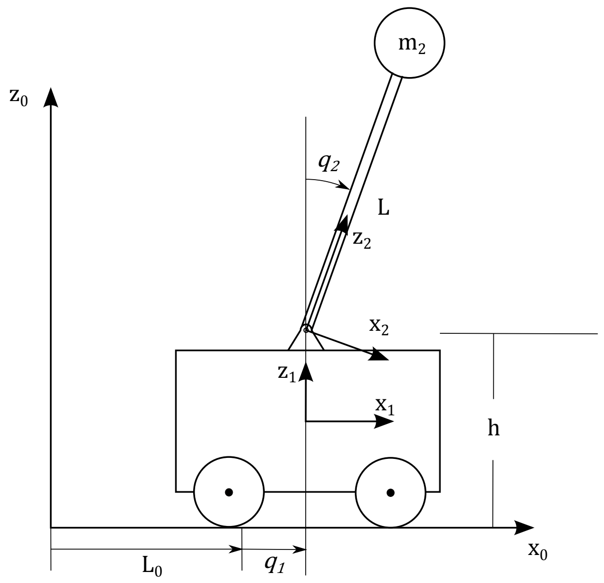


Figure 2.9: Schematic diagram of the inverted pendulum on rolling wagon.

In a similar manner, the angular velocity of body $i$ is found as

$$\begin{aligned}\boldsymbol{\omega}^0_{i/0} &= \begin{bmatrix} \boldsymbol{\mathcal{J}}^{\omega_i}_{(\dot{q}_1)} & \boldsymbol{\mathcal{J}}^{\omega_i}_{(\dot{q}_2)} & \cdots & \boldsymbol{\mathcal{J}}^{\omega_i}_{(\dot{q}_i)} & \mathbf{0}_{3\times(n-i)} \end{bmatrix} \dot{\boldsymbol{q}} \\ &\triangleq \boldsymbol{J}^{\omega}_i(\boldsymbol{q})\dot{\boldsymbol{q}}\end{aligned} \tag{2.63}$$

The linear velocity of the center of gravity of body $i$, and the angular velocity of body $i$ are now collected into a single vector as

$$\boldsymbol{v}_i = \begin{bmatrix} \boldsymbol{v}^0_{cg_i/0} \\ \boldsymbol{\omega}^0_{i/0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}^v_{cg_i}(\boldsymbol{q}) \\ \boldsymbol{J}^{\omega}_i(\boldsymbol{q}) \end{bmatrix} \dot{\boldsymbol{q}} \triangleq \boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.64}$$

**Example 2.4.** The method for finding the relevant positions and velocities of a system of rigid bodies in spatial motion is now demonstrated through an example. In particular we shall find the relevant positions and velocities of the inverted pendulum of example 2.2, using the framework presented above. We also show how this can be done using the symbolic mathematical software Maple.

A schematic diagram of the system is shown in figure 2.9, where the various forces and springs are omitted. The wagon can displace along the $x_0$-axis with the same direction as the unit normal vector $\boldsymbol{e}_1$. Therefore we can consider the wagon to be linked to the ground through a prismatic lower pair joint. The link between the pendulum and the wagon is a lower pair revolute joint, rotating about an axis with the same direction as the unit vector $\boldsymbol{e}_2$. Notice also from the figure that two additional reference frames are defined. First, the reference frame $x_1y_1z_1$ is fixed to the wagon. The orientation of the reference frame is always identical to that of the $x_0y_0z_0$ reference frame. Next, the frame $x_2y_2z_2$ is defined. This reference frame is attached to pendulum, with its origin in the joint between the wagon and the pendulum.

The unit vector along which the wagon displace, $\boldsymbol{e}_1$, and the unit vector about which the pendulum rotate, $\boldsymbol{e}_2$, can be expressed in terms of their local reference frames as

$$\begin{aligned} \boldsymbol{e}_1^1 &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \\ \boldsymbol{e}_2^2 &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \end{aligned} \tag{2.65}$$

The coordinate of the reference frame 1, relative to the origin of the 0-frame, expressed in terms of the 0-frame, given that $q_1 = 0$, is denoted $\boldsymbol{r}_{1o/0}^0$, and defined as

$$\boldsymbol{r}_{1o/0}^0 = \begin{bmatrix} L_0 & 0 & h/2 \end{bmatrix}^T \tag{2.66}$$

Then the coordinates of the reference frame 1 relative to 0 for arbitrary values of $q_1$ is found as

$$\boldsymbol{r}_{1/0}^0 = q_1\boldsymbol{e}_1^0 + \boldsymbol{r}_{1o/0}^0 = \begin{bmatrix} L_0 + q_1 \\ 0 \\ h/2 \end{bmatrix} \tag{2.67}$$

The coordinates of the origin of reference frame 2 relative to 1, expressed in terms of reference frame 1 is

$$\boldsymbol{r}_{2/1}^1 = \begin{bmatrix} 0 & 0 & h/2 \end{bmatrix}^T \tag{2.68}$$

and the coordinates of the concentrated mass at the end of the pendulum relative to the origin of reference frame 2, in terms of reference frame 2 is

$$\boldsymbol{r}_{cg2/2}^2 = \begin{bmatrix} 0 & 0 & L \end{bmatrix}^T \tag{2.69}$$

We now define the two rotation matrices $\boldsymbol{R}_1^0 = \boldsymbol{I}_{3\times3}$, and $\boldsymbol{R}_2^1(q_2) = \boldsymbol{R}_y(q_2)$, where $\boldsymbol{R}_y$ is defined in (2.48). The first rotation matrix is the $3 \times 3$ identity matrix because the orientation of the 1-frame relative to the 0-frame is constant and identical. It is however included in order to keep a certain consistency of notation. The second rotation matrix represents a rotation about the $y$-axis between the 1-frame and the 2-frame. With these rotation matrices we find the position of the concentrated mass at the end of the pendulum, relative to the origin of the 0-frame, in terms of the

0-frame, as

$$\boldsymbol{r}^0_{cg2/0} = \boldsymbol{r}^0_{1/0} + \boldsymbol{R}^0_1\boldsymbol{r}^1_{2/1} + \boldsymbol{R}^0_1\boldsymbol{R}^1_2(q_2)\boldsymbol{r}^2_{cg2/2} = \begin{bmatrix} L_0 + q_1 + L\sin(q_2) \\ 0 \\ h + L\cos(q_2) \end{bmatrix} \tag{2.70}$$

Similarly, the unit vectors about or along which the joints of the system displace, can be expressed in terms of the 0-frame as

$$\begin{aligned} \boldsymbol{e}^0_1 &= \boldsymbol{R}^0_1\boldsymbol{e}^1_1 \\ \boldsymbol{e}^0_2 &= \boldsymbol{R}^0_1\boldsymbol{R}^1_2(q_2)\boldsymbol{e}^2_2 \end{aligned} \tag{2.71}$$

We now find the linear velocity of the wagon and the concentrated pendulum mass. The contribution to the linear velocity of the wagon from $\dot{q}_1$ is

$$\boldsymbol{v}^{(\dot{q}_1)}_{1/0} = \boldsymbol{i}\,\dot{q}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \dot{q}_1 = \boldsymbol{\mathcal{J}}^{\boldsymbol{v}_1}_{\dot{q}_1}\dot{q}_1 \tag{2.72}$$

There are no contribution to the linear velocity of the wagon from $\dot{q}_2$. Thus, the velocity of the wagon can be expressed as

$$\begin{aligned} \boldsymbol{v}^0_{1/0} &= \begin{bmatrix} \boldsymbol{\mathcal{J}}^{\boldsymbol{v}_1}_{\dot{q}_1} & \boldsymbol{0}_{3\times1} \end{bmatrix}\dot{\boldsymbol{q}} = \boldsymbol{J}_1\dot{\boldsymbol{q}} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{2.73}$$

which is in accordance with the velocity found in example 2.2. The contribution to the linear velocity of the concentrated mass of the pendulum from $\dot{q}_1$ is also given by

$$\boldsymbol{v}^{(\dot{q}_1)}_{cg2/0} = \boldsymbol{i}\,\dot{q}_1 = \boldsymbol{\mathcal{J}}^{\boldsymbol{v}_{cg2}}_{\dot{q}_1}\dot{q}_1 \tag{2.74}$$

while the contribution from $\dot{q}_2$ is

$$\boldsymbol{v}^{(\dot{q}_2)}_{cg2/0} = (\boldsymbol{e}^0_2 \times \boldsymbol{r}^0_{cg2/2})\dot{q}_2 = \boldsymbol{\mathcal{J}}^{\boldsymbol{v}_{cg2}}_{\dot{q}_2}\dot{q}_2 \tag{2.75}$$

We can then express the linear velocity of the pendulum concentrated mass as

$$\begin{aligned} \boldsymbol{v}^0_{cg2/0} &= \begin{bmatrix} \boldsymbol{\mathcal{J}}^{\boldsymbol{v}_{cg2}}_{\dot{q}_1} & \boldsymbol{\mathcal{J}}^{\boldsymbol{v}_{cg2}}_{\dot{q}_2} \end{bmatrix}\dot{\boldsymbol{q}} = \boldsymbol{J}_2(q_2)\dot{\boldsymbol{q}} \\ &= \begin{bmatrix} 1 & L\cos(q_2) \\ 0 & 0 \\ 0 & -L\sin(q_2) \end{bmatrix}\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 + L\cos(q_2)\dot{q}_2 \\ 0 \\ -L\sin(q_2)\dot{q}_2 \end{bmatrix} \end{aligned} \tag{2.76}$$

which also is in accordance with the velocity found in example 2.2. This example is concluded with the algorithm 2.1 for doing these calculations in the symbolic mathematical software Maple.

**Algorithm 2.1** Kinematics of inverted pendulum on wagon: Maple 17

```
restart:
with(LinearAlgebra):

#Constant Coordinates
e1_1 := Vector([1, 0, 0]):
e2_2 := Vector([0, 1, 0]):
r0_1o_0 := Vector([L0, 0, (1/2)*h]):
r1_2_1 := Vector([0, 0, (1/2)*h]):
r2_cg2_2 := Vector([0, 0, L]):

#Rotation Matrices
R0_1 := IdentityMatrix(3):
R1_2 := Matrix([[cos(q2), 0, sin(q2)],[0, 1, 0], [-sin(q2), 0, cos(q2)]]):
R0_2 := R0_1 R1_2:

#Coordinates to CG's
r0_1_0 := r0_1o_0 + R0_1 e1_1*q1:
r0_cg2_0 := r0_1_0 + R0_1 r1_2_1 + R0_2 r2_cg2_2:

#Contributions to the Linear Vel. of Wagon
jcm1_q1 := Vector([1, 0, 0]):
jcm1_q2 := Vector([0, 0, 0]):

#Contributions to Lin. Vel. of Pendulum Mass
jcm2_q1 := Vector([1, 0, 0]):
jcm2_q2 := CrossProduct(R0_2 e2_2, R0_2 r2_cg2_2):

#Linear Velocities
dq := Vector([dq1, dq2]): #time rate of generalized coords.
J1 := Matrix([jcm1_q1, jcm1_q2]):
J2 := Matrix([jcm2_q1, jcm2_q2]):
v0_1_0 := J1 dq:
v0_cg2_0 := J2 dq:
```

# Chapter 3

# Bond Graph Modelling of Systems of Bodies in Spatial Motion

The bond graph language simplify the task of modelling complex multidimensional systems because it provides a graphical interface to the problem, mapping the energetic structure, rather than considering the interaction of forces and torques. This may contribute in making the problem more transparent, and allow for a more structured modelling approach. However, for increasingly large and complex systems, the bond graph grow large, and the number of occurrences of differential causality can make the computer implementation of the bond graph slow. This chapter presents a modelling framework, utilizing the power of the bond graph language as well as that of the Lagrangian mechanics. In particular, a bond graph template for the basic dynamics of the system, based on Lagrangian mechanics is created, such that issues of differential causality are eliminated, while still utilizing the virtues of the bond graph language. To this effect we shall use the IC-field technique, which we briefly discussed in section 2.2, in order to implement the Lagrangian equations of motion for a system of bodies in spatial motion in a bond graph. This method may seem to undermine the flexibility and modularity of the bond graph, as the basic dynamics of the system will be hidden within the IC-field. We shall however see how various interfaces to the system can be established within the bond graph, and as such regain the flexibility and modularity of the bond graph. In addition to presenting the theoretical approach, an effective manner in which to implement the model in the bond graph software 20-sim is presented.

In the following section, the Lagrangian equations of motion is rewritten into the Lagrange-Hamiltonian state space form, or the Hamiltonian canonical equations, discussed in e.g. Ginsberg (1995), Meirovitch (2003) and Pedersen and Engja (2008). This form, we shall see, is convenient when implementing the system in the bond graph language, both because it is in state space form, i.e., a set of first order differential equations, and because the states are the generalized momentum and the generalized coordinates, which are easily related to the bond graph. After implementing this model in a bond graph, using the IC-field technique as is done in Karnopp et al. (2006), and Pedersen and Engja (2008), it is shown, in a schematic manner, how useful interfaces to the basic dynamics can be found. Finally, we present an effective method for implementing the state space model in the bond graph software *20-sim*. The main problem addressed by this method, is how to efficiently develop and update the large expressions that occur for complicated systems. As such, this method may be useful in implementing the above mentioned state space model within any ordinary differential equation (ODE) solver, such as those provided by Matlab, though the focus in this text is the bond graph software 20-sim.

## 3.1 Lagrange Hamiltonian Equations of Motion

In this section, we show how the $n$ second order differential equations of the Lagrangian formulations (2.14), can be rewritten to state space form, i.e., a set of $2n$ first order differential equations, with the generalized coordinate displacements and the associated momentum as states, and the generalized forces as input. Recall that we found the kinetic energy of the $i$-th body of the system as

$$T_i(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{1}{2} m_i \boldsymbol{v}_{cg_i}^T \boldsymbol{v}_{cg_i} + \frac{1}{2}\boldsymbol{\omega}_i^T \boldsymbol{I}_{bi} \boldsymbol{\omega}_i \tag{3.1}$$

Recall also that we found the linear velocity of the center of gravity, and the angular velocity, of the $i$-th body as the compact expressions

$$\begin{aligned} \boldsymbol{v}_{cm_i/0}^0 &= \boldsymbol{J}_{cm_i}^v(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \boldsymbol{\omega}_{i/0}^0 &= \boldsymbol{J}_i^\omega(\boldsymbol{q})\dot{\boldsymbol{q}} \end{aligned} \tag{3.2}$$

which could be collected in the single expression

$$\boldsymbol{v}_i = \begin{bmatrix} \boldsymbol{v}_{cg_i/0}^0 \\ \boldsymbol{\omega}_{cg_i/0}^0 \end{bmatrix} = \boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.3}$$

We can collect the two terms on the left hand side of (3.1) in a matrix formulation as

$$T_i(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{1}{2}\boldsymbol{v}_i^T \begin{bmatrix} m_i \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{bi} \end{bmatrix} \boldsymbol{v}_i \tag{3.4}$$

where $\boldsymbol{I}_{bi}$ is the inertia tensor of the $i$-th body and $\boldsymbol{I}_{3\times 3}$ is the $3\times 3$ identity matrix. By substituting $\boldsymbol{v}_i$ for the right hand side in (3.3), and denoting the above matrix as $\boldsymbol{M}_i$, we get

$$T_i(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{J}_i^T(\boldsymbol{q})\boldsymbol{M}_i\boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}} = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{B}_i(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.5}$$

where $\boldsymbol{B}_i(\boldsymbol{q}) = \boldsymbol{J}_i^T(\boldsymbol{q})\boldsymbol{M}_i\boldsymbol{J}_i(\boldsymbol{q})$ is the mass-inertia matrix of the $i$-th body. By using that the total kinetic energy of the system is the sum of all contributions, we find

$$T(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T\left(\sum_{i=1}^{n}\boldsymbol{B}_i(\boldsymbol{q})\right)\dot{\boldsymbol{q}} = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.6}$$

where $\boldsymbol{B}(\boldsymbol{q})$ is the positive definite and symmetric system mass-inertia matrix.

As we now shall rewrite the Lagrangian equations of motion into state space form, recall that

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}}\right) - \left(\frac{\partial T}{\partial \boldsymbol{q}} - \frac{\partial V}{\partial \boldsymbol{q}}\right) = \boldsymbol{\tau} \tag{3.7}$$

The potential energy differentiated with respect to the generalized coordinates is the vector of *restoring forces*, $\boldsymbol{g}(\boldsymbol{q})$(Sciavicco and Siciliano, 2000). These are the weight of the rigid body, as well as any spring forces acting on the body. Thus

$$\frac{\partial V}{\partial \boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q}) \tag{3.8}$$

We recognize, by using (3.6), that the first term in the Lagrangian equations of motion can be found as

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}}\right) = \frac{d}{dt}\left(\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}\right) \tag{3.9}$$

We can in the same manner rewrite

$$\frac{\partial T}{\partial \boldsymbol{q}} = \left(\frac{1}{2}\dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right) \tag{3.10}$$

Inserting (3.8), (3.9), and (3.10) into (3.7), we get

$$\frac{d}{dt}\left(\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}\right) - \left(\frac{1}{2}\dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q})\right) = \boldsymbol{\tau} \tag{3.11}$$

We recognize $\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}$ as the momentum associated with the displacement of the generalized coordinates, i.e., the generalized momentum $\boldsymbol{p}$. Thus

$$\boldsymbol{p} = \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} \Leftrightarrow \dot{\boldsymbol{q}} = \boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{p} \tag{3.12}$$

By substituting $\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{p}$ in (3.11), we find

$$\begin{aligned}\dot{\boldsymbol{p}} &= \frac{1}{2}\dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}\\ &= \frac{1}{2}\boldsymbol{p}^T\boldsymbol{B}^{-1}(\boldsymbol{q})\frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{p} - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}\\ &\triangleq \boldsymbol{f}_p(\boldsymbol{p},\boldsymbol{q}) + \boldsymbol{\tau}\end{aligned} \tag{3.13}$$

where $\boldsymbol{f}_p : \mathcal{R}^{2n} \rightarrow \mathcal{R}^n$ is defined as

$$\boldsymbol{f_p}(\boldsymbol{p},\boldsymbol{q}) = \frac{1}{2}\boldsymbol{p}^T\boldsymbol{B}^{-1}(\boldsymbol{q})\frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{p} - \boldsymbol{g}(\boldsymbol{q}) \tag{3.14}$$

We have in the second line in (3.13) substituted (3.12) for $\dot{\boldsymbol{q}}$, and used the following proposition:

**Proposition 3.1.** *Since the mass-inertia matrix is symmetric, also the inverse mass-inertia matrix is symmetric, and we can write* $(\boldsymbol{B}^{-1})^T = \boldsymbol{B}^{-1}$.

*Proof.* As $\boldsymbol{B}$ is symmetric, we can write $\boldsymbol{B} = \boldsymbol{B}^T$. It then follows that $\boldsymbol{B}^{-1} = (\boldsymbol{B}^T)^{-1} = (\boldsymbol{B}^{-1})^T$. Then $\boldsymbol{B}^{-1} = (\boldsymbol{B}^{-1})^T$, and $\boldsymbol{B}^{-1}$ must be symmetric. □

Combining (3.12) and (3.13), we have the state space model

$$\begin{aligned}\dot{\boldsymbol{q}} &= \boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{p} \\ \dot{\boldsymbol{p}} &= \boldsymbol{f}_p(\boldsymbol{p},\boldsymbol{q}) + \boldsymbol{\tau}\end{aligned} \tag{3.15}$$

with the vector of generalized forces as input.

**Example 3.2.** In example 2.2, the second order Lagrangian equations of motion for an inverted pendulum on a moving wagon was found. The linear velocities of the centres of gravity of the two bodies in the system was found using geometric Jacobian matrices in example 2.4. In this example, the equations of motion for the inverted pendulum is developed in the Lagrange Hamiltonian state space form.

Using the geometric Jacobian matrices developed for the system in example 2.4 together with (3.5), we find the individual mass matrices for the system by using

$$\begin{aligned}T_1(\boldsymbol{q},\dot{\boldsymbol{q}}) &= \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{J}_1^T\boldsymbol{M}_1\boldsymbol{J}_1\dot{\boldsymbol{q}} = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{B}_1\dot{\boldsymbol{q}} \\ T_2(\boldsymbol{q},\dot{\boldsymbol{q}}) &= \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{J}_2^T(q_2)\boldsymbol{M}_2\boldsymbol{J}_2(q_2)\dot{\boldsymbol{q}} = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{B}_2(q_2)\dot{\boldsymbol{q}}\end{aligned} \tag{3.16}$$

where $\boldsymbol{M}_1 = m_1\boldsymbol{I}_{3\times3}$, $\boldsymbol{M}_2 = m_2\boldsymbol{I}_{3\times3}$, $T_1(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the kinetic energy associated to the wagon, and $T_2(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the kinetic energy associated to the pendulum. Notice that we in this example are not concerned with angular momentum, and as such not angular velocity since the wagon do not rotate and the pendulum is considered as a concentrated mass. This is also the reason why the term *mass matrix,* as opposed to *mass-inertia matrix* is used in this example.

The system mass matrix can be found by summing together individual mass matrices according to (3.6). Doing this yields

$$\begin{aligned}\boldsymbol{B}(q_2) = \boldsymbol{B}_1 + \boldsymbol{B}_2(q_2) &= \begin{bmatrix} m_1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} m_2 & Lm_2\cos(q_2) \\ Lm_2\cos(q_2) & L^2 m_2 \end{bmatrix} \\ &= \begin{bmatrix} m_1 + m_2 & Lm_2\cos(q_2) \\ Lm_2\cos(q_2) & L^2 m_2 \end{bmatrix}\end{aligned} \tag{3.17}$$

Notice that the system mass matrix is positive definite and symmetric. Notice also that expression for the total kinetic energy

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{B}(q_2)\dot{\boldsymbol{q}} = \frac{1}{2}m_1\dot{q}_1^2 + \frac{1}{2}m_2\left(L^2\dot{q}_2^2 + 2L\cos(q_2)\dot{q}_1\dot{q}_2 + \dot{q}_1^2\right) \tag{3.18}$$

is in agreement with the expression found for the total kinetic energy in example 2.2. With the mass matrix defined, the state equations for the generalized displacement can be found. In order to find the state equation for the generalized momentum, it is necessary to differentiate the mass matrix with respect to the generalized coordinates. Doing so yields

$$\begin{aligned}\frac{\partial \boldsymbol{B}(q_2)}{\partial q_1} &= \boldsymbol{0}_{2\times 2} \\ \frac{\partial \boldsymbol{B}(q_2)}{\partial q_2} &= \begin{bmatrix} 0 & -Lm_2\sin(q_2) \\ -Lm_2\sin(q_2) & 0 \end{bmatrix}\end{aligned} \tag{3.19}$$

The first term in the function $\boldsymbol{f}_p(\boldsymbol{p}, \boldsymbol{q})$ from (3.14) can now be found as

$$\begin{aligned}&\frac{1}{2}\boldsymbol{p}^T\boldsymbol{B}^{-1}(q_2)\frac{\partial \boldsymbol{B}(q_2)}{\partial \boldsymbol{q}}\boldsymbol{B}^{-1}(q_2)\boldsymbol{p} \\ =&\frac{1}{2}\dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{B}(q_2)}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \frac{1}{2}\begin{bmatrix} \dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{B}(q_2)}{\partial q_1} \\ \dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{B}(q_2)}{\partial q_2} \end{bmatrix}\dot{\boldsymbol{q}} \\ =&\frac{1}{2}\begin{bmatrix} 0 & 0 \\ -Lm_2\sin(q_2)\dot{q}_2 & -Lm_2\sin(q_2)\dot{q}_1 \end{bmatrix}\dot{\boldsymbol{q}} \\ =&\begin{bmatrix} 0 \\ -Lm_2\sin(q_2)\dot{q}_1\dot{q}_2 \end{bmatrix}\end{aligned} \tag{3.20}$$

The second term in $\boldsymbol{f}_p(\boldsymbol{p}, \boldsymbol{q})$, i.e., the restoring forces are found from the expression for the potential energy. Recall from example 2.2 that we found an expression for the potential energy differentiated with respect to the generalized coordinates. Recall also from the section above that this corresponds to the vector of restoring forces. Thus

$$\boldsymbol{g}(\boldsymbol{q}) = \begin{bmatrix} k_1 q_1 \\ -Lm_2 g\sin(q_2) + k_2 q_2 \end{bmatrix} \tag{3.21}$$

Inserting (3.20) and (3.21) in (3.14), we find

$$\boldsymbol{f}_p(\boldsymbol{p}, \boldsymbol{q}) = \begin{bmatrix} -k_1 q_1 \\ L m_2 \sin(q_2)\,(g - \dot{q}_1 \dot{q}_2) - k_2 q_2 \end{bmatrix} \tag{3.22}$$

Using that that the inverse mass matrix is

$$\boldsymbol{B}^{-1}(q_2) = \frac{1}{m_2(\cos^2(q_2) - 1) - m_1} \begin{bmatrix} -1 & \frac{\cos(q_2)}{L} \\ \frac{\cos(q_2)}{L} & \frac{m_1 + m_2}{m_2 L^2} \end{bmatrix} \tag{3.23}$$

and writing out the state space formulation (3.15), we find

$$\begin{aligned} \dot{q}_1 &= -\frac{p_1}{m_2(\cos^2(q_2) - 1) - m_1} + \frac{\cos(q_2) p_2}{L m_2(\cos^2(q_2) - 1) - L m_1} \\ \dot{q}_2 &= \frac{\cos(q_2) p_2}{L m_2(\cos^2(q_2) - 1) - L m_1} - \frac{(m_1 + m_2) p_2}{L^2 m_2(\cos^2(q_2) - 1) - L^2 m_1} \\ \dot{p}_1 &= -k_1 q_1 + \tau_1 - \tau_{d1} \\ \dot{p}_2 &= L m_2 \sin(q_2)\,(g - \dot{q}_1 \dot{q}_2) - k_2 q_2 + \tau_2 - \tau_{d2} \end{aligned} \tag{3.24}$$

where $p_1$ and $p_2$ are the generalized momentums such that $\boldsymbol{p} = [p_1,\, p_2]^T$. The damping force are in this example included as $\tau_{d1} = d_1 \dot{q}_1$ and $\tau_{d2} = d_2 \dot{q}_2$ in order to simplify the notation. As an alternative, the expressions for $\dot{q}_1$ and $\dot{q}_2$ could be substituted for the right hand side of $\dot{\boldsymbol{q}} = \boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{p}$.

This example is concluded with the algorithm 3.1 for finding the mass matrix and its derivatives with respect to the generalized coordinates in Maple 17. The algorithm is a continuation of algorithm 2.1, presented in example 2.4.

**Algorithm 3.1** Mass matrix of inverted pendulum on wagon: Maple 17

```
#System Mass Matrix
J1T := Transpose(J1):
J2T := Transpose(J2):
M1 := m1*IdentityMatrix(3):
M2 := m2*IdentityMatrix(3):
B1 := J1T M1 J1:
B2 := J2T M2 J2:
B = B1 + B2:

#Differentiate wrt. q
dBdq1 := map(diff, B, q1):
dBdq2 := map(diff, B, q2):
```

**Discussion on the state space model**

It might be of interest to compare the state space model (3.15) to a form that some readers will find more familiar. In particular, it might be of interest to investigate how the various terms relate to each other. In general, systems of rigid bodies in spatial motion can be modelled on the form presented in e.g. Sciavicco and Siciliano (2000) for manipulators, or in Fossen (2011) for marine vehicles. This model takes the form

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{3.25}$$

where all non-conservative efforts such as friction and actuator forces are included in the vector $\boldsymbol{\tau}$. The vector $\boldsymbol{g}(\boldsymbol{q})$, we recognize as the restoring forces, and the term $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ are the Coriolis and centrifugal forces.

This model can be presented in state space form as

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{\nu} \\ \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{\nu}} &= -\boldsymbol{C}(\boldsymbol{q}, \boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau} \end{aligned} \tag{3.26}$$

The generalized momentum of the system is the mass-inertia matrix multiplied by the rate of the generalized coordinates, i.e. $\boldsymbol{p} = \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}$. Then the generalized momentum can then be differentiated with respect to time, in order to obtain

$$\begin{aligned} \dot{\boldsymbol{p}} &= \frac{d}{dt}\left(\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}\right) = \boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{B}}(\boldsymbol{q})\dot{\boldsymbol{q}} \\ &= \boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T \frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} \end{aligned} \tag{3.27}$$

where the chain rule of differentiation is used in the last term. Substituting this equation for $\dot{\boldsymbol{p}}$ in (3.15), inserting (3.14) for $\boldsymbol{f}_p$, and substituting $\boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{p} = \dot{\boldsymbol{q}}$, we get

$$\begin{aligned} \boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T \frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} &= \frac{1}{2}\dot{\boldsymbol{q}}^T \frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau} \\ \Rightarrow \boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} &= -\frac{1}{2}\dot{\boldsymbol{q}}^T \frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau} \end{aligned} \tag{3.28}$$

By comparing (3.28) to (3.26), we see that the term we obtained from differentiating the kinetic energy with respect to the generalized coordinates, is in fact the Coriolis and centrifugal forces $\boldsymbol{C}(\boldsymbol{q}, \boldsymbol{\nu})\boldsymbol{\nu}$.

**Example 3.3.** In this example, the state space model found in example 3.2 for the moving wagon with an inverted pendulum, is transformed to the form of (3.26).

The mass matrix of the system was found as

$$\boldsymbol{B}(q_2) = \begin{bmatrix} m_1 + m_2 & Lm_2\cos(q_2) \\ Lm_2\cos(q_2) & L^2 m_2 \end{bmatrix} \tag{3.29}$$

Recall also that the Coriolis and centrifugal forces can be found from differentiating the kinetic energy with respect to the generalized coordinates. We then have

$$\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\nu})\boldsymbol{\nu} = \frac{1}{2}\boldsymbol{\nu}^T \frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}}\boldsymbol{\nu} \tag{3.30}$$

where $\boldsymbol{\nu} = \dot{\boldsymbol{q}}$. In example 3.2, in (3.20), we found the kinetic energy differentiated with respect to the generalized coordinates. Combining this with the preceding equation, we find

$$\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\nu})\boldsymbol{\nu} = \frac{1}{2}\left[\begin{array}{cc} 0 & 0 \\ -Lm_2\sin(q_2)\nu_2 & -Lm_2\sin(q_2)\nu_1 \end{array}\right]\left[\begin{array}{c} \nu_1 \\ \nu_2 \end{array}\right] \tag{3.31}$$

where $\nu_1 = \dot{q}_1$, and $\nu_2 = \dot{q}_2$. The restoring forces are already found in (3.21) as

$$\boldsymbol{g}(\boldsymbol{q}) = \left[\begin{array}{c} k_1 q_1 \\ -Lm_2 g\sin(q_2) + k_2 q_2 \end{array}\right] \tag{3.32}$$

The state space form (3.26) can now be found by inserting (3.29), (3.31), and (3.32), into (3.26).

## Lagrange Hamiltonian Equations for Quasi-Coordinates

Recall that the Lagrangian equations of motion for quasi-coordinates could be expressed as

$$\frac{d}{dt}\left(\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}}\right) + \boldsymbol{\beta}^T\boldsymbol{\gamma}\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} - \boldsymbol{\beta}^T\frac{\partial \bar{T}}{\partial \boldsymbol{q}} + \boldsymbol{\beta}^T\frac{\partial V}{\partial \boldsymbol{q}} = \boldsymbol{\beta}^T\boldsymbol{\tau} \tag{3.33}$$

In order to get these equations in the Lagrange-Hamiltonian form, we define the momentum associated to the quasi-coordinates as

$$\boldsymbol{p} = \frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} \tag{3.34}$$

Inserting (3.34) into (3.33) and rearranging yields

$$\dot{\boldsymbol{p}} = -\boldsymbol{\beta}^T\boldsymbol{\gamma}\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} + \boldsymbol{\beta}^T\frac{\partial \bar{T}}{\partial \boldsymbol{q}} - \boldsymbol{\beta}^T\frac{\partial V}{\partial \boldsymbol{q}} + \boldsymbol{\beta}^T\boldsymbol{\tau} \tag{3.35}$$

Recall that the kinetic energy for a system of rigid bodies could be expressed in terms of the generalized coordinates and the time rate of the generalized coordinates as

$$T(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.36}$$

Making the substitution $\dot{\boldsymbol{q}} = \boldsymbol{\beta}\boldsymbol{\omega}$ in (3.36) yields

$$T(\boldsymbol{q},\boldsymbol{\beta}\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\beta}^T\boldsymbol{B}(\boldsymbol{q})\boldsymbol{\beta}\boldsymbol{\omega} \tag{3.37}$$

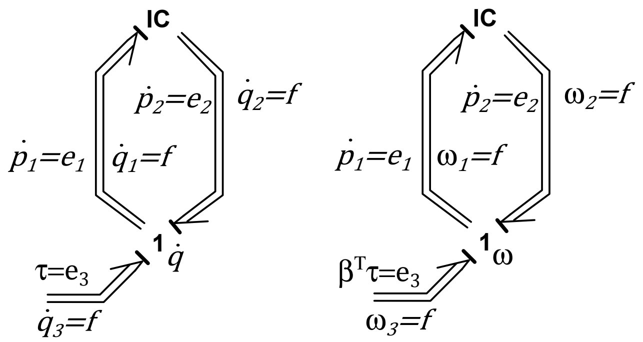


Figure 3.1: Basic bond graph implementations of the Lagrangian equations of motion for both generalized and quasi-coordinates.

By now defining the mass-inertia matrix for quasi-coordinates as

$$\bar{\boldsymbol{B}}(\boldsymbol{q}) = \boldsymbol{\beta}^T \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\beta} \tag{3.38}$$

we can define the kinetic energy in terms of quasi-coordinates as

$$\bar{T}(\boldsymbol{q},\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^T \bar{\boldsymbol{B}}(\boldsymbol{q})\boldsymbol{\omega} \tag{3.39}$$

By combining (3.39) and (3.34), we find an expression for the momentum associated to the quasi-coordinates as

$$\boldsymbol{p} = \bar{\boldsymbol{B}}(\boldsymbol{q})\boldsymbol{\omega} \tag{3.40}$$

Rearranging this equation yields an expression for the quasi-coordinates. Combining this with (3.35) give the state space model

$$\begin{aligned}
\boldsymbol{\omega} &= \bar{\boldsymbol{B}}^{-1}(\boldsymbol{q})\boldsymbol{p} \\
\dot{\boldsymbol{p}} &= -\boldsymbol{\beta}^T\boldsymbol{\gamma}\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} + \boldsymbol{\beta}^T\frac{\partial \bar{T}}{\partial \boldsymbol{q}} - \boldsymbol{\beta}^T\frac{\partial V}{\partial \boldsymbol{q}} + \boldsymbol{\beta}^T\boldsymbol{\tau} \\
&= -\boldsymbol{\beta}^T\boldsymbol{\gamma}\bar{\boldsymbol{B}}\boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\omega}^T\frac{\partial \bar{\boldsymbol{B}}}{\partial \boldsymbol{q}}\boldsymbol{\omega} - \boldsymbol{\beta}^T\boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\beta}^T\boldsymbol{\tau} \\
&= \bar{\boldsymbol{f}}_p(\boldsymbol{q},\boldsymbol{\omega}) + \boldsymbol{\beta}^T\boldsymbol{\tau}
\end{aligned} \tag{3.41}$$

## 3.2 Bond Graph Implementation

We now implement (3.15) and (3.41) in basic bond graphs, and show how, in a conceptual manner, the basic bond graphs can be extended to provide useful interfaces to the basic model.

We start by placing a 1-junction representing $\dot{\boldsymbol{q}} = \boldsymbol{f}$ in the case of generalized coordinates, or $\boldsymbol{\omega} = \boldsymbol{f}$, in the case of quasi-coordinates, and an IC-field, as seen in figure 3.1. The constitutive relations of the 1-junction are

$$\begin{aligned} \dot{\boldsymbol{q}}_1 &= \dot{\boldsymbol{q}}_2 = \dot{\boldsymbol{q}}_3 = \dot{\boldsymbol{q}} = \boldsymbol{f} \\ \dot{\boldsymbol{p}}_1 &= \dot{\boldsymbol{p}}_2 + \boldsymbol{\tau} \end{aligned} \tag{3.42}$$

if generalized coordinates are used, and

$$\begin{aligned} \boldsymbol{\omega}_1 &= \boldsymbol{\omega}_2 = \boldsymbol{\omega}_3 = \boldsymbol{\omega} = \boldsymbol{f} \\ \dot{\boldsymbol{p}}_1 &= \dot{\boldsymbol{p}}_2 + \boldsymbol{\beta}^T \boldsymbol{\tau} \end{aligned} \tag{3.43}$$

if quasi-coordinates are used. Note that the subscripts in this case refer to the number of the power bond, not to the $i$-th element of the vectors. Note also that in the following, we shall assume that we use generalized coordinates, and not quasi-coordinates. The method presented is however similar in both cases, and examples of quasi-coordinate implementation is given in both section 4.2 and in the appendix A.1.

By studying the causality of these bond graph, we see that power bond 1 sets the effort in the IC-field, while power bond 2 sets the flow. As such, we can consider the effort $\dot{\boldsymbol{p}}_1$ and the flow $\dot{\boldsymbol{q}}_2$ as inputs to the element, and $\dot{\boldsymbol{p}}_2$ and $\dot{\boldsymbol{q}}_1$ as outputs. Then the constitutive relation of the IC-field can be defined as

$$\begin{aligned} \dot{\boldsymbol{q}}_1 &= \boldsymbol{B}^{-1}(\boldsymbol{q}_2)\boldsymbol{p}_1 \\ \dot{\boldsymbol{p}}_2 &= \boldsymbol{f}_p(\boldsymbol{p}_1, \boldsymbol{q}_2) \end{aligned} \tag{3.44}$$

By comparing (3.42) and (3.44), we recognize the Lagrangian equations of motion in state space form as

$$\begin{aligned} \dot{\boldsymbol{q}}_1 &= \dot{\boldsymbol{q}} = \boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{p} \\ \dot{\boldsymbol{p}}_1 &= \dot{\boldsymbol{p}} = \dot{\boldsymbol{p}}_2 + \boldsymbol{\tau} = \boldsymbol{f}_p(\boldsymbol{p}, \boldsymbol{q}) + \boldsymbol{\tau} \end{aligned} \tag{3.45}$$

The bond graph in figure 3.1 does not add much functionality to the system (3.15). In both the bond graph implementation and in (3.15), any subsystems interfacing the system of bodies, must be in the form of generalized forces. One benefit of the bond graph implementation is however that the system providing the generalized forces can be drawn directly in the bond graph, something which certainly is convenient if the system is e.g. a multidisciplinary actuator system.

In order to make the bond graph in figure 3.1 more versatile, recall that the velocity of any point $p$ on the system of rigid bodies can be expressed as functions of the generalized coordinates and their time derivatives as $\boldsymbol{\nu}_p = \boldsymbol{J}_(\boldsymbol{q})\dot{\boldsymbol{q}}$. Thus, using the

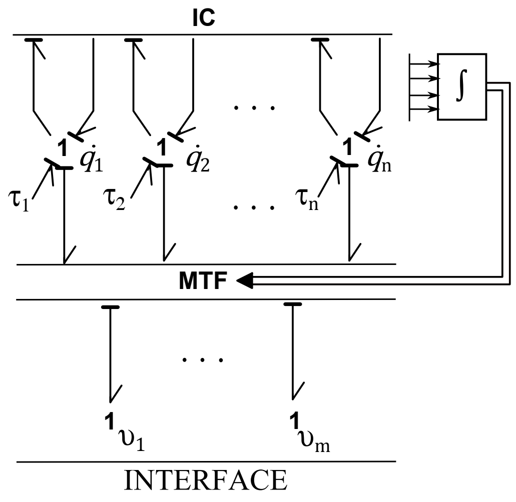


Figure 3.2: Extended bond graph implementation of the Lagrangian equations of motion for a system of bodies in spatial motion.

modulated transformer element with $\boldsymbol{q}$ as input, and the constitutive relations

$$\begin{aligned}\boldsymbol{\nu}_p &= \boldsymbol{J}_p(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} &= \boldsymbol{J}_p^T(\boldsymbol{q})\boldsymbol{e}_p\end{aligned} \tag{3.46}$$

where $\boldsymbol{e}_p$ is the output effort, we can place 1-junctions representing the velocity $\boldsymbol{\nu}_p$ at the point $p$. This provides an interface to any systems interfacing the real system at the point $p$. Figure 3.2 show schematically how this can be done. Note that we have also here separated the generalized coordinates into individual 1-junctions. This might be beneficial if different kinds of systems interface the bond graph model trough the generalized forces.

## 3.3 An Effective Method for Computer Implementation

The state space model (3.15) is well suited for computer implementation, either through bond graph software, using the bond graph implementation from section 3.2, or through some ordinary differential equation (ODE) solver in e.g. *Matlab*. For complicated systems however, it is convenient to find an effective method for producing the system equations. It is also important in some applications that the implementation is computationally efficient. In this section, such a method is proposed.

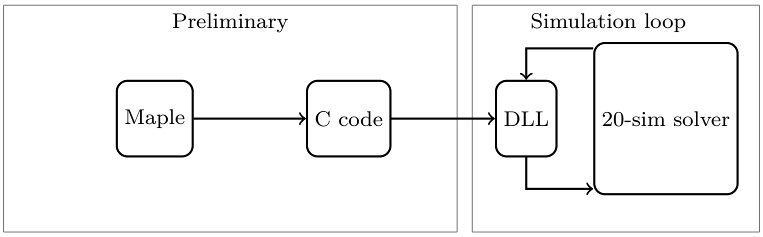


Figure 3.3: Diagram showing general layout of implementation.

### 3.3.1 General Layout

For complicated systems the elements of the mass matrix $\boldsymbol{B}(\boldsymbol{q})$ tends to become large expressions, and as seen from (3.11), the partial derivative of the mass-inertia matrix, with respect to the generalized coordinates, need to be calculated. Furthermore, the restoring forces $\boldsymbol{g}(\boldsymbol{q})$, which was the partial derivative or the potential energy with respect to the generalized coordinates, also need to be found. To this effect it is convenient to use symbolic software such as *Maple*. In the following, we propose a method that is efficient both in way of producing the dynamic model, and computationally efficient. The general layout of the method is shown in figure 3.3. The symbolic software Maple is utilized in order to find symbolic expressions for the system mass inertia matrix, its derivatives with respect to each of the generalized coordinates, and the restoring forces. Then C-codes for calculating these expressions can be auto-generated in Maple. This C-code can then, after being restructured somewhat, be compiled as a dynamic link library file (DLL). The functions within the DLL file can the be accessed as external functions from the bond graph software 20-sim in order to update the variables in question for each time step in the numerical integration.

### 3.3.2 Software

Before going into details about the implementation, short introductions on the software used, and its relevance in this thesis, are provided.

#### Maple

Maple is a commercial computer program, developed by Maplesoft, a division in Waterloo Maple Inc. This program can be used in order to, among other things, perform symbolic calculations on a computer. Maple also offer a code generation package, which allows for automatic code generation in a variety of languages, among which are C. The code generation package also enables code optimization such that the auto generated code is optimized with respect to computational efficiency (Maplesoft, 2014). Examples on how Maple is useful in relation to the work of this thesis are already presented in the algorithms 2.1 and 3.1. In the example 3.5 we shall utilize Maple further.

#### C

C is a relatively low-level programming language, meaning that it mainly deals with characters, numbers and addresses, rather than objects such as strings, lists and sets (Kernighan, 1988). The main purpose of C, in the context of this thesis, is to define a set of functions that update the relevant expressions during model simulations. These functions are programmed in C, and from the C source code, DLL-files are built, from which the bond graph software access the functions.

#### Cygwin

Cygwin is an open source Linux-like environment for Windows (Hunt and Turney, 2000). Among other things, it provides the famous GCC-compiler for windows users. This is relevant for the work presented here because this compiler possess the capability of building object files from c-code, and DLL files from object files.

#### 20-sim

20-sim is a modelling and simulation program, where bond graphs, block diagrams or combinations of those can be drawn directly in the program graphical editor(Klein et al., 2013). The program provides the standard bond graph elements, as well as a variety of signalling blocks. These can be placed directly in the graphical editor as sub models. Every sub model can in turn be composed of lower level sub models. At the bottom of any hierarchy of sub models, is a set of equations. These are implemented through the SIDOPS+ programming language.

### 3.3.3 Symbolic Expressions

Recall the Lagrange Hamiltonian state space models found in section 3.1 for generalized coordinates and quasi-coordinates. Consider now that one of these models are to be implemented, as described in section 3.2, and simulated in a bond graph. In both cases, the mass-inertia matrix needs to be updated for each time step. Furthermore, the partial derivatives of the mass inertia matrix with respect to each of the generalized coordinates, as well as the vector of restoring forces, i.e., the potential energy expression differentiated with respect to each of the generalized coordinates, needs to be updated for each time step. If quasi-coordinates are used, one does in addition need to update the matrices $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$. The algorithms 2.1 and 3.1 demonstrated, through the inverted pendulum on moving wagon example, how symbolic expressions for the mass-inertia matrix as well as its partial derivatives with respect to the generalized coordinates can be found. We have however yet to see how to find symbolic expressions for the potential energy differentiated with respect to the generalized coordinates, i.e., the restoring forces, and how to export the relevant expressions to optimized C-code.

When exporting expressions from Maple as C-code, a series of choices must be made. First, the general structure of the resulting C-code must be considered. In this work, it is decided to define a separate function for each of the elements of the various matrices and vectors. At a later stage, functions who interface the bond graph software, and in turn call the relevant functions in order to calculate the values of each element of the desired vectors and matrices, are defined. These functions are presented in section 3.3.4. An other thing to consider is whether to export a symbolic expression for the mass-inertia matrix or the inverted mass-inertia matrix. If the former is chosen, the matrix must be inverted numerically in the bond graph software for each time step. The latter choice however, is in some cases not practical because of the complexity of the symbolic expressions for the elements of the mass matrix. In this work it is chosen to numerically invert the matrix on-line in the bond graph software.

Finally, it is worth while to remember that the mass inertia matrix is symmetric. It is then necessary to export only the upper triangular elements of the matrix. According to proposition 3.4, this is also true for the matrices resulting from differentiation the mass matrix with respect to the generalized coordinates.

**Proposition 3.4.** *The $n \times n$ matrix $\frac{\partial \boldsymbol{B}}{\partial q_i}$, where $\boldsymbol{B}$ is a system mass inertia matrix, and $q_i$ is the $i$-th generalized coordinate, is symmetric for $i \in [1, n]$*

*Proof.* For any element $b_{jk}$ in the symmetric matrix $\boldsymbol{B}$, the following holds

$$b_{jk} = b_{kj} \tag{3.47}$$

It then follows that

$$\frac{\partial b_{jk}}{\partial q_i} = \frac{\partial b_{kj}}{\partial q_i} \tag{3.48}$$

which proves that the matrix $\frac{\partial \boldsymbol{B}}{\partial q_i}$ is symmetric. □

**Example 3.5.** We again consider the inverted pendulum on the moving wagon. The purpose of this example is to show how Maple 17 can be utilized in order to export the expressions $\boldsymbol{B}(q_2)$, $\partial \boldsymbol{B}(q_2)/\partial q_2$, and $\boldsymbol{g}(\boldsymbol{q})$. Recall that the matrix $\partial \boldsymbol{B}(q_2)/\partial q_1 = \boldsymbol{0}$. We therefore do not export this. In algorithm 3.1, the two first of these expressions were found. Algorithm 3.2 show how to find symbolic expressions for the restoring forces, while algorithm 3.3 show how to export the to optimized C-code.

**Algorithm 3.2** Restoring forces for inverted pendulum on wagon: Maple 17

```
#Potential Energy
G:=Vector([0,0,-g])#acceleration of gravity
V1:= -m1*G^%T r0_1_0 + 1/2*k1*q1^2:
V2:= -m2*G^%T r0_cg2_0 + 1/2*k2*q2^2:
V := V1 + V2:

#Differentiate wrt. q
g1:= diff(V,q1):#first element of restoring vector
g2:= diff(V,q2):#second element of restoring vector
```

**Algorithm 3.3** Export symbolic expressions to C-code: Maple 17

```
#Export mass matrix
for i from 1 to 2 do
    for j from i to 2 do
        F:= makeproc(B[i,j]): #Make proclamation in order to output a function
        C(F,resultname=cat("b",i,j),output="inv_pend.txt",optimize):
    end do
end do

#Export mass matrix partial derivative wrt. q2
for i from 1 to 2 do
    for j from i to 2 do
        F:= makeproc(dBdq2[i,j]):
        C(F,resultname=cat("dbdq2_",i,j),output="inv_pend.txt",optimize):
    end do
end do

#Export restoring forces
F:= makeproc(g1):
C(F,resultname="g1",output="inv_pend.txt",optimize):
F:= makeproc(g2):
C(F,resultname="g2",output="inv_pend.txt",optimize):
```

### 3.3.4 C-code Structure

In order to adapt the C-code exported from Maple such that it efficiently can communicate with 20-sim, we must first consider the requirements from the external function interface of 20-sim. This interface require that the external functions are defined as

```
int extFunc(double *inarr, int inputs, double *outarr, int outputs, int major)
```

The first argument **inarr* is a pointer to an input array of doubles. The input array have the number *inputs* of elements. Similarly, the third argument **outarr* is an output array of length *outputs*, of doubles. The final argument *major* equals 1 if the integrator in 20-sim performs a major step, and 0 if the integrator performs a minor step (Klein et al., 2013). If the external function returns 0, 20-sim interprets this as a successful call.

The C-code exported from 20-sim is a set of functions for each of the relevant matrices and vectors. As an example, the $n \times n$ mass-inertia matrix is symmetric, such that only the upper triangle of the matrix need to be calculated. The upper triangle consists of $0.5(n^2 + n)$ elements. Thus the C-code consists of $0.5(n^2 + n)$ functions for updating the mass-inertia matrix. In order for the external interface of 20-sim to be able to utilize these functions, a function after the prototype, *extFunc()* above is defined for the purpose of updating the mass inertia matrix. This function, when called from 20-sim, in turn calls each of the $0.5(n^2 + n)$ functions for the individual elements of the system mass-inertia matrix, using the pointers to the arrays of inputs and outputs as argument. Thus, a C-code for updating the mass-inertia matrix of a system of $n$ degrees of freedom can be structured according to

```
// Calculations
//---------------------------------------------
// Update element 1,1 of mass-inertia matrix
int b11 (double *inarr, double *outarr)
{
  Calculate element [1,1] of the mass inertia matrix
  and assign to outarr[0]
  return 0;
}
// Update element 1,2 of mass-inertia matrix
int b12 (double *inarr, double *outarr)
{
  Calculate element [1,2] of the mass inertia matrix
  and assign to outarr[1]
  return 0;
}
    .
    .
    .
// Update element 1,n of mass-inertia matrix
int b1n (double *inarr, double *outarr)
{
  Calculate element [1,n] of the mass inertia matrix
  and assign to outarr[n-1]
  return 0;
}
// Update element 2,2 of mass-inertia matrix
int b22 (double *inarr, double *outarr)
{
  Calculate element [2,2] of the mass inertia matrix
  and assign to outarr[n]
  return 0;
}
    .
    .
    .
// Update element n,n of mass-inertia matrix
int bnn (double *inarr, double *outarr)
{
  Calculate element [n,n] of the mass inertia matrix
  and assign to outarr[0.5(n^2+n)-1]
  return 0;
}
//---------------------------------------------

// Interface 20-sim
//---------------------------------------------
int updateB(double *inarr, int inputs, double *outarr, int outputs, int major)
{
  // Update upper triangular elements of matrix B
  b11(inarr, outarr); //update elemetn [1,1]
  b12(inarr, outarr); //update elemetn [1,2]
      .
      .
      .
  b1n(inarr, outarr); //update elemetn [1,n]
  b22(inarr, outarr); //update elemetn [2,2]
      .
      .
      .
  bnn(inarr, outarr); //update elemetn [n,n]
  return 0; // return success
}
```

As the mass inertia matrix is a function of the generalized coordinates $\boldsymbol{q}$, the array of inputs pointed to by **inarr*, contains the generalized coordinates, such that $innarr[0] = q_1$, $innarr[1] = q_2$, etc.. The output array pointed to by **outarr*, is in the case above the array containing all upper triangular elements of the mass-inertia matrix, $\boldsymbol{B}(\boldsymbol{q})$, such that $outarr[0] = \boldsymbol{B}[1,1]$, $outarr[1] = \boldsymbol{B}[1,2]$, etc.. When the function *updateB* calls e.g. *b11* with the pointer to the input array and the output array, as arguments, the array of inputs, i.e., the generalized coordinates, are used to calculate $\boldsymbol{B}[1,1]$. The result is then stored in $outarr[0]$, from where 20-sim gets the value.

The functions *b11*, *b12*, etc. in the code above, are those exported from Maple 17. The direct export does however need to be edited somewhat in order to obtain the structure presented above. The typical direct export from Maple, for updating a single expression, might have the structure

```
double someExpression(void)
{
  return(some calculations(q1,q2,...,qn));
}
```

Firstly, it is desired that the function takes in the pointers to the arrays of inputs and outputs as arguments. Secondly, the function should use $inarr[0]$, $inarr[1]$, ..., $inarr$[n-1], as opposed to q1, q2, ..., qn as input arguments in the calculations. Third, the function should store the calculated value to the appropriate location in the output array, and return 0. This is achieved by editing the above code to

```
double someExpression(double *inarr, double *outarr)
{
  outarr[i] = some calculations(inarr)
  return 0;
}
```

In order to make these notions somewhat clearer, the inverted pendulum on wagon example is continued.

**Example 3.6.** In this example, the direct output exported by the Maple algorithm 3.3, is presented. Next, the edited C-code, which can be utilized by 20-sim in order to update the mass-matrix, the mass matrix partial derivative with respect to $q_2$, and the vector of generalized forces, is presented. Recall that the mass matrix partial derivative with respect to $q_1$ is identically equal to 0, and is therefore not necessary to update.

The raw export from the Maple algorithm 3.3, only altered by adding three comments, is

```
// Mass matrix
double b11 (void)
{
  return(m1 + m2);
}
double b12 (void)
{
  double t1;
  t1 = cos(q2);
  return(t1 * L * m2);
}
double b22 (void)
{
  double t1;
  double t2;
  double t3;
  double t6;
  double t7;
  t1 = cos(q2);
  t2 = t1 * t1;
  t3 = L * L;
  t6 = sin(q2);
  t7 = t6 * t6;
  return(m2 * t2 * t3 + m2 * t3 * t7);
}
// Mass matrix partial derivative wrt. q2
int dbdq2_11 (void)
{
  return(0);
}
double dbdq2_12 (void)
{
  double t1;
  t1 = sin(q2);
  return(-t1 * L * m2);
}
int dbdq2_22 (void)
{
  return(0);
}
// Restoring forces
double g1 (void)
{
  return(k1 * q1);
}
double g2 (void)
{
  double t2;
  t2 = sin(q2);
  return(-L * g * m2 * t2 + k2 * q2);
}
```

One problem not yet addressed are the parameters of the system. In the direct output, notice that the constant parameters such as $m1$ and $m2$ appear. These are not yet assigned values. Two possible methods for handling this is proposed. First, one might define these parameters in 20-sim, and include them in the array of inputs. This approach allows for flexibility in that it will be easy to alter the parameters from within 20-sim, without recompiling the C-code. An other possibility, which is used in this example, is to code the parameters in various data structures defined in the beginning of the C-code. Structuring this code according to the above, and creating the interfaces to 20-sim, results in the code

```
/*
---------------------------------------------
    inverted_pendulum.c
---------------------------------------------
*/

#include<stdio.h>
#include<math.h>
#define g 9.81
typedef struct
{
  float m1;
  float m2;
  float L;
  float k1;
  float k2;
} PARAM;

// Define constant parameters
//                   m1  m2  L    k1  k2
PARAM parameters = {2,   1,  0.3, 3,  0.5};

// Calculations
//=============================================
// Mass matrix
int b11 (double *inarr, double *outarr)
{
  outarr[0] = parameters.m1 + parameters.m2;
return 0;
 }
int b12 (double *inarr, double *outarr)
{
  double t1;
  t1 = cos(inarr[1]);
  outarr[1] =t1 * parameters.L * parameters.m2;
return 0;
 }
int b22 (double *inarr, double *outarr)
{
  double t1;
  double t2;
  double t3;
  double t6;
  double t7;
  t1 = cos(inarr[1]);
  t2 = t1 * t1;
  t3 = parameters.L * parameters.L;
  t6 = sin(inarr[1]);
  t7 = t6 * t6;
  outarr[2] =parameters.m2 * t2 * t3 + parameters.m2 * t3 * t7;
return 0;
 }
// Mass matrix partial derivative wrt. inarr[1]
int dbdq2_11 (double *inarr, double *outarr)
{
  outarr[0] =0;
return 0;
 }
int dbdq2_12 (double *inarr, double *outarr)
{
  double t1;
  t1 = sin(inarr[1]);
  outarr[1] =-t1 * parameters.L * parameters.m2;
return 0;
 }
int dbdq2_22 (double *inarr, double *outarr)
{
  outarr[2] =0;
return 0;
 }
// Restoring forces
int g1 (double *inarr, double *outarr)
{
  outarr[0] =parameters.k1 * inarr[0];
return 0;
 }
int g2 (double *inarr, double *outarr)
{
  double t2;
  t2 = sin(inarr[1]);
  outarr[1] =-parameters.L * g * parameters.m2 * t2 + parameters.k2 * inarr[1];
return 0;
 }
//==========================================

// 20-sim interface
//==========================================
int updateB(double *inarr, int inputs, double *outarr, int outputs, int major)
{
  //update upper triangular elements of mass matrix
  b11(inarr, outarr);
  b12(inarr, outarr);
  b22(inarr, outarr);
  return 0; // return success
}
int update_dBdq2(double *inarr, int inputs, double *outarr, int outputs, int major
    )
{
  //update upper triangular elements of dB/dq2
  dbdq2__11(inarr, outarr);
  dbdq2__12(inarr, outarr);
  dbdq2__22(inarr, outarr);
  return 0; // return success
}
int update_g(double *inarr, int inputs, double *outarr, int outputs, int major)
{
  //update restoring force vector
  g1(inarr, outarr);
  g2(inarr, outarr);
  return 0; // return success
}
```

### Building dynamic link library file in Cygwin

The C-code for updating the various expressions, can be used in order to build a DLL-file. Using the Cygwin software, this is achieved by using the commands

```
>> gcc -c c_code_name.c
>> gcc -shared -o dll_name.dll c_code_name.o
```

where *c_code_name.c* is the name of the file containing the C-code, and *dll_name.dll* is the name of the resulting DLL-file.

### 3.3.5 20-sim Implementation

Recall the figure 3.1, where the basic dynamics of some system of rigid bodies is implemented in bond graph, either based on generalized coordinates or quasi-coordinates. Recall also the constitutive relation for the IC-field stated in (3.44), for generalized coordinates. The input effort, denoted $\dot{\boldsymbol{p}}_1$ in (3.44), shall now be denoted $\boldsymbol{e}_{in}$, and the output effort is $\boldsymbol{e}_{out}$. Furthermore, the input flow, denoted $\dot{\boldsymbol{q}}_2$ in (3.44), shall now be denoted $\boldsymbol{f}_{in}$, and the output flow $\boldsymbol{f}_{out}$. Writing out the constitutive relation for the IC-field, we now obtain

$$\begin{aligned} \boldsymbol{q} &= \int_{\tau=0}^{t} \boldsymbol{f}_{in} d\tau \\ \boldsymbol{f}_{out} &= \boldsymbol{B}^{-1}(\boldsymbol{q}) \int_{\tau=0}^{t} \boldsymbol{e}_{in} d\tau \\ \boldsymbol{e}_{out} &= \frac{1}{2}\boldsymbol{f}_{in}^T \frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}} \boldsymbol{f}_{in} - \boldsymbol{g}(\boldsymbol{q}) \end{aligned} \tag{3.49}$$

if the generalized coordinates approach is used. For the quasi coordinate approach we get

$$\begin{aligned} \boldsymbol{q} &= \int_{\tau=0}^{t} \boldsymbol{f}_{in} d\tau \\ \boldsymbol{f}_{out} &= \bar{\boldsymbol{B}}^{-1}(\boldsymbol{q}) \int_{\tau=0}^{t} \boldsymbol{e}_{in} d\tau \\ \boldsymbol{e}_{out} &= -\boldsymbol{\beta}^T \boldsymbol{\gamma} \bar{\boldsymbol{B}} \boldsymbol{f}_{in} + \frac{1}{2}\boldsymbol{\beta}^T \boldsymbol{f}_{in}^T \frac{\partial \bar{\boldsymbol{B}}}{\partial \boldsymbol{q}} \boldsymbol{f}_{in} - \boldsymbol{\beta}^T \boldsymbol{g}(\boldsymbol{q}) \end{aligned} \tag{3.50}$$

Exactly which expression to implement in the DLL-file, and which to code directly in the underlying code of the IC-field depends on both the system and the preferences of the modeller. In general however, for complex systems, the expressions for the mass-inertia matrix $\boldsymbol{B}(\boldsymbol{q})$, and its partial derivative with respect to the generalized coordinates, are large and complex, and convenient to generate in Maple, and update trough the DLL-file. Often this is also true for the restoring forces. For quasi-coordinates, the matrix $\boldsymbol{\gamma}$ is also typically convenient to generate and implement in this manner. In the following, it is assumed that generalized coordinates are used, and that the mass-inertia matrix, its partial derivatives with respect to the generalized coordinates, and the restoring forces are updated in 20-sim through the DLL-file *sim_lib.dll*. This library provides the function *updateB* for updating the mass inertia matrix, *update_dBdq1* for updating the partial derivative of the mass-inertia matrix with respect to the first generalized coordinate, etc., and the function *update_g* for updating the vector of restoring forces. In the underlying code of the IC-field, these functions can be accessed through the commands

```
f_in = p2.f; //input flow
// assign values to array of inputs for the external functions
q = int(f_in); // integrate input velocity to obtain generalized coordinates
x = q;
// update elements of mass-inertia matrix
B_out = dll('sim_lib.dll','updateB',x);
// update elements of partial derivative of mass-inertia matrix wrt q1
dBdq1_out = dll('sim_lib.dll','update_dBdq1',x);
        ...
// update elements of partial derivative of mass-inertia matrix wrt q2
dBdqn_out = dll('sim_lib.dll','update_dBdqn',x);
// update elements of restoring vector
g_out = dll('sim_lib.dll','update_g',x);
```

where the arrays *B_out*, *dBdq1_out*, *dBdqn_out*, and *g_out*, are those pointed to by the pointer **outarr* in each command. Similarly, the pointer *inarr* points to the array $x$. The output arrays can now be assigned to the matrices and vectors in question as

```
// assign elements of output array B_out to mass-inertia matrix
for i = 1 to n do // n is number of degrees of freedom
  for j = i to n do
    B[i,j] = B_out[(j)+(n-1)*(i-1)-(i-1)*(i-2)/2];
    B[j,i] = B_out[(j)+(n-1)*(i-1)-(i-1)*(i-2)/2];
  end;
end;
k = 1;
// assign elements of output array dBdq1_out to partial derivative of mass-
     inertia matrix
for i = 1 to n do // n is number of degrees of freedom
  for j = i to n do
    dBdq1[i,j] = dBdq1_out[j+(n-1)*(i-1)-(i-1)*(i-2)/2];
    dBdq1[j,i] = dBdq1_out[j+(n-1)*(i-1)-(i-1)*(i-2)/2];
  end;
end;

...

//restoring forces
g = g_out;
```

where the expression

$$j + (n-1)(i-1) - \frac{(i-1)(i-2)}{2} \tag{3.51}$$

calculates the correct index. After the matrices and vectors are updated, the constitutive relations can be implemented as

```
e_in = p1.e; //input effort
// state equation for generalized velocity
p1.f = inverse(B)int(e_in); // assign output flow
// Coriolis and centrifugal matrix
C[1,1:n] = transpose(f_in)*dBdq1;
C[2,1:n] = transpose(f_in)*dBdq2;
        ...
C[n,1:n] = transpose(f_in)*dBdqn;
// state equation for generalized momentum
p2.e = C*f_in - g; //assign value to the output effort
```

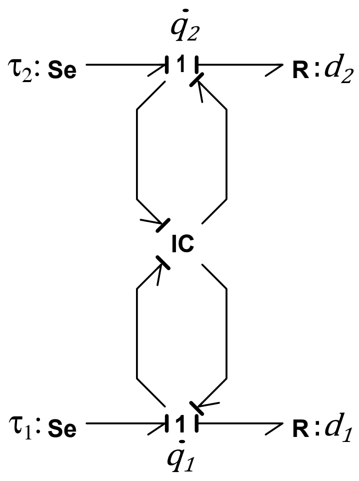


Figure 3.4: IC-field implementation of inverted pendulum on moving wagon.

where *p1.e* and *p2.e* are the input and output efforts of the IC-field, and *p2.f* and *p1.f* are the input and output velocities.

**Example 3.7.** Recall the bond graph implementation of the inverted pendulum on the moving wagon in example 2.3. We now develop a new bond graph implementation for the system, according to the method presented above. Figure 3.4 show the new bond graph for the system.

The underlying code for the IC-field is presented in algorithm 3.4. This code utilize the DLL-file *sim_lib.dll*, which is built from the final C-code of example 3.6.

**Algorithm 3.4** IC-field code: 20-sim.

```
parameters
  string dll_name = 'sim_lib.dll';
variables
  real x[2]; // array of inputs arguments to dll−functions
  real B_out[3]; // array of outputs from external function to update mass matrix
  real dBdq2_out[3]; // array of outputs from external function to update partial
       derivative of mass matrix wrt. q2
  real g_out[2]; // array of outputs from external function to update generalized
       forces
  real B[2,2]; // mass matrix
  real dBdq1[2,2]; //partial derivative of mass matrix with respect to q1
  real dBdq2[2,2]; //partial derivative of mass matrix with respect to q2
  real g[2];  //Vector of testoring forces
  real C[2,2]; // coriolis and centrifugal matrix
  real i, j; // counters
  real f_in[2]; // input velocities
  real f_out[2]; // output velocities
  real e_in[2]; // input efforts
  real e_out[2]; // output efforts
initialequations
  // partial derivative of mass matrix with respect to q1 predefined
  // as it is always equal to 0
  dBdq1 = 0;
equations
  // collect input efforts and flows
  f_in[1] = pw_2.f; //flow on second bond connected to 1−junction of wagon
  f_in[2] = pp_2.f; //flow on second bond connected to 1−junction of pendulum
  e_in[1] = pw_1.e; //effort on first bond connected to 1−junction of wagon
  e_in[2] = pp_1.e; //effort on first bond connected to 1−junction of pendulum
  // assign values to array of inputs to dll functions
  x = int(f_in); //integrate input flow to obtain generalized coordinates
  // update mass matrix
  B_out = dll(dll_name, 'updateB',x);
  for i = 1 to 2 do
    for j = i to 2 do
      B[i,j] = B_out[(j)+(2−1)∗(i−1)−(i−1)∗(i−2)/2];
      B[j,i] = B_out[(j)+(2−1)∗(i−1)−(i−1)∗(i−2)/2];
    end;
  end;
  // update parital derivative of mass matrix with respect to q2
  dBdq2_out = dll(dll_name, 'update_dBdq2',x);
  for i = 1 to 2 do
    for j = i to 2 do
      dBdq2[i,j] = dBdq2_out[i+j −1];
      dBdq2[j,i] = dBdq2_out[i+j −1];
    end;
  end;
  // update restoring force vector
  g_out = dll(dll_name,'update_g',x);
  g = g_out;
  // state equations for generalized coordinates
  f_out = inverse(B) ∗ int(e_in);
  // coriolis and centripeta matrix
  C[1,1:2] = 0.5∗transpose(f_in)∗dBdq1;
  C[2,1:2] = 0.5∗transpose(f_in)∗dBdq2;
  // state equations for generalized momentum
  e_out = C∗f_in − g;

  // assign values to outputs
  pw_1.f = f_out[1]; //flow on 1st bond from 1−junction of wagon
  pp_1.f = f_out[2]; //flow on 1st bond from 1−junction of eondulum
  pw_2.e = e_out[1]; //effort on 2nd bond from 1−junction of wagon
  pp_2.e = e_out[2]; //effort on 2nd bond from 1−junction of pendulum
```

# Chapter 4

# Case Studies

In the previous chapter, an effective method for producing and implementing dynamic models of systems of rigid bodies in spatial motion was presented. In this chapter two applications of this method is demonstrated through two case studies. These case studies will both serve as examples on the theory presented in the preceding chapters, as well as show how this bond graph approach is convenient in that advanced models of subsystems easily and seamlessly can be connected to the basic dynamics. In the first case study, a simulator for the Titan 4 manipulator with hydraulic actuators is developed, while in the second case study, a simulation model of a remotely operated vehicle with a manipulator is created.

## 4.1 Case Study 1 - Titan 4 Simulator

In this case study, the modelling framework developed in this masters thesis is utilized in order to create a simulator for the Titan 4 manipulator. The simulator takes user input from the *Logitech Extreme 3D Pro* joystick, which is used in order to control each of the six manipulator joints. The response of the manipulator is displayed for the user as a 3D animation through the 20-sim 3D animation toolbox.

The Titan 4 manipulator is a seven degrees of freedom manipulator with an open chain of linked bodies structure. As can be seen from figure 4.1, six of the degrees of freedom corresponds to lower pair revolute joints, whereas the seventh corresponds to the gripping motion of the claw, i.e., the end effector. In the simulator, we shall not be concerned with the end effector gripping motion, and as such view the system as

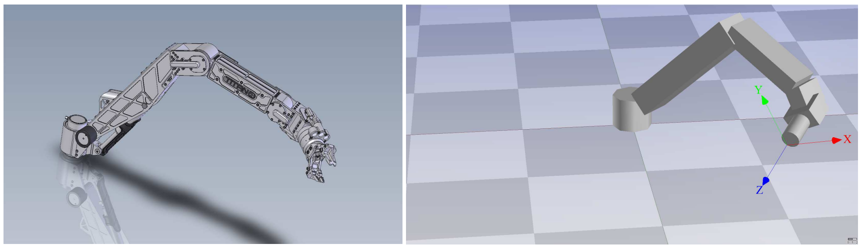


Figure 4.1: The Titan 4 manipulator. To the left is a CAD-figure, and to the right is a screen shot from the 20-sim 3D animation toolbox.

a six degrees of freedom system. The six manipulator joints of interest, are actuated by a hydraulic system with five hydraulic motors and one hydraulic linear actuator, i.e., a piston. The linear actuator is connected to the second joint from the base. Each actuator is controlled by a servo valve.

In the following section, the basic dynamics of the system is modelled according to the method presented in this thesis. Next, the hydraulic actuator system is developed and connected to the basic model, before a control system is designed according to the method presented in A.2. In section 4.1.4, a guidance system, taking in the joystick signals, and filtering them into signals suitable for the controller is developed. Finally, some simulation results and evaluations of the simulator is presented.

### 4.1.1 Basic Dynamics Modelling

The purpose of this section is to first derive the expressions that is necessary to export from Maple in order to create a suitable DLL-file. Secondly, the basic dynamics bond graph implementation is presented.

The generalized coordinates of the manipulator is chosen as the angular displacements of the six revolute joints, numbered from the base and outwards. Next, a local reference frame is placed in each joint according to figure 4.2. An inertial reference frame is also placed such that it is identical to the reference frame $x_1y_1z_1$ when the innermost joint angle $q_1 = 0$, such that the coordinate of the origin of the reference frame $x_1y_1z_1$, relative to the origin of the inertial reference frame is given as $\boldsymbol{r}_{1/0} = [0,\ 0,\ 0]^T$.

Figure 4.2 can also be used in order to construct the necessary rotation matrices. In order to construct the matrix $\boldsymbol{R}_i^{i-1}(q_i)$, we identify a sequence of rotations which can be applied to the reference frame of joint $i-1$, in order for it to obtain the same orientation as the reference frame of joint $i$. To this effect, note that the axis

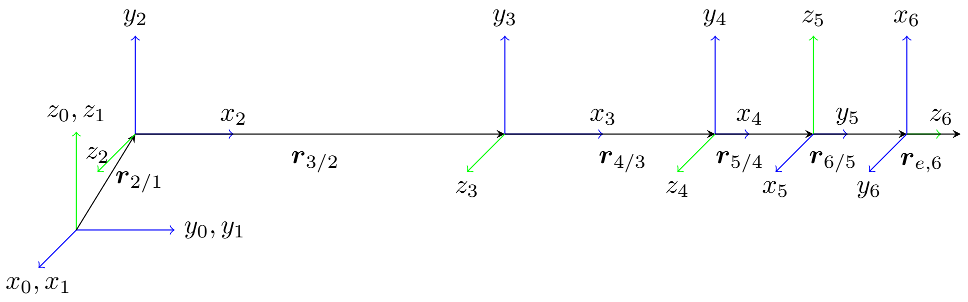


Figure 4.2: Titan 4 kinematics with all joint angles set to zero.

about which joint $i$ revolve is in figure 4.2 identified as the green axis of the reference frame $x_i y_i z_i$. As an example, consider the rotation matrix $\boldsymbol{R}_1^0(q_1)$. The inertial reference frame, can be rotated an angle $q_1$ about the $z_0$ axis in order to align it with the reference frame of joint 1. Similarly, the matrix $\boldsymbol{R}_2^1(q_2)$ is identified by observing that the reference frame of joint 1 can be rotated first an angle of 90° about the $z_1$-axis to obtain the intermediate reference frame $x_1^{'} y_1^{'} z_1^{'}$. This frame can in turn be rotated an angle of 90° about the $x_1^{'}$-axis in order to obtain the intermediate reference frame $x_1^{''} y_1^{''} z_1^{''}$. Finally, this reference frame can be rotated an angle $q_2$ about the $z_1^{''}$-axis, resulting in a reference frame with the same orientation as the reference frame of joint 2. The rest of the rotation matrices can be identified in a similar manner, resulting in

$$\begin{aligned}
\boldsymbol{R}_1^0(q_1) &= \boldsymbol{R}_z(q_1) \\
\boldsymbol{R}_2^1(q_2) &= \boldsymbol{R}_z(90°)\boldsymbol{R}_x(90°)\boldsymbol{R}_z(q_2) \\
\boldsymbol{R}_3^2(q_3) &= \boldsymbol{R}_z(q_3) \\
\boldsymbol{R}_4^3(q_4) &= \boldsymbol{R}_z(q_4) \\
\boldsymbol{R}_5^4(q_5) &= \boldsymbol{R}_y(-90°)\boldsymbol{R}_x(90°)\boldsymbol{R}_z(q_5) \\
\boldsymbol{R}_6^5(q_5) &= \boldsymbol{R}_x(-90°)\boldsymbol{R}_z(-90°)\boldsymbol{R}_z(q_6)
\end{aligned} \tag{4.1}$$

where the principal rotation matrices $\boldsymbol{R}_z$, $\boldsymbol{R}_y$ and $\boldsymbol{R}_x$ are given in (2.48).

Recall from chapter 2 that it is necessary to find the linear and angular velocity of each manipulator body center of gravity, as well as the coordinates of this point relative to some common point of reference. To this effect, the velocities and the coordinates in question shall be expressed relative to, and in terms of, the inertial reference frame. The coordinates of the origin of each reference frame relative to the origin of the preceding reference frame, expressed in terms of the local reference frame, are known and stated in appendix B. Using this, the coordinates of the $i$-th joint, relative to origin of, and expressed in terms of the inertial reference frame, can be found as

$$\boldsymbol{r}^0_{i/0} = \boldsymbol{R}^0_1 \boldsymbol{r}^1_{2/1} + \boldsymbol{R}^0_1 \boldsymbol{R}^1_2 \boldsymbol{r}^2_{3/2} + \cdots + \boldsymbol{R}^0_1 \boldsymbol{R}^1_2 \cdots \boldsymbol{R}^{i-2}_{i-1} \boldsymbol{r}^{i-1}_{i/i-1} \tag{4.2}$$

Similarly, the coordinates of joint $i$ relative to joint $k$ for $k < i$, can be found as

$$\boldsymbol{r}^0_{i/k} = \boldsymbol{R}^0_1 \boldsymbol{R}^1_2 \cdots \boldsymbol{R}^{k-1}_k \boldsymbol{r}^k_{k+1/k} + \cdots + \boldsymbol{R}^0_1 \boldsymbol{R}^1_2 \cdots \boldsymbol{R}^{1-2}_{i-1} \boldsymbol{r}^{i-1}_{i/i-1} \tag{4.3}$$

Given the coordinates of joint $i$ relative to the joint $k$, the coordinates of the center of gravity for body $i$ relative to joint $k$ is found as

$$\boldsymbol{r}^0_{cgi/k} = \boldsymbol{r}^0_{k+1/k} + \boldsymbol{r}^0_{k+2/k+1} + \cdots + \boldsymbol{r}^0_{i/i-1} + \boldsymbol{R}^0_1 \boldsymbol{R}^1_2 \cdots \boldsymbol{R}^{i-1}_i \boldsymbol{r}^i_{cgi/i} \tag{4.4}$$

where the vectors $\boldsymbol{r}^i_{cgi/i}$ are constant and given in the appendix B.

With all relevant coordinates $\boldsymbol{r}^0_{cgi/k}$ identified, the linear and angular velocities $\boldsymbol{v}^0_{cg1/0}$ and $\boldsymbol{\omega}^0_{i/0}$, and the corresponding geometric Jacobian matrices $\boldsymbol{J}_i(\boldsymbol{q})$, can be found according to (2.56) through (2.64).

Next, we define the six $6 \times 6$ local mass-inertia matrices

$$\boldsymbol{M}_i = \begin{bmatrix} m_i \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}^0_i \end{bmatrix} \tag{4.5}$$

where $m_i$ is the mass of the $i$-th body and $\boldsymbol{I}^0_i = \boldsymbol{R}^0_i \boldsymbol{I}^i_i \boldsymbol{R}^i_0$ is the inertia tensor of the $i$-th body expressed in terms of the inertial reference frame. Using the local mass-inertia matrices and the geometric Jacobian matrices, the system mass-inertia matrix can be found as

$$\boldsymbol{B}(\boldsymbol{q}) = \sum_{i=1}^{6} \boldsymbol{B}_i(\boldsymbol{q}) \tag{4.6}$$

where

$$\boldsymbol{B}_i(\boldsymbol{q}) = \boldsymbol{J}^T_i(\boldsymbol{q}) \boldsymbol{M}_i \boldsymbol{J}_i(\boldsymbol{q}) \tag{4.7}$$

The potential energy of the system is found according to (2.2) and (2.3). With the potential energy and the system mass-inertia matrix found, Maple can be used in order to symbolically partially differentiate both expressions with respect to each of the generalized coordinates, and export the expressions to optimized C-code. A Maple algorithm for do so is found in appendix B.

Figure 4.3 show a possible manner in which to implement the basic dynamics of the Titan 4 manipulator in bond graph. Each 1-junction in the figure represent a generalized coordinate displacement rate. The underlying code for the IC-field of this bond graph is attached in appendix B.

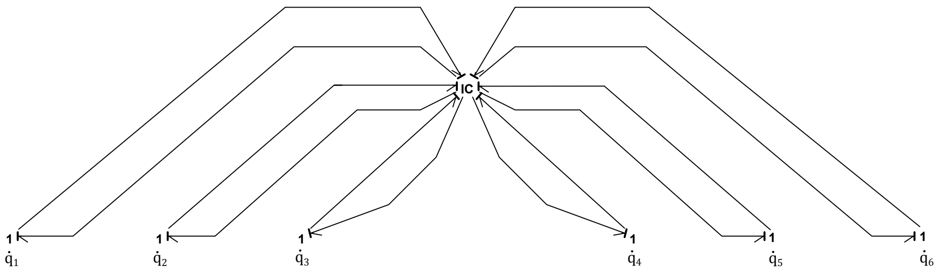


Figure 4.3: Bond graph of the Titan 4 basic dynamics.

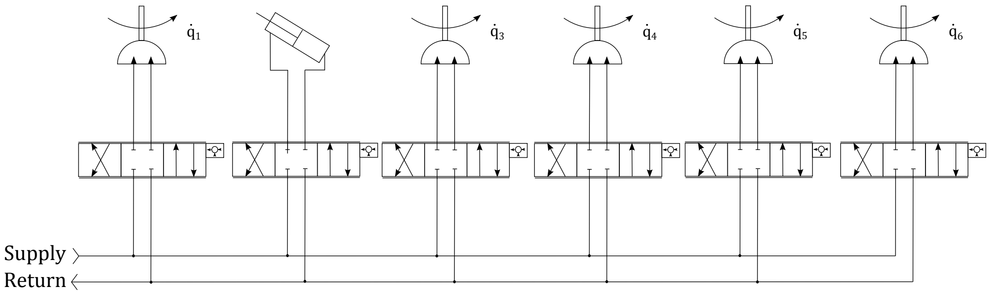


Figure 4.4: Hydraulic diagram of Titan 4 actuator system.

### 4.1.2 Hydraulic Actuator System

The Titan 4 manipulator is equipped with hydraulic actuators. On the second joint is a linear piston actuator, while the rest of the joints are actuated by hydraulic motors. Each actuator is controlled by a servo valve, and supplied from a common pump, as illustrated in figure 4.4. The following discussion starts by presenting a bond graph model of the hydraulic system, where the servo valves and the actuators are shown as sub models. The mapping from actuator output effort to the resulting torque on the manipulator joints are discussed before the servo valve models and the actuator models are presented.

A bond graph model of the hydraulic actuator system, and its connection to the bond graph of the basic dynamics of the manipulator, is shown in figure 4.5. Notice first that the hydraulic supply pressure and return pressure are modelled as the effort sources to the left in the figure, such that these pressures are assumed to be constant. The servo valves are all connected to both the supply and return line. To the right of the servo valves are the actuators, whose output efforts are connected to

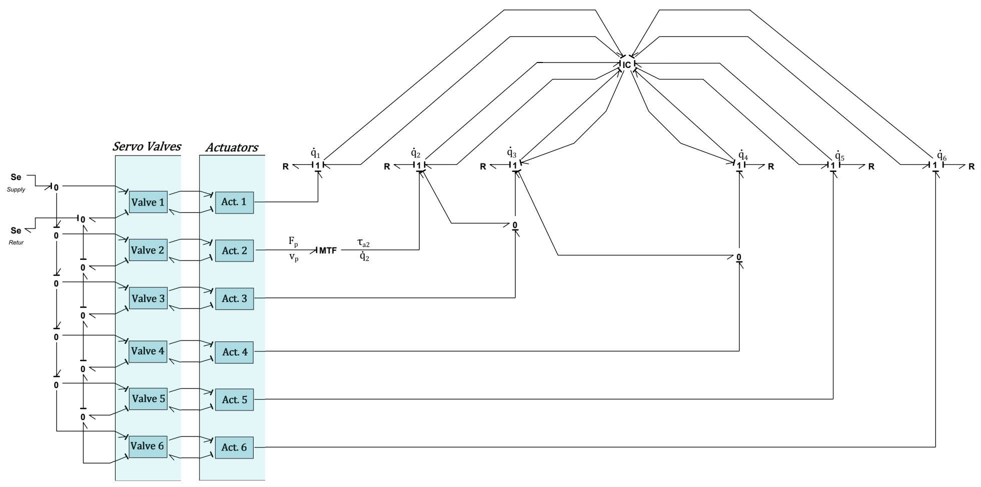


Figure 4.5: Bond graph of Titan 4 with hydraulic actuator system.

the corresponding joint through the 1-junction representing the angular displacement rate of the joint in question. For all joints, but the second, the actuator efforts are torque outputs, acting directly on the corresponding joints generalized displacements as generalized coordinates. The second actuator, i.e., the linear actuator, generates a linear force $F_p$, which must be transformed into the generalized force $\tau_{a2}$, which is the torque about the second joint due to the actuator. This is done with the modulated transformer situated to the right of the second actuator in figure 4.5. The constitutive relations for this transformer is given as

$$\begin{aligned}\tau_{a2} &= \mathcal{J}_2^a(\boldsymbol{q})F_p \\ \dot{q_2} &= \frac{1}{\mathcal{J}_2^a(\boldsymbol{q})}v_p\end{aligned} \tag{4.8}$$

where

$$\mathcal{J}_2^a(\boldsymbol{q}) = L\sin(\phi) \tag{4.9}$$

and $L$ and $\phi$ are given in figure 4.6. By inspecting the figure, it can be seen that the angle $\phi$ and the length of the linear actuator, $L_p$, are dependent on the joint angle $q_2$. The length of the actuator can be related to $q_2$ by using the cosine rule as

$$L_c(q_2) = \sqrt{L^2 + L_1^2 - 2LL_1\cos(\beta)} \tag{4.10}$$

where $\beta = \beta_0 + q_2$, and $L_1$ and $\beta_0$ are defined in the figure. Using $L_p(q_2)$, the angle

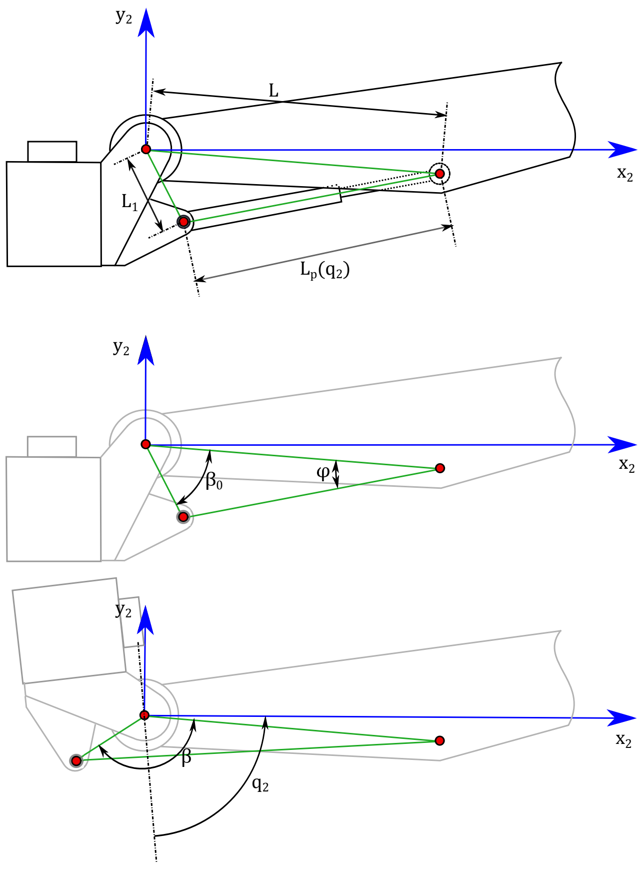


Figure 4.6: Geometric definitions for the linear actuator.

$\phi$ can be found by again applying the cosine rule as

$$L_1^2 = L^2 + L_p(q_2)L\cos(\phi)$$
$$\Rightarrow \phi = \cos^{-1}\left(\frac{L^2 + L_p^2(q_2) - L_1^2}{2LL_p(q_2)}\right) \tag{4.11}$$

Notice also from figure 4.5 that the reaction torque from the third actuator acts on the second body through the power bond from the 1-junction representing $\dot{q}_1$ to the 0-junction on the power bond from the third actuator to the 1-junction representing $\dot{q}_3$. The same relation can also be found between the fourth actuator and the third body. The reason for this is that the second, third and forth joints all revolve about parallel axes, where as any other joint on the manipulator, revolve about an axis that is normal to that of the preceding or subsequent joint. This can be seen from figure 4.2.

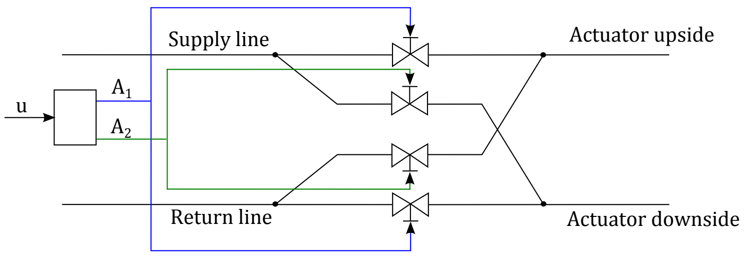


Figure 4.7: Model of servo valve.

### Servo valves and actuators

In figure 4.5 we see that the servo valves are connected to the supply pressure line and the return line on one side, and the upside and down side line of the actuators on the other side. The servo valves should have two functions. First, they should be able to control the amount of hydraulic fluid flowing through, by restricting the area through which the fluid can flow. Second, they should be able to switch the up and down side lines of the actuators in order to reverse their directions.

Figure 4.7, show a conceptual model of the servo valve, which provides both mentioned functions. The servo valve conceptual model consists of four simple valves, able to restrict the flow area of the lines passing through them. The valve connected to the supply line and the actuator upside, and the valve connected to the actuator downside and the return line, restricts the flow area to $A_1$, while the two remaining valves restrict the flow area to $A_2$. Consider now a signal $u$ controlling the valve opening areas $A_1$ and $A_2$ such that

$$A_1 = \begin{cases} u, & \text{if } u \geq 0 \\ 0, & \text{else} \end{cases}, \quad A_2 = \begin{cases} 0, & \text{if } u \geq 0 \\ -u, & \text{else} \end{cases} \tag{4.12}$$

Then $u$ can be thought of as the valve opening area of the servo valve. In the case where $u \geq 0$, the valve is open in the forward direction, where as when $u < 0$, the valve is open in the reverse direction. In the first case, we have $A_1 = u$ and $A_2 = 0$, such that the pressure supply is connected to the upside of the actuator, and in the latter case, $A_1 = 0$ and $A_2 = u$, such that the supply pressure is connected to the actuator downside.

Figure 4.7 can be modelled as a bond graph as shown in figure 4.8, which depicts the content in the sub models representing the servo valves in figure 4.5. The four modulated transformers represents the four valves in figure 4.7, with the constitutive

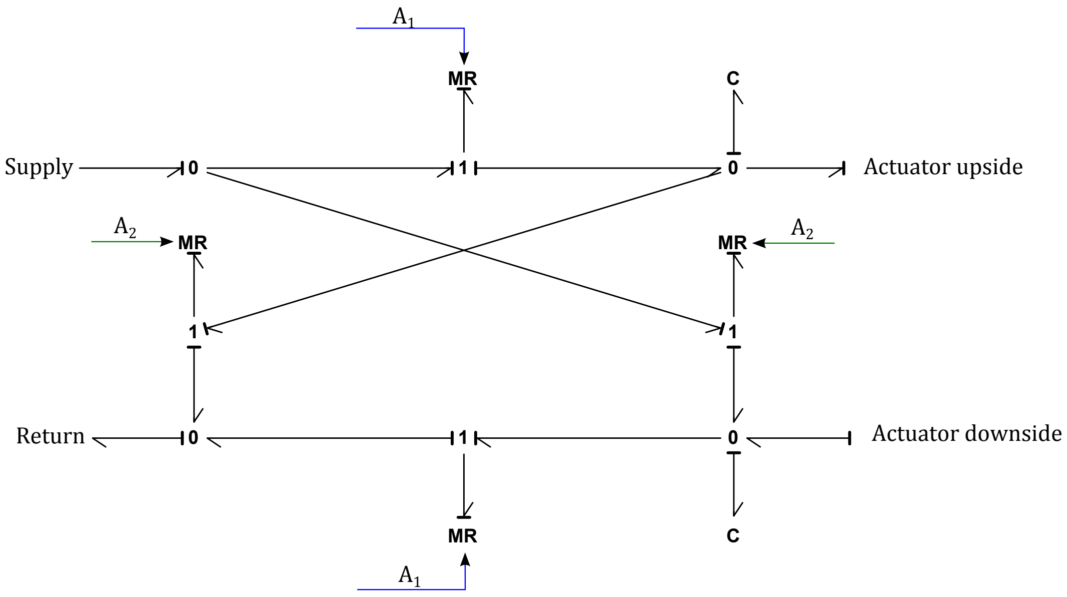


Figure 4.8: Bond graph model of Titan 4 servo valve.

relations (Pedersen and Engja, 2008)

$$\dot{V} = C_d A \text{sgn}(P)\sqrt{\frac{2}{\rho}|P|} \tag{4.13}$$

where $\dot{V}$ is the flow, i.e., the volumetric rate of the hydraulic fluid, of the valve in question, $P$ is the effort, i.e., the hydraulic pressure loss over the valve in question, $C_d$ is the hydraulic discharge coefficient, $\rho$ is the density of the hydraulic fluid, and $A$ is the modulator of the element, i.e., either $A_1$ or $A_2$. The two C-elements in the bond graph represents the compliance of the hydraulic fluid, modelled as two accumulators with the constitutive relations

$$P = \frac{\beta}{V_0}\int_{\tau=0}^{t} \dot{V} d\tau + P_0 \tag{4.14}$$

where $\beta$ is the effective bulk modulus, accounting for the compliance of both the hydraulic fluid and the tubing, and $P_0$ is the initial volume. Notice that the servo valve opening area is assumed to be instantaneous. This is a simplification made on the assumption that the valve dynamics are much faster than the rest of the system. This assumption is investigated in A.2 for hydraulic systems of various stiffness, and found to be good given that the system is not to stiff. In the simulator of this case study, the hydraulic parameters, in particular the bulk modulus and

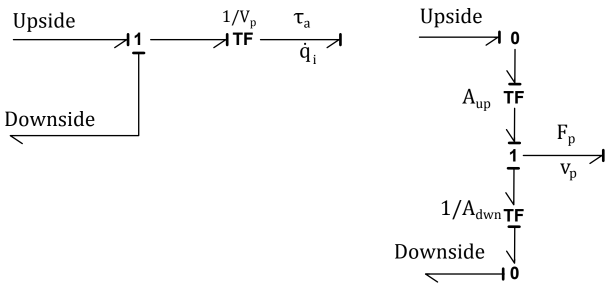


Figure 4.9: Bond graph of Titan 4 hydraulic actuators.

the initial volumes, are chosen such that the hydraulics safely can be assumed to be much slower than the control valves.

Bond graph models of the hydraulic actuators are shown in figure 4.9. The model of the hydraulic motor is shown to the left, and a model for the linear actuator is shown to the right in the figure. For both models, the pressure $P_{up}$ of the upside of the actuator and the pressure $P_{dwn}$ of the down side are inputs, while the mechanical effort, i.e., the torque in the case of motor, and force in the case of linear actuator, are output. The motor transforms the pressure loss $P_L = P_{up} - P_{dwn}$ into a torque $\tau_a$. This is represented by the transformer element with the constitutive relation

$$\begin{aligned} P_L &= \frac{1}{V_p}\tau_a \\ \dot{V} &= V_p\dot{q} \end{aligned} \tag{4.15}$$

where $V_p$ is the volumetric displacement of the hydraulic motor per radian.

The linear actuator relates the force $F_p$ exerted by the piston, to the pressure loss over the actuator. On the upside of the actuator, the pressure $P_{up}$ works on the piston area $A_{up}$, resulting in the force $F_{up} = P_{up}A_{up}$. Likewise, the downside pressure $P_{dwn}$ acts on the area $A_{dwn} = A_{up} - A_{rod}$, where $A_{rod}$ is the area occupied by the piston rod on the downside. This results in the force $F_{dwn} = P_{dwn}A_{dwn}$. These transformations are represented by the transformer elements with the moduli $A_{up}$ and $1/A_{dwn}$ respectively. As seen from the figure, the resulting piston force $F_p$ is the difference between $F_{up}$ and $F_{dwn}$.

The hydraulic actuators provides significant friction to the manipulator joints. This friction is represented by the R-elements connected to the 1-junctions representing the angular displacement rate of the joints, as shown in figure 4.5. In this case study, the friction is assumed to be linear, such that the friction torque is given by $\tau_{fi} = d_{fi}\dot{q}_i$ for joint $i$ where $d_{fi}$ is the friction coefficient. An alternative friction model could be the LuGre friction model, as presented in e.g. Zeng and Sepehri (2006).

### 4.1.3 Controller

Recall from the previous section that the servo valve models were structured such that they could take in a control signal $u$, representing the desired valve opening area. In this section, the controller proposed in A.2 is implemented to the simulator, in order to generate the signal $u$. The controller is based on a simplified model of the system, referred to as the control plant model. The control plant model is derived and validated against a more advanced model in A.2.

The controller to be implemented is a joint angle trajectory tracking controller, meaning that the controller seeks to make the measured joint angle signals follow a trajectory representing the corresponding desired joint angles. To this effect the controller utilize measurements of the joint angles $\boldsymbol{q}_m$, the joint angular rates $\dot{\boldsymbol{\omega}}_m = \dot{\boldsymbol{q}}_m$, and the pressure loss over the actuators $\boldsymbol{P}_m$. It is in this case study assumed that these measurements are available and noiseless. The controller also utilize signals representing the desired joint angles $\boldsymbol{q}_d$, the desired joint angular rates $\dot{\boldsymbol{q}}_d$, the desired joint acceleration $\ddot{\boldsymbol{q}}_d$, and the desired joint acceleration rate $\dddot{\boldsymbol{q}}_d$. These signals are provided by the guidance system, which is discussed in the next section. Figure 4.10 show the layout of the system.

As the controller is derived and proven stable in A.2, we only summarize the controller equations in this section. The joint angular error and the angular velocity error is defined as

$$\begin{aligned} \boldsymbol{e}_1 &= \boldsymbol{q}_m - \boldsymbol{q}_d \\ \boldsymbol{e}_2 &= \boldsymbol{\omega}_m - \dot{\boldsymbol{q}}_d \end{aligned} \tag{4.16}$$

The control law is then given as

$$\boldsymbol{u} = \frac{V_0\sqrt{\rho}}{2\beta C_d}\boldsymbol{\Gamma}^{-1}\boldsymbol{v}_d \tag{4.17}$$

where the $6 \times 6$ diagonal matrix $\boldsymbol{\Gamma}$ is defined as

$$\boldsymbol{\Gamma} = \text{diag}(\boldsymbol{\nu}) \tag{4.18}$$

and $\boldsymbol{\nu} = [\nu_1, \nu_2, ..., \nu_6]^T$. The function $\nu_i$ for the $i$-th servo valve is

$$\nu_i = \sqrt{P_s - \text{sgn}(u_i)P_{Li}} \tag{4.19}$$

The vector $\boldsymbol{v}_d$ from (4.17) can be expressed as

$$\boldsymbol{v}_d = \frac{2\beta}{V_0}\boldsymbol{E}\boldsymbol{\omega}_m + \dot{\boldsymbol{\alpha}}_1 - \boldsymbol{E}\boldsymbol{e}_2 - \boldsymbol{K}_3\boldsymbol{e}_3 \tag{4.20}$$

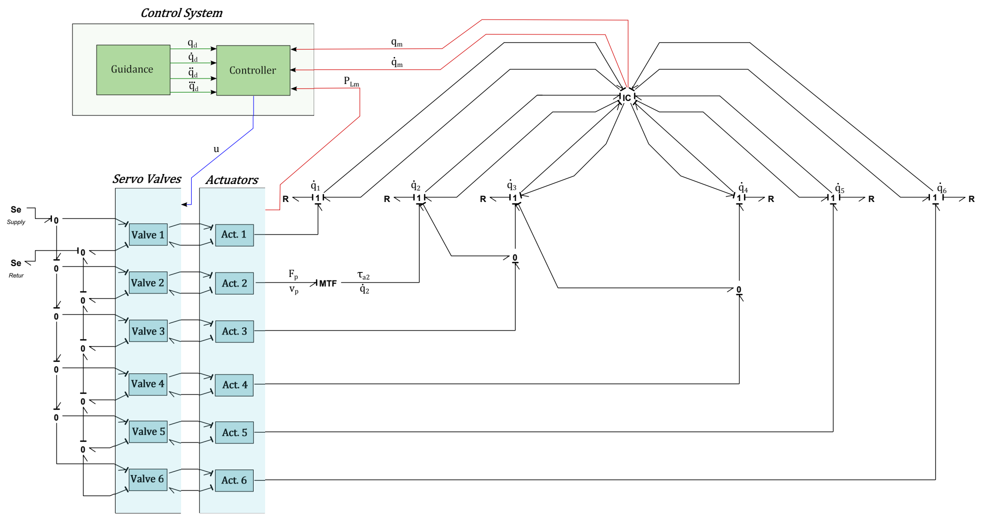


Figure 4.10: Bond graph of Titan 4 manipulator with hydraulic actuator system and control system.

where

$$\begin{aligned} \boldsymbol{E} &= \operatorname{diag}(\boldsymbol{e}) \\ \boldsymbol{e} &= [e_1,\, e_2 \,...,\, e_6]^T \\ e_i &= \begin{cases} V_{pi} & \text{if } i \in \{1,3,4,5,6\} \\ \mathcal{J}_i^a(\boldsymbol{q}_m)A_{pi} & \text{if } i = 2 \end{cases} \end{aligned} \tag{4.21}$$

The vector $\boldsymbol{\alpha}_1$ from (4.20) is the vector of desired pressure losses over the actuators,. The difference between the desired pressure losses and the measured ones are denoted $\boldsymbol{e}_3$. Then the diagonal and positive definite design matrix $\boldsymbol{K}_3$ is a gain matrix to this error. The desired pressure loss is

$$\boldsymbol{\alpha}_1 = \boldsymbol{E}^{-1}(\boldsymbol{C}(\boldsymbol{q}_m, \boldsymbol{\omega}_m)\dot{\boldsymbol{q}}_d + \boldsymbol{\tau}_f + \boldsymbol{g}(\boldsymbol{q}_m) + \boldsymbol{B}(\boldsymbol{q}_m)\ddot{\boldsymbol{q}}_d + \boldsymbol{\alpha}_0) \tag{4.22}$$

where $\boldsymbol{\tau}_f$ is the friction forces. These are easy to calculate exactly in this case study, but in general they should be estimated. It should also be noted that the controller stability properties are retained if this term is not included, as the friction always will contribute to increased damping, i.e, a negative term in any Lyapunov function time derivative along the system trajectories. The term $\boldsymbol{\alpha}_0$ is given as

$$\boldsymbol{\alpha}_0 = -\boldsymbol{K}_p\boldsymbol{e}_1 - \boldsymbol{K}_d\boldsymbol{e}_2 \tag{4.23}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are positive definite diagonal design matrices. In particular, the former is the proportional gain matrix and the latter is the derivative gain matrix.

Recall from (4.20) that the time derivative of $\boldsymbol{\alpha}_1$ appeared. In order to find an expression for this, assume that the function $\mathcal{J}(\boldsymbol{q}_m)A_{p2} \approx m$, where $m$ is a constant. Doing so yields $\dot{\boldsymbol{E}} = \boldsymbol{0}$, such that

$$\begin{aligned}\dot{\boldsymbol{\alpha}}_1 &= \dot{\boldsymbol{E}}^{-1}(\boldsymbol{C}(\boldsymbol{q}_m, \boldsymbol{\omega}_m)\dot{\boldsymbol{q}}_d + \boldsymbol{\tau}_f + \boldsymbol{g}(\boldsymbol{q}_m) + \boldsymbol{B}(\boldsymbol{q}_m)\ddot{q}_d + \boldsymbol{\alpha}_0) \\ &+ \boldsymbol{E}^{-1}\frac{d}{dt}\left(\boldsymbol{C}(\boldsymbol{q}_m, \boldsymbol{\omega}_m)\dot{\boldsymbol{q}}_d + \boldsymbol{\tau}_f + \boldsymbol{g}(\boldsymbol{q}_m) + \boldsymbol{B}(\boldsymbol{q}_m)\ddot{\boldsymbol{q}}_d + \boldsymbol{\alpha}_0\right) \\ &= \boldsymbol{0} + \boldsymbol{E}^{-1}(\dot{\boldsymbol{C}}\dot{\boldsymbol{q}}_d + \boldsymbol{C}\ddot{q}_d + \dot{\boldsymbol{\tau}}_f + \dot{\boldsymbol{g}} + \dot{\boldsymbol{B}}\ddot{\boldsymbol{q}}_d + \boldsymbol{B}\dddot{\boldsymbol{q}}_d + \dot{\boldsymbol{\alpha}}_0)\end{aligned} \tag{4.24}$$

where

$$\dot{\boldsymbol{\alpha}}_0 = -\boldsymbol{K_p e_2} + \boldsymbol{K}_d\boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{e}_2 + \boldsymbol{\alpha}_0 + \boldsymbol{E}\boldsymbol{e}_3 \tag{4.25}$$

and $\dot{\boldsymbol{C}}$, $\dot{\boldsymbol{\tau}}$, and $\dot{\boldsymbol{g}}$ are found by numeric differentiation, and $\dot{\boldsymbol{B}}$ can be found as

$$\dot{\boldsymbol{B}}(\boldsymbol{q}_m) = \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}_m = \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{q}}\boldsymbol{\omega}_m \tag{4.26}$$

such that

$$\dot{\boldsymbol{B}}\ddot{\boldsymbol{q}}_d = \boldsymbol{\omega}_m^T\frac{\partial \boldsymbol{B}}{\partial \boldsymbol{q}}\ddot{\boldsymbol{q}}_d \tag{4.27}$$

In addition to the expressions presented above, it is decided to include an integrator in the controller, in order to compensate for external constant disturbances. Stability of this is not proved, and the integrator comes as an addition to the controller derived in A.2. The integration error can be defined as

$$\boldsymbol{e}_0 = \int_{\tau=0}^{t} \boldsymbol{e}_1 d\tau \tag{4.28}$$

Doing so, and including the term $-\boldsymbol{K}_i\boldsymbol{e}_0$ to $\boldsymbol{\alpha}_0$, where $\boldsymbol{K}_i$ is the positive definite diagonal integral gain matrix, adds integral effect to the controller.

The set of equations described above are implemented in the equations box with the label *Controller* in figure 4.10. Simulation results showing the performance of the controller are presented in section 4.1.6, after the simulator is fully developed. As a final note, the inversion of the matrix $\boldsymbol{\Gamma}$ in (4.17) requires that all $\nu_i \neq 0$. Furthermore, the square root in this functions results in the restriction $\nu_i \geq 0$. Thus, we must require that $\nu_i > 0$ for all $i$, which means that the pressure loss over the actuators never can exceed the supply pressure. For a high pressure system, this requirement is usually meet. It can however be violated if some extreme external load acts on the manipulator. This is however never the case in the simulator designed here.

### 4.1.4 Guidance System

In this section, the guidance system is discussed. The task of the guidance system is twofold. First the system should take in signals from the user through a joystick, and generate a joint angle reference signal $\boldsymbol{q}_r$. Second, the joint angle reference signal should be filtered through a reference model, outputting the necessary input signal $\boldsymbol{q}_d$, $\dot{\boldsymbol{q}}_d$, $\ddot{\boldsymbol{q}}_d$, and $\dddot{\boldsymbol{q}}_d$ for the controller. This filter shall be referred to as the reference model.

20-sim have a predefined sub model taking input from the *Logitech Extreme 3D Pro* joystick, allowing signal bonds, representing the various joystick output signals, to be taken directly from the sub model. This sub model and joystick is used in the simulator. The joystick have three analogous axes in addition to a number of buttons, whereas the Titan 4 manipulator model have six degrees of freedom. As it is desirable to use analogous axes to control all the manipulator joints, we lack for three analogous axes. This problem is solved by utilizing one of the joystick buttons such that the three innermost joins are controller by default, but when the button is pressed, the three remaining joints can be controlled instead. The drawback of this approach is that only three joints can be controlled simultaneously.

It is found that the easiest manner of controlling the simulator, is to let the direct input from the joystick represent the reference velocity. If the operator let go of the joystick in this case, the joystick signal goes to zero, and the joint angles are constant. The reference model however, take the reference position $\boldsymbol{q}_r$ as input. Therefore, the joystick output is integrated before entering the filter, such that

$$\boldsymbol{q}_r = \int_{\tau=0}^{t} \boldsymbol{S}\boldsymbol{i}d\tau \tag{4.29}$$

where $\boldsymbol{i}$ is the output of the joystick, defined such that $i_i \in [-1, 1]$ for $i = 1$, 2, ..., 6. As $\boldsymbol{i}$ now represents the reference velocity of the joints, this signal is scaled by the matrix $\boldsymbol{S} = \text{diag}([\dot{q}_{1max}, \dot{q}_{2max}, ..., \dot{q}_{6max}]^T)$, such that the maximum joystick output $i_{imax}$ for joint $i$ is $i_{imax} = \dot{q}_{imax}$. Figure 4.11 illustrates the relationship between the scaled input $\boldsymbol{S}\boldsymbol{i}$, and the reference position $\boldsymbol{q}_r$ for one single joint. Here, the joystick signal is scaled such that its maximum range is $[-0.3, 0.3]$.

Figure 4.12 show a block diagram representation of the guidance system. The leftmost box, *joystick*, is the default joystick sub model from 20-sim. The signal $\boldsymbol{i}$ from the joystick goes from the joystick box to the *signal mapping* box. In this box, the joystick signal is scaled, and the logics of switching between the three innermost joints and the three outermost joints by the button, is handled. Next, the scaled signal is integrated to obtain $\boldsymbol{q}_r$. To the right of this integrator, a third order differential equation, representing the reference model is implemented. In equation form, the reference model can be stated as (Fossen, 2011)

$$\dddot{\boldsymbol{q}}_d + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}\ddot{\boldsymbol{q}}_d + (2\boldsymbol{\Delta} + \boldsymbol{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{q}}_d + \boldsymbol{\Omega}^3\boldsymbol{q}_d = \boldsymbol{\Omega}^3\boldsymbol{q}_r \tag{4.30}$$

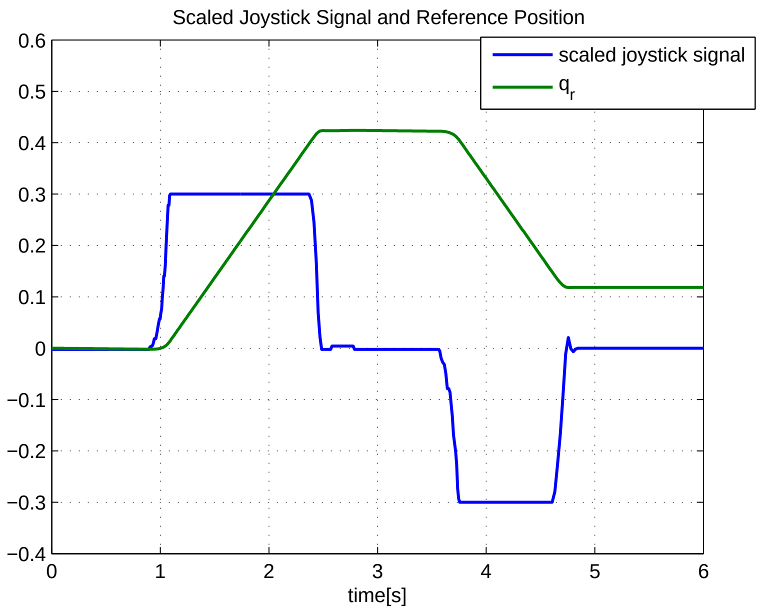


Figure 4.11: Scaled joystick signal and resulting reference position.

where

$$\begin{aligned}\mathbf{\Delta} &= \text{diag}([\xi_1,\, \xi_2,\, ...,\, \xi_6]^T)\\ \mathbf{\Omega} &= \text{diag}([\omega_{n1},\, \omega_{n2},\, ...,\, \omega_{n6}]^T)\end{aligned} \tag{4.31}$$

and $\xi_i$ and $\omega_{ni}$ are the relative damping ratio and the natural frequency of joint $i$. Figure 4.13 show the reference signal $q_r$ and the desired joint angle signal for one single joint, $q_d$. We see from the figure that there are some time delay between the joystick and the desired joint angle. This is because the reference model need some time to accelerate, whereas the joystick signal can accelerate instantaneously. There will however in any case be some delay from the joystick to the manipulator, as the manipulator is subject to physical laws, and can thus not follow the instantaneous joystick signal without delay.

### 4.1.5 Physical Bumpers

The Titan 4 manipulator joints are not able to move to any desired angle. That is, each joint is restricted to move within a certain range, with the exception of the sixth joint, which can rotate to infinity. These restrictions should be reflected in the simulator. In this case study, the restrictions are modelled as physical bumpers.

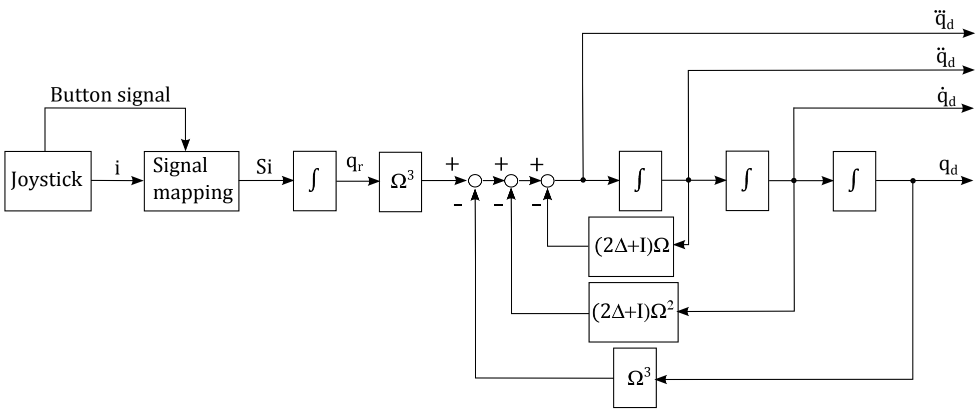


Figure 4.12: Block diagram representation of guidance system.

## Bond graph modelling

We define the allowed range for the Titan 4 manipulator joints as $\boldsymbol{q} \in [\boldsymbol{q}_{dwn}, \boldsymbol{q}_{up}]$, where $\boldsymbol{q}_{dwn} = [q_{1dwn}, q_{2dwn}, ..., q_{5dwn}, -\infty]^T$, $\boldsymbol{q}_{up} = [q_{1up}, q_{2up}, ..., q_{5up}, \infty]^T$, and $q_{idwn}$ and $q_{iup}$ are the minimum and maximum angular displacement of joint $i$. Consider now that a joint angle is approaching either the upper or the lower limit of its range, and hits the limit at the time $t = t_1$. Then, after a while, at the time $t = t_2 > t_1$, the joint angle moves back within the allowed range. Then, for any time $t_1 < t < t_2$, the joint can be considered to be connected to the preceding link, or the base in case of link 1, by an angular spring. However, for any time when the joint is within the allowed range, it should be unaffected by this spring. This can be modelled in the bond graph by connecting each of the joint angle rates that are subject to a restricted range to a C-element. The C-element then have to monitor whether or not the joint angle in question is within or out of the allowed range. The constitutive relation for the C-element can then be stated as

$$\bar{q} = \begin{cases} q - q_{up}, & \text{if } q \geq q_{up} \\ q - q_{dwn}, & \text{if } q \leq q_{up} \\ q, & \text{else} \end{cases} \tag{4.32}$$
$$e_C = \begin{cases} k_b \bar{q}, & \text{if q is out of allowed range} \\ 0, & \text{if not} \end{cases}$$

where $e_C$ is the effort associated to the C-element in question, and $q$ is the corresponding displacement, i.e., the joint angle. In addition to the compliance of the bumper, we shall add damping effect. This can be modelled in a MR-element with

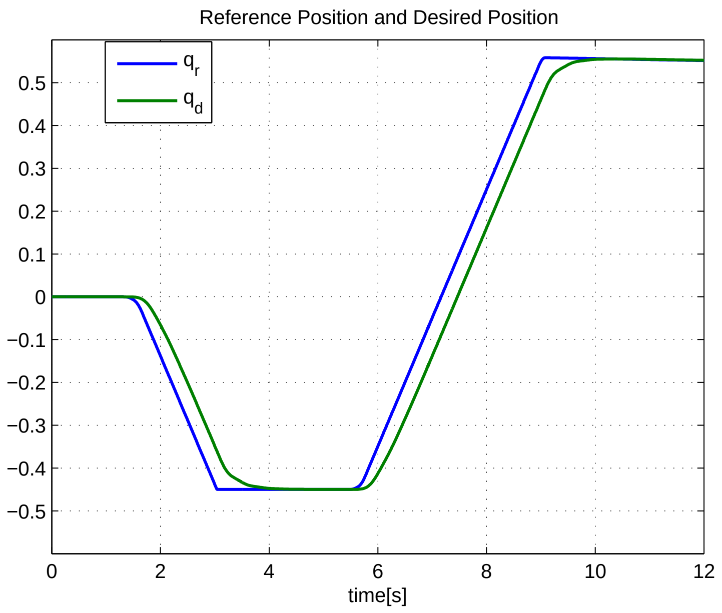


Figure 4.13: Time series showing the reference joint angle and the desired joint angle.

the constitutive relation

$$e_R = \begin{cases} d_b f, & \text{if q is out of allowed range} \\ 0, & \text{if not} \end{cases} \tag{4.33}$$

where $f$ is the flow, i.e. the joint angular rate, and $e_r$ is the damping effort. The angular displacement of the joint is taken in as a signal. In order to reduce the number of power bonds associated to the 1-junctions representing the angular rate of the joints, these two elements are joined to the single bumper elements seen in figure 4.14, with the constitutive relation $e = e_C + e_R$.

### Effects on the guidance system

If the bumpers are implemented and not taken into account in the guidance system, this results in two unfortunate effects. First, the reference signal $\boldsymbol{q}_r$ and the desired joint angles $\boldsymbol{q}_d$ are able to be out side of the allowed range, while the joint angles are not. In this case, the manipulator will not respond properly to the reference signal. Secondly, if a manipulator joint rests against its bumper, while the corresponding signal $q_d$ is outside the range, the corresponding integration error in the controller will continue to grow. This can result in an unstable controller. In order to resolve these issues, the integrator between $\boldsymbol{Si}$ and $\boldsymbol{q}_r$ in figure 4.12 is limited such that $\boldsymbol{q}_r$, and

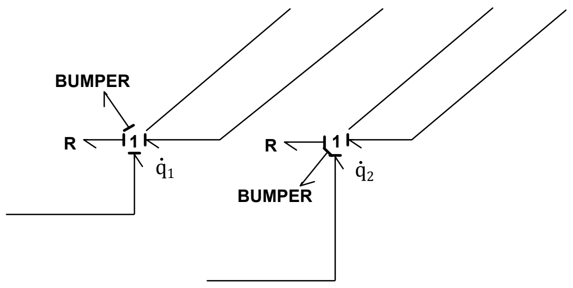


Figure 4.14: Bumpers of the first and second joint.

consequently $\boldsymbol{q}_d$, cannot move outside of the allowed range. The SIDOPS+ language of 20-sim provides a suitable function for this, namely the function *limint(x,min, max)*. This function integrates the variable $x$, such that the output is limited between *min* and *max*. Utilizing this function as the integrator between the limited joystick signal and the reference signal, as

$$\boldsymbol{q}_r = \text{limint}(\boldsymbol{S}\boldsymbol{i}, \boldsymbol{q}_{dwn} + \boldsymbol{\epsilon}, \boldsymbol{q}_{up} - \boldsymbol{\epsilon}) \tag{4.34}$$

where $\boldsymbol{\epsilon}$ is a $6 \times 1$ vector of small numbers, the reference signal will always be within the allowed range, with the safety buffer $\boldsymbol{\epsilon}$.

### 4.1.6 Real Time 3D Animation

In this section, a short discussion on the 3D visualization for the simulator is presented, along with some simulation results and a corresponding series of screen shots from the 3D animation.

The response of the manipulator to joystick inputs are in this simulator presented to the user through a real time 3D animation. This 3D animation is built using the 20-sim 3D animation toolbox. The toolbox allows for reference frames to be placed in the animation. The orientation of the reference frames can be defined through rotation matrices, and the position through $3 \times 1$ vectors. Once one reference frame is defined, a new reference frame can be defined relative to the first. The technique used in this case study is to first place an inertial reference frame. Then the 1-frame is placed with the orientation defined from the rotation matrix $\boldsymbol{R}_1^0(q_1)$, and the position $\boldsymbol{r}_{1/0}^0$. Similarly, reference frame 2 is defined relative to reference frame 1 with the orientation and position defined from $\boldsymbol{R}_2^1(q_2)$ and $\boldsymbol{r}_{2/1}^1$. The remaining local frames are placed in the same manner. Then, within each of the local reference frames, a cylinder or a cube is drawn, depending on the shape of the body in question.

Figure 4.15 show a series of screen shots during a real time simulation with joystick

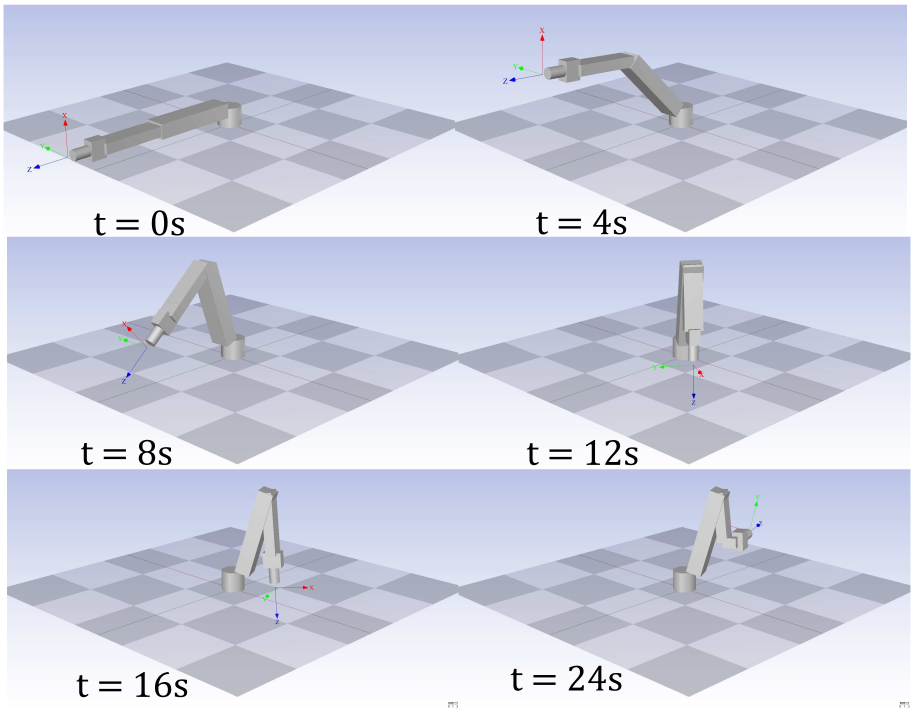


Figure 4.15: Screen shots of a simulator operation.

input, while figure 4.16, show time series for the same simulation. The screen shots are included in order to present the 3D-visualization, while the time series is included in order to show the performance of the control system. On the left hand side in the time series, the joint angles $\boldsymbol{q}$, the reference angles $\boldsymbol{q}_r$, and the filtered reference angles $\boldsymbol{q}_d$ are plotted, while on the right hand side, some points of interest are magnified. From this figure, we recognise the time delay between the reference angles $\boldsymbol{q}_r$ and the filtered angles $\boldsymbol{q}_d$. Overall, the performance of the controller is satisfactory. The only exception is the second joint angle when its reference signal hits the bumper. This is magnified on the right hand plot in the second row, where it can be seen that the angular displacement use about two seconds in order to converge properly towards the reference signal. The reason for this is assumed to be twofold. First, in the simulation model, the second joint experience the reaction torque from the third actuator. This is not accounted for in the control plant model. Secondly, the function mapping the force of the linear actuator to a torque about the second joint, where assumed to be a constant in the control plant model. Together, these simplifications in the control plant model results in a bias for the second joint, particularly when the third actuator is active. This bias is then corrected by the integral effect after

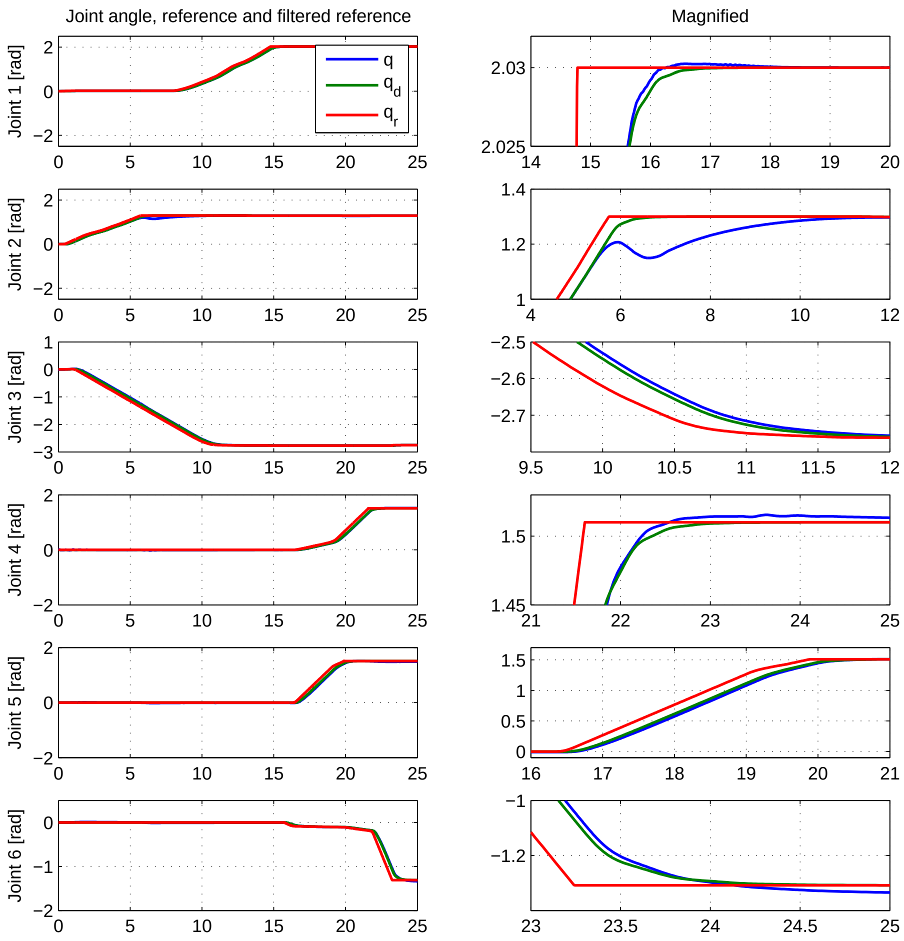


Figure 4.16: Time series of a simulator operation.

approximately two seconds.

## 4.2 Case Study 2 - Manipulator and Remotely Operated Vehicle

The purpose of this case study is twofold. First it is included in order to demonstrate a possible application of the modelling framework presented in this thesis. Secondly, it demonstrates that the manipulator, and its interface to the external environment have a severe impact on the vehicle dynamics. To this effect we shall study a lifting operation performed with a working class remotely operated vehicle and a robotic manipulator. In particular, the system, i.e., the vehicle with the manipulator, will position it self in front of an object, pick it up, reposition, and place the object.

In order to perform simulations of the lifting operation it is necessary to model the basic dynamics of the system, actuators for both the manipulator and the vehicle, motion control systems for both the manipulator and the vehicle, as well as a model of the object to be lifted, and an interface between this model and the end effector of the manipulator. In order to further demonstrate some of the possibilities offered by the basic modelling framework, the vehicle actuator models, i.e., the thruster system, is modelled according to the method proposed in Healey et al. (1995). Furthermore, hydrodynamic damping and hydrodynamic added mass is included in the model. The vehicle to be modelled in this case study is a working class remotely operated vehicle with 6 thrusters. For the manipulator, the three innermost bodies of the Titan 4 manipulator studied in the previous case study, is used. We do however in this case assume electrical actuators for the manipulator.

The basic model is built according to the method presented in appendix A.1. A brief summary of this is presented in the following section. After the basic model is described, we extend it to include added mass, hydrodynamic damping, actuator systems, an interface to the object to be lifted, and finally, motion control systems for both the vehicle and the manipulator.

A simulation of the lifting operation is created. The simulation scenario is that there are two pillars on the seabed, one of which have an object situated on top. The underwater vehicle and the manipulator are then supposed to positions it self in front of the object, lift it, relocate to the other pillar and place the object atop it. The parameters used for this simulation are summarized in appendix C. It is however difficult to visualize the operation based on time series plots alone, and in order to give the reader a better understanding of the operation, a video animation is created. This can be found in the appended files. Time series of the location and orientation of the vehicle center of gravity, along with the manipulator joint angles and the location and orientation of the object is provided.

### 4.2.1 Basic ROV-Manipulator Model

In this case study the Lagrangian dynamics, and as such the IC-field, will only account for the dynamics related to the kinetic energy of the system, i.e., the motion due to inertia forces and Coriolis and centrifugal forces. The dynamics related to the potential energy, i.e., the restoring forces, are accounted for directly in the bond graph, as is done in appendix A.1.

#### Inertia, Coriolis and centrifugal forces

The generalized coordinates of the system is defined as $\boldsymbol{q} = [(\boldsymbol{r}^0_{b/0})^T,\, \boldsymbol{\Theta}^T, \boldsymbol{q}_e^T]$. The position $\boldsymbol{r}^0_{b/0} = [n,\, e,\, d]^T$ is the north, east, and down position of the vehicle body fixed reference frame origin, relative to, and expressed in terms of, the inertial reference frame. We place the vehicle body fixed reference frame in the vehicle center of gravity. The vector $\boldsymbol{\Theta} = [\phi,\, \theta,\, \psi]^T$ is the Euler angles of the vehicle, and $\boldsymbol{q}_e = [q_{e1},\, q_{e2},\, q_{e3}]^T$ is the joint angles of the three manipulator joints. Furthermore, we find the vector of quasi-coordinates $\boldsymbol{\omega} = [u,\, v,\, w,\, p,\, q,\, r,\, \dot{q}_{e1},\, \dot{q}_{e2},\, \dot{q}_{e3}]^T$ as

$$\boldsymbol{\omega} = \boldsymbol{\alpha}^T \dot{\boldsymbol{q}} \tag{4.35}$$

where the transformation matrix $\boldsymbol{\alpha}^T$ is defined in appendix A.1, and $u$, $v$, $w$, $p$, $q$, and $r$ are the surge, sway, heave, roll, pitch and yaw velocity of the vehicle respectively. Expressing the kinetic energy of the system in terms of the quasi-coordinates as

$$\bar{\boldsymbol{T}}(\boldsymbol{q},\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^T \bar{\boldsymbol{B}}(\boldsymbol{q})\boldsymbol{\omega}^T \tag{4.36}$$

the Lagrangian equations of motion are given as (Meirovitch, 2003)

$$\frac{d}{dt}\left(\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}}\right) + \boldsymbol{\beta}^T\boldsymbol{\gamma}\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} - \boldsymbol{\beta}^T\frac{\partial \bar{T}}{\partial \boldsymbol{q}} = \boldsymbol{\beta}^T\boldsymbol{\tau} \tag{4.37}$$

where $\boldsymbol{\beta} = (\boldsymbol{\alpha}^T)^{-1}$, and $\boldsymbol{\gamma}$ is given in appendix A.1. These equations can be expressed in the state space form as

$$\begin{aligned} \boldsymbol{\omega} &= \bar{\boldsymbol{B}}^{-1}(\boldsymbol{q})\boldsymbol{p} \\ \dot{\boldsymbol{p}} &= \bar{\boldsymbol{f}}_p(\boldsymbol{q},\boldsymbol{\omega}) + \boldsymbol{\beta}^T\boldsymbol{\tau} \end{aligned} \tag{4.38}$$

where

$$\bar{\boldsymbol{f}}_p(\boldsymbol{q},\boldsymbol{\omega}) = -\boldsymbol{\beta}^T\boldsymbol{\gamma}\bar{\boldsymbol{B}}(\boldsymbol{q})\boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\omega}^T\frac{\partial \bar{\boldsymbol{B}}(\boldsymbol{q})}{\partial \boldsymbol{q}}\boldsymbol{\omega} \tag{4.39}$$

and the generalized forces can be structured as $\boldsymbol{\beta}^T\boldsymbol{\tau} = [\boldsymbol{\tau}_v^T,\, \boldsymbol{\tau}_\omega^T,\, \boldsymbol{\tau}_{qe}^T]^T$, where $\boldsymbol{\tau}_v$ are the generalized forces associated to the linear motion of the vehicle, $\boldsymbol{\tau}_\omega$ are the generalized forces associated to the angular motion of the vehicle, and $\boldsymbol{\tau}_{qe}$ are the generalized forces associated to the angular motion of the manipulator joints. This state space model can be implemented in a bond graph as shown in figure 4.17.

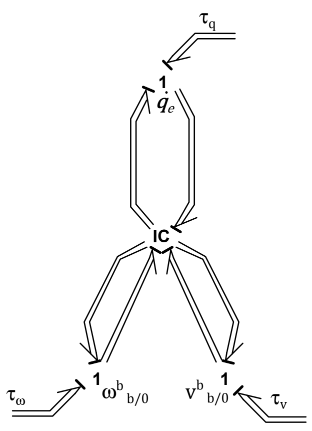


Figure 4.17: Bond graph implementation of the basic dynamics related to the kinetic energy of the system.

## Restoring forces

In order to include restoring forces to the system we follow the approach outlined in A.1. We divide the problem into three parts. First we consider the restoring forces connected to the linear motion of the vehicle, then we look at the restoring forces related to the angular motion of the vehicle, and finally we consider the linear and angular restoring forces of the manipulator bodies.

## Linear restoring forces on the vehicle

The linear restoring forces due to the vehicle is the difference between the buoyancy force and the weight of the vehicle. We shall assume that the vehicle is always fully submerged, and as such, these forces are constant. Expressed in terms of the inertial reference frame, which z-axis points straight down towards the earth center, we find the linear restoring force

$$\boldsymbol{f}_v^0 = \boldsymbol{f}_g^0 + \boldsymbol{f}_b^0 = \begin{bmatrix} 0 \\ 0 \\ (m_{rov} - \rho_w \nabla_{rov}) g \end{bmatrix} \tag{4.40}$$

where $m_{rov}$ is the mass of the vehicle, $\rho_w$ is the density of the sea water, $\nabla_{rov}$ is the displaced volume of the vehicle, and $g$ is the acceleration of gravity. Furthermore, $\boldsymbol{f}_g^0 = [0,\, 0,\, m_{rov} g]^T$ is the weight of the vehicle and $\boldsymbol{f}_b^0 = [0,\, 0,\, -\rho_w \nabla_{rov} g]^T$ is the buoyancy force, both of which are expressed in terms of the inertial reference frame.

In order to interface this force to the basic bond graph, we place a 1-junction

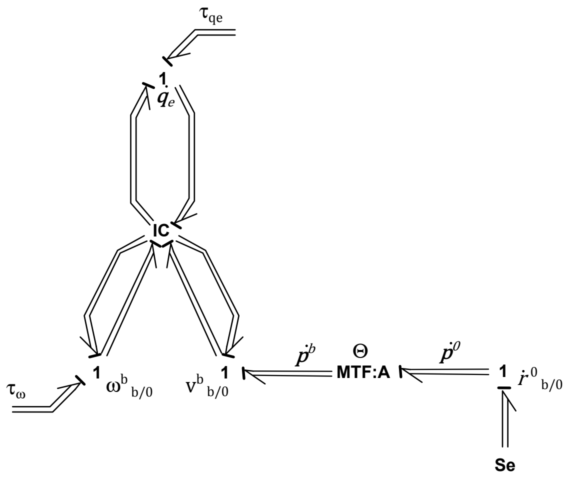


Figure 4.18: Bond graph implementation with linear and angular restoring of the vehicle.

representing the linear velocity of the vehicle expressed in terms of the inertial reference frame, $\dot{\boldsymbol{r}}^0_{b/0}$, and model the force as an effort source. This is shown in figure 4.18 where the modulated transformer *MTF:A* have the constitutive relation

$$\begin{aligned}\dot{\boldsymbol{p}}^b &= \boldsymbol{R}^b_0(\boldsymbol{\Theta})\dot{\boldsymbol{p}}^0 \\ \dot{\boldsymbol{r}}^0_{b/0} &= \boldsymbol{R}^0_b(\boldsymbol{\Theta})\boldsymbol{v}^b_{b/0}\end{aligned} \tag{4.41}$$

and the interpretation of $\dot{\boldsymbol{p}}^0$ and $\boldsymbol{p}^b$ is clear from the figure.

## Angular restoring forces of the vehicle

Angular restoring forces occur when the center of gravity and the center of buoyancy is not located at the same point, and the orientation of the vehicle is such that the buoyancy force and the weight do not act along the same line. In such cases the force pair induce torques on the vehicle, namely the angular restoring forces. In order to find an expression for these forces, we define the coordinate $\boldsymbol{r}^b_{CB/b}$ as the location of the center of buoyancy relative to the origin of the vehicle body fixed reference frame. Then we find the restoring force, expressed in terms of the vehicle body fixed reference frame as

$$\boldsymbol{f}^b_\omega = \boldsymbol{r}^b_{CB/b} \times \boldsymbol{f}^b_b = \boldsymbol{r}^b_{CB/b} \times \boldsymbol{R}^b_0(\boldsymbol{\Theta})\boldsymbol{f}^0_b \tag{4.42}$$

This equation is suitable to implement in a C-element connected to the 1-junction representing the angular velocity of the vehicle, where the time integral of the flow is taken as

$$\boldsymbol{\Theta} = \int_0^t \boldsymbol{T}_\Theta \boldsymbol{\omega}^b_{b/0} d\tau \tag{4.43}$$

and the transformation matrix $\boldsymbol{T}_\Theta(\boldsymbol{\Theta})$ is defined in appendix A.1. This is illustrated in figure 4.19.

**Restoring of the manipulator bodies**

For the manipulator of this case study, we assume that the center of gravity and the center of buoyancy is located at the same point in all the bodies. This eliminates the angular restoring on the manipulators. Note that the procedure for including angular restoring is similar to that of the vehicle, and is described in appendix A.1.

In order to connect the linear restoring forces, we place 1-junctions representing the linear velocity of each of the three manipulator bodies center of gravity, $\boldsymbol{v}^b_{cmi/0}$. These velocities are, as described in A.1, related to the quasi-coordinates as

$$\boldsymbol{v}^b_{cgi/0} = \boldsymbol{J}^v_i(\boldsymbol{q})\boldsymbol{\omega} \tag{4.44}$$

Consulting figure 4.19, we see that the modulated transformers *MTF:B1*, *MTF:B2*, and *MTF:B3*, with the constitutive relations

$$\begin{aligned}\dot{\boldsymbol{p}}^b_{i,1} &= [\boldsymbol{J}^v_i(\boldsymbol{q})]^T\dot{\boldsymbol{p}}^b_{i,2} \\ \boldsymbol{v}^b_{cgi/0} &= \boldsymbol{J}^v_i(\boldsymbol{q})\boldsymbol{\omega}\end{aligned} \tag{4.45}$$

maps the quasi-coordinates into the linear velocities of the center of gravity of body $i$, expressed in terms of the vehicle body fixed reference frame. The notation $\dot{\boldsymbol{p}}^b_{i,1}$ and $\dot{\boldsymbol{p}}^b_{i,1}$ are the input effort and output effort of the transformer respectively. The velocity of the center of gravity of manipulator body $i$ can now be expressed in terms of the inertial reference frame in the same manner as we did for the linear velocity of the vehicle, as illustrated by the modulated transformers denoted *MTF:C1*, *MTF:C2*, and *MTF:C3*, enabling us to place the rightmost 1-junctions in figure 4.19. The linear restoring of the manipulator bodies can now be modelled by connecting effort sources with the effort

$$\boldsymbol{f}^0_{cgi} = \begin{bmatrix} 0 \\ 0 \\ (m_i - \rho_w \nabla_i)g \end{bmatrix} \tag{4.46}$$

where $m_i$ is the mass of manipulator body $i$, and $\nabla_i$ is the volume displacement of body $i$.

### 4.2.2 Added Mass and Potential Damping

The added mass effects due to the motion of an object is strongly depended on the shape of the object(Faltinsen, 1993). By considering the vehicle and the manipulator as a single object, the shape of the object, and as such the added mass, is dependent on the joint displacements of the manipulator, making it more complicated to

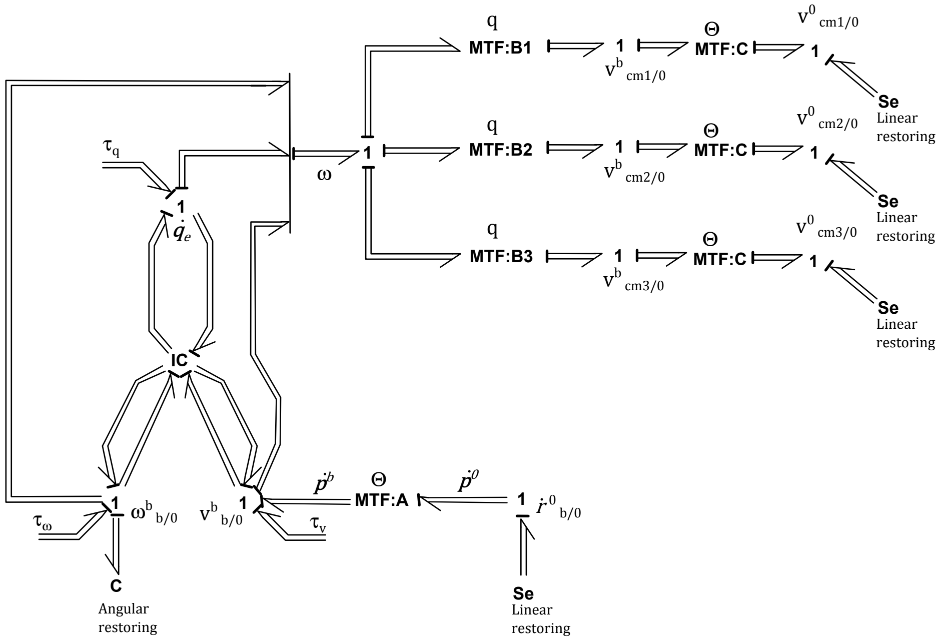


Figure 4.19: Bond graph implementation of the basic dynamics of the system, including restoring forces.

calculate. In order to circumvent this problem we consider the added mass of each body independently. To this effect we assume that the added mass of one body is not affected by the proximity of other bodies. The added mass also strongly depend on the wave excitation frequency (Faltinsen, 1993). We shall however assume that the system is submerged sufficiently to render the wave excitations, which decay exponentially along the inertial z-axis, negligible, and as such, assume frequency independent added mass. Finally, we assume that the added mass is decoupled, meaning that motion in any degree of freedom contribute only to the added mass in the degree of freedom in question. The assumptions stated above is summarized as

**Assumption 1.** The added mass related to a body is independent of any other bodies of the system.

**Assumption 2.** The system is submerged sufficiently to use frequency independent added mass.

**Assumption 3.** The added mass is decoupled.

According to the appendix A.1, we can include the effects of the added mass in the basic equations as

$$\begin{aligned} \boldsymbol{\omega} &= [\bar{\boldsymbol{B}}(\boldsymbol{q}) + \boldsymbol{B}_A(\boldsymbol{q})]^{-1}\boldsymbol{p} \\ \dot{\boldsymbol{p}} &= \bar{\boldsymbol{f}}_p(\boldsymbol{q},\boldsymbol{\omega}) + \boldsymbol{f}_{pA}(\boldsymbol{q},\boldsymbol{\omega}) + \boldsymbol{\tau} \end{aligned} \tag{4.47}$$

where $\boldsymbol{B}_A$ is the added mass matrix of the vehicle, and $\boldsymbol{f}_{pA}(\boldsymbol{q},\boldsymbol{\omega}) = \boldsymbol{C}_A(\boldsymbol{\omega})\boldsymbol{\omega}$ are the Coriolis and centripetal forces resulting from the added mass. We shall use two different approaches to implement this to the vehicle and to the manipulator bodies. We start with the vehicle

## Added mass for the vehicle

For the vehicle alone, we can from A.1 see that the mass matrix is independent of the generalized coordinates and constant. Therefore, with the assumptions 1 and 2, the added mass matrix is also constant. We define the added mass matrix for the vehicle, according to Fossen (2011), as

$$\begin{aligned} \boldsymbol{B}_A &= \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \mathbf{0}_{3\times3} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & 0_{3\times3} \end{bmatrix} \\ &= \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} & \mathbf{0}_{1\times3} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} & \mathbf{0}_{1\times3} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} & \mathbf{0}_{1\times3} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} & \mathbf{0}_{1\times3} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} & \mathbf{0}_{1\times3} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} & \mathbf{0}_{1\times3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times3} \end{bmatrix} \end{aligned} \tag{4.48}$$

and the Coriolis and centripetal matrix as

$$\boldsymbol{C}_A(\boldsymbol{\omega}) = \begin{bmatrix} \mathbf{0}_{3\times3} & -S(\boldsymbol{A}_{11}\boldsymbol{v}^b_{b/0} + \boldsymbol{A}_{12}\boldsymbol{\omega}^b_{b/0}) & \mathbf{0}_{3\times3} \\ -S(\boldsymbol{A}_{11}\boldsymbol{v}^b_{b/0} + \boldsymbol{A}_{12}\boldsymbol{\omega}^b_{b/0}) & -S(\boldsymbol{A}_{21}\boldsymbol{v}^b_{b/0} + \boldsymbol{A}_{22}\boldsymbol{\omega}^b_{b/0}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{4.49}$$

where the matrices are augmented to be compatible with the 9 degrees of freedom system, with zero contributions to the three manipulator bodies. The variable $X_{\dot{u}}$ is the added mass in surge direction due to motion in the same direction, and $X_{\dot{v}}$ is the added mass in the surge direction due to sway motion etc.. The operator $S(\cdot)$ is the cross product operator defined as

$$S(\boldsymbol{a}) = \begin{bmatrix} a & -a_1 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \quad \boldsymbol{a} = [a_1,\, a_2,\, a_3]^T \tag{4.50}$$

Using assumption 3, we find that all off-diagonal elements in the added mass matrix for the vehicle is zero. Thus

$$\boldsymbol{A}_{11} = \begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 \\ 0 & 0 & Z_{\dot{w}} \end{bmatrix}, \quad \boldsymbol{A}_{12} = \boldsymbol{A}_{21} = \boldsymbol{0}, \quad \boldsymbol{A}_{22} = \begin{bmatrix} K_{\dot{p}} & 0 & 0 \\ 0 & M_{\dot{q}} & 0 \\ 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{4.51}$$

The added mass for the vehicle can now be included according to (4.47) as

$$\begin{aligned} \boldsymbol{\omega} &= [\boldsymbol{B}(\boldsymbol{q}) + \boldsymbol{B}_A]^{-1}\boldsymbol{p} \\ \dot{\boldsymbol{p}} &= \bar{\boldsymbol{f}}_p(\boldsymbol{q}, \boldsymbol{\omega}) + \boldsymbol{C}_A(\boldsymbol{\omega})\boldsymbol{\omega} + \boldsymbol{\tau} \end{aligned} \tag{4.52}$$

### Added mass for the manipulator

For the manipulator bodies, by the assumptions 1, 2, and 3, we can include the added mass by a constant scaling factor of the manipulator mass inertia matrices. For manipulator body $i$ we can then define the combined mass-inertia and added mass matrix as

$$\tilde{\boldsymbol{B}}_i(\boldsymbol{q}) = \bar{\boldsymbol{B}}_i(\boldsymbol{q}) + \boldsymbol{B}_{Ai}(\boldsymbol{q}) = \boldsymbol{J}_i^T(\boldsymbol{q}) \begin{bmatrix} m_i\boldsymbol{\Gamma}_i & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \tilde{\boldsymbol{I}}_i^b \end{bmatrix} \boldsymbol{J}_i(\boldsymbol{q}) \tag{4.53}$$

where $\boldsymbol{\Gamma}_i = \mathrm{diag}(\gamma_{i1},\, \gamma_{i2},\, \gamma_{i3})$. Then $\gamma_{i1}$ is a scaling factor that includes the added mass of the $i$-th body in the first principal direction of the local reference frame of the body, due to motion in the same direction. In the same manner, $\gamma_{i2}$ and $\gamma_{i3}$ are the scaling factors including the added mass in the second and third principal directions of the local reference frame due to motion in those directions. The $3 \times 3$ matrix $\tilde{\boldsymbol{I}}_i^b$ is the inertia tensor of the $i$-th body, expressed in terms of the vehicle body fixed reference frame, corrected for added mass, such that

$$\tilde{\boldsymbol{I}}_i^b = \boldsymbol{I}_i^b + \boldsymbol{I}_{Ai}^b \tag{4.54}$$

where $\boldsymbol{I}_{Ai}^b$ is the added mass matrix for the angular degrees off freedom of the $i$-th body. The added mass-corrected inertia tensor can be expressed in terms of the local reference frame as(Ginsberg, 1995)

$$\tilde{\boldsymbol{I}}_i^0 = \boldsymbol{R}_i^b \tilde{\boldsymbol{I}}_i^b \boldsymbol{R}_i^b \tag{4.55}$$

Using that the added mass is decoupled, i.e., assumption 3, the locally expressed added mass-corrected inertia tensor can be expressed as

$$\tilde{\boldsymbol{I}}_i^b = \begin{bmatrix} \gamma_{i4} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & \gamma_{i5} I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & \gamma_{i6} I_{zz} \end{bmatrix} \tag{4.56}$$

where $\gamma_{i4}$ is the scaling factor correcting for the added mass in motion about the first local principal axis due to motion about this axis for the $i$-th body, and $\gamma_{i5}$ and $\gamma_{i6}$ are the correction factor for added mass about the second and third principal axis due to motion about those axes, respectively.

### 4.2.3 Hydrodynamic Damping

There are a number of different physical phenomena causing hydrodynamic damping. In this case study we consider skin friction, and damping due to vortex shedding, using simplified models. The skin friction is assumed to be linear, while the damping due to vortex shedding is assumed to obey the friction term from Morrison's equation.

#### Vehicle damping

Both the angular velocity and the linear velocity of the vehicle is assumed to be associated to linear skin friction. For the linear velocity we have

$$\boldsymbol{f}_{sf}^{v} = \boldsymbol{D}_v \boldsymbol{v}_{b/0}^b \tag{4.57}$$

where $\boldsymbol{D}_v$ is the $3 \times 3$ matrix of friction coefficients associated to the linear velocity. For the angular velocity we find the skin friction force

$$\boldsymbol{f}_{sf}^{\omega} = \boldsymbol{D}_\omega \boldsymbol{\omega}_{b/0}^b \tag{4.58}$$

where $\boldsymbol{D}_\omega$ is the $3 \times 3$ matrix of friction coefficients associated to the angular velocity. By the assumption of decoupled skin friction, the two matrices $\boldsymbol{D}_v$ and $\boldsymbol{D}_{omega}$ are diagonal and described by

$$\begin{aligned} \boldsymbol{D}_v &= \text{diag}(d_u,\, d_v,\, d_w) \\ \boldsymbol{D}_\omega &= \text{diag}(d_p,\, d_q,\, d_r) \end{aligned} \tag{4.59}$$

where $d_u$, $d_v$, $d_w$, $d_p$, $d_q$, and $d_r$ are the damping coefficients in surge, sway, heave, roll, pitch, and yaw motion, respectively. These damping forces are represented by R-elements connected to the 1-junctions representing the linear and angular velocity of the vehicle in terms of vehicle body fixed coordinates.

In this case study, it is assumed that the damping due to vortex shedding is associated only to the linear velocity of the vehicle. According to Faltinsen (1993), these forces can be modelled as the friction term in Morrison's equation, such that

$$\begin{aligned} f_{vs}^{u}(u) &= \frac{1}{2}\rho C_d A_{pu}|u|u \\ f_{vs}^{v}(v) &= \frac{1}{2}\rho C_d A_{pv}|v|v \\ f_{vs}^{w}(w) &= \frac{1}{2}\rho C_d A_{pw}|w|w \end{aligned} \tag{4.60}$$

where $f_{vs}^{u}(u)$, $f_{vs}^{v}(v)$, and $f_{vs}^{w}(w)$ are the friction forces in surge, sway and heave motion respectively, while $A_{pu}$, $A_{pv}$, and $A_{pw}$ are the projected areas in the three directions. The drag coefficient $C_d$ is assumed constant in this case study. The model (4.60) can be implemented in the bond graph as a R-element connected to the linear velocity of the vehicle in terms of vehicle body fixed coordinates.

### Manipulator damping

The manipulator is also assumed to be affected by linear skin friction and damping due to vortex shedding. This can be modelled using strip theory as proposed in McLain and Rock (1998). The basic idea is illustrated in figure 4.20. Each manipulator body is divided into $s$ strips with equal length $dL = L/s$, where $L$ is the total length of the body. The friction force acting on each strip is the sum of the linear skin friction force and the damping force due to vortex shedding. The resulting forces from each of the strips on body $i$, results in a torque exerted on the $i$-th joint. Summing up the resulting torques from each strip of the $i$-th body yields the total hydrodynamic damping associated to joint $i$.

The velocity $\boldsymbol{v}_{i,k/0}$, illustrated in figure 4.20, is the linear velocity at the center of the $k$-th strip on the $i$-th manipulator body, relative to the inertial reference frame. For the manipulator of this case study, the local reference frames are defined such that one of the principal axis is parallel to the longitudinal direction of the manipulator body to which the local frame belong, while two principal axis are normal to the longitudinal direction. Thus, if the linear velocity $\boldsymbol{v}_{i,k/0}$ is expressed in terms of the local reference frame, one of the principal velocity components are parallel to the longitudinal direction, while two is not. Linear skin friction is associated to motion i all three principal directions, and can be expressed as

$$\boldsymbol{f}_{sf}^{ik} = \boldsymbol{D}_{i,k}\boldsymbol{v}_{\boldsymbol{i,k/0}}^{\boldsymbol{i}} \tag{4.61}$$

where $\boldsymbol{D}_{i,k} = d_{sf,i}\boldsymbol{I}_{3,3}$ is the diagonal $3 \times 3$ matrix of damping coefficients for strip $k$ on body $i$. The damping force due to vortex shedding is associated only to the velocity components that are not parallel to the longitudinal direction of the manipulator body. This is because the two cross sections of the strip normal to this direction

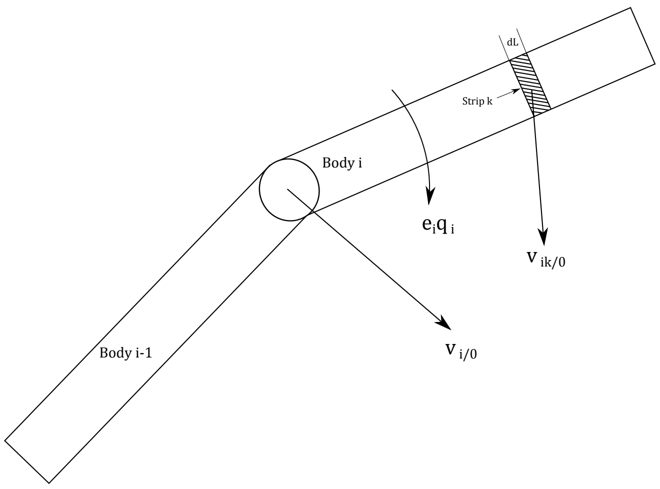


Figure 4.20: Illustration of strip theory scheme applied to a manipulator.

does not face the fluid, but rather the next and the previous strip cross section. As such, one can say that the projected area to the fluid is zero. Note that this is not necessarily the case for the first and final strip of a body. We shall however assume that this is the case. Defining $\boldsymbol{v}^i_{i,k/0} = [u_{i,k},\, v_{i,k},\, w_{i,k}]^T$, the damping force due to vortex shedding on strip $k$ of manipulator body $i$ is

$$\begin{aligned} f_{vs}^{x_{i,k}}(u_{i,k}) &= \frac{1}{2}\rho C_{d,i} A_{(i,k)x} |u_{i,k}| u_{i,k} \\ f_{vs}^{y_{i,k}}(v_{i,k}) &= \frac{1}{2}\rho C_{d,i} A_{(i,k)y} |v_{i,k}| v_{i,k} \\ f_{vs}^{z_{i,k}}(w_{i,k}) &= \frac{1}{2}\rho C_{d,i} A_{(i,k)z} |w_{i,k}| w_{i,k} \end{aligned} \tag{4.62}$$

Here, $C_{d,i}$ is the drag coefficient associated to manipulator body $i$, and $A_{(i,k)x}$, $A_{(i,k)y}$, and $A_{(i,k)z}$ are the projected areas for each principal direction in the local reference frame.

The damping forces for each strip can be implemented in bond graph as R-elements connected to 1-junctions representing the linear velocity of the strip expressed in terms of local reference frames. Figure 4.21 show how the damping forces can be included for body $i$ divided into $s$ strips.

### 4.2.4 Actuator Systems

We have already, in the previous case study, seen how to develop a high fidelity actuator system for a manipulator. In this case study we use simple electrical motors,

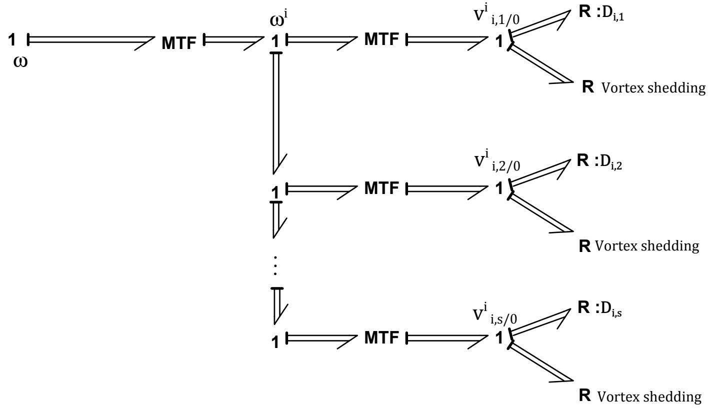


Figure 4.21: Bond graph implementation of damping forces for manipulator body $i$, using $s$ strips.

which we assume can be modelled as effort sources applied directly as generalized forces on each manipulator joint. For the vehicle however, we shall endeavour to develop a high fidelity thruster system after the method presented in Healey et al. (1995), where the drag and lift forces generated by the propeller are coupled with the fluid inertia inside the thruster ducts.

Before developing sub models for the thruster dynamics however, the thruster set up, and the mapping between individual thrust forces and the resulting force and torque acting on the vehicle is defined. The thrust delivered by thruster $i$ is denoted $T_i$, and the thrust of all $k$ thrusters can be collected in the vector $\boldsymbol{T} = [T_1, T_2, ..., T_k]^T$. The thrust $\boldsymbol{T}$ results in the forces and torques $\boldsymbol{\tau}_T = [\boldsymbol{\tau}_v^T, \tau_\omega^T]^T$ on the vehicle, expressed in terms of the vehicle body fixed reference frame. The vector $\boldsymbol{\tau}_v = [\tau_u, \tau_v, \tau_w]^T$ are the forces in surge, sway and heave direction, while $\boldsymbol{\tau}_\omega = [\tau_p, \tau_q, \tau_r]$ are the torques in roll, pitch and yaw. The relation between the thrust vector and the resulting force and torque vector is given as

$$\boldsymbol{\tau}_T = \boldsymbol{H}\boldsymbol{T} \tag{4.63}$$

where $\boldsymbol{H}$ is the $k \times 6$ thrust allocation matrix. When, at a later stage, a control system is designed for the vehicle, the thruster mapping (4.63) must be solved for $\boldsymbol{T}$ in order to find the thrust command for each thruster. In order to simplify this task, the vehicle is outfitted with $k = 6$ thrusters such that the thrust allocation matrix is square and invertible.

Figure 4.22 show the thruster set up for the vehicle of this case study, where the position and orientation of the thrusters relative to the vehicle body fixed reference frame is given by the parameters $d_1$, $d_2$, ..., $d_7$, and $\alpha$. We now consider the force

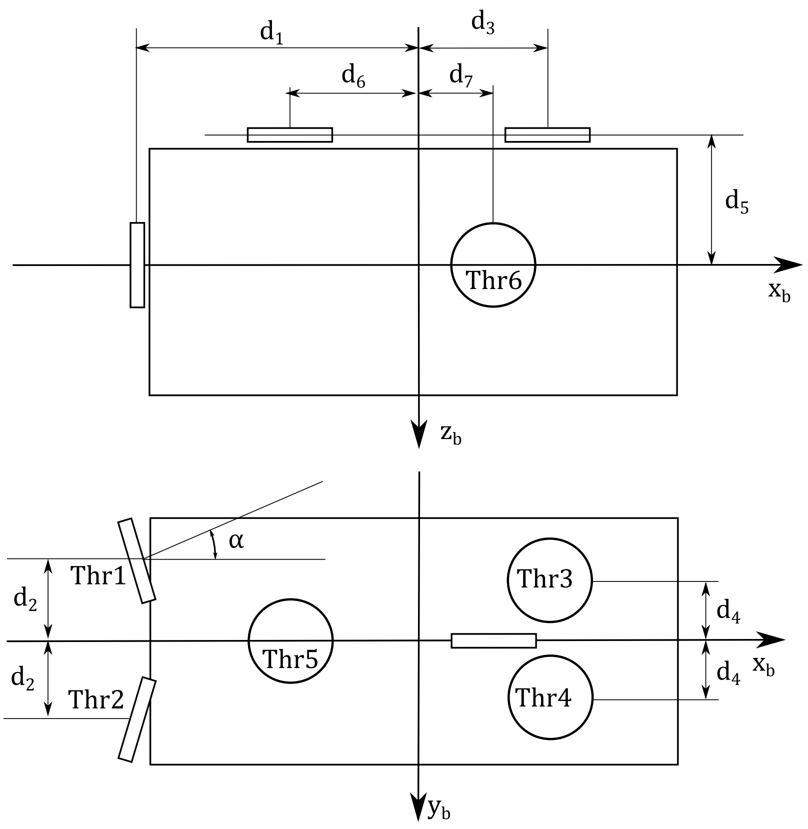


Figure 4.22: Thruster set-up for remotely operated vehicle.

and torque exerted by each thruster on the vehicle. By inspection of figure 4.22, it can be seen that the first and second thrusters generate the force and thrust vectors

$$\boldsymbol{\tau}_{T1} = \begin{bmatrix} T_1\cos(\alpha) \\ -T_1\sin(\alpha) \\ 0 \\ 0 \\ 0 \\ T_1(d_1\sin(\alpha)+d_2\cos(\alpha)) \end{bmatrix}, \boldsymbol{\tau}_{T2} = \begin{bmatrix} T_2\cos(\alpha) \\ T_2\sin(\alpha) \\ 0 \\ 0 \\ 0 \\ T_2(-d_1\sin(\alpha)-d_2\cos(\alpha)) \end{bmatrix} \quad (4.64)$$

where the sub scripts $T1$ and $T2$ indicates that we consider the contribution from thruster 1 and 2, to the vector $\boldsymbol{\tau}_T$. The third, fourth, fifth and sixth thrusters generates the forces and torques

$$\boldsymbol{\tau}_{T3} = \begin{bmatrix} 0 \\ 0 \\ T_3 \\ -d_4 T_3 \\ -d_3 T_3 \\ 0 \end{bmatrix}, \boldsymbol{\tau}_{T4} = \begin{bmatrix} 0 \\ 0 \\ T_4 \\ d_4 T_4 \\ -d_3 T_4 \\ 0 \end{bmatrix}, \boldsymbol{\tau}_{T5} = \begin{bmatrix} 0 \\ 0 \\ T_5 \\ 0 \\ d_6 T_5 \\ 0 \end{bmatrix}, \boldsymbol{\tau}_{T6} = \begin{bmatrix} 0 \\ T_6 \\ 0 \\ 0 \\ 0 \\ d_7 T_6 \end{bmatrix} \tag{4.65}$$

The total force and torques exerted by the thruster system is the sum of all contributions. Thus, we can write

$$\begin{aligned} \boldsymbol{\tau}_T &= \boldsymbol{\tau}_{T1} + \boldsymbol{\tau}_{T2} + \cdots + \boldsymbol{\tau}_{T6} \\ &= \begin{bmatrix} (T_1 + T_2)\cos(\alpha) \\ (-T_1 + T_2)\sin(\alpha) + T_6 \\ T_3 + T_4 + T_5 \\ (-T_3 + T_4)d_4 \\ (-T_3 - T_4)d_3 + T_5 d_6 \\ T_1(d_1 \sin(\alpha) + d_2 \cos(\alpha)) - T_2(d_1 \sin(\alpha) + d_2 \cos(\alpha)) + T_6 d_7 \end{bmatrix} \end{aligned} \tag{4.66}$$

This equation can be used in order to find the thrust allocation matrix as

$$\boldsymbol{H} = \begin{bmatrix} \cos(\alpha) & \cos(\alpha) & 0 & 0 & 0 & 0 \\ -\sin(\alpha) & \sin(\alpha) & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & -d_4 & d_4 0 & 0 & \\ 0 & 0 & -d_3 & -d_3 & d_6 & 0 \\ d_1 \sin(\alpha) + d_2 \cos(\alpha) & -d_1 \sin(\alpha) - d_2 \cos(\alpha) & 0 & 0 & 0 & d_7 \end{bmatrix} \tag{4.67}$$

such that $\boldsymbol{\tau}_T = \boldsymbol{H}\boldsymbol{T}$. Notice that this matrix is invertible for all values of $\alpha$ satisfying $\cos(\alpha) \neq 0$ and $\sin(\alpha) \neq 0$, something which reflects the fact that the vehicle is actuated in all degrees of freedom, given that the restrictions for $\alpha$ are not violated. This statement is substantiated by the following proof.

*Proof.* Consider the case where we want a certain force and torque vector $\boldsymbol{\tau}_{Td}$ from the thruster system. Then we can choose the thrust vector $\boldsymbol{T} = \boldsymbol{H}^{-1}\boldsymbol{\tau}_{Td}$. Given that the thrust allocation matrix $\boldsymbol{H}$ is square, the only case where a thrust vector $\boldsymbol{T}$ cannot be found is when the inverse of the thrust allocation matrix is singular. Thus, if the thrust allocation matrix always is invertible, it follows that the vehicle can be actuated in any degree of freedom. $\square$

Before proceeding to develop models for the thruster dynamics, we shall see how the thruster force mapping can be implemented in the bond graph. To this effect we shall, for now, consider the thrust vector $\boldsymbol{T}$ be delivered by an effort source, as shown in figure 4.23. At this stage we define the $6 \times 1$ vector $\boldsymbol{u}_T = [u_{T1}, u_{T2}, ..., u_{T6}]^T$, where $u_{Ti}$ is the linear velocity component of thruster $i$, in the thrust direction of the $i$-th thruster. This velocity is composed of the contribution $\boldsymbol{u}_T^{(v)}$, from the linear velocity of the vehicle, and the contribution $\boldsymbol{u}_T^{(\omega)}$ from the angular velocity of the vehicle, such that $\boldsymbol{u}_T = \boldsymbol{u}_T^{(v)} + \boldsymbol{u}_T^{(\omega)}$.

Both transformer elements in figure 4.23 have the input effort $\boldsymbol{T}$. The output effort of the right hand side transformer is the vector $\boldsymbol{\tau}_v$, and the output effort of the left hand side transformer is $\boldsymbol{\tau}_\omega$. Similarly, the input flow on the right hand side transformer is the linear velocity of the vehicle, in vehicle body fixed coordinates, while the output flow is the velocity component $\boldsymbol{u}_T^{(v)}$. The input flow of the left hand side transformer is the angular velocity of the vehicle, and the output flow is $\boldsymbol{u}_T^{(\omega)}$. In order to find the constitutive relations for the transformers, we partition the thrust allocation matrix into the two $3 \times 6$ matrices

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_v \\ \boldsymbol{H}_\omega \end{bmatrix} \tag{4.68}$$

Using this, the relation between the thrust vector $\boldsymbol{T}$ and the resulting force on the vehicle is

$$\boldsymbol{\tau}_v = \boldsymbol{H}_v \boldsymbol{T} \tag{4.69}$$

and the relation between the linear velocity of the vehicle and the contribution to the thruster velocities $\boldsymbol{u}_T^{(v)}$, along the thrust axis, from the linear velocity of the vehicle is

$$\boldsymbol{u}_T^{(v)} = \boldsymbol{H}_v^T \boldsymbol{v}_{b/0}^b \tag{4.70}$$

Thus, the constitutive relations for the right hand side transformer in figure 4.23 are given by (4.69) and (4.70).

For the left hand side transformer, the relation between the thrust vector and the resulting torque is

$$\boldsymbol{\tau}_\omega = \boldsymbol{H}_\omega \boldsymbol{T} \tag{4.71}$$

The contribution to the linear velocity component along the thrust axis of each thruster from the angular velocity of the vehicle, $\boldsymbol{u}_T^{(\omega)}$, is

$$\boldsymbol{u}_T^{(\omega)} = \boldsymbol{H}_\omega^T \boldsymbol{\omega}_{b/0}^b \tag{4.72}$$

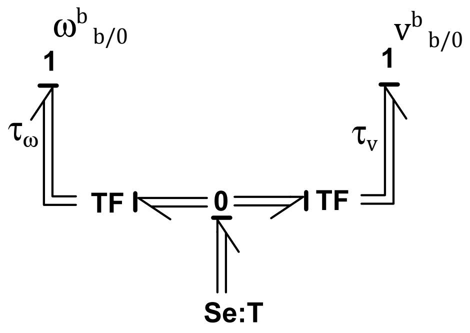


Figure 4.23: Bond graph implementation of thrust force mapping.

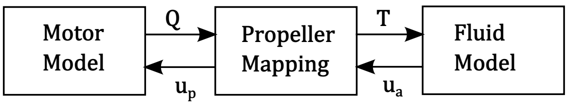


Figure 4.24: Schematic model of thrust dynamics.

## Thruster models

We now exchange the effort source generating the thrust vector in figure 4.23 by a more realistic model. Healey et al. (1995) proposes the thruster model which is shown schematically in figure 4.24. The basic idea of this model is to first model a motor, which generates a torque $Q$ on the propeller shaft. The propeller shaft then rotates at an angular velocity $\omega_p$, resulting in a tangential speed of $u_p = 0.7R$, measured, according to convention(Healey et al., 1995), at $0.7R$, where $R$ is the propeller radius. The propeller blade also experience an incoming velocity $u_a$ on the fluid particles in the thruster duct. Together these velocity components results in a drag force and a lift force on the propeller blade. These in turn combines into the torque $Q$ and the thrust force $T$. Finally, the thrust force accelerate the water inside the duct, resulting in a force on the thruster.

The motor of the thrusters are in this case study modelled as effort sources. This implies the assumption that the motor can instantly deliver the desired torque to the propeller shaft. We do however include linear friction in the bearings of the propeller shaft, as well as the inertia of the propeller and the shaft. Thus the propeller and the motor can be modelled in bond graph according to figure 4.25, where $J_p$ is the moment of inertia for the propeller and shaft, and $d_p$ is the friction coefficient in the bearings.

Consider now a thruster moving through the water with the linear velocity $\boldsymbol{v}_T$ as

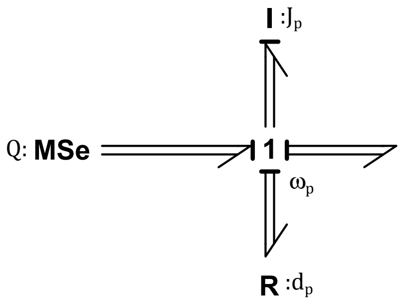


Figure 4.25: Bond graph of the propeller and motor.

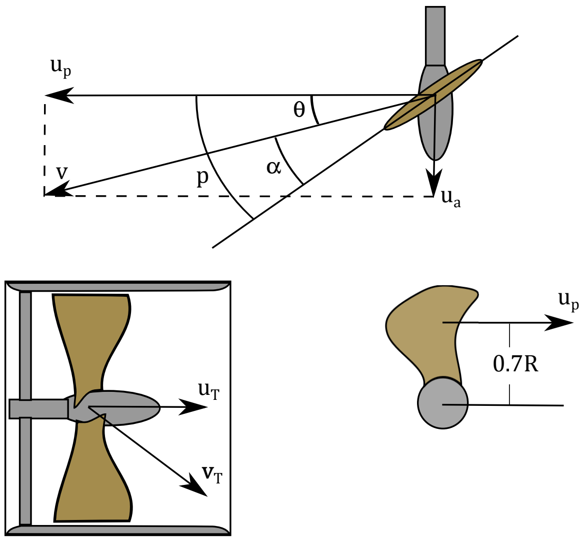


Figure 4.26: Schematic drawing of thruster.

shown in figure 4.26. The component of $\boldsymbol{v}_T$ along the direction of the thrust, is the velocity $u_T$. We assume in the following that when the thruster move through water, the flow through the thruster duct due to the velocity $\boldsymbol{v}_T$ of the thruster, is the component $u_T$. In other terms, the only velocity component of the water in the duct, is that which is normal to the cross section area of the duct. The water inside the duct can be accelerated such that the propeller experience an incoming velocity different than $u_T$. Let this velocity be denoted $u_a$. The point located $0.7R$ out on the propeller blade, experiences the velocity $v = \sqrt{u_p^2 + u_a^2}$, coming in at the angle $\alpha$ relative to the pitch $p$ of the blade, as illustrated in figure 4.26. The lift force $L$, and

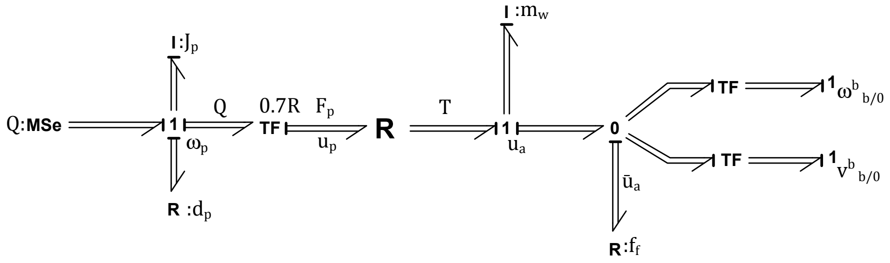


Figure 4.27: Bond graph of thruster system connected to vehicle

drag force $D$ are then (Healey et al., 1995)

$$\begin{aligned} L &= \frac{1}{2}\rho v^2 A C_L \sin(2\alpha) \\ D &= \frac{1}{2}\rho v^2 A C_D (1 - \cos(2\alpha)) \end{aligned} \tag{4.73}$$

where $\rho$ is the water density, $A$ is the propeller duct cross section area, $C_L$ is the lift coefficient, and $C_D$ is the drag coefficient. The lift force and drag force, assumed to act at $0.7R$, results in a torque $Q$ on the propeller shaft and a thrust force $T$. These are found as

$$\begin{aligned} Q &= 0.7R(L\sin(\theta) + D\cos(\theta)) \\ T &= L\cos(\theta) - D\sin(\theta) \end{aligned} \tag{4.74}$$

The thrust force accelerates the water in the thruster duct with mass $m_w$. This results in a relative velocity $\bar{u}_a = u_a - u_T$ between the thruster and the water. In equation form, this can be expressed as

$$m_w \dot{u}_a + f_f(\bar{u}_a) = T \tag{4.75}$$

where the friction force $f_f(\bar{u}_a)$ can be expressed as

$$f_f(\bar{u}_a) = 2\rho A|\bar{u}_a|\bar{u}_a \tag{4.76}$$

Figure 4.27 show how the bond graph of the motor and propeller can be expanded according to the equations presented above, and connected to the vehicle. The transformer element to the left with modulus $0.7R$ transforms the angular velocity of the propeller to the tangential velocity at $0.7R$ out on the propeller, and the shaft torque $Q$ to the corresponding force $F_p$. The R-field element sets the force $F_p$ and the thrust force $T$ according to 4.74. On the right hand side of the R-field, the inertia of the water accelerated by the thrust is represented by the I-element, and the friction term $f_f(\bar{u}_a)$ is represented by the rightmost R-element.

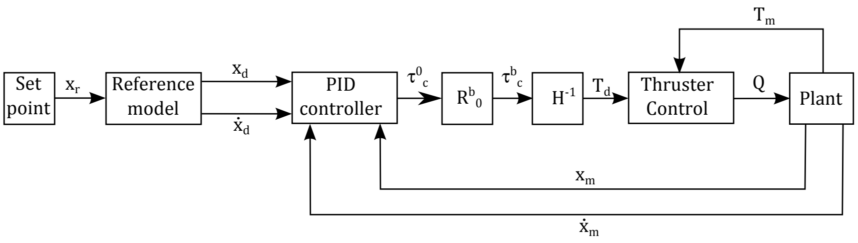


Figure 4.28: Vehicle control system layout.

### 4.2.5 Vehicle Control System

We now equip the vehicle with control system. This will enable sending desired position and orientation commands rather than commanding the individual thruster motors directly.

The general control layout is shown in figure 4.28. The user inputs are taken as the set points. These are the signals in the $6 \times 1$ vector $\boldsymbol{x}_r$, representing the reference position and orientation for the vehicle. The reference position is given as coordinates along the axis of the inertial reference frame, i.e., north, east and down position, whereas the orientation is given in terms of Euler angles, i.e roll, pitch and yaw angle. The reference signal $\boldsymbol{x}_r$ is input to the reference model. This is a second order filter with velocity saturation given by (Fossen, 2011)

$$\begin{aligned} \boldsymbol{M}^{rov}_{ref}\ddot{\boldsymbol{x}}_d + \boldsymbol{D}^{rov}_{ref}\dot{\boldsymbol{x}}_d + \boldsymbol{K}^{rov}_{ref}\boldsymbol{x}_d &= \boldsymbol{K}^{rov}_{ref}\boldsymbol{x}_r \\ \dot{\boldsymbol{x}}_d &= \text{sat}(\dot{\boldsymbol{x}}_{min},\, \dot{\boldsymbol{x}}_{max}) \end{aligned} \tag{4.77}$$

which ensures a smooth signal for the desired position and orientation, as well as for the desired linear and angular velocity. The signal vector $\boldsymbol{x}_d$ is the desired position and orientation, and the $6 \times 6$ diagonal matrices $\boldsymbol{M}^{rov}_{ref}$, $\boldsymbol{D}^{rov}_{ref}$ and $\boldsymbol{K}^{rov}_{ref}$, are design matrices. The function *sat(·)* is the saturation function applied to the velocity in order to make sure that the vehicle can keep up with the reference model.

The outputs $\boldsymbol{x}_d$ and $\dot{\boldsymbol{x}}_d$, from the reference model, are the inputs to the controller of the vehicle, along with the corresponding measured states $\boldsymbol{x}_m$ and $\dot{\boldsymbol{x}}_m$. Note that it is here assumed that there exists perfect measurements of the position, orientation and velocity. This is generally not the case, and an observer would usually provided estimates to these states (Fossen, 2011). The controller is, as seen in figure 4.28, a proportional integral derivative (PID) controller, given by the equations

$$\begin{aligned}\boldsymbol{e}_1 &= \boldsymbol{x}_d - \boldsymbol{x}_m \\ \boldsymbol{e}_2 &= \dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}_m \\ \boldsymbol{e}_3 &= \int_{\tau=0}^{t} \boldsymbol{e}_1 d\tau \\ \boldsymbol{\tau}_c^0 &= \boldsymbol{K}_p^{rov}\boldsymbol{e}_1 + \boldsymbol{K}_d^{rov}\boldsymbol{e}_2 + \boldsymbol{K}_i^{rov}\boldsymbol{e}_3\end{aligned} \tag{4.78}$$

where $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, and $\boldsymbol{e}_3$, are the position-orientation error, the velocity error, and the integral error. The gain matrices $\boldsymbol{K}_p^{rov}$, $\boldsymbol{K}_d^{rov}$, and $\boldsymbol{K}_i^{rov}$, are the corresponding $6 \times 6$ diagonal and positive definite gain matrices. Finally, $\boldsymbol{\tau}_c^0$ is the commanded force and torque vector to the thruster system, expressed in terms of the inertial reference frame. The thruster system does however operate in terms of the vehicle body fixed reference frame, and it is necessary to transform the representation of the commanded force and torque to this reference frame. This is achieved with the transformation

$$\boldsymbol{\tau}_c^b = \begin{bmatrix} \boldsymbol{R}_0^b & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{R}_0^b \end{bmatrix} \boldsymbol{\tau}_c^0 \tag{4.79}$$

The commanded forces and torques $\boldsymbol{\tau}_c^b$ to be delivered by the thruster system, now need to be translated into the signal of individual thruster commands, $\boldsymbol{T}_d$. These are the desired individual thrust forces input to the thruster control systems. In order to design the thruster control system, it is assumed that the thrust produced by each thruster is measurable. This measurement, denoted $\boldsymbol{T}_m$, is also input to the thruster control system. This controller is characterized as a proportional integral (PI) controller given as

$$\begin{aligned}\boldsymbol{e}_T &= \boldsymbol{T}_d - \boldsymbol{T}_m \\ \boldsymbol{Q} &= \boldsymbol{K}_{pT}\boldsymbol{e}_T + \boldsymbol{K}_{iT}\int_{\tau=0}^{t} \boldsymbol{e}_T d\tau\end{aligned} \tag{4.80}$$

where $\boldsymbol{e}_T$ is the error between the desired and measured thrust, $\boldsymbol{Q}$ is the commanded torque for the thruster motors, which in this case study are exactly the produced torques as the motors are modelled as effort sources, and the $6 \times 6$ diagonal and positive definite gain matrices for the proportional and integral gains are $\boldsymbol{K}_{pT}$ and $\boldsymbol{K}_{iT}$.

### 4.2.6 Manipulator Control System

A control system is also designed for the manipulator. Recall from the previous case study that a control system were designed for the Titan 4 manipulator with hydraulic actuators. The control design for the electrically actuated manipulator of this case study is less complicated. Figure 4.29 show the general layout of the control system.

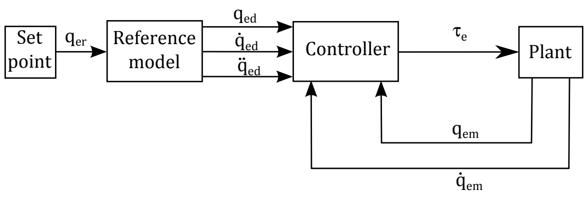


Figure 4.29: Manipulator control system layout.

The user input to this control system is the vector of reference joint angles, $\boldsymbol{q}_{ed}$. These are filtered through the reference model given by Fossen (2011) as

$$\begin{aligned} \boldsymbol{M}_{ref}^{man}\ddot{\boldsymbol{q}}_{ed} + \boldsymbol{D}_{ref}^{man}\dot{\boldsymbol{q}}_{ed} + \boldsymbol{K}_{ref}^{man}\boldsymbol{q}_{ed} &= \boldsymbol{K}_{ref}^{man}\boldsymbol{q}_{er} \\ \dot{\boldsymbol{q}}_{ed} &= \text{sat}(\dot{\boldsymbol{q}}_{e,min}, \dot{\boldsymbol{q}}_{e,max}) \end{aligned} \tag{4.81}$$

where $\boldsymbol{M}_{ref}^{man}$, $\boldsymbol{D}_{ref}^{man}$, and $\boldsymbol{K}_{ref}^{man}$ are $3 \times 3$ matrices of reference model parameters, and $\dot{\boldsymbol{q}}_{e,min}$ and $\dot{\boldsymbol{q}}_{e,max}$ are the vectors of minimum and maximum angular rate of the manipulator joints.

The manipulator controller block in figure 4.29 takes the desired joint angles, $\boldsymbol{q}_{ed}$, the desired joint angular rate $\dot{\boldsymbol{q}}_{ed}$, the desired joint angular acceleration $\ddot{\boldsymbol{q}}_{ed}$, along with the measurements $\boldsymbol{q}_{em}$ and $\dot{\boldsymbol{q}}_{em}$, as inputs. A suitable controller law is found by following the first steps of the controller design in the appendix A.2. The manipulator model, isolated from the vehicle dynamics, can be expressed as

$$\begin{aligned} \dot{\boldsymbol{q}}_e &= \boldsymbol{\omega}_e \\ \dot{\boldsymbol{\omega}}_e &= \boldsymbol{B}_e^{-1}(\boldsymbol{q}_e)(-\boldsymbol{C}_e(\boldsymbol{q}_e, \boldsymbol{\omega}_e)\boldsymbol{\omega}_e - \boldsymbol{\tau}_f - \boldsymbol{g}_e(\boldsymbol{q}_e) + \boldsymbol{\tau}_e) \end{aligned} \tag{4.82}$$

where $\boldsymbol{q}_e$ are the vector of joint angles, $\boldsymbol{B}_e(\boldsymbol{q}_e)$ is the manipulator mass-inertia matrix, $\boldsymbol{C}_e(\boldsymbol{q}_e, \boldsymbol{\omega}_e)$ is the Coriolis and centrifugal matrix , $\boldsymbol{g}_e(\boldsymbol{q}_e)$ is the restoring forces of the manipulator, $\boldsymbol{\tau}_f$ are the friction forces, and $\boldsymbol{\tau}_e$ are the actuator torques. The error states can then be defined as

$$\begin{aligned} \boldsymbol{e}_{e1} &= \boldsymbol{q}_e - \boldsymbol{q}_{ed} \\ \boldsymbol{e}_{e2} &= \boldsymbol{\omega}_e - \dot{\boldsymbol{q}}_{ed} \end{aligned} \tag{4.83}$$

The error dynamics are found by time differentiating the error states, which yields

$$\begin{aligned} \dot{\boldsymbol{e}}_{e1} &= \boldsymbol{\omega}_e - \dot{\boldsymbol{q}}_e \\ \dot{\boldsymbol{e}}_{e2} &= \dot{\boldsymbol{\omega}}_e - \ddot{\boldsymbol{q}}_e \end{aligned} \tag{4.84}$$

By substituting $\dot{\boldsymbol{\omega}}_e$ for (4.82) in (4.84), we obtain

$$\dot{\boldsymbol{e}}_{e2} = \boldsymbol{B}_e^{-1}(\boldsymbol{q}_e)(-\boldsymbol{C}_e(\boldsymbol{q}_e, \boldsymbol{\omega}_e)\boldsymbol{\omega}_e - \boldsymbol{\tau}_f - \boldsymbol{g}_e(\boldsymbol{q}_e) + \boldsymbol{\tau}_e) + \ddot{q}_{ed} \tag{4.85}$$

The control law is now chosen as

$$\begin{aligned}\boldsymbol{\tau}_e &= \boldsymbol{C}_e(\boldsymbol{q}_{em}, \dot{\boldsymbol{q}}_{em})\dot{\boldsymbol{q}}_{ed} + \boldsymbol{g}_e(\boldsymbol{q}_{em}) + \boldsymbol{B}_e(\boldsymbol{q}_{em})\ddot{\boldsymbol{q}}_{ed} \\ &\quad + \boldsymbol{K}_p^{man}\boldsymbol{e}_{e1} - \boldsymbol{K}_d^{man}\boldsymbol{e}_{e2} - \boldsymbol{K}_i^{man}\int_{\tau=0}^{t} \boldsymbol{e}_{e1} d\tau\end{aligned} \tag{4.86}$$

Assuming that the measurements $\boldsymbol{q}_{em}$ and $\dot{\boldsymbol{q}}_{em}$ are perfectly correct, this control law result in the error dynamics

$$\begin{aligned}\dot{\boldsymbol{e}}_{e1} &= \boldsymbol{\omega}_e - \dot{\boldsymbol{q}}_{ed} \\ \dot{\boldsymbol{e}}_{e2} &= \boldsymbol{B}_e^{-1}(-\boldsymbol{C}_e(\boldsymbol{q}_e, \boldsymbol{\omega}_e) - \boldsymbol{\tau}_f \\ &\quad - \boldsymbol{K}_p^{man}\boldsymbol{e}_{e1} - \boldsymbol{K}_d^{man}\boldsymbol{e}_{e2} - \boldsymbol{K}_i^{man}\int_{\tau=0}^{t} \boldsymbol{e}_{e1} d\tau)\end{aligned} \tag{4.87}$$

### 4.2.7 External Object Interface

Recall that the main purpose of this case study is to show how the manipulator end effector can interface an external object such that the ROV-manipulator system can lift the object and drop it at a new location. In the following, a method for achieving this in the bond graph framework is presented.

We start by establishing an interface for the manipulator end effector by placing 1-junctions representing the velocity of the end effector. The linear and angular velocity of the end effector, in terms of the inertial reference frame, can be found as

$$\begin{aligned}\boldsymbol{v}_{ee/0}^{0} &= \boldsymbol{J}_{ee}^{v}(\boldsymbol{q})\boldsymbol{\omega} \\ \boldsymbol{\omega}_{ee/0}^{0} &= \boldsymbol{J}_{ee}^{\omega}(\boldsymbol{q})\boldsymbol{\omega}\end{aligned} \tag{4.88}$$

where the subscript *ee* point to the end effector, $\boldsymbol{\omega}$ is the vector of quasi coordinates, and $\boldsymbol{J}_{ee}^{v}(\boldsymbol{q})$ and $\boldsymbol{J}_{ee}^{\omega}$ are geometric Jacobian matrices, found according to the method presented in 2.3. Using these transformations, 1-junctions representing the linear and angular velocity respectively, can be placed in the bond graph. Figure 4.30 show how this can be done, where the geometric Jacobian matrices of (4.88) are implemented in the usual fashion in the transformer elements denoted *MTF:D1* and *MTF:D2*.

In order to connect the object to the end effector, it is necessary to introduce some elasticity between the end effector and the object. To this effect we model springs between the end effector and the object. This can be thought of as the elasticity in some handle by which the end effector grip the object. In order to avoid vibrations, dampers are modelled along with the springs. The springs and dampers are also used in order to define whether or not the object is gripped by the end effector. This is achieved by setting the spring stiffness to zero whenever the end effector does not grip the object.

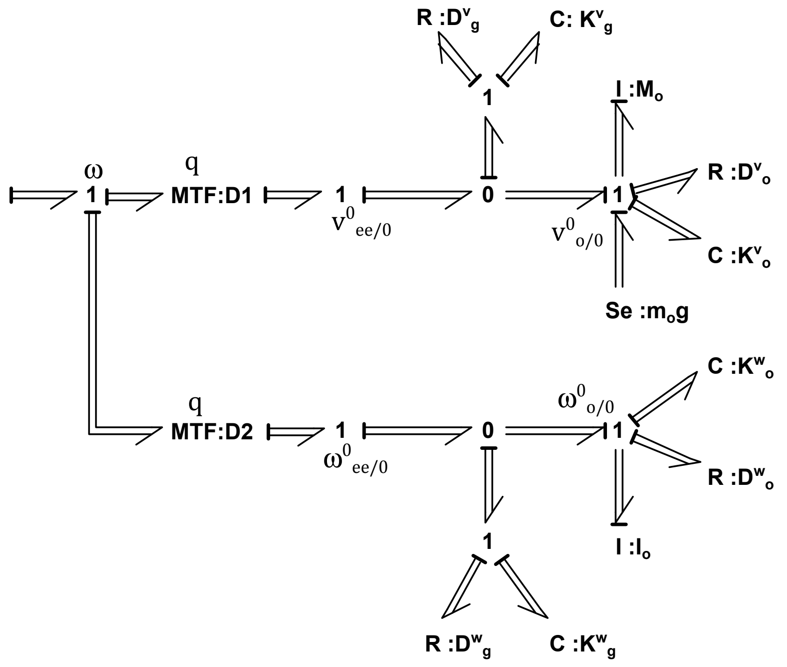


Figure 4.30: Bond graph of end effector and external object to be lifted.

Finally, the object itself is modelled. This is done in the bond graph by placing two 1-junctions. One representing the linear velocity of the object, and the other representing the angular velocity. These can be seen as the 1-junctions denoted $\boldsymbol{v}^0_{o/0}$ and $\boldsymbol{\omega}^0_{o/0}$ in figure 4.30. The springs and dampers representing the elasticity in the grip of the end effector, can be seen connected to the 0-junctions between the end effector and the object velocities. The matrices $\boldsymbol{D}^v_g$ and $\boldsymbol{D}^\omega_g$ are the $3 \times 3$ diagonal matrices of damping coefficients for the relative velocity between the end effector and the object, while the matrices $\boldsymbol{K}^v_g$ and $\boldsymbol{K}^\omega_g$ are the matrices of spring stiffness coefficients for the relative displacement between the object and the end effector. In this case study it is assumed that during the time when the object is lifted, it is allowed to rotate almost freely, as if lifted by a short rope, fixed at the center of gravity, while the linear displacement is associated to great stiffness.

We also model the elasticity of the of the support on which the object rests when it is not lifted by the manipulator. In this case study, this support is modelled as a surface in the inertial $xy$-plane. In the C-elements and R-elements in figure 4.30 connected to the 1-junctions representing the linear and angular velocity of the object, logic operators monitor the $z$ position of the object. Once the object comes in contact with the defined surface, the spring stiffness coefficients and the damping coefficients of the diagonal $3 \times 3$ matrices $\boldsymbol{K}_o^v$, $\boldsymbol{K}_o^\omega$, $\boldsymbol{D}_o^v$, and $\boldsymbol{D}_o^\omega$ are activated, i.e., they go from zero to appropriate values. The matrices $\boldsymbol{M}_o$ and $\boldsymbol{I}_o$ of the two I-elements on the 1-junctions representing the linear and angular velocity of the object, are the mass and inertia matrices of the object.

### 4.2.8 Simulations

Figure 4.31 show the final bond graph, as implemented in 20-sim, of the system, including all sub models described in the preceding sections. A simulation of an underwater lifting operation is carried out using this model. The parameters used in the simulation are summarized in appendix C. The main visual presentation of the simulation results is a video animation which can be found in the appended files of this thesis. Figure 4.32 show a screen shot of this video animation. This animation includes the remotely operated vehicle with thrust vectors, the manipulator, the object to be lifted, and the pillars which the object is moved between. The 3D animation is created using the 3D animation toolbox of 20-sim, similarly as for the Titan 4 simulator of the previous case study.

In addition to the video animation, time series of the vehicle position and orientation, along with the corresponding reference signals to the vehicle control system, are presented in figure 4.33. The manipulator joint angles along with the corresponding reference signal for the manipulator controller, are presented in figure 4.34. In both time series, the blue vertical line represent the time instance in which the external object is gripped by the manipulator, while the red vertical line represents the time instance in which the object is placed down. The effect of the object is apparent in both figures. Notice in particular that the vehicle is not able to maintain its altitude when lifting and dropping the object. Also the roll and pitch angle is affected by the weight of the object. The time series also show that all three manipulator joints are affected by the object. The innermost joint is affected by the additional inertia to the system by the object when the vehicle alter the yaw angle, while the two subsequent joints are affected by the weight of the object. Notice that the integrator on the controller of the manipulator just manage to bring the third joint angle back to the the reference value, before dropping the mass, while the second joint angle is not able follow the reference value during the lift.

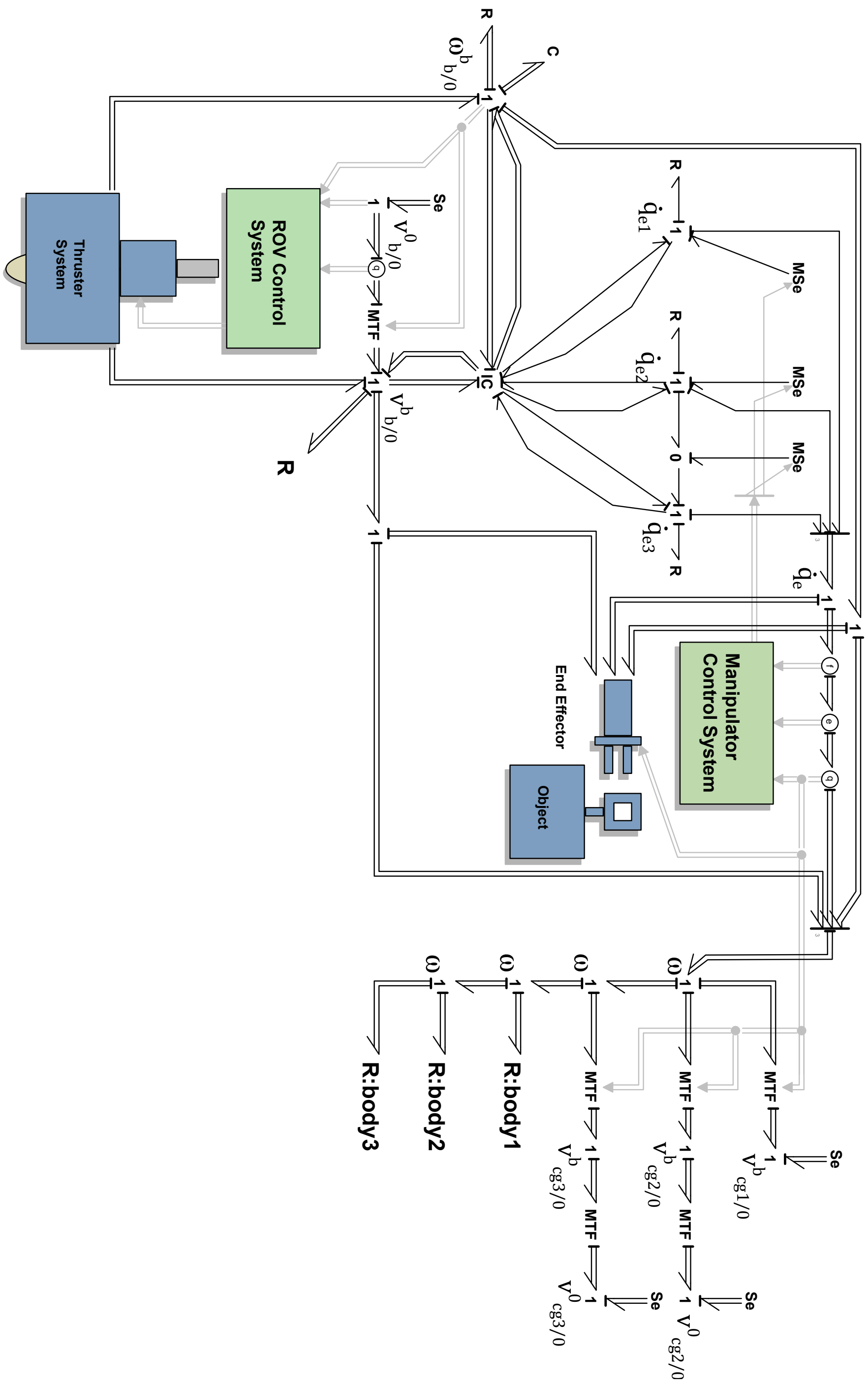


Figure 4.31: Final bond graph as implemented in 20-sim.

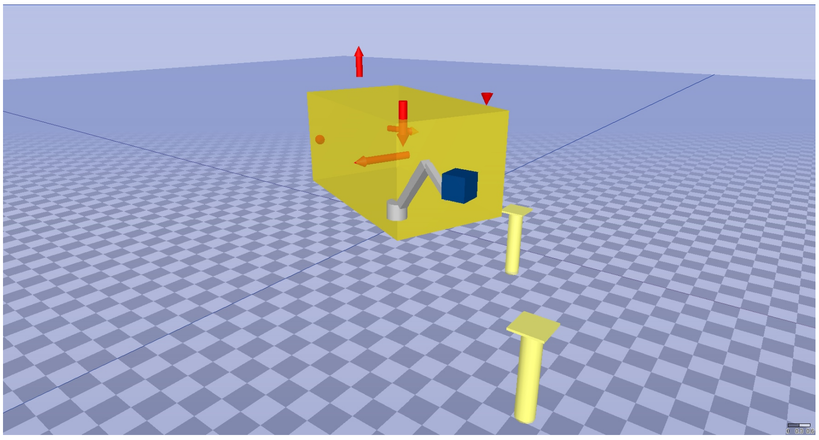


Figure 4.32: Screen shot from video animation of underwater lifting operation.

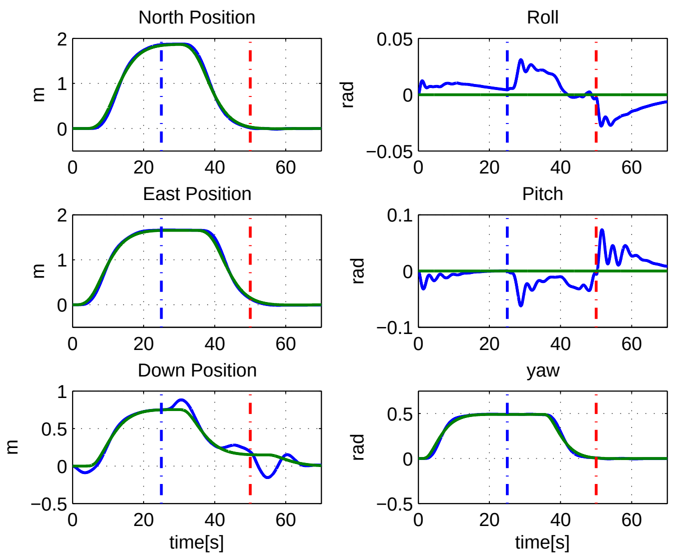


Figure 4.33: Vehicle position, orientation and reference signals during lifting operation. The blue lines are the states, while the green lines are the filtered reference signals.

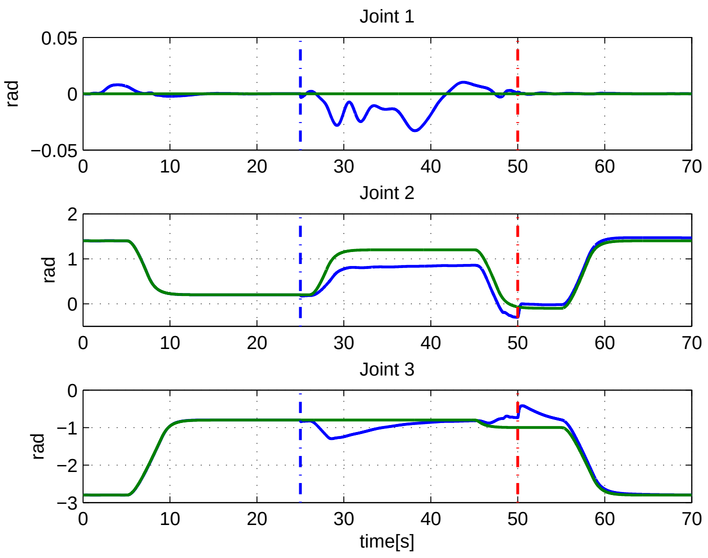


Figure 4.34: Manipulator joint angles and joint angle reference signal during lifting operation. The blue lines are the actual states while the green lines are the filtered reference signals.

Chapter 5

# Conclusion and Further Work

This thesis has focused on developing an efficient and flexible manner in which to produce and implement dynamical models of systems of rigid bodies in spatial motion on computers. The method presented was based on Lagrangian dynamics and bond graph modelling, and implemented in the bond graph software 20-sim. In order to achieve this, symbolic expressions were found in the symbolic software Maple and exported to optimized C-code. The C-code was used in order to build a dynamic link library file which was utilized by 20-sim in order to update the relevant variables for each integration step in the numerical integration of the model. The theoretical approach of the method was derived and applied to a simple example system with an inverted pendulum situated on a moving wagon. Then the method for efficiently implementing the model in 20-sim were explained and demonstrated by continuing on the same example, showing maple algorithm, C-code and the relevant code from the 20-sim implementation. A focus of this thesis have been effective production and implementation of dynamics models in bond graph software. At this stage, one of the more time consuming tasks in implementing the model on computers, is the conversion of the raw C-code output from Maple to the finished C-code from which the DLL-file can be built. In order to simplify this task, it is suggested that a program for doing the necessary alterations are made. As an example, the program could query about the number of generalized coordinates and their name in the Maple code, and replace any occurrence of the Maple names in the code by *inarr[0], inarr[1], ..., inarr[n-1]*. The program could also make inquiries about the functions to be defined for communicating with the 20-sim external interface, and print these functions to the new C-code.

Real applications of the method were demonstrated in two case studies. First a high fidelity simulator for the robotic manipulator Titan 4 were developed. This model run real-time, taking input from a joystick. This example demonstrated how a hydraulic actuator system conveniently could be connected to the simulator, using the bond graph language. In the second case study, a dynamic model of a remotely operated vehicle, equipped with a manipulator were developed. During this process, the virtues of this approach was further demonstrated by connecting various sub systems such as a vehicle thruster system to the basic model. The purpose of the models developed for the case studies was not to include any conceivable dynamical effect. Rather, the purpose was to demonstrate how some of the dynamical effects commonly modelled for such systems could be included by using the framework presented in this thesis. However, both the models developed provides a solid base for further development of both a Titan 4 manipulator model, and an interconnected underwater vehicle and manipulator model. If the models are to be further developed, one of the most important subsystems to upgrade are the manipulator joint friction models. It would also be of interest to include current loads and a more advanced model for the hydrodynamic added mass and damping to the vehicle and manipulator model. Furthermore, the Titan 4 simulator could be modelled under water by including buoyancy forces, as well as hydrodynamic added mass and damping in the same manner as is done in case study two. For further development of this model as a simulator, it might also be of interest to replace the current joystick by the Titan 4 master controller. This is a joystick which resembles the manipulator, making the control easier. This however would require the development of an interface between 20-sim and the Titan 4 master controller. A second alternative would be to develop the inverse kinematics of the manipulator, such that the three analogous axis of the current joystick could be used in order to control the position of the end effector directly.

# Bibliography

Angue-Mintsa, H., Venugopal, R., Kenné, J.-P., and Belleau, C. (2011). Adaptive position control of an electrohydraulic servo system with load disturbance rejection and friction compensation. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 133(6). cited By (since 1996)0.

Berger, R., M., ElMaraghy, H., A., and ElMaraghy, W., H. (1990). The analysis of simple robots using bond graphs. *Journal of Manufacturing Systems.*

Faltinsen, O. (1993). *Sea Loads on Ships and Offshore Structures.* Cambridge Ocean Technology Series. Cambridge University Press.

Fossen, T. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control.* 1 edition.

Ginsberg, J. (1995). *Advanced Engineering Dynamics.* 2 edition.

Healey, A., Rock, S., Cody, S., Miles, D., and Brown, J. (1995). Toward an improved understanding of thruster dynamics for underwater vehicles. *Oceanic Engineering, IEEE Journal of*, 20(4):354–361.

Hunt, L., H. and Turney, J. (2000). *Cygwin User's Guide.*

Karnopp, D., C., Margolis, D., L., and Rosenberg, R., C. (2006). *System Dynamics: Modelling and Simulation of Mechatronic Systems.* 4 edition.

Karnopp, D. (1969). Power-conserving transformations: physical interpretations and applications using bond graphs. *Journal of the Franklin Institute*, 288(3).

Kernighan, B. W. (1988). *The C Programming Language.* Prentice Hall Professional Technical Reference, 2nd edition.

Khalil, H., K. (2002). *Nonlinear Systems.* Prentice Hall PTR.

Kim, J., Chung, W. K., and Yuh, J. (2003). Dynamic analysis and two-time scale control for underwater vehicle-manipulator systems. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 577–582 vol.1.

Klein, C., Groothuis, M., A., and Differ, H., G. (2013). *20-sim 4.3 Reference Manual, Enscheede, Controllab Products B.V.*

Maplesoft (2014). *Maple User Manual. Toronto: Maplesoft, a division of Waterloo Maple Inc.*

McLain, T. W. and Rock, S. M. (1998). Development and experimental validation of an underwater manipulator hydrodynamic model. *I. J. Robotic Res.*, 17(7):748–759.

Meirovitch, L. (2003). *Methods of Analytical Dynamics.* 1 edition.

Pedersen, E. (2012). Bond graph modeling of marine vehicle dynamics. *Bond Graph Modeling: Theory and Practice Symposium at the 7th Vienna International Conference on Mathematical Modeling.*

Pedersen, E. and Engja, H. (2008). *Mathematical Modelling and Simulation of Physical Systems : Lecture Notes in course TMR4275 Modelling, simulation and analysis of dynamic systems.*

Sciavicco, L. and Siciliano, B. (2000). *Modelling and Control of Robot Manipulators.* 1 edition.

Soylu, S., Firmani, F., Buckham, B., and Podhorodeski, R. (2010). Comprehensive underwater vehicle-manipulator system teleoperation. In *OCEANS 2010*, pages 1–8.

Vaz, A., Kansal, H., and Singla, A. (2003). Some aspects in the bond graph modelling of robotic manipulators: angular velocities from symbolic manipulation of rotation matrices. *TENCON 2003. Conference on Convergent Technologies for the Asia-Pacific Region.*

Zeng, H. and Sepehri, N. (2006). Adaptive backstepping control of hydraulic manipulators with friction compensation using lugre model. volume 2006, pages 3164–3169. cited By (since 1996)8.

# Appendix A

# Appended Papers

In the following sections the two papers written as a part of this Masters Thesis are attached.

## A.1 Paper 1 - Bond Graph Modelling of Marine Vehicle with Manipulator Equipment

# Bond Graph Modelling of Marine Vehicle with Manipulator Equipment

**B. Rokseth** * **E. Pedersen** **

* *Department of Marine Technology, Norwegian University of Science and Technology (NTNU),7491 Trondheim, Norway (e-mail: borgerok@stud.ntnu.no)*
** *Department of Marine Technology, Norwegian University of Science and Technology (NTNU),7491 Trondheim, Norway (e-mail: eilif.pedersen@ntnu.no)*

**Abstract:** Both marine surface vehicles and underwater vehicles are often equipped with cranes, robotic manipulators or similar equipment. Much attention is awarded to modelling of both the dynamics of marine vehicles and the dynamics of manipulators and cranes. However, less attention is given to the interconnected behaviour of the vehicle and the equipment, even though such equipment may have a serious impact on the vehicle dynamic behaviour, or conversely, the vehicle may have a serious impact on the equipment dynamic behaviour. In this paper, we develop a method for modelling the interconnected dynamics of such systems. The basic dynamics of the system is modelled using the Lagrange method. The resulting equations are then implemented in the bond graph language. This bond graph can be used as a basis in modelling of both surface and underwater vehicles with various equipment, typical examples being remotely operated vehicle (ROV) with robotic manipulator, or ship with deck crane. The level of fidelity of the basic model can be enhanced by connecting or enhancing various sub models to the bond graph such as thruster models, hydrodynamic damping models, equipment actuators models, friction models, and wave and current load models.

*Keywords:* Bond Graphs, Dynamic Modelling, Marine Vehicles, Lagrange Dynamics, Quasi-coordinates

## 1. INTRODUCTION

The purpose of this paper is to develop a bond graph frame work for mathematical modelling of the interconnected dynamics of a marine vehicle together with equipment such as cranes or robotic manipulators. This frame work can be applied to any type of marine, or space vehicle, with any type of equipment. We will focus the paper on marine vehicles with lower pair jointed, open link structured equipment. With this we mean equipment constituting an open chain of linked bodies where each body is linked to the next through a joint with one degree of freedom.

There are numerous applications in which this kind of interconnected model can be useful. In a ROV-manipulator context, such models can be used in order to develop unified model based controllers for the interconnected dynamics. Real ROV simulators may also be designed in order to train personnel for high precision tasks. An other application can be for ships with heavy decks cranes, where the model can be used to investigate the ship reaction to heavy lifting in various weather conditions. In this case an interconnected model can be used as a tool for determining in what configurations the crane can safely operate under given weather conditions.

Even though interconnected models are developed for ROV and manipulator in e.g. Soylu et al. (2010) and Kim et al. (2003), we believe that a bond graph approach to the problem will be useful in that model developers easily can connect different sub models to the system in order to enhance the basic model to the desired level. As an example, hydrodynamic damping models and environmental loads such as waves and currents from e.g Faltinsen (1993) and Fossen (2011) can easily be connected to both the vehicle and the equipment within the bond graph frame work. Similarly, actuators systems for both the vehicle and the equipment can be interfaced to the modelling frame work developed here.

Dynamic equations will be derived using the Lagrangian method with quasi-coordinates, as derived in Meirovitch (2003). Only the dynamics associated to the kinetic energy will be modelled in the Lagrange equations, because the dynamics associated to the potential energy conveniently can be accounted for directly in the bond graph. The Lagrangian equations will be implemented through an IC-field in the bond graph, using the quasi-state momentum and the generalized displacement as states as shown in Karnopp et al. (2006).

In the following section we model the dynamics of the vehicle in a manner similar to that presented in Pedersen (2012), where the resulting equations are well suited for bond graph implementation. During this section, the purpose of the before mentioned quasi coordinates will be made clear. In section 3, we extend the model to include the equipment dynamics, while keeping the convenient structure of the model. In section 4 we proceed to implement the model in the bond graph language. This bond graph is then extended by introducing restoring forces to both the vehicle and the equipment. These are the forces that comes from the potential energy terms in the Lagrangian method. As these forces are vitally important for any vehicle and equipment, we derive these in as much detail as possible while still retaining the level of generality of the basic model. Finally we establish interfaces to some of the most common sub systems and hydrodynamic forces. The purpose of this is to show how different sub systems can be connected to the basic model, and as such, we do not go into details about the various systems.

## 2. MARINE VEHICLE DYNAMICS

In this section we seek to find equations of motion for the marine vehicle, using momentum and displacement as states. A state space model expressed in terms of these states is convenient for bond graph implementation, as will be seen later. Besides, this state space model is far easier to derive from the Lagrangian equations than are the traditional state space model, with displacement and displacement rates as states. This is mainly because we avoid the tedious task of time differentiating the mass-inertia matrix when using momentum, as opposed to displacement rate.

### *2.1 Some Kinematic Considerations*

Let the position and orientation of the vehicle be given relative to an inertial reference frame, denoted by 0. We attach a second reference frame to the vehicle body and denote it $b$. Then the position of the vehicle is given by the vector $\boldsymbol{r}^0_{b/0}$, where the superscript indicate that the vector is expressed in terms of the inertial reference frame, while the subscript $b/0$ indicate that the vector give the position to the origin of the vehicle body fixed reference frame relative to the origin of the inertial reference frame. The orientation of the the vehicle is given by the Euler angles $\boldsymbol{\Theta} = [\phi,\, \theta,\, \psi]^T$. In this paper the Euler angles are defined such that if the vehicle is rotated an angle $\phi$ about its x-axis, an angle $\theta$ about the resulting y-axis, and finally an angle $\psi$ about the resulting z-axis, then the body fixed coordinate frame have the same orientation as the inertial reference frame. Using this, we can find an expression for the angular velocity of the vehicle, expressed in terms of the body fixed reference frame as

$$\boldsymbol{\omega}^b_{b/0} = \boldsymbol{i}_b\dot{\phi} + \boldsymbol{j}^{'}_b\dot{\theta} + \boldsymbol{k}^{''}_b\dot{\psi} = \boldsymbol{T}^{-1}_{\Theta}(\boldsymbol{\Theta})\dot{\boldsymbol{\Theta}} \tag{1}$$

where $\boldsymbol{i}_b$ is the unit normal vector along the x-axis of the vehicle body fixed reference frame, $\boldsymbol{j}^{'}_b$ is the unit normal vector along the y-axis of the reference frame resulting from the rotation $\phi$, and $\boldsymbol{k}^{''}_b$ is the unit normal vector along the z-axis of the reference frame resulting from the rotation $\theta$. The $3 \times 3$ matrix $\boldsymbol{T}^{-1}_{\Theta}$ is then defined as $\boldsymbol{T}^{-1}_{\Theta} = [\boldsymbol{i}_b,\, \boldsymbol{j}^{'}_b,\, \boldsymbol{k}^{''}_b]$. Expressions for the unit normal vectors along the axis of the intermediate reference frames can be found by using the principal rotation matrices for the sequence of rotations described above. Consider first a coordinate, $\boldsymbol{c}^b$, expressed in terms of the vehicle body fixed reference frame. This coordinate can be expressed in terms of the first intermediate reference frame, i.e., the reference frame resulting from the rotation $\phi$ about the body fixed x-axis, by using the principal rotation matrix $\boldsymbol{R}_x(\phi)$ as

$$\boldsymbol{c}^{'} = \boldsymbol{R}_x(\phi)\boldsymbol{c}^b \tag{2}$$

Because the rotation matrix is orthogonal, we can write the inverse of the matrix as $\boldsymbol{R}^{-1}_{\boldsymbol{x}} = \boldsymbol{R}^T_{x}$. Using this, an expression for the unit normal vector along the y-axis of the first intermediate reference frame can be expressed in terms of the body fixed reference frame as

$$\boldsymbol{j}^{'}_b = \boldsymbol{R}^T_x(\phi)\boldsymbol{j}_b \tag{3}$$

where $\boldsymbol{j}_b = [0,\, 1,\, 0]^T$ is the unit normal vector along the y-axis of the body fixed reference frame. Similarly, the coordinate $\boldsymbol{c}^{'}$, can be expressed in terms of the second intermediate reference frame, resulting from the rotation $\theta$ about $\boldsymbol{j}^{'}_b$, by using the rotation matrix $\boldsymbol{R}_y(\theta)$, such that

$$\boldsymbol{c}^{''} = \boldsymbol{R}_y(\theta)\boldsymbol{c}^{'} \tag{4}$$

Using this expression, we find that the unit normal vector $\boldsymbol{k}^{''}_b$ can be expressed as

$$\boldsymbol{k}^{''}_b = \boldsymbol{R}^T_y(\theta)\boldsymbol{k}^{'}_b = \boldsymbol{R}^T_x\boldsymbol{R}^T_y\boldsymbol{k}_b \tag{5}$$

where $\boldsymbol{k}_b = [0,\, 0,\, 1]^T$. With these transformations defined, we can express the transformation matrix of (1) as

$$\boldsymbol{T}^{-1}_{\Theta}(\boldsymbol{\Theta}) = \left[\, \boldsymbol{i}_b,\ \boldsymbol{R}^T_x\boldsymbol{j}_b,\ \boldsymbol{R}^T_x\boldsymbol{R}^T_y\boldsymbol{k}_b \,\right] \tag{6}$$

The final principal rotation matrix $\boldsymbol{R}_z(\psi)$ can be used in order to transform a coordinate expressed in terms of the second intermediate reference frame, to be expressed in terms of the inertial reference frame. We can now design the rotation matrix transforming a coordinate representation from the vehicle body fixed reference frame to the inertial reference frame as

$$\boldsymbol{R}^0_b = \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi) \tag{7}$$

with

$$\begin{aligned}\boldsymbol{R}_z(\psi) &= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \boldsymbol{R}_y(\theta) = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \\ \boldsymbol{R}_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix}\end{aligned} \tag{8}$$

and $s_x = sin(x)$, and $c_x = cos(x)$. We can now write

$$\boldsymbol{c}^0 = \boldsymbol{R}^0_b\boldsymbol{c}^b \tag{9}$$

This rotation matrix, as with the principal rotation matrices, is orthogonal such that

$$\left(\boldsymbol{R}^0_b\right)^{-1} = \left(\boldsymbol{R}^0_b\right)^T = \boldsymbol{R}^b_0 \tag{10}$$

### *2.2 Kinetic Energy of the Vehicle*

The kinetic energy of the vehicle can be expressed as

$$T = \frac{1}{2}\left((\boldsymbol{v}^b_{cg/0})^T\boldsymbol{M}\boldsymbol{v}_{cg/0} + (\boldsymbol{\omega}^b_{b/0})^T\boldsymbol{I}_g\boldsymbol{\omega}^0_{b/0}\right) \tag{11}$$

where $\boldsymbol{M} = m\boldsymbol{I}_{3\times3}$, $m$ is the mass of the vehicle, $\boldsymbol{I}_{3\times3}$ is the identity matrix, $\boldsymbol{I}_g$ is the vehicle inertia tensor, and $\boldsymbol{v}_{cg/0}$ is the linear velocity of the vehicle center of gravity relative to the inertial reference frame. However, using the Lagrangian approach, the kinetic energy should be expressed in terms of a set of generalized coordinates and their rates. The generalized coordinates are a set of coordinates that uniquely define the position and orientation of the vehicle, and are in this paper chosen as

$$\boldsymbol{q} = \left[ \left( \boldsymbol{r}^0_{b/0} \right)^T, \boldsymbol{\Theta}^T \right]^T \tag{12}$$

The linear velocity of the vehicle center of gravity can be expressed in terms of the generalized coordinates as

$$\begin{aligned} \boldsymbol{v}^b_{cg} &= \boldsymbol{v}^b_{b/0} + \boldsymbol{\omega}^b_{b/0} \times \boldsymbol{r}^b_{cg/b} \\ &= \boldsymbol{R}^b_0(\boldsymbol{\Theta})\dot{\boldsymbol{r}}^0_{b/0} + \boldsymbol{T}^{-1}_\Theta(\boldsymbol{\Theta})\dot{\boldsymbol{\Theta}} \times \boldsymbol{r}^b_{cg/b} \end{aligned} \tag{13}$$

where $\boldsymbol{v}^b_{b/0}$ is the velocity of the origin of the vehicle body fixed reference frame, and $\boldsymbol{r}^b_{cg/b}$ is the vector from the origin of the vehicle body fixed reference frame to the center of gravity.

By substituting (1) and (13) in (11), the kinetic energy takes the form $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$. We do however seek to replace the dependency on $\dot{\boldsymbol{q}}$ by the quasi coordinates

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{v}^b_{b/0} \\ \boldsymbol{\omega}^b_{b/0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^b_0 & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}^{-1}_\Theta \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{r}}^0_{b/0} \\ \dot{\boldsymbol{\Theta}} \end{bmatrix} = \boldsymbol{\alpha}^T \dot{\boldsymbol{q}} \tag{14}$$

because this will make the resulting equations of motion dependent on the body fixed linear and angular velocity, rather than the linear velocity in terms of the inertial frame and the Euler angle rates. The inverse of (14) is

$$\dot{\boldsymbol{q}} = \boldsymbol{\beta}\boldsymbol{\omega} \tag{15}$$

where

$$\boldsymbol{\beta} = (\boldsymbol{\alpha}^T)^{-1} = \begin{bmatrix} \boldsymbol{R}^0_b & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}_\Theta \end{bmatrix} \tag{16}$$

Substituting (15) into the expression for $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$, yields the expression $T(\boldsymbol{q}, \boldsymbol{\beta}\boldsymbol{\omega}) = \bar{T}(\boldsymbol{q}, \boldsymbol{\omega})$, which can be found explicitly by finding the linear velocity of the vehicle center of gravity expressed in terms of the quasi-coordinates. This is recognized as the first expression in (13), and can be expressed compactly as

$$\begin{aligned} \boldsymbol{v}^b_{cg/0} &= \left[ \boldsymbol{I}_{3\times3} \ \boldsymbol{i}_b \times \boldsymbol{r}^b_{cg/0} \ \boldsymbol{j}_b \times \boldsymbol{r}^b_{cg/0} \ \boldsymbol{k}_b \times \boldsymbol{r}^b_{cg/0} \right] \boldsymbol{\omega} \\ &\triangleq \boldsymbol{J}^v_b \boldsymbol{\omega} \end{aligned} \tag{17}$$

where $\boldsymbol{J}^v_b$ is the geometric Jacobian matrix for the linear velocity of the center of mass of the vehicle. More trivially, the angular velocity can be expressed in matrix form as

$$\begin{aligned} \boldsymbol{\omega}^b_{b/0} &= [\, \boldsymbol{0}_{3\times3} \ \boldsymbol{I}_{3\times3} \,] \boldsymbol{\omega} \\ &\triangleq \boldsymbol{J}^\omega_b \boldsymbol{\omega} \end{aligned} \tag{18}$$

Now we define the vector $\boldsymbol{v}_b = [(\boldsymbol{v}^b_{cg/0})^T, (\boldsymbol{\omega}^b_{cg/0})^T]^T$, i.e., the linear velocity of the vehicle center of gravity, and the angular velocity of the body collected in the vector $\boldsymbol{v}_b$. This can be expressed compactly as

$$\boldsymbol{v}_b = \begin{bmatrix} \boldsymbol{J}^v_b \\ \boldsymbol{J}^\omega_b \end{bmatrix} \boldsymbol{\omega} = \boldsymbol{J}_b \boldsymbol{\omega} \tag{19}$$

With this, we find the kinetic energy in terms of quasi coordinates as

$$\begin{aligned} \bar{T}_b(\boldsymbol{q}, \boldsymbol{\omega}) &= \frac{1}{2} \boldsymbol{\omega}^T \boldsymbol{J}^T_b \begin{bmatrix} \boldsymbol{M} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_g \end{bmatrix} \boldsymbol{J}_b \boldsymbol{\omega} \\ &\triangleq \frac{1}{2} \boldsymbol{\omega}^T \boldsymbol{B}_b \boldsymbol{\omega} \end{aligned} \tag{20}$$

where $\boldsymbol{B}_b$ is the symmetric and positive definite vehicle mass-inertia matrix.

*2.3 Equations of Motion*

In the traditional Lagrange method, in which the kinetic energy is expressed in terms of generalized coordinates and rates, as opposed to generalized coordinates and quasi-coordinates, the equations of motion takes the form

$$\frac{d}{dt}\left( \frac{\partial T}{\partial \dot{\boldsymbol{q}}} \right) - \frac{\partial T}{\partial \boldsymbol{q}} = \boldsymbol{\tau} \tag{21}$$

where $\boldsymbol{\tau}$ is the vector of generalized coordinates. Note that the potential energy of the system is not included here. When introducing quasi-coordinates, the chain rule must be used when differentiating because the quasi-coordinates are functions of the generalized coordinates and rates. From Meirovitch (2003), we have that the quasi-equations of motion becomes

$$\frac{d}{dt}\left( \frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} \right) + \boldsymbol{\beta}^T \boldsymbol{\gamma} \frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} - \boldsymbol{\beta}^T \frac{\partial \bar{T}}{\partial \boldsymbol{q}} = \boldsymbol{\beta}^T \boldsymbol{\tau} \tag{22}$$

where the $n \times n$ matrix $\boldsymbol{\gamma}$ of (22) is given as

$$\boldsymbol{\gamma} = \begin{bmatrix} \xi_{11} & \cdots & \xi_{1n} \\ \vdots & \ddots & \vdots \\ \xi_{n1} & \cdots & \xi_{nn} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\omega}^T \boldsymbol{\beta}^T \dfrac{\partial \boldsymbol{\alpha}}{\partial q_1} \\ \vdots \\ \boldsymbol{\omega}^T \boldsymbol{\beta}^T \dfrac{\partial \boldsymbol{\alpha}}{\partial q_n} \end{bmatrix} \tag{23}$$

where

$$\xi_{ij} = \boldsymbol{\omega}^T \boldsymbol{\beta}^T \frac{\partial \alpha_{ij}}{\partial \boldsymbol{q}} \tag{24}$$

Note that $\partial\boldsymbol{\alpha}/\partial q_i$ is a square matrix, in which each element $\alpha_{ij}$ are differentiated with respect to $q_i$, whereas $\partial\alpha_{ij}/\partial\boldsymbol{q}$ is a column vector in which the element $\alpha_{ij}$ is differentiated with respect to each of the generalized coordinates.

The kinetic energy differentiated with respect to the velocity constitutes the momentum of the system in question. Thus

$$\dot{\boldsymbol{p}} = \frac{d}{dt}\left( \frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} \right) \tag{25}$$

where $\boldsymbol{p}$ is the momentum of the quasi states, i.e., the momentum expressed in terms of the vehicle body fixed reference frame. Going back to (20), we find that

$$\frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} = \boldsymbol{B}\boldsymbol{\omega} \tag{26}$$

Inverting (26) and substituting $\boldsymbol{p} = \partial\bar{T}/\partial\boldsymbol{\omega}$, yields

$$\boldsymbol{\omega} = \boldsymbol{B}^{-1}\boldsymbol{p} \tag{27}$$

We also find, by comparing (22) and (25), that

$$\begin{aligned} \dot{\boldsymbol{p}} &= -\boldsymbol{\beta}^T \boldsymbol{\gamma} \frac{\partial \bar{T}}{\partial \boldsymbol{\omega}} + \boldsymbol{\beta}^T \frac{\partial \bar{T}}{\partial \boldsymbol{q}} + \boldsymbol{\beta}^T \boldsymbol{\tau} \\ &= \boldsymbol{\beta}^T \boldsymbol{\gamma} \boldsymbol{B} \boldsymbol{\omega} + \frac{1}{2} \boldsymbol{\beta}^T \boldsymbol{\omega}^T \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{q}} \boldsymbol{\omega} + \boldsymbol{\beta}^T \boldsymbol{\tau} \\ &= \boldsymbol{f}_p(\boldsymbol{q}, \boldsymbol{\omega}) + \boldsymbol{\beta}^T \boldsymbol{\tau} \end{aligned} \tag{28}$$

where it is used that

$$\frac{\partial \bar{T}}{\partial \boldsymbol{q}} = \frac{1}{2} \boldsymbol{\omega}^T \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{q}} \boldsymbol{\omega} \tag{29}$$

and

$$\boldsymbol{\omega}^T \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{q}} \boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{\omega}^T \dfrac{\partial \boldsymbol{B}}{\partial q_1} \\ \vdots \\ \boldsymbol{\omega}^T \dfrac{\partial \boldsymbol{B}}{\partial q_n} \end{bmatrix} \boldsymbol{\omega} \tag{30}$$

Note that, in the case of the marine vehicle, the system mass-inertia matrix is not a function of the generalized coordinates, so that $\partial T/\partial \boldsymbol{q} = 0$. We have however included the expression because this will in general not be the case when equipment is added to the system.

Combining (27) and (28) we obtain a state space model describing the basic dynamics of the vehicle, as

$$\begin{aligned} \boldsymbol{\omega} &= \boldsymbol{B}^{-1}\boldsymbol{p} \\ \dot{\boldsymbol{p}} &= \boldsymbol{f}_p(\boldsymbol{q},\boldsymbol{\omega}) + \boldsymbol{\beta}^T\boldsymbol{\tau} \end{aligned} \tag{31}$$

## 3. EXPANDING THE MODEL TO INCLUDE EQUIPMENT DYNAMICS

The system, now defined as the vehicle and the equipment, move in $n = 6 + k$ degrees of freedom, where the vehicle move in 6 degrees of freedom, and the equipment in $k$. In case of lower pair jointed, open chain structured equipment, this means that the equipment have $k$ joints. For such equipment, the obvious choice for generalized coordinates are the joint displacements, denoted $\boldsymbol{q}_e = [q_{e1}, q_{e2}, ..., q_{ek}]^T$. The system vector of generalized coordinates are thus the $n \times 1$ vector $\boldsymbol{q} = [(\boldsymbol{r}^0_{b/0})^T, \boldsymbol{\Theta}^T, \boldsymbol{q}_e^T]^T$. The quasi-coordinates of the equipment are defined simply as the rate of the generalized coordinates of the equipment, such that the system vector of quasi-coordinates are $\boldsymbol{\omega} = [(\boldsymbol{v}^b_{b/0})^T, (\boldsymbol{\omega}^b_{b/0})^T, \dot{\boldsymbol{q}}_e^T]^T$. With these augmented vectors of generalized coordinates and quasi-coordinates, it is necessary to augment the transformation matrices $\boldsymbol{\alpha}^T$, and $\boldsymbol{\beta}$. Recall that we had $\boldsymbol{\omega} = \boldsymbol{\alpha}^T\dot{\boldsymbol{q}}$. Using the expression (14), together with the notation $\dot{\boldsymbol{q}}_e = \boldsymbol{I}_{k\times k}\dot{\boldsymbol{q}}_e$, we find that the augmented $n \times n$ transformation matrix $\boldsymbol{\alpha}^T$ is given as

$$\boldsymbol{\alpha}^T(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{R}^b_0 & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times k} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}^{-1}_\Theta & \boldsymbol{0}_{3\times k} \\ \boldsymbol{0}_{k\times3} & \boldsymbol{0}_{k\times3} & \boldsymbol{I}_{k\times k} \end{bmatrix} \tag{32}$$

The augment inverse transformation matrix is then

$$\boldsymbol{\beta}(\boldsymbol{q}) = \left(\boldsymbol{\alpha}^T\right)^{-1} = \begin{bmatrix} \boldsymbol{R}^0_b & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times k} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}_\Theta & \boldsymbol{0}_{3\times k} \\ \boldsymbol{0}_{k\times3} & \boldsymbol{0}_{k\times3} & \boldsymbol{I}_{k\times k} \end{bmatrix} \tag{33}$$

Before deriving the equations of motion, we shall investigate the kinematics of the system. In particular, we seek to find expressions for the velocity of the center of mass of each of the equipment bodies, as functions of the generalized coordinates and the quasi-coordinates. This in order to find an expression for the system kinetic energy. To this effect, it is necessary to find expressions for the coordinates of each of the bodies center of mass, relative to the preceding joints and the body fixed reference frame

Figure 1 show some equipment with an open chain structure, e.g. a robotic manipulator. In this case there are two revolute joints, and one prismatic joint. In each joint, there is a reference frame, attached to the corresponding body, such that body $i$ is attached to reference frame $i$. If joint $i$ is a revolute joint, body $i$ rotate about the vector $\boldsymbol{e}_i$, and if joint $i$ is a prismatic joint, body $i$ displace along the vector $\boldsymbol{e}_i$. For convenience, we place the reference frames such that the rotation or displacement of joint i is about or along one of the principal axis of the local reference frame.

In the following, we assume that the location of the center of mass of each link, relative to the link reference frame

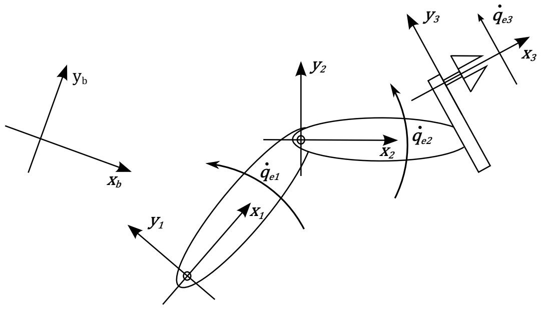


Fig. 1. Kinematics of an open chain of linked bodies.

origin, is known. For link $i$ these coordinates are denoted $\boldsymbol{r}^i_{cmi/i}$. We also define the coordinates of joint $i+1$ relative to joint $i$, in terms of reference frame $i$, as $\boldsymbol{r}^i_{i+1/i}$. In the case when joint $i$ is a revolute joint, these coordinates are constant, and in the case of prismatic joints, the coordinates are dependent on the displacement $q_{e(i+1)}$. In order to find the coordinates $\boldsymbol{r}^i_{i+1/i}$ in this case, we define the coordinate $\boldsymbol{r}^i_{zi/i}$, as the point where reference frame $i+1$ is located for $q_{e(i+1)} = 0$, relative to reference frame $i$. An expression for the vector $\boldsymbol{r}_{i+1/i}$ in the case of joint $i+1$ being prismatic is then

$$\boldsymbol{r}_{i+1/i} = \boldsymbol{r}_{zi/i} + \boldsymbol{e}_{i+1}q_{i+1} \tag{34}$$

With the coordinates $\boldsymbol{r}_{cmi/i}$, $\boldsymbol{r}_{i+1/i}$, and the coordinate of the first link relative to the origin of the body fixed reference frame $\boldsymbol{r}_{1/b}$, we can find the coordinates of any center of mass, relative to any joint, as well as relative to the body fixed reference frame of the vehicle. As an example, the position of the center of mass for body $i$, relative to the origin of the body fixed reference frame is

$$\boldsymbol{r}_{cmi/b} = \boldsymbol{r}_{1/b} + \boldsymbol{r}_{2/1} + ... + \boldsymbol{r}_{i/i-1} + \boldsymbol{r}_{cmi/i} \tag{35}$$

We do however need to express all the terms in (35) in terms of the same reference frame. To this effect, we develop rotation matrices as functions of the generalized coordinates, mapping vectors expressed in terms of any of the local reference frames, into a reference frame with the same orientation as the vehicle body fixed reference frame. Next, we investigate the differential kinematics of the system in order to develop geometric Jacobian matrices as functions of the generalized coordinates. These matrices, one for each body of the equipment, map the quasi-coordinates into the angular and linear velocity of the center of mass of the given body, equivalent to (19), where the vehicle geometric Jacobian is defined.

### *3.1 Coordinate Transformations*

Consider a vector $\boldsymbol{c}^1$ expressed in terms of the reference frame of the innermost joint of the equipment. The orientation of this reference frame, relative to the vehicle body fixed reference frame, will vary with $q_{e1}$ if the first joint is revolute. We can identify a sequence of principal rotations, where the last one can be dependent on $q_{e1}$, that can be applied to the vehicle body fixed reference frame in order to give it an orientation identical to the orientation of reference frame 1. With this sequence identified, the principal rotation matrices of (8) can be applied in order to construct the rotation matrix $\boldsymbol{R}_1^b(q_{e1})$, mapping a vector expressed in terms of reference frame 1 into a reference frame with the same orientation as the vehicle body fixed reference frame. With this matrix defined, we can write

$$\boldsymbol{c}^b = \boldsymbol{R}_1^b(q_{e1})\boldsymbol{c}^1 \tag{36}$$

In the same manner, we can find a matrix $\boldsymbol{R}_2^1(q_{e2})$ that maps a vector expressed in terms of reference frame 2 into a reference frame with the same orientation as frame 1. We can then write

$$\boldsymbol{c}^b = \boldsymbol{R}_1^b(q_{e1})\boldsymbol{c}^1 = \boldsymbol{R}_1^b(q_{e1})\left(\boldsymbol{R}_2^1(q_{e2})\boldsymbol{c}^2\right) \tag{37}$$

In order to get a compact notation, we define the rotation matrices

$$\boldsymbol{R}_i^b(\boldsymbol{q}_e) = \boldsymbol{R}_1^b(q_{e1})\boldsymbol{R}_2^1(q_{e2})\cdots\boldsymbol{R}_i^{i-1}(q_i) \tag{38}$$

for $i \in (2, k)$. These matrices possess the same properties as the matrix $\boldsymbol{R}_b^0(\boldsymbol{\Theta})$ such that the inverse operation is equivalent to the transpose-operation. With (38), we can rewrite (37) to

$$\boldsymbol{c}^b = \boldsymbol{R}_2^b(\boldsymbol{q}_e)\boldsymbol{c}^2 \tag{39}$$

With the rotation matrices, we are able to express all the local coordinates for the center of mass locations, and the subsequent joints in terms of the vehicle body fixed reference frame. Doing so, allows us to find the position of the center of mass for any body, relative to any joint.

### *3.2 Differential Kinematics*

Both the linear and angular velocities of the various bodies of the equipment, are explicitly dependent on the velocity of the vehicle, and the rates of the preceding joints. We define the contribution to the linear velocity of the center of mass of body $i$ from the linear velocity of the vehicle as

$$\begin{aligned}\boldsymbol{v}_{cmi/0}^{(\boldsymbol{v}_{b/0})} &= \boldsymbol{v}_{b/0}^b = \boldsymbol{I}_{3\times3}\boldsymbol{v}_{b/0}^b \\ &\triangleq \boldsymbol{\mathcal{J}}_{\boldsymbol{v}_b}^{\boldsymbol{v}_{cmi}}\boldsymbol{v}_{b/0}^b\end{aligned} \tag{40}$$

where the superscript in parenthesis denote from where the given contribution comes.

The contribution to the same velocity, by the angular velocity of the vehicle is defined as

$$\begin{aligned}\boldsymbol{v}_{cmi/0}^{(\boldsymbol{\omega}_{b/0})} &= \boldsymbol{\omega}_{b/0}^b \times \boldsymbol{r}_{cmi/b}^b \\ &= \left(\boldsymbol{\omega}_{b/0}^b\boldsymbol{i}_b + \boldsymbol{\omega}_{b/0}^b\boldsymbol{j}_b + \boldsymbol{\omega}_{b/0}^b\boldsymbol{k}_b\right) \times \boldsymbol{r}_{cmi/b}^b \\ &= \left[\boldsymbol{i}_b \times \boldsymbol{r}_{cmi/b}^b, \boldsymbol{j}_b \times \boldsymbol{r}_{cmi/b}^b, \boldsymbol{k}_b \times \boldsymbol{r}_{cmi/b}^b\right]\boldsymbol{\omega}_{b/0}^b \\ &\triangleq \boldsymbol{\mathcal{J}}_{\boldsymbol{\omega}_b}^{\boldsymbol{v}_{cmi}}\boldsymbol{\omega}_{b/0}^b\end{aligned} \tag{41}$$

where $\boldsymbol{r}_{cmi/b}$ is the coordinate of the center of mass of body $i$ relative to the origin of the vehicle body fixed reference frame. The contribution to the linear velocity of the $i$-th center of mass from the rate of joint $p$ for $p \leq i$, depends on whether the joint is revolute or prismatic. We define

$$\begin{aligned}\boldsymbol{v}_{cmi/0}^{(\dot{q}_{ep})} &\triangleq \boldsymbol{\mathcal{J}}_{\dot{q}_{ep}}^{\boldsymbol{v}_{cmi}}\dot{q}_{ep} \\ &= \begin{cases}(\boldsymbol{e}_p^b \times \boldsymbol{r}_{cmi/p}^b)\dot{q}_{ep}, & \text{for revolute} \\ \boldsymbol{e}_p^b\dot{q}_{ep} & \text{for prismatic}\end{cases}\end{aligned} \tag{42}$$

where $\boldsymbol{r}_{cmi/p}$ is the coordinate of the center of mass of body $i$ relative to the origin of reference frame $p$, and $\boldsymbol{e}_p^b$ is the vector about, or along, which body $p$ revolve of translate, in terms of the vehicle body fixed reference frame. In the case where $p > i$, the contribution is zero. Note that the vehicle body fixed representation of the position vectors and the rotation, and translation directional vectors defined above, can be found using the rotation matrices defined in (38).

Using (40) through (42), we can find the linear velocity of the center of mass of link $i$, expressed in terms of the vehicle body fixed reference frame, as a function of the generalized coordinates and the quasi-coordinates, as

$$\begin{aligned}\boldsymbol{v}_{cmi/0}^b &= \left[\boldsymbol{\mathcal{J}}_{\boldsymbol{v}_b}^{\boldsymbol{v}_{cmi}}, \boldsymbol{\mathcal{J}}_{\boldsymbol{\omega}_b}^{\boldsymbol{v}_{cmi}}, \boldsymbol{\mathcal{J}}_{\dot{q}_{e1}}^{\boldsymbol{v}_{cmi}}, \ldots, \boldsymbol{\mathcal{J}}_{\dot{q}_{ei}}^{\boldsymbol{v}_{cmi}}, \boldsymbol{0}\right]\boldsymbol{\omega} \\ &\triangleq \boldsymbol{J}_i^v(\boldsymbol{q})\boldsymbol{\omega}\end{aligned} \tag{43}$$

where the dimensions of the zero matrix $\boldsymbol{0}$ is $3 \times (k - i)$.

We now proceed to find the various contributions to the angular velocity of body $i$. There is no contribution to this velocity from the linear velocity of the vehicle. Thus, we can define

$$\begin{aligned}\boldsymbol{\omega}_i^{(\boldsymbol{v}_b)} &= \boldsymbol{0}_{3\times3}\boldsymbol{v}_{b/0}^b \\ &\triangleq \boldsymbol{\mathcal{J}}_{\boldsymbol{v}_b}^{\boldsymbol{\omega}_i}\boldsymbol{v}_{b/0}^b\end{aligned} \tag{44}$$

The contribution from the angular velocity of the vehicle can be formulated as

$$\begin{aligned}\boldsymbol{\omega}_i^{(\boldsymbol{\omega}_{b/0})} &= \boldsymbol{I}_{3\times3}\boldsymbol{\omega}_{b/0}^b \\ &\triangleq \boldsymbol{\mathcal{J}}_{\boldsymbol{\omega}_b}^{\boldsymbol{\omega}_i}\boldsymbol{\omega}_{b/0}^b\end{aligned} \tag{45}$$

Finally, the contribution to the angular velocity from the joint displacement rate $\dot{q}_{ep}$, given that $p \leq i$, is

$$\begin{aligned}\boldsymbol{\omega}_i^{(\dot{q}_{ep})} &\triangleq \boldsymbol{\mathcal{J}}_{\dot{q}_{ep}}^{\boldsymbol{\omega}_i}\dot{q}_{ep} \\ &= \begin{cases}\boldsymbol{e}_p^b, & \text{for revolute} \\ \boldsymbol{0}_{3\times1} & \text{for prismatic}\end{cases}\end{aligned} \tag{46}$$

The total angular velocity of body $i$ of the equipment, can be found by summing up the contributions stated in (44) through (46) as

$$\begin{aligned}\boldsymbol{\omega}_{i/0}^b &= \left[\boldsymbol{\mathcal{J}}_{\boldsymbol{v}_b}^{\boldsymbol{\omega}_i}, \boldsymbol{\mathcal{J}}_{\boldsymbol{\omega}_b}^{\boldsymbol{\omega}_i}, \boldsymbol{\mathcal{J}}_{\dot{q}_{e1}}^{\boldsymbol{\omega}_i}, \ldots, \boldsymbol{\mathcal{J}}_{\dot{q}_{ei}}^{\boldsymbol{\omega}_i}, \boldsymbol{0}\right]\boldsymbol{\omega} \\ &\triangleq \boldsymbol{J}_i^\omega(\boldsymbol{q})\boldsymbol{\omega}\end{aligned} \tag{47}$$

where the zero matrix is of dimension $3 \times (k - i)$.

We now define the $6 \times 1$ vector $\boldsymbol{v}_i = [(\boldsymbol{v}_{cmi/0}^b)^T, (\boldsymbol{\omega}_{i/0}^b)^T]^T$, where the linear and angular velocity of the center of mass of body $i$ is stacked together. Furthermore, we define the $6 \times n$ geometric Jacobian matrix for the velocity of body $i$ as

$$\boldsymbol{J}_i(\boldsymbol{q}) = \begin{bmatrix}\boldsymbol{J}_i^v(\boldsymbol{q}) \\ \boldsymbol{J}_i^\omega(\boldsymbol{q})\end{bmatrix} \tag{48}$$

Using this, a compact expression for the velocity of the center of mass for body $i$ is

$$\boldsymbol{v}_i = \boldsymbol{J}_i(\boldsymbol{q})\boldsymbol{\omega} \tag{49}$$

### *3.3 Kinetic Energy of System*

The kinetic energy the system can be found by summing up the contributions from each body in the system. In (20), the contribution to the total kinetic energy from the vehicle is found. It is however necessary to augment this expression, as $\boldsymbol{q}$ and $\boldsymbol{\omega}$ have been augmented. This is achieved by augmenting the geometric Jacobian matrix found in (19) to

$$\boldsymbol{J}_b = \begin{bmatrix} \boldsymbol{J}_b^v & \boldsymbol{0}_{3\times k} \\ \boldsymbol{J}_b^\omega & \boldsymbol{0}_{3\times k} \end{bmatrix} \tag{50}$$

in order to make it compatible to the new vector of quasi-coordinates.

The kinetic energy of the equipment body $i$ can be found, in the same manner as that of the vehicle, as

$$\begin{aligned} \bar{T}_i(\boldsymbol{q},\boldsymbol{\omega}) &= \frac{1}{2}\boldsymbol{\omega}^T \boldsymbol{J}_i^T(\boldsymbol{q}) \begin{bmatrix} \boldsymbol{M}_i & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{I}_i^b \end{bmatrix} \boldsymbol{J}_i(\boldsymbol{q})\boldsymbol{\omega} \\ &\triangleq \frac{1}{2}\boldsymbol{\omega}^T \boldsymbol{B}_i(\boldsymbol{q})\boldsymbol{\omega} \end{aligned} \tag{51}$$

where $\boldsymbol{M}_i = m_i \boldsymbol{I}_{3\times 3}$, $m_i$ is the mass of body $i$, and $\boldsymbol{I}_i^b = \boldsymbol{R}_i^b \boldsymbol{I}_i \boldsymbol{R}_b^i$ is the inertia tensor of body $i$, expressed in terms of the vehicle body fixed reference frame. The matrix $\boldsymbol{I}_i$ is the locally expressed inertia tensor, and $\boldsymbol{B}_i(\boldsymbol{q})$ is the equipment body $i$ mass-inertia matrix, which also is symmetric and positive definite.

To find the system kinetic energy, we sum up all the contributions as

$$\begin{aligned} \bar{T}(\boldsymbol{q},\boldsymbol{\omega}) &= \bar{T}_b(\boldsymbol{q},\boldsymbol{\omega}) + \sum_{i=1}^{k} \left(\bar{T}_i(\boldsymbol{q},\boldsymbol{\omega})\right) \\ &= \frac{1}{2}\boldsymbol{\omega}^T \left( \boldsymbol{B}_b + \sum_{i=1}^{k} \left(\boldsymbol{B}_i(\boldsymbol{q})\right) \right) \boldsymbol{\omega} \\ &\triangleq \frac{1}{2}\boldsymbol{\omega}^T \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\omega} \end{aligned} \tag{52}$$

where the symmetric and positive definite system mass-inertia matrix $\boldsymbol{B}(\boldsymbol{q})$ is the sum of the vehicle and the equipment bodies mass-inertia matrices.

Using the equations (27) and (28), we find a state space model for the complete system as

$$\begin{aligned} \boldsymbol{\omega} &= \boldsymbol{B}^{-1}\boldsymbol{p} \\ \dot{\boldsymbol{p}} &= \boldsymbol{f}_p(\boldsymbol{q},\boldsymbol{\omega}) + \boldsymbol{\beta}^T\boldsymbol{\tau} \end{aligned} \tag{53}$$

## 4. BOND GRAPH IMPLEMENTATION

We now have a set of equations describing the basic dynamics of the system, i.e., the dynamics of the system without restoring forces. This set of equations are well suited for implementation in the bond graph language. After creating a basic bond graph of the system, i.e, implementing the basic equations, gravity and restoring forces will be introduced to the system. Finally, we establish interfaces to other dynamical loads and systems.

### *4.1 Basic Model*

The equations (53) can be implemented in a bond graph as shown in figure 2. The equation set is dependent on the generalized coordinates $\boldsymbol{q}$, the quasi coordinates $\boldsymbol{\omega}$, and the momentum $\boldsymbol{p}$. The implementation to the left in figure 2, show three vector power bonds sharing the same 1-junction. By letting the effort $\boldsymbol{e}_1 = \dot{\boldsymbol{p}}_1$, and the flow $\boldsymbol{f}_2 = \boldsymbol{\omega}_2$, be input ports to the IC-field, we seek to find expressions for the outputs $\dot{\boldsymbol{p}}_2$ and $\boldsymbol{\omega}_1$. As all three power bonds are connected to the same 1-junction, we have that $\boldsymbol{\omega}_1 = \boldsymbol{\omega}_2 = \boldsymbol{\omega}$. We also see from the 1-junction that

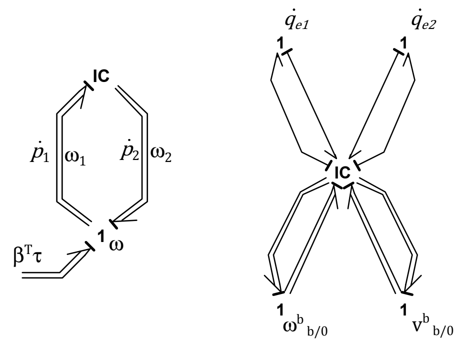


Fig. 2. Left figure: Basic bond graph of the system with single 1-junction. Right figure: Basic bond graph with flows separated into several 1-junctions

$\dot{\boldsymbol{p}}_1 = \dot{\boldsymbol{p}}_2 + \boldsymbol{\beta}^T\boldsymbol{\tau} = \dot{\boldsymbol{p}}$. Thus, the constitutive relations for the IC-field are

$$\begin{aligned} \boldsymbol{\omega}_1 &= \boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{p}_1 \\ \dot{\boldsymbol{p}}_2 &= \boldsymbol{f}_p(\boldsymbol{q},\boldsymbol{\omega}_1) \end{aligned} \tag{54}$$

where the vector of generalized coordinates, are found by integrating the equation

$$\dot{\boldsymbol{q}} = \boldsymbol{\beta}\boldsymbol{\omega} \tag{55}$$

In order to conveniently develop and interface extensions to this basic model, we partition the quasi-coordinate vector into the linear velocity of the vehicle, $\boldsymbol{v}_{b/0}^b$, the angular velocity of the vehicle, $\boldsymbol{\omega}_{b/0}^b$, and the joint rates of the equipment $\dot{\boldsymbol{q}}_e$. Furthermore, it might be convenient to partition the vector of joint rates into $k$ separate velocities $\dot{q}_{e1}$, $\dot{q}_{e2}$, ..., $\dot{q}_{ek}$. We can now create separate 1-junctions, representing each of these velocity components, and connect each to the IC-field as shown to the right in figure 2.

### *4.2 Introducing Restoring Forces*

Restoring forces are the forces and torques resulting from the gravity forces, i.e., the weight of the vehicle, or body in question, and the buoyancy forces. The linear restoring force, i.e., the restoring force associated to the linear motion of the vehicle, is the resulting force from the difference between the weight and the buoyancy, while the torques appear when the center of gravity, and the center of buoyancy do not coincide, and the body orientation is not in its equilibrium. We shall in the following expand the bond graph shown to the right in figure 2, to include restoring forces for both the vehicle and the equipment.

We denote the linear restoring force acting on the vehicle body as $\boldsymbol{F}_{R,b}^0$. This force is expressed in terms of the inertial reference frame, and is the resultant force of the weight $\boldsymbol{f}_g^0$, and the buoyancy $\boldsymbol{f}_b^0$. For underwater vehicles, this force is typically constant. In such cases, the force can be modelled using an effort source. However, for a surface vehicles, the buoyancy force increase proportionally to the displaced volume of water. In this case, energy will be stored as a function of the vertical position of the vehicle relative to the water surface, and as such, a C-element is the natural choice.

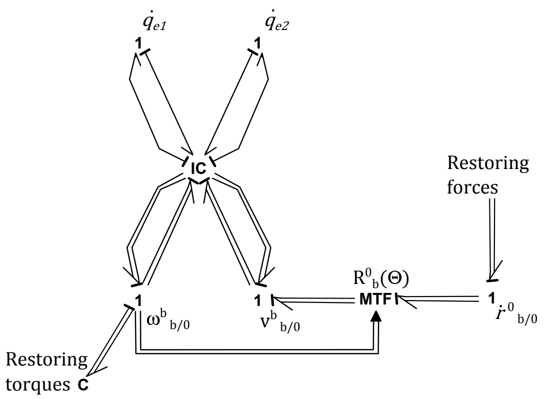


Fig. 3. Bond graph extended to interface restoring forces on the vehicle.

As the linear restoring force of the vehicle is expressed in terms of the inertial reference frame, we expand the bond graph shown to the right in figure 2 to the one shown in figure 3, in order to provide an interface to the force. This is achieved by introducing a 1-junction representing the linear velocity of the vehicle, $\dot{\boldsymbol{r}}^0_{b/0}$. To this effect we can use the transformation

$$\dot{\boldsymbol{r}}^0_{b/0} = \boldsymbol{R}^0_b(\boldsymbol{\Theta})\boldsymbol{v}^b_{b/0} \tag{56}$$

which can be modelled using the transformer element in figure 3, where the Euler angles are found by integrating the equation

$$\dot{\boldsymbol{\Theta}} = \boldsymbol{T}_\Theta(\boldsymbol{\Theta})\boldsymbol{\omega}^b_{b/0} \tag{57}$$

The restoring torques acting on the vehicle body are denoted $\boldsymbol{\tau}^b_{R,b}$. These torques are expressed in terms of the vehicle body fixed reference frame and can in general be found as

$$\boldsymbol{\tau}^b_{R,b} = \boldsymbol{r}^b_{cg/b} \times \boldsymbol{R}^b_0\boldsymbol{f}^0_g + \boldsymbol{r}^b_{cb/b} \times \boldsymbol{R}^b_0\boldsymbol{f}^0_b \tag{58}$$

where $\boldsymbol{r}^b_{cg/b}$ and $\boldsymbol{r}^b_{cb/b}$ are the coordinates of the the vehicle center of mass and center of buoyancy relative to the origin of the vehicle body fixed reference frame, and $\boldsymbol{f}^0_g$ and $\boldsymbol{f}^0_b$ are the weight and buoyancy of the vehicle. Energy will be stored as a function of the vehicle displacement due to the restoring torques, and the C-element is thus a suitable implementation. In figure 3, we see the restoring torque of the vehicle body as the C-element connected to the 1-junction representing the body fixed angular velocity.

In order to apply restoring forces to the equipment bodies, we place 1-junctions representing the linear and angular velocity of the center of mass of each body. Figure 4 show how our bond graph can be extended to include these velocities, using the transformations given in (43) and (47). We have omitted the signal bonds to transformer elements in this figure, but is implied that we take in $\boldsymbol{\omega}$, and integrate (55) in order to find $q$.

The linear restoring force of body $i$ is denoted $\boldsymbol{F}^0_{R,i}$, and can be implemented in the same manner as the linear restoring of the vehicle body. This implies that we use a C-element if the force is variable, as is the case if the body is partially submerged, or an effort source if the force is constant. The elements representing the linear

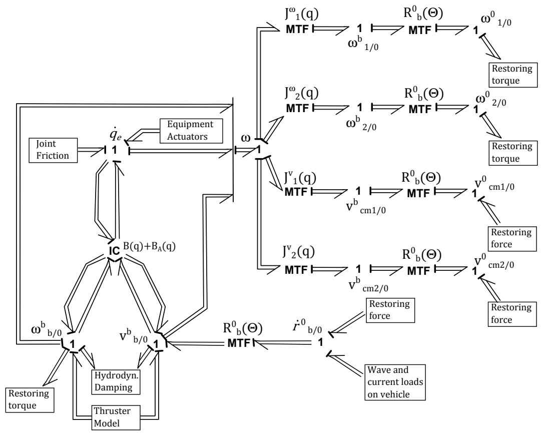


Fig. 4. Bond graph extended to interface restoring forces on the equipment.

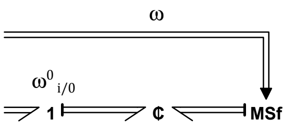


Fig. 5. Using C-field and modulated effort source to model the restoring torque.

restoring force can be connected directly to the 1-junctions representing $\boldsymbol{v}^0_{cmi/0}$.

The restoring torque of equipment body $i$ is denoted $\boldsymbol{\tau}^0_{R,i}$. This is the torque about the center of mass of the body in question, and is dependent on the location of the center of buoyancy relative to the center of mass, as well as the magnitude of the buoyancy. In general we can write

$$\boldsymbol{\tau}^0_{R,i} = \boldsymbol{r}^0_{cbi/cmi} \times \boldsymbol{f}^0_{b,i} \tag{59}$$

where $\boldsymbol{r}^0_{cbi/cmi} = \boldsymbol{R}^0_b(\boldsymbol{\Theta})\boldsymbol{R}^b_i(\boldsymbol{q}_e)\boldsymbol{r}^i_{cbi/cmi}$ is the coordinates of the center of buoyancy relative the the center of mass of body $i$, and $\boldsymbol{f}^0_{b,i}$ is the buoyancy force of the $i$-th body. This equation is however not suitable for implementation in a C-field, as is the case for the restoring torque of the vehicle body. This is due to the fact that we have no means by which to find the signal state $\boldsymbol{q}$, as we cannot solve for $\boldsymbol{\omega}$ in (47), and thus cannot integrate (55). A solution to this is shown in figure 5, where we use a modulated flow source, taking in the quasi-coordinates as modulator, together with an two-port C-element. We then connect port one to the modulated flow source, and port two to the 1-junction representing the angular velocity of body $i$. For signal state one, we use

$$\boldsymbol{q}_{C1} = \int \boldsymbol{\beta\omega} dt = \boldsymbol{q} \tag{60}$$

Then the effort of port two can be found as

$$\boldsymbol{e}_2 = \boldsymbol{\tau}^0_{R,i}(\boldsymbol{q}_{C1}) \tag{61}$$

### *4.3 Interfacing External Environment and Thrusters*

There exists numerous possible extensions to include in this kinds of models. We propose some possibilities in figure 4, in order to demonstrate how this bond graph may be utilized to interface sub models. In the following, we comment these shortly.

The equipment actuators may, as an example, consist of hydraulic motors and pistons, which in turn rely upon a hydraulic system with pumps, control valves and such. Other types of equipment may be actuated by electrical systems. Common for most systems however, is that the actuators impose an effort on the joints, which can be connected to the 1-junction $\dot{\boldsymbol{q}}_e$ as shown in figure 4.

The marine vehicle may be actuated by a variety of different propulsion devices. The surface vehicle can typically be actuated by regular propellers along with rudder and a variety of other thrusters such as azimuth thrusters and tunnel thrusters. Underwater vehicles may be actuated by e.g. a set of thrusters, along with fins and rudders. Regardless of the type however, all actuator systems for marine vehicles must be able to deliver forces and torques in the relevant degrees of freedom. As the vehicle actuators are somehow attached to the vehicle in question, the forces and torques delivered will be expressed in terms of the vehicle body fixed reference frame, and as such, the actuator system force and torque outputs are connected to the 1-junctions representing the linear and angular velocity of the vehicle.

There are many places in this kind of systems where friction and damping forces act. In figure 4, we have included joint friction for the equipment, and hydrodynamic damping for the vehicle. The hydrodynamic damping is set by the vehicle body fixed angular and linear velocities, and is thus connected to the respective 1-junctions, while the joint friction is set by the joint rates $\dot{\boldsymbol{q}}_e$. For an underwater vehicle with robotic manipulator, it might also be relevant to connect hydrodynamic damping models to the velocities of the different manipulator links.

We have also included a sub model for wave and current loads. These loads act on the linear motion of the vehicle. It is also convenient to express these loads in terms of the inertial reference frame. Thus, the wave and current sub model effort output is connected to the 1-junction representing the vehicle velocity in terms of the inertial reference frame.

Finally we treat the effects of the hydrodynamic phenomenon, added mass. We can rewrite the basic equations of motion (53), to the common form of e.g. Faltinsen (1993). Denoting the forces resulting from the added mass as $\boldsymbol{\tau}_A$, and assuming that no other external forces that those due to addad mass act on the system, we may write

$$\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{\omega}} + \boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{\omega} = \boldsymbol{\tau}_A \tag{62}$$

where the term $\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{\omega}$ is known as the Coriolis and centrifugal forces. The added mass forces can be expanded to the form

$$\boldsymbol{\tau}_A = -\boldsymbol{B}_A(\boldsymbol{q})\dot{\boldsymbol{\omega}} - \boldsymbol{C}_A(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{\omega} \tag{63}$$

The added momentum is then

$$\boldsymbol{p}_A = \boldsymbol{B}_A(\boldsymbol{q})\boldsymbol{\omega} \tag{64}$$

and the rate of the added momentum is

$$\dot{\boldsymbol{p}}_A = \boldsymbol{B}_A(\boldsymbol{q})\dot{\boldsymbol{\omega}} + \boldsymbol{C}_A(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{\omega} = \boldsymbol{f}_{pA}(\boldsymbol{q},\boldsymbol{\omega}) \tag{65}$$

By now redefining the momentum of the quasi-states, $\boldsymbol{p}$, such that it incorporates the added momentum, we can write the constitutive relation of the IC-field as

$$\begin{aligned} \boldsymbol{\omega} &= \left[\boldsymbol{B}(\boldsymbol{q}) + \boldsymbol{B}_A(\boldsymbol{q})\right]^{-1}\boldsymbol{p} \\ \dot{\boldsymbol{p}} &= \boldsymbol{f}_p(\boldsymbol{q},\boldsymbol{\omega}) + \boldsymbol{f}_{pA}(\boldsymbol{q},\boldsymbol{\omega}) \end{aligned} \tag{66}$$

## 5. CONCLUSION

Lagrangian mechanics and bond graph modelling have in this paper been applied in order to develop a framework for dynamic modelling of marine vehicles with equipment such as cranes and manipulators. The framework provides possibilities for seamless integration of various bond graph sub models to the system. As a closing remark, we may note that the model developed here was limited to equipment with an open chain of linked bodies structure, with lower pair joints. This is however not because it is problematic to use this approach on other kinds of equipment, say parallel manipulators, or devices without lower pair joints, but rather because we wished to provide an unambiguous method for connecting the equipment to the vehicle. This is difficult to do for any conceivable type of equipment that could be mounted on a marine vehicle, thus some restrictions had to be used in this text. As the open structure with lower pair joints is a typical structure for both robotic manipulators used for subsea tasks, and for various crane equipment on surface vessels, these particular restrictions were chosen. The modeller using the methods presented here may however choose to apply other theory for modelling the equipment, than what is presented in section 3 in this paper, and still be able to use the general method provided in this paper in order to model marine vehicles with equipment that does not satisfy the restrictions given here.

## REFERENCES

Faltinsen, O. (1993). *Sea Loads on Ships and Offshore Structures.* Cambridge Ocean Technology Series. Cambridge University Press.

Fossen, T., I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control.* 1 edition.

Karnopp, D., C., Margolis, D., L., and Rosenberg, R., C. (2006). *System Dynamics: Modelling and Simulation of Mechatronic Systems.* 4 edition.

Kim, J., Chung, W.K., and Yuh, J. (2003). Dynamic analysis and two-time scale control for underwater vehicle-manipulator systems. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, 577–582 vol.1.

Meirovitch, L. (2003). *Methods of Analytical Dynamics.*

Pedersen, E. (2012). Bond graph modeling of marine vehicle dynamics. *Bond Graph Modeling: Theory and Practice Symposium at the 7th Vienna International Conference on Mathematical Modeling.*

Soylu, S., Firmani, F., Buckham, B., and Podhorodeski, R. (2010). Comprehensive underwater vehicle-manipulator system teleoperation. In *OCEANS 2010*, 1–8.

## A.2 Paper 2 - Backstepping Controller for Manipulator with Hydraulic Actuator System

# Backstepping Controller for Manipulator with Hydraulic Actuators

**B. Rokseth** * **E. Pedersen** **

* *Department of Marine Technology, Norwegian University of Science and Technology (NTNU),7491 Trondheim, Norway (e-mail: borgerok@stud.ntnu.no)*
** *Department of Marine Technology, Norwegian University of Science and Technology (NTNU),7491 Trondheim, Norway (e-mail: eilif.pedersen@ntnu.no)*

**Abstract:** In this paper, a trajectory tracking backstepping controller for robotic manipulators with hydraulic actuators is derived. The controller is proven to be globally asymptotically stable. In the literature found on hydraulic system control design, there are two model simplifications commonly applied in order to simplify the physical model of the system. These simplifications are also applied here, but a focus in this paper is to investigate, through model evaluations and numerical simulations, to what degree, and under which circumstances, the simplifications are valid. To this effect, simplified models are compared to a high fidelity bond graph model. The first simplification to be investigated is based on the assumption that the dynamics of the servo valves in the system are much faster than the dynamics of the rest of the system. This assumption is used in order to apply the simplification of instantaneous valve dynamics. The second simplification commonly applied, is that of reducing the two hydraulic pressure states on the upside and downside of each actuator to the single differential pressure state.

*Keywords:* Manipulator, Hydraulics, Control, Backstepping, Modelling, Robotics, Bond Graph

## 1. INTRODUCTION

Manipulators with hydraulic actuators are widely used due to large power to weight ratios and fast response times (Yao and Wang, 2012). The highly non linear nature of the hydraulic system does however make model based control design a challenge for these manipulators. Furthermore, as opposed to manipulators with electric actuators, where the actuator efforts can be controlled directly, the effort exerted by the hydraulic actuators are proportional to the pressure difference over the actuator. The pressure difference is controlled through the volumetric flow of hydraulic fluid through the control valve, which in turn is related to the time derivatives of the upside and downside hydraulic pressure. As such, one can only directly control the time derivative of the actuator efforts.

There are a great deal of scientific work done both on control of hydraulic actuators in general, and on manipulators with hydraulic actuators. Several different modelling approaches are used in order to develop physical models of the hydraulic systems, and several different control schemes are used. Feedback linearization, back-stepping and sliding mode control schemes are commonly applied. The feedback linearizing controllers proposed in Angue-Mintsa et al. (2011) and Seo et al. (2007) both utilize a mathematical model in which the two independent pressure states on the upside and down side of the actuator are replaced by the differential pressure. This is a major simplification of the system, but is in general required if the feedback linearizing method is used because the model with both upside and downside pressure is not minimum phase. On the other hand, Wang et al. (2011) propose a sliding mode controller, where the hydraulic part of the system is modelled by two states, namely the upside and downside pressure. In Bu and Yao (2000) a back-stepping controller is proposed, also based on a mathematical model with two pressure states, while Zeng and Sepehri (2006), propose a back-stepping controller using a model where the hydraulics are modelled by only the differential pressure.

An other modelling choice which has to be made, is whether or not to model the valve dynamics. Both Zeng and Sepehri (2006), and Wang et al. (2011) assumes that the control valves are so fast, compared to the rest of the system, that they can be modelled as instantaneous. However, due to the great stiffness of the hydraulic system, fast time constants are likely to be present under certain conditions. As such, the assumption of an instantaneous control valve can not be taken for granted.

The main purpose of this paper is to develop a backstepping controller for hydraulic manipulators. However, in doing so, the need to make informed decisions regarding what model simplifications that can be applied becomes apparent. In order to address this issue, we start by deriving high fidelity simulation models for single hydraulic actuator systems, on model with a hydraulic motor, and one model with a linear actuator. Next, a method for comparing the frequency response of the valve to that of the hydraulics, is presented. This can be used as an aid in making informed choices regarding the necessity of valve

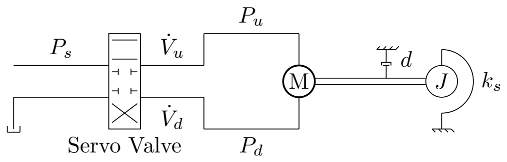


Fig. 1. Diagram of hydraulic system with servo valve, rotational actuator and mass-spring-damper load.

dynamics. Then a simplified model, like those which the controllers of (Angue-Mintsa et al., 2011), Seo et al. (2007), and Zeng and Sepehri (2006) are based on, is derived. By numerical simulations, we compare the dynamics of the simulation model and the simplified model. Based on this we can evaluate whether we must use a model based on upside and downside pressure, or if the simplified model based on differential pressure suffices.

In the following section, a hybrid simulation model of a hydraulic system with a control valve and a rotational hydraulic motor is derived. We use bond graphs in order to model the system. The state equations for the systems are extracted from the bond graphs, and rewritten into a form recognized in most of the literature references where the model with two hydraulic pressure states are used. In section 3, we investigate the dynamics of this model further in order to say something about time constants of the system at various points of operation, which in turn can be used to determine whether valve dynamics are necessary or not. In section 4, we proceed to simplify this model into the form where only the differential pressure is considered. This simplified model is utilized to run simulations, with the purpose of comparing the simplified model to the simulations model.

## 2. SIMULATION MODEL OF SIMPLE ELECTRO-HYDRAULIC SYSTEM

In this section a simulation model for a hydraulic system with a servo valve, a hydraulic motor and a simple loading system, as shown in figure 1, is derived. It is also shown how the hydraulic motor can be replaced by a linear actuator. We start with an instantaneous servo valve, i.e., a servo valve without dynamics. Next, the servo valve dynamics are incorporated to the model as a first order differential equation.

The system in figure 1 consists of a pressure source, a control valve, and a hydraulic motor. The hydraulic motor shaft is connected to a mass with inertia $J$, attached to the ground through a spring with stiffness $k_s$. The damper with damping coefficient $d$ represents the friction of the motor and any bearings for the shaft.

Figure 2 show a bond graph of the system in figure 1. Starting from the left, there are two effort sources representing the hydraulic supply pressure and the return pressure, both of which are assumed to be constant. The control valve is modelled as a set of four R-elements. Two of which have the constitutive relation $f = v_1(e, u)$, and two of which with the constitutive relation $f = v_2(e, u)$, where $f$ is the fluid volumetric flow on the power bond associated

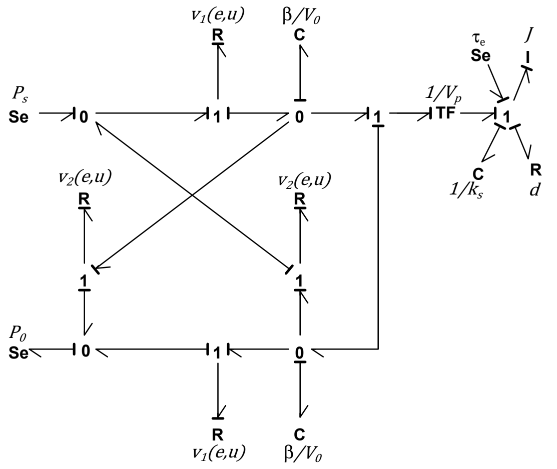


Fig. 2. Bond graph of hydraulic system with control valve and motor, connected to mass-damper-spring system.

to the R-element in question, $e$ is the hydraulic pressure of the power bond, and $u$ is a control signal. When the control valve is open in the forward direction, $v_1(e, u) > 0$ and $v_2(e, u) = 0$. Conversely, when the control valve is open in the reverse direction, we have $v_1(e, u) = 0$, and $v_2(e, u) > 0$. The two C-elements to the right of the control valve represents the fluid compressibility and any elastic nature of the tubing etc., described by the constitutive relation $e = (\beta/V_0)q$. The effort $e$ is the hydraulic pressure of the power bond associated to the C-element in question, $\beta$ is the total bulk modulus of the oil, including any elastic nature of the tubing, $V_0$ is the initial fluid volume, and $q$ is the integral of the flow variable associated to the power bond. The hydraulic motor is represented by the transformer element with modulus $1/V_p$, where $V_p$ is the volumetric displacement per radian of the motor. It is assumed that there are no loss or back flow through the pump. The mass inertia, the spring and the friction, are modelled as the I-element, the C-element and the R-element to the right in the bond graph. Finally, an external torque, $\tau_e$ is included.

By extracting the state equations directly from the bond graph, as in Karnopp et al. (2006), and taking the return pressure as zero, we get

$$\begin{aligned}
\dot{q} &= \frac{1}{J}p \\
\dot{p} &= -k_s q - \frac{d}{J}p + \frac{V_p}{V_0}\beta(V_u - V_d) + \tau_e \\
\dot{V}_u &= -\frac{V_p}{J}p + v_1\left(Ps - \frac{\beta}{V_0}V_u,\, u\right) - v_2\left(\frac{\beta}{V_0}V_u,\, u\right) \\
\dot{V}_d &= \frac{V_p}{J}p - v_1\left(\frac{\beta}{V_0}V_d,\, u\right) - v_2\left(\frac{\beta}{V_0}V_d - P_s,\, u\right)
\end{aligned} \tag{1}$$

where $q$ is the angular displacement of the mass of the loading system, $p$ is the momentum of the mass, and $V_u$ and $V_d$ are the volumes of compressed fluid on the upside and downside of the motor respectively. The hydraulic part of this state space model is expressed in terms of the volume states $V_u$ and $V_d$. In the reference literature of this paper however, the hydraulic models are expressed in terms of the corresponding pressure states, rather than the volume states. In order to get the model (1) on this form, consider the two volume states. These can be transformed into pressure states by assuming a linear relationship between the compressed volumes and the corresponding pressures as

$$P_u = \frac{\beta}{V_0}V_u, \quad P_d = \frac{\beta}{V_0}V_d \tag{2}$$

Thus, we can multiply both sides of the two last equations in (1), by $\beta/v_0$, and substitute (2) for the arguments in the functions $v_1$ and $v_2$ to obtain

$$\begin{aligned} \dot{P}_u &= -\frac{\beta}{V_0}\left(\frac{V_p}{J}p + v_1\left(Ps - P_u,\, u\right) - v_2\left(P_u,\, u\right)\right) \\ \dot{P}_d &= \frac{\beta}{V_0}\left(\frac{V_p}{J}p - v_1\left(P_d,\, u\right) - v_b\left(P_d - P_s,\, u\right)\right) \end{aligned} \tag{3}$$

The functions $v_1$ and $v_2$, describing the flow through the control valve can be expressed as

$$\begin{aligned} v_1(z,u) &= \begin{cases} C_d sgn(z) u\sqrt{\frac{2}{\rho}|z|}, & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ v_2(z,u) &= \begin{cases} 0, & \text{if } u \geq 0 \\ C_d sgn(z) u\sqrt{\frac{2}{\rho}|z|} & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

where $C_d$ is the hydraulic discharge coefficient, $\rho$ is the density of the hydraulic fluid and $sgn(z)$ is the signum function, returning 1 if $z > 0$, 0 if $z = 0$, and $-1$ for $z < 0$. By realizing that $v_2(z,u) = -v_2(-z,u)$, we can rewrite the last state equation to

$$\dot{P}_d = \frac{\beta}{V_0}\left(\frac{V_p}{J}p - v_1\left(P_d,\, u\right) + v_2\left(P_s - P_d,\, u\right)\right) \tag{5}$$

From (4), we see that either $v_1$ or $v_2$ is always zero. By using this, and defining the pressures

$$\begin{aligned} P_1 &= \begin{cases} P_s - P_u, & \text{if } u \geq 0 \\ P_u & \text{otherwise} \end{cases} \\ P_2 &= \begin{cases} P_d, & \text{if } u \geq 0 \\ P_s - P_d & \text{otherwise} \end{cases} \end{aligned} \tag{6}$$

we can rewrite the state space model to

$$\begin{aligned} \dot{q} &= \frac{1}{J}p \\ \dot{p} &= -k_s q - \frac{d}{J}p + V_p(P_u - P_d) + \tau_e \\ \dot{P}_u &= -\frac{\beta V_p}{JV_0}p + \frac{\beta}{V_0}C_d sgn(P_1)\sqrt{\frac{2}{\rho}|P_1|}\, u \\ \dot{P}_d &= \frac{\beta V_p}{JV_0}p - \frac{\beta}{V_0}C_d sgn(P_2)\sqrt{\frac{2}{\rho}|P_2|}\, u \end{aligned} \tag{7}$$

A system with a linear actuator, i.e., a piston instead of a motor, as shown in figure 3, is quite similar to the system (7). The only difference lies in the interface between the mechanical and the hydraulic systems. We define the function $\mathcal{J}(q)$ as the function mapping the force on the

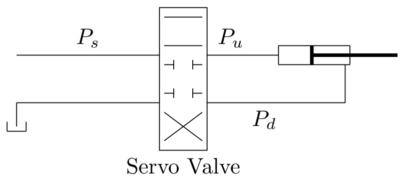


Fig. 3. Diagram of hydraulic system with linear actuator.

piston rod to the force or torque exerted on joint. As an example, the piston may be placed between two bodies, linked together by a pin, such that the bodies rotate relative to each other when the piston extend. In this case the piston force is mapped into a torque by the function $\mathcal{J}(q)$. A simpler case occur when the piston is induce a direct linear motion on the joint in question. Then we obtain simply $\mathcal{J}(q) = \mathcal{J} = 1$. In any case, the state space model describing a hydraulic system with a linear actuator can be expressed as

$$\begin{aligned} \dot{q} &= \frac{1}{J}p \\ \dot{p} &= -k_s q - \frac{d}{J}p + \mathcal{J}(q)(A_u P_u - A_d P_d) + \tau_e \\ \dot{P}_u &= -\frac{\beta A_u}{JV_0}\mathcal{J}(q)p + \frac{\beta}{V_0}C_d sgn(P_1)\sqrt{\frac{2}{\rho}|P_1|}\, u \\ \dot{P}_d &= \frac{\beta A_d}{JV_0}\mathcal{J}(q)p - \frac{\beta}{V_0}C_d sgn(P_2)\sqrt{\frac{2}{\rho}|P_2|}\, u \end{aligned} \tag{8}$$

where $A_u$ and $A_d$ are the upside and downside piston areas respectively.

Finally, the models (7) and (8) can be augmented to include valve dynamics. Assume that the valve opening area $u$ is now controlled by the valve control signal $v$, and that the dynamics follow the first order model

$$\dot{u} = -\frac{1}{T_a}\left(u - k_a v\right) \tag{9}$$

where $T_a$ is the time constant of the valve, and $k_a$ is the valve gain. Then (9) can be added as a fifth state equation to the system (7) and (8).

## 3. FREQUENCY RESPONSE

In the literature, the valve dynamics are often neglected by assuming that they are much faster than the rest of the system. In order to obtain a better foundation for evaluating the necessity of modelling valve dynamics, we study the dynamics of the simulation model (7). In particular, we study the frequency response of the differential pressure over the motor, to valve opening area perturbations, by performing experiments on the model. We compare this to the frequency response of the valve opening area dynamics as described by (9).

In general, servo valves are delivered with bode-diagrams, so that the frequency response of the valve dynamics are known. So by experimentally developing a frequency response diagram of the hydraulic system, as we show in the following, allows for a direct comparison between the valve and the hydraulics. Then the common assumption that valve dynamics are much faster than the rest of the

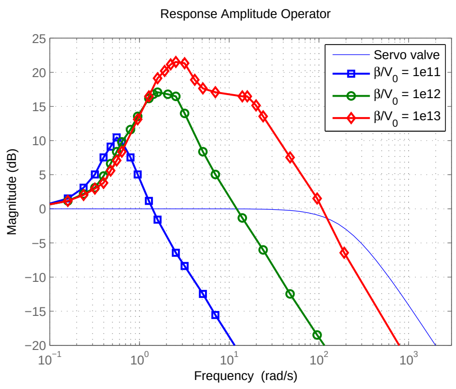


Fig. 4. Frequency response for three different bulk-initial volume ratios and servo valve.

system, and as such can be neglected, can be either verified or rejected for a particular system.

In order to study the frequency response of the highly non-linear model (7), let the valve opening area follow a sinus curve with frequency $\omega$, such that

$$u = A_{max} sin(\omega t) \tag{10}$$

where $A_{max}$ is the opening area when the valve is fully opened. Note that a negative value for $u$ represents that the valve is opened in the reverse direction. Next, define the amplitude ratio $\eta$ as

$$\eta = \frac{|P_L|}{P_{L(\omega \to 0)}} \tag{11}$$

where $|P_L|$ is the amplitude of the differential pressure $P_L = P_u - P_d$ over the motor, and $P_{L(\omega\to 0)}$ is the differential pressure when the frequency goes to zero. By running a series of simulations over a range of frequencies, with the input defined in (10), and noting the amplitude ratio $\eta$ for each simulation, a frequency response diagram can be designed by plotting $\eta$ against the frequency.

Table 1. Parameters used in frequency response simulations.

| Parameter | Unit | Value |
|---|---|---|
| $J$ | $kgm^2$ | 150 |
| $k_s$ | $Nm$ | 0 |
| $d$ | $kgm/s$ | 300 |
| $V_p$ | $m^3$ | 1e-4 |
| $C_d$ | $-$ | 0.9 |
| $\rho$ | $kg/m^3$ | 950 |
| $Ps$ | $Pa$ | 307e5 |
| $A_{max}$ | $m^2$ | 1e-6 |
| $T_a$ | $s$ | 1e-4 |
| $K_a$ | | 1e-6 |

As an example, consider figure 4, where the frequency response amplitude in Decibels are plotted against the frequency in Hertz for both the system (7), and the servo valve described by (9). The frequency response for the differential pressure is considered for three cases of the ratio $\beta/V_0$, as can be seen in the figure. The values of the rest of the parameters are presented in table 3. For this

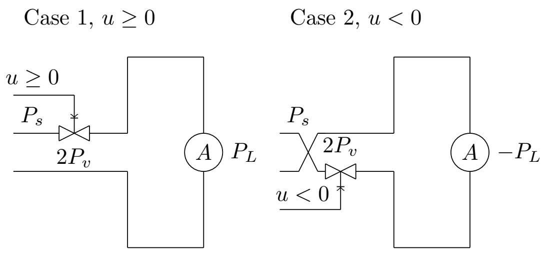


Fig. 5. Two different simplified models. One for the case of forward flow and one for the case ow backward flow.

particular example, we see that the assumption of fast valve dynamics, as compared to the rest of the system, safely can be made for the cases where $\beta/V_0 = 1e11$ and $\beta/V_0 = 1e12$. For the stiffest simulation case with $\beta/V_0 = 1e13$ however, the assumption of much faster valve dynamics would be highly questionable.

## 4. MODEL REDUCTION

We now proceed to derive simplified models for the hydraulic systems described by (7) and (8), using only the differential pressure state in stead of the upside and downside pressure states, as well as instantaneous valve dynamics. The models (7) and (8), shall hence forth be referred to as *simulation plant models*, and the simplified models to be derived shall be referred to as *control plant models*. We also assume instantaneous valve dynamics. After deriving the simplified model, we compare the simulation plant models to the control plant models by investigating the step responses of the systems.

By studying the second equation in (7), the term $V_p(P_u - P_d)$ can be recognized as the torque delivered by the hydraulic motor. Because $P_u - P_d$ is the differential pressure over the actuator, we conclude that the actuator torque is directly dependent only on the differential pressure. This motivates the idea of replacing the upside and downside pressure states by the differential pressure. In order to do the same for the system with linear actuator, i.e. system (8), we need to assume that the upside and down side piston areas are equal. Doing so, we define $A_p = A_u = A_d$, and simplify $\mathcal{J}(q)(A_u P_u - A_d P_d) = \mathcal{J}(q) A_p P_L$. However, by defining the differential pressure rate as

$$\begin{aligned}\dot{P}_L &= \dot{P}_u - \dot{P}_d \\ &= -\frac{2\beta V_p}{JV_0} p \\ &+ \frac{\beta C_d}{V_0}\left(sgn(P_1)\sqrt{\frac{2}{\rho}|P_1|} + sgn(P_2)\sqrt{\frac{2}{\rho}|P_2|}\right) u\end{aligned} \tag{12}$$

we see that the valve flow dynamics can not be expressed accurately solely by the differential pressure. In order to remove this dependency on the upside and downside pressures, we simplify the model into the two cases shown in figure 5, where the circle denoted $A$ represents either the hydraulic motor or the linear actuator. As can be seen from the figure, this simplification require the assumption that the valve imposes no restriction on the flow on the downstream side of the actuator. We must also assume that the volume rate $\dot{V}_v$ through the valve is equal to the volume rate $\dot{V}_a$ through the actuator, as we shall see shortly. This in order to be able to express the volume rate in terms of the pressure loss over the motor. This is in general not true because the fluid between the valve and the motor can compress slightly. It should also be noted that in order to keep the relation (2), we directly contradict the assumption of equal flow through the valve and the actuator, as we here introduce elasticity to the fluid. We do nevertheless make the assumption.

The pressure loss over the valve is denoted $2P_v$, where the factor of two comes in because the fluid in the real system need to pass through the valve twice. From figure 5, we see that the pressure loss through the valve is related to the supply pressure and the pressure loss over the actuator by

$$2P_v = \begin{cases} P_s - P_L, & \text{if } u \geq 0 \\ P_s + P_L, & \text{otherwise} \end{cases} \tag{13}$$

These two cases can be collected into one expression as

$$P_v = \frac{1}{2}(P_s - sgn(u)P_L) \tag{14}$$

The volumetric flow over the valve can be expressed as

$$\begin{aligned} \dot{V}_v &= C_d\sqrt{\frac{2}{\rho}|P_v|}\; u \\ &= C_d\sqrt{\frac{1}{\rho}|P_s - sgn(u)P_L|}\; u = \dot{V}_a \end{aligned} \tag{15}$$

where we inserted (14) for $P_v$, and used the assumption that the flow over the valve $\dot{V}v$ equals the flow over the motor $\dot{V}_a$. By applying (2) and doubling the bulk modulus of the hydraulic fluid, we get

$$\dot{P}_L = \frac{2\beta}{V_0}C_d\sqrt{\frac{1}{\rho}|P_s - sgn(u)P_L|}\; u \tag{16}$$

The reason why the bulk modulus is doubled, is that the fluid can be compressed only at one side at the time, according to the simplified model. This can be compared to a mechanical system with two parallel springs which are exchanged by one spring with double stiffness.

By connecting the mechanical part of the system as before, we get the reduced state space model

$$\begin{aligned} \dot{q} &= \frac{1}{J}p \\ \dot{p} &= -k_s q - \frac{d}{J}p + V_p P_L + \tau_e \\ \dot{P}_L &= -\frac{2\beta V_p}{JV_0}p + \frac{2\beta}{V_0}C_d\sqrt{\frac{1}{\rho}|P_s - sgn(u)P_L|}\; u \end{aligned} \tag{17}$$

for the system with hydraulic motor, and

$$\begin{aligned} \dot{q} &= \frac{1}{J}p \\ \dot{p} &= -k_s q - \frac{d}{J}p + \mathcal{J}(q)A_p P_L + \tau_e \\ \dot{P}_L &= -\frac{2\beta A_p}{JV_0}\mathcal{J}(q)p + \frac{2\beta}{V_0}C_d\sqrt{\frac{1}{\rho}|P_s - sgn(u)P_L|}\; u \end{aligned} \tag{18}$$

for the system with linear actuator.

### *4.1 Model Comparison*

Figure 6 show step responses for the control plant model, and the simulation plant model with first order valve dynamics, and hydraulic motors. The parameters from table 3 are used on both models, the only exceptions being that the spring stiffness is changed to $k_s = 1500Nm$ so that the systems are allowed to build up pressure, and that the linear damping coefficient is altered to $d = 150kgm/s$. The step input occur at $t = 0.5s$ where the input take a step from 0 to $A_{max}$.

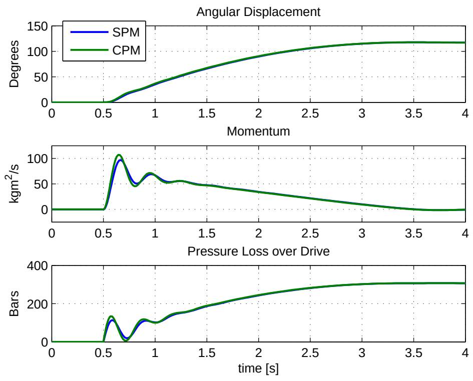


Fig. 6. Simulation of hydraulic motor system step response for both the simulation plant model (SPM), and the control plant model (CPM).

It is clear from the figure that the control plant model follows the simulation plant model accurately enough for control purposes, for this particular system. The simulation plant model does however appear to be slightly slower and more damped than the control plant model.

Figure 7 show similar simulations for the control plant model and the simulation plant model, both with linear actuators. In these simulations, the linear actuator is connected to a mechanical system as shown in figure 8. The parameters are the same as for the simulations shown in figure 6. The volumetric displacement of the motor, $V_p$ is however not relevant for this simulations, and are replaces by the piston areas $A_u$ = 3e-4, $A_d$ = 1.5e-4, and $A_p$ = 2.25e-5, such that the piston area of the control plant model is taken as the mean of the two actual areas.

From these simulations, notice that the control plant model for the system with linear actuator agrees to a lesser degree with the corresponding simulation plant model, than does the control plant model with the hydraulic motor. In particular, notice that the assumption of equal piston area on both sides of the actuator results in an error in the displacement of the loading system for the control plane model. On the other hand, the rise time and the periods of the oscillations are similar for both the control plant model and the simulation plant model.

## 5. DYNAMICS OF MANIPULATOR WITH HYDRAULIC ACTUATOR SYSTEM

In this section, the hydraulic control plant model is connected to a model of a robotic manipulator. The robotic manipulator model with $n$ joints can be expressed as

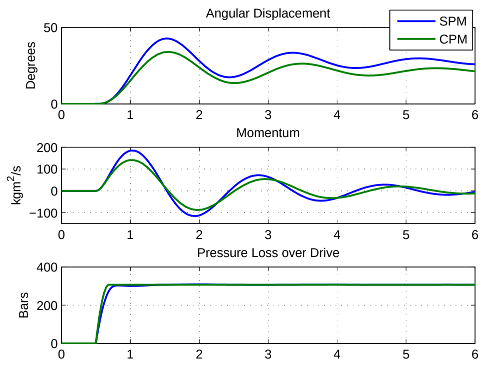


Fig. 7. Simulation of linear actuator system step response for both the simulation plant model (SPM), and the control plant model (CPM).

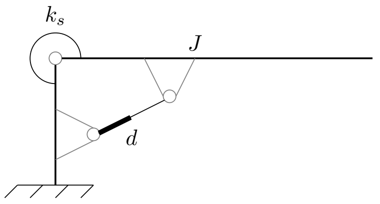


Fig. 8. Diagram of linear actuator connected to mechanical system.

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{\omega} \\ \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{\omega}} &= -\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{\omega} - \boldsymbol{\tau}_f(\boldsymbol{q},\boldsymbol{\omega}) - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_a \end{aligned} \tag{19}$$

where $\boldsymbol{q}$ is the $n \times 1$ vector of manipulator joint displacements, and $\boldsymbol{\omega} = \dot{\boldsymbol{q}}$ is the vector of joint displacement rates. The symmetric and positive definite $n \times n$ matrix $\boldsymbol{B}(\boldsymbol{q})$ is the mass-inertia matrix of the manipulator, and $\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})$ is the Coriolis and centrifugal force matrix, such that $\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{\omega}$ accounts for these forces. The weight of the manipulator is accounted for by the term $\boldsymbol{g}(\boldsymbol{q})$, and the friction forces are collected in the term $\boldsymbol{\tau}_f(\boldsymbol{q},\boldsymbol{\omega})$. The vector $\boldsymbol{\tau}_a$ represents the actuator forces. The Coriolis and centrifugal matrix is found as

$$\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^T \frac{\partial \boldsymbol{B}(\boldsymbol{q})}{\partial \boldsymbol{q}} \tag{20}$$

The weight vector of the manipulator is found as

$$\boldsymbol{g}(\boldsymbol{q}) = \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}} \tag{21}$$

where $V(\boldsymbol{q})$ is the potential energy of the manipulator.

The actuator effort $\boldsymbol{\tau}_a$ must now be related to the differential pressure. To this effect we define the function

$$e_i = \begin{cases} V_{pi}, & \text{if motor} \\ \mathcal{J}_i(\boldsymbol{q})A_{pi}, & \text{if linear actuator} \end{cases} \tag{22}$$

for each manipulator joint $i \in [1, n]$. Defining the $n \times n$ matrix $\boldsymbol{E} = diag(\boldsymbol{e})$, where $\boldsymbol{e} = [e_1,\, e_2,\, ...,\, e_n]^T$, we can express the actuator efforts as

$$\boldsymbol{\tau}_a = \boldsymbol{E}\boldsymbol{P}_L \tag{23}$$

where $\boldsymbol{P}_L$ is the column vector of the $n$ differential pressures. The effect on the differential pressure rate from the momentum of the manipulator, can be expressed as

$$\begin{aligned} \boldsymbol{f}_r &= -\frac{2\beta}{V_0}\boldsymbol{E}\boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{p} \\ &= -\frac{2\beta}{V_0}\boldsymbol{E}\boldsymbol{B}^{-1}(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})\boldsymbol{\omega} \\ &= -\frac{2\beta}{V_0}\boldsymbol{E}\boldsymbol{\omega} \end{aligned} \tag{24}$$

where $\boldsymbol{p} = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\omega}$ is the joint momenta vector. The valve flow functions for each actuator is still as defined in (15). In order to get this on a compact form, we define the function

$$\nu_i(P_{Li}, u_i) = \sqrt{P_s - sgn(u_i)P_{Li}} \tag{25}$$

where $P_{Li}$ and $u_i$ are the $i$-th element of the differential pressure vector and the input vector respectively. Defining the matrix $\boldsymbol{\Gamma}(\boldsymbol{P}_L, \boldsymbol{u}) = diag(\boldsymbol{\nu})$, where $\boldsymbol{\nu} = [\nu_1,\, \nu_2,\, ...,\, \nu_n]^T$, the differential pressure rate due to the servo valve can be expressed as

$$\boldsymbol{v}(\boldsymbol{P}_L, \boldsymbol{u}) = \frac{2\beta C_d}{V_0\sqrt{\rho}}\boldsymbol{\Gamma}\boldsymbol{u} \tag{26}$$

Thus, the control plant state space model of the manipulator with hydraulic actuators can be expressed as

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} &= \boldsymbol{B}^{-1}(\boldsymbol{q})\left(-\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{\omega} - \boldsymbol{\tau}_f - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{E}\boldsymbol{P}_L\right) \\ \dot{\boldsymbol{P}}_L &= -\frac{2\beta}{V_0}\boldsymbol{E}\boldsymbol{\omega} + \frac{2\beta C_d}{V_0\sqrt{\rho}}\boldsymbol{\Gamma}\boldsymbol{u} \end{aligned} \tag{27}$$

## 6. CONTROLLER DESIGN

We proceed to derive a backstepping controller for hydraulic manipulators, based on (27). In order for the manipulator joint displacements to follow trajectories, we look at the error dynamics of the system. Let $\boldsymbol{q_d}$ be the vector of desired joint displacements, such that

$$\boldsymbol{e}_1 = \boldsymbol{q} - \boldsymbol{q}_d \tag{28}$$

is the vector of joint tracking errors. Time differentiating the tracking error yields

$$\dot{\boldsymbol{e}}_1 = \boldsymbol{\omega} - \dot{\boldsymbol{q}}_d \tag{29}$$

which is the joint rate tracking error, $\boldsymbol{e}_2$. Thus,

$$\dot{\boldsymbol{e}}_1 = \boldsymbol{e}_2 \tag{30}$$

The dynamics of the joint rate tracking error is found by time differentiating $\boldsymbol{e}_2$, and substituting $\dot{\boldsymbol{\omega}}$ for the second expression in (27). This yields the expression

$$\dot{\boldsymbol{e}}_2 = \boldsymbol{B}^{-1}(\boldsymbol{q})\left(-\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{\omega} - \boldsymbol{\tau}_f - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{E}\boldsymbol{P}_L\right) - \ddot{\boldsymbol{q}}_d \tag{31}$$

We now define the differential pressure tracking error as

$$\boldsymbol{e}_3 = \boldsymbol{P_L} - \boldsymbol{\alpha}_1 \tag{32}$$

and consider $\boldsymbol{\alpha}_1$ to be a virtual input to the system. Choosing this virtual input as

$$\boldsymbol{\alpha_1} = \boldsymbol{E}^{-1}\left(\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})\dot{\boldsymbol{q}}_d + \boldsymbol{\tau}_f + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}}_d + \boldsymbol{\alpha}_0\right) \tag{33}$$

and substituting (32) and (33) in (31), we find

$$\dot{\boldsymbol{e}}_2 = \boldsymbol{B}^{-1}(\boldsymbol{q})\left(-\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega})\boldsymbol{e}_2 + \boldsymbol{E}\boldsymbol{e}_3 + \boldsymbol{\alpha}_0\right) \tag{34}$$

where $\boldsymbol{\alpha}_0$ is the PD controller described by

$$\boldsymbol{\alpha}_0 = -\boldsymbol{K}_p\boldsymbol{e}_1 - \boldsymbol{K}_d\boldsymbol{e}_2 \tag{35}$$

With the virtual input defined as above, the differential pressure error dynamics an be expressed as

$$\dot{\boldsymbol{e}}_3 = -\frac{2\beta}{V_0}\boldsymbol{E}\boldsymbol{\omega} + \boldsymbol{v}(\boldsymbol{P}_L, \boldsymbol{u}) - \dot{\boldsymbol{\alpha}}_1 \tag{36}$$

where $\boldsymbol{v}(\boldsymbol{P}_L, \boldsymbol{u})$ is defined in (26). We now take $\boldsymbol{v}(\boldsymbol{P}_L, \boldsymbol{u})$ as a new virtual input. If a suitable expression for this virtual input is found, $\boldsymbol{v}(\boldsymbol{P}_L, \boldsymbol{u})$, can be solved for the real input vector $\boldsymbol{u}$. In order to find a suitable expression for the virtual input, consider the Lyapunov function candidate

$$\begin{aligned} V(\boldsymbol{e}) =& \frac{1}{2}\boldsymbol{e}_2^T\boldsymbol{B}(\boldsymbol{q})\boldsymbol{e}_2 + \frac{1}{2}\boldsymbol{e}_1^T\boldsymbol{K}_p\boldsymbol{e}_1 \\ &+ \epsilon\boldsymbol{e}_1^T\boldsymbol{B}(\boldsymbol{q})\boldsymbol{e}_2 + \frac{1}{2}\boldsymbol{e}_3^T\boldsymbol{e}_3 \end{aligned} \tag{37}$$

where $\boldsymbol{e} = [\boldsymbol{e}_1, \boldsymbol{e}_2^T, \boldsymbol{e}_3^T]^T$. This Lyapunov function candidate is positive definite if a sufficiently small value for the constant $\epsilon > 0$ is chosen, and the proportional gain matrix is positive definite. We also note that the function is radially unbounded. Thus, if we can find a virtual input $\boldsymbol{v}(\boldsymbol{P}_L, \boldsymbol{u})$, such that the Lyapunov function candidate time derivative along the system trajectories, $\dot{V} < 0$, the candidate is indeed a Lyapunov function, and the closed loop system is globally asymptotically stable (Khalil, 2002). If the virtual input is chosen as

$$\begin{aligned} \boldsymbol{v}(\boldsymbol{P}_L, \boldsymbol{u}) =& \frac{2\beta}{V_0}\boldsymbol{E}\boldsymbol{\omega} + \dot{\boldsymbol{\alpha}}_1 - \boldsymbol{E}\boldsymbol{e}_2 \\ &- \epsilon\boldsymbol{E}\boldsymbol{e}_1 - \boldsymbol{K}_3\boldsymbol{e}_3 \\ =& \boldsymbol{v}_d \end{aligned} \tag{38}$$

where $\boldsymbol{K}_3$ is a positive definite gain matrix,

$$\dot{\boldsymbol{\alpha}}_1 \approx \boldsymbol{E}^{-1}(\dot{\boldsymbol{C}}\dot{\boldsymbol{q}}_d + \boldsymbol{C}\ddot{q}_d + \dot{\boldsymbol{g}} + \dot{\boldsymbol{\tau}}_f + \dot{\boldsymbol{B}}\ddot{q}_d + \boldsymbol{B}\dddot{q}_d + \dot{\boldsymbol{\alpha}}_0) \tag{39}$$

and

$$\dot{\boldsymbol{\alpha}}_0 = -\boldsymbol{K}_p\boldsymbol{e}_2 - \boldsymbol{K}_d\boldsymbol{B}^{-1}(-\boldsymbol{C}\boldsymbol{e}_2) + \boldsymbol{\alpha}_0 + \boldsymbol{E}\boldsymbol{e}_3 \tag{40}$$

we find that

$$\begin{aligned} \dot{V} &= -\boldsymbol{e}^T \begin{bmatrix} \epsilon\boldsymbol{K}_p & \epsilon\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega}) & 0 \\ \epsilon(\boldsymbol{K}_d - \dot{\boldsymbol{B}}(\boldsymbol{q})) & \boldsymbol{K}_d - \epsilon\boldsymbol{B}(\boldsymbol{q}) & 0 \\ 0 & 0 & \boldsymbol{K}_3 \end{bmatrix} \boldsymbol{e} \\ &= \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e} \end{aligned} \tag{41}$$

where it is used that $\boldsymbol{z}^T(\dot{\boldsymbol{B}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\omega}))\boldsymbol{z} = 0$ for any $\boldsymbol{z}$. As the matrix $\boldsymbol{P}$ is positive definite for sufficiently small values of $\epsilon$ and positive definite gain matrices $\boldsymbol{K}_p$, and $\boldsymbol{K}_d$, $\dot{V} < 0$. Thus, the system is asymptotically stable. As the Lyapunov function $V(\boldsymbol{e})$ is radially unbounded, this result is global, and the system is globally asymptotically stable.

In order to find the control input $\boldsymbol{u}$ that yields the virtual input $\boldsymbol{v}_d$, we solve the virtual input for $\boldsymbol{u}$. Combining (26) and (38), we find

$$\boldsymbol{u} = \frac{V_0\sqrt{\rho}}{2\beta C_d}\boldsymbol{\Gamma}^{-1}\boldsymbol{v}_d \tag{42}$$

where $\boldsymbol{v}_d$ is defined in (38).

### *6.1 Closing Remarks*

In order to be able to calculate an input $\boldsymbol{u}$, it is required that $\boldsymbol{\Gamma}^{-1}$ exists. This is guaranteed as long as $\nu_i > 0$ for all $i$, where $\nu_i$ is defined in (25). This in turn requires that $P_s - sgn(u_i)P_{Li} > 0$ for all $i$. For a high pressure system, this is reasonable to assume. An other requirement for this controller is that the reference positions $\boldsymbol{q}_d$ need to be three times continuously differentiable, such that $\dddot{\boldsymbol{q}}_d$ exists. It is also worth noting that the stability proof provided above is based on the control plant model, and is as such only valid under the assumption that the real system behave according to the control plant model.

## 7. CONCLUSION

We have in this paper derived simulation models for electrohydraulical systems with both linear and angular actuators, and presented a manner in which to use the simulation models in order to decide whether valve dynamics modelling are needed for a control model design of a particular system. Furthermore, we have derived a typical simplified control plant model, dependent only on the differential pressure, as opposed to both the upside and downside pressure of the actuator. The control plant model have been compared to the simulation plant model through simulations, and found to be good a approximation for that particular system. The control plant models for the electrohydraulic systems were connected to a robotic manipulator model in a compact matrix form, and finally, a back-stepping trajectory tracking control law for the hydraulic manipulator was designed and proven globally asymptotically stable.

## REFERENCES

Angue-Mintsa, H., Venugopal, R., Kenn, J.P., and Belleau, C. (2011). Adaptive position control of an electrohydraulic servo system with load disturbance rejection and friction compensation. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 133(6). Cited By (since 1996)0.

Bu, F. and Yao, B. (2000). Observer based coordinated adaptive robust control of robot manipulators driven by single-rod hydraulic actuators. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, 3034–3039 vol.3. doi: 10.1109/ROBOT.2000.846488.

Karnopp, D., C., Margolis, D., L., and Rosenberg, R., C. (2006). *System Dynamics: Modelling and Simulation of Mechatronic Systems*. 4 edition.

Khalil, H., K. (2002). *Nonlinear Systems*. Prentice Hall PTR.

Seo, J., Venugopal, R., and Kenn, J.P. (2007). Feedback linearization based control of a rotational hydraulic drive. *Control Engineering Practice*, 15(12), 1495–1507. Cited By (since 1996)18.

Wang, X., Sun, X., and Ye, H. (2011). Finite-time position tracking control of rigid hydraulic manipulators based on high-order terminal sliding mode. *Journal of Systems and Control Engineering*.

Yao, J. and Wang, C. (2012). Model reference adaptive control for a hydraulic underwater manipulator. *JVC/Journal of Vibration and Control*, 18(6), 893–902. Cited By (since 1996)4.

Zeng, H. and Sepehri, N. (2006). Adaptive backstepping control of hydraulic manipulators with friction compensation using lugre model. volume 2006, 3164–3169. Cited By (since 1996)8.

# Appendix B

# Case Study 1

## B.1 Parameters

This section presents the most important parameters used in the Titan 4 simulator. The leftmost column in B.1 states to which system the parameters belong, while the center column provides the denotation and a description. A few observations and notes are in place before the parameters are presented.

- The local inertia matrices for each body is assumed to be diagonal, such that all of-diagonal elements are zero.
- The moments of inertia about the $x_1$ and $y_1$ axis for the first body are irrelevant because the base can only revolve about the local z-axis.
- In order to avoid vector notion in the parameter list we define the vectors $\boldsymbol{r}^i_{i+1/i} = [x_{i+1/i},\, y_{i+1/i},\, z_{i+1/i}]^T$ and $\boldsymbol{r}^i_{cgi/i} = [x_{cgi},\, y_{cgi},\, z_{cgi}]^T$.
- All six servo valves are identical.
- All controller gain matrices are diagonal such that
  $\boldsymbol{K}_i = \text{diag}(k_{i1},\, k_{i2},\, ...,\, k_{i6})$
  $\boldsymbol{K}_p = \text{diag}(k_{p1},\, k_{p2},\, ...,\, k_{p6})$
  $\boldsymbol{K}_d = \text{diag}(k_{d1},\, k_{d2},\, ...,\, k_{d6})$
  $\boldsymbol{K}_3 = \text{diag}(k_{31},\, k_{32},\, ...,\, k_{36})$.

Table B.1: Parameters and coefficients used in the bond graph model

| **System** | **Description** | **Value** |
|---|---|---|
| Body 1 | $m_1$, Mass of body | 15 kg |
| | $I_z$, Moment of inertia about $z_1 - axis$ | $0.108\,kgm^2$ |
| | $x_{2/1}$, Distance along $x_1$ axis between 1 and 2 frame | $0\,m$ |
| | $y_{2/1}$, Distance along $y_1$ axis between 1 and 2 frame | $0.121\,m$ |
| | $z_{2/1}$, Distance along $z_1$ axis between 1 and 2 frame | $0.195\,m$ |
| | $x_{cg1}$, Distance along $x_1$ axis between 1 frame and CG | $0\,m$ |
| | $y_{cg1}$, Distance along $y_1$ axis between 1 frame and CG | $0\,m$ |
| | $z_{cg1}$, Distance along $z_1$ axis between 1 frame and CG | $0.1\,m$ |
| Body 2 | $m_2$, Mass of body | 40 kg |
| | $I_x$, Moment of inertia about $x_2 - axis$ | $0.216\,kgm^2$ |
| | $I_y$, Moment of inertia about $y_2 - axis$ | $2.522\,kgm^2$ |
| | $I_z$, Moment of inertia about $z_2 - axis$ | $2.522\,kgm^2$ |
| | $x_{3/2}$, Distance along $x_2$ axis between 2 and 3 frame | $0.851\,m$ |
| | $y_{3/2}$, Distance along $y_2$ axis between 2 and 3 frame | $0\,m$ |
| | $z_{3/2}$, Distance along $z_2$ axis between 2 and 3 frame | $0\,m$ |
| | $x_{cg2}$, Distance along $x_2$ axis between 2 frame and CG | $0.39\,m$ |
| | $y_{cg2}$, Distance along $y_2$ axis between 2 frame and CG | $0\,m$ |
| | $z_{cg2}$, Distance along $z_2$ axis between 2 frame and CG | $0\,m$ |
| Body 3 | $m_3$, Mass of body | 17 kg |
| | $I_x$, Moment of inertia about $x_3 - axis$ | $0.082\,kgm^2$ |
| | $I_y$, Moment of inertia about $y_3 - axis$ | $0.372\,kgm^2$ |
| | $I_z$, Moment of inertia about $z_3 - axis$ | $0.241\,kgm^2$ |
| | $x_{4/3}$, Distance along $x_3$ axis between 3 and 4 frame | $0.483\,m$ |
| | $y_{4/3}$, Distance along $y_3$ axis between 3 and 4 frame | $0\,m$ |
| | $z_{4/3}$, Distance along $z_3$ axis between 3 and 4 frame | $0\,m$ |
| | $x_{cg3}$, Distance along $x_3$ axis between 3 frame and CG | $0.241\,m$ |
| | $y_{cg3}$, Distance along $y_3$ axis between 3 frame and CG | $0\,m$ |
| | $z_{cg3}$, Distance along $z_3$ axis between 3 frame and CG | $0\,m$ |
| Body 4 | $m_4$, Mass of body | 8 kg |
| | $I_x$, Moment of inertia about $x_4 - axis$ | $0.034\,kgm^2$ |
| | $I_y$, Moment of inertia about $y_4 - axis$ | $0.029\,kgm^2$ |
| | $I_z$, Moment of inertia about $z_4 - axis$ | $0.363\,kgm^2$ |
| | $x_{5/4}$, Distance along $x_4$ axis between 4 and 5 frame | $0.133\,m$ |
| | $y_{5/4}$, Distance along $y_4$ axis between 4 and 5 frame | $0\,m$ |
| | $z_{5/4}$, Distance along $z_4$ axis between 4 and 5 frame | $0\,m$ |

| | | |
|---|---|---|
| | $x_{cg4}$, Distance along $x_4$ axis between 4 frame and CG | $0.065\,m$ |
| | $y_{cg4}$, Distance along $y_4$ axis between 4 frame and CG | $0\,m$ |
| | $z_{cg4}$, Distance along $z_4$ axis between 4 frame and CG | $0\,m$ |
| Body 5 | $m_5$, Mass of body | 8 kg |
| | $I_x$, Moment of inertia about $x_5 - axis$ | $0.025\,kgm^2$ |
| | $I_y$, Moment of inertia about $y_5 - axis$ | $0.034\,kgm^2$ |
| | $I_z$, Moment of inertia about $z_5 - axis$ | $0.029\,kgm^2$ |
| | $x_{6/5}$, Distance along $x_5$ axis between 5 and 6 frame | $0\,m$ |
| | $y_{6/5}$, Distance along $y_5$ axis between 5 and 6 frame | $0.107\,m$ |
| | $z_{6/5}$, Distance along $z_5$ axis between 5 and 6 frame | $0\,m$ |
| | $x_{cg5}$, Distance along $x_5$ axis between 5 frame and CG | $0\,m$ |
| | $y_{cg5}$, Distance along $y_5$ axis between 5 frame and CG | $0.053\,m$ |
| | $z_{cg5}$, Distance along $z_5$ axis between 5 frame and CG | $0\,m$ |
| Body 6 | $m_6$, Mass of body | 12 kg |
| | $I_x$, Moment of inertia about $x_6 - axis$ | $0.025\,kgm^2$ |
| | $I_y$, Moment of inertia about $y_6 - axis$ | $0.034\,kgm^2$ |
| | $I_z$, Moment of inertia about $z_6 - axis$ | $0.029\,kgm^2$ |
| | $x_{cg6}$, Distance along $x_6$ axis between 5 frame and CG | $0\,m$ |
| | $y_{cg6}$, Distance along $y_6$ axis between 5 frame and CG | $0\,m$ |
| | $z_{cg6}$, Distance along $z_6$ axis between 5 frame and CG | $0.083\,m$ |
| Hydraulics | $\rho$, Density of hydraulic fluid | $950\,{}^{\mathrm{kg}}/_{m^3}$ |
| | $C_d$, Hydraulic discharge coefficient | $0.9\,-$ |
| | $\beta$, Total bulk modulus | $1e8\,{}^{\mathrm{N}}/_{m^2}$ |
| | $V_0$, Initial volume of hydraulic fluid. | 1e-3 $m^3$ |
| | $P_S$, Hydraulic supply pressure | 187e5 Pa |
| | $P_0$, Hydraulic return pressure | 0 Pa |
| Actuator 1 | $V_p$, Volumetric displacement | 1e-4 ${}^{m^3}/_{\mathrm{rad}}$ |
| Actuator 2 | $A_{up}$, Upside piston area | 9.6e-4 $m^2$ |
| | $A_{dwn}$, Downside piston area. | $7.8\,m^2$ |
| | $L$, Length of torque arm for actuator | 0.425 m |
| | $L_1$, Length parameter | 0.117 m |
| | $\beta_0$, Angle parameter | 1.149 rad |
| Actuator 3 | $V_p$, Volumetric displacement | 1e-4 ${}^{m^3}/_{\mathrm{rad}}$ |
| Actuator 4 | $V_p$, Volumetric displacement | 1e-5 ${}^{m^3}/_{\mathrm{rad}}$ |
| Actuator 5 | $V_p$, Volumetric displacement | 1e-5 ${}^{m^3}/_{\mathrm{rad}}$ |
| Actuator 6 | $V_p$, Volumetric displacement | 1e-5 ${}^{m^3}/_{\mathrm{rad}}$ |
| Joints | $d_1$, Linear friction coefficient joint 1 | $100\,{}^{kgm^2}/_{\mathrm{s}}$ |

| | | |
|---|---|---|
| | $d_2$, Linear friction coefficient joint 2 | 80 $kgm^2$/s |
| | $d_3$, Linear friction coefficient joint 3 | 50 $kgm^2$/s |
| | $d_4$, Linear friction coefficient joint 4 | 20 $kgm^2$/s |
| | $d_5$, Linear friction coefficient joint 5 | 5 $kgm^2$/s |
| | $d_6$, Linear friction coefficient joint 6 | 2.5 $kgm^2$/s |
| Bumpers | $q_{1up}$, Upper displacement limit joint 1 | 2.09 $rad$ |
| | $q_{1dwn}$, Lower displacement limit joint 1 | -2.09 rad |
| | $q_{2up}$, Upper displacement limit joint 2 | 1.36 rad |
| | $q_{2dwn}$, Lower displacement limit joint 2 | -0.45 rad |
| | $q_{3up}$, Upper displacement limit joint 3 | 1.87 rad |
| | $q_{3dwn}$, Lower displacement limit joint 3 | -2.84 rad |
| | $q_{4up}$, Upper displacement limit joint 4 | 1.57 rad |
| | $q_{4dwn}$, Lower displacement limit joint 4 | -1.57 rad |
| | $q_{5up}$, Upper displacement limit joint 5 | 1.57 rad |
| | $q_{5dwn}$, Lower displacement limit joint 5 | -1.57 rad |
| Controller | $k_{i1}$, Joint 1 integrator gain | 5e4 Nm/s |
| | $k_{i2}$, Joint 2 integrator gain | 5e4 Nm/s |
| | $k_{i3}$, Joint 3 integrator gain | 5e4 Nm/s |
| | $k_{i4}$, Joint 4 integrator gain | 5e2 Nm/s |
| | $k_{i5}$, Joint 5 integrator gain | 100 Nm/s |
| | $k_{i6}$, Joint 6 integrator gain | 100 Nm/s |
| | $k_{p1}$, Joint 1 proportional gain | 5e4 N/m |
| | $k_{p2}$, Joint 2 proportional gain | 5e4 N/m |
| | $k_{p3}$, Joint 3 proportional gain | 3e4 N/m |
| | $k_{p4}$, Joint 4 proportional gain | 5e2 N/m |
| | $k_{p5}$, Joint 5 proportional gain | 200 N/m |
| | $k_{p6}$, Joint 6 proportional gain | 250 N/m |
| | $k_{d1}$, Joint 1 derivative gain | 0 - |
| | $k_{d2}$, Joint 2 derivative gain | 0 - |
| | $k_{d3}$, Joint 3 derivative gain | 0 - |
| | $k_{d4}$, Joint 4 derivative gain | 0 - |
| | $k_{d5}$, Joint 5 derivative gain | 0 - |
| | $k_{d6}$, Joint 6 derivative gain | 0 - |
| | $k_{31}$, Joint 1 pressure gain | 5 1/s |
| | $k_{32}$, Joint 2 pressure gain | 0 1/s |
| | $k_{33}$, Joint 3 pressure gain | 0 1/s |
| | $k_{34}$, Joint 4 pressure gain | 0.1 1/s |
| | $k_{35}$, Joint 5 pressure gain | 1 1/s |

| | | |
|---|---|---|
| | $k_{36}$, Joint 6 pressure gain | 1 $^1/_s$ |
| Guidance | $\dot{q}_{1max}$, Joint 1 rate limit | 0.4 $^{rad}/_s$ |
| | $\dot{q}_{2max}$, Joint 2 rate limit | 0.3 $^{rad}/_s$ |
| | $\dot{q}_{3max}$, Joint 3 rate limit | 0.3 $^{rad}/_s$ |
| | $\dot{q}_{4max}$, Joint 4 rate limit | 0.5 $^{rad}/_s$ |
| | $\dot{q}_{5max}$, Joint 5 rate limit | 0.5 $^{rad}/_s$ |
| | $\dot{q}_{6max}$, Joint 6 rate limit | 0.8 $^{rad}/_s$ |
| | $\epsilon$, Displacement limit safety buffer | 0.06 rad |
| | $\xi_1$, Joint 1 relative damping ratio | 1 - |
| | $\xi_2$, Joint 2 relative damping ratio | 1 - |
| | $\xi_3$, Joint 3 relative damping ratio | 1 - |
| | $\xi_4$, Joint 4 relative damping ratio | 1 - |
| | $\xi_5$, Joint 5 relative damping ratio | 1 - |
| | $\xi_6$, Joint 6 relative damping ratio | 1 - |
| | $\omega_{n1}$, Joint 1 natural frequency | 10 $^{rad}/_s$ |
| | $\omega_{n2}$, Joint 2 natural frequency | 10 $^{rad}/_s$ |
| | $\omega_{n3}$, Joint 3 natural frequency | 10 $^{rad}/_s$ |
| | $\omega_{n4}$, Joint 4 natural frequency | 10 $^{rad}/_s$ |
| | $\omega_{n5}$, Joint 5 natural frequency | 10 $^{rad}/_s$ |
| | $\omega_{n6}$, Joint 6 natural frequency | 10 $^{rad}/_s$ |

## B.2 Codes and Algorithms

This section provides the Maple code and the underlying code for the IC-field for the Titan 4 Simulator. The purpose of the Maple code is to first generate symbolic expressions for the mass-inertia matrix, its partial derivatives with respect to the generalized coordinates and the restoring forces. Second, the Maple code outputs optimized C-code, which is used in order to build a dynamic link library file which the 20-sim code can access in order to update the expressions in question. The Maple code is presented first, and the IC-field code next.

*restart; with(VectorCalculus) : with (Student[LinearAlgebra]) : with(CodeGeneration) :*
*with(LinearAlgebra) : with(ArrayTools) :*

# Titan 4 Manipulator Dynamic Model.

By Boerge Rokseth 02.06.2014

pi := Pi

**Rotation matrices**

$R0\_1 := Matrix([[\cos(q1), -\sin(q1), 0], [\sin(q1), \cos(q1), 0], [0, 0, 1]])$ :

$R1\_2 := Matrix\left(\left[\left[\cos\left(\frac{pi}{2}\right), -\sin\left(\frac{pi}{2}\right), 0\right], \left[\sin\left(\frac{pi}{2}\right), \cos\left(\frac{pi}{2}\right), 0\right], [0, 0, 1]\right]\right) Matrix\left(\left[[1, 0, 0], \left[0, \cos\left(\frac{pi}{2}\right), -\sin\left(\frac{pi}{2}\right)\right], \left[0, \sin\left(\frac{pi}{2}\right), \cos\left(\frac{pi}{2}\right)\right]\right]\right) Matrix([[\cos(q2), -\sin(q2), 0], [\sin(q2), \cos(q2), 0], [0, 0, 1]])$ :

$R2\_3 := Matrix([[\cos(q3), -\sin(q3), 0], [\sin(q3), \cos(q3), 0], [0, 0, 1]])$ :

$R4\_4 := Matrix([[\cos(q4), -\sin(q4), 0], [\sin(q4), \cos(q4), 0], [0, 0, 1]])$ :

$R4\_5 := Matrix\left(\left[\left[\cos\left(-\frac{pi}{2}\right), 0, \sin\left(-\frac{pi}{2}\right)\right], [0, 1, 0], \left[-\sin\left(-\frac{pi}{2}\right), 0, \cos\left(-\frac{pi}{2}\right)\right]\right]\right) Matrix\left(\left[[1, 0, 0], \left[0, \cos\left(-\frac{pi}{2}\right), -\sin\left(-\frac{pi}{2}\right)\right], \left[0, \sin\left(-\frac{pi}{2}\right), \cos\left(-\frac{pi}{2}\right)\right]\right]\right) Matrix([[\cos(q5), -\sin(q5), 0], [\sin(q5), \cos(q5), 0], [0, 0, 1]])$ :

$R5\_6 := Matrix\left(\left[[1, 0, 0], \left[0, \cos\left(-\frac{pi}{2}\right), -\sin\left(-\frac{pi}{2}\right)\right], \left[0, \sin\left(-\frac{pi}{2}\right), \cos\left(-\frac{pi}{2}\right)\right]\right]\right)$
$Matrix\left(\left[\left[\cos\left(-\frac{pi}{2}\right), -\sin\left(-\frac{pi}{2}\right), 0\right], \left[\sin\left(-\frac{pi}{2}\right), \cos\left(-\frac{pi}{2}\right), 0\right], [0, 0, 1]\right]\right) Matrix([[\cos(q6), -\sin(q6), 0], [\sin(q6), \cos(q6), 0], [0, 0, 1]])$ :

**Define local vectors (joint i to i+1) in terms of i**

$r0\_1\_0 := Vector([0, 0, 0])$ :
$r1\_2\_1 := Vector([0, 0.121, 0.195])$ :
$r2\_3\_2 := Vector([0.851, 0, 0])$ :
$r3\_4\_3 := Vector([0.483, 0, 0])$ :
$r4\_5\_4 := Vector([0.133, 0, 0])$ :
$r5\_6\_5 := Vector([0, 0.107, 0])$ :

**Local vectors in terms of base frame**

$r0\_2\_1 := R0\_1\, r1\_2\_1$ :
$r0\_3\_2 := R0\_1\, R1\_2\, r2\_3\_2$ :
$r0\_4\_3 := R0\_1\, R1\_2\, R2\_3\, r3\_4\_3$ :
$r0\_5\_4 := R0\_1\, R1\_2\, R2\_3\, R3\_4\, r4\_5\_4$ :
$r0\_6\_5 := R0\_1\, R1\_2\, R2\_3\, R3\_4\, R4\_5\, r5\_6\_5$ :

**Position of each joint from base**

$r0\_2\_0 := r0\_1\_0 + r0\_2\_1$ :
$r0\_3\_0 := r0\_2\_0 + r0\_3\_2$ :
$r0\_4\_0 := r0\_3\_0 + r0\_4\_3$ :
$r0\_5\_0 := r0\_4\_0 + r0\_5\_4$ :
$r0\_6\_0 := r0\_5\_0 + r0\_6\_5$ :

**Unit vectors that each link revolve about (in terms of base frame):**

$k := Vector([0, 0, 1])$ :
$e1 := R0\_1\, k$ :
$e2 := R0\_1\, R1\_2\, k$ :
$e3 := R0\_1\, R1\_2\, R2\_3\, k$ :
$e4 := R0\_1\, R1\_2\, R2\_3\, R3\_4\, k$ :
$e5 := R0\_1\, R1\_2\, R2\_3\, R3\_4\, R4\_5\, k$ :
$e6 := R0\_1\, R1\_2\, R2\_3\, R3\_4\, R4\_5\, R5\_6\, k$ :

**Vector from joint i to cm of link i:**

$r0\_cm1 := r0\_1\_0 + R0\_1\, Vector([0, y\_cm1, z\_cm1])$ :
$r0\_cm2 := r0\_2\_0 + R0\_1\, R1\_2\, Vector([x\_cm2, y\_cm2, 0])$ :
$r0\_cm3 := r0\_3\_0 + R0\_1\, R1\_2\, R2\_3\, Vector([x\_cm3, 0, 0])$ :
$r0\_cm4 := r0\_4\_0 + R0\_1\, R1\_2\, R2\_3\, R3\_4\, Vector([x\_cm4, 0, 0])$ :
$r0\_cm5 := r0\_5\_0 + R0\_1\, R1\_2\, R2\_3\, R3\_4\, R4\_5\, Vector([0, y\_cm5, 0])$ :
$r0\_cm6 := r0\_6\_0 + R0\_1\, R1\_2\, R2\_3\, R3\_4\, R4\_5\, R5\_6\, Vector([0, 0, z\_cm6])$ :

**Geometric Jacobian for each center of mass**
**-link 1**

```
r0_cm1_1 := r0_cm1 :
J1 := Matrix([[CrossProduct(e1, r0_cm1_1), ZeroMatrix(3, Size(qdot)[1]-1)], [e1 , ZeroMatrix(3, Size(qdot)[1]-1)]]) :
```

**-link 2**

```
r0_cm2_1 := r0_cm2 − r0_1_0 :
r0 _cm2_2 := r0_cm2 − r0_2_0 :
J2 := Matrix([[CrossProduct(e1, r0_cm2_1), CrossProduct(e2, r0 _cm2_2) , ZeroMatrix(3, Size(qdot)[1] − 2)], [e1, e2, ZeroMatrix(3, Size(qdot)[1]
    − 2)]]) :
```

**-link 3**

```
r0_cm3_1 := r0_cm3 − r0_1_0 :
r0_cm3_2 := r0_cm3 - r0_2_0 :
r0_cm3_3 := r0_cm3 − r0_3_0 :
J3 := Matrix([[CrossProduct(e1, r0_cm3_1), CrossProduct(e2, r0_cm3_2) , CrossProduct(e3, r0_cm3_3),  ZeroMatrix(3, Size(qdot)[1] − 3)], [e1, e2, e3,
      ZeroMatrix(3, Size(qdot)[1] − 3)]]) :
```

**-link 4**

```
r0 _cm4_1 := r0_cm4 − r0_1_0 :
r0_cm4_2 := r0_cm4 − r0_2_0 :
r0_cm4_3 := r0_cm4 − r0_3_0 :
r0_cm4_4 := r0_cm4 − r0_4_0 :
 J4 := Matrix([[CrossProduct(e1, r1J4), CrossProduct(e2, r2J4) , CrossProduct(e3, r3J4), CrossProduct(e4, r4J4) , ZeroMatrix(3, Size(qdot)[1] − 4)],
      [e1, e2, e3, e4, ZeroMatrix(3, Size(qdot)[1] − 4)]]) :
```

**-link 5**

```
r0_cm5_1 := r0_cm5 - r0_1_0 :
r0_cm5_2 := r0_cm5 − r0_2_0 :
r0_cm5_3 := r0_cm5 −  r0_3_0 :
r0_cm5_4 := r0_cm5 − r0_4_0 :
r0_cm5_5 := r0_cm5 − r0_5_0 :
J5 := Matrix([[CrossProduct(e1, r0_cm5_1), CrossProduct(e2, r0_cm5_2) , CrossProduct(e3, r0_cm5_3), CrossProduct(e4, r0_cm5_4) , CrossProduct(e5,
      r0_cm5_5) , ZeroMatrix(3, Size(qdot)[1] − 5)], [e1, e2, e3, e4, e5, ZeroMatrix(3, Size(qdot)[1] − 5)]]) :
```

**-link 6**

```
r0 _cm6_1 := r0_cm6 − r0_1_0 :
r0_cm6_2 := r0_cm6 − r0_2_0 :
r0_cm6_3 := r0_cm6 − r0_3_0 :
r0_cm6_4 := r0_cm6 − r0_4_0 :
r0_cm6_5 := r0_cm6 − r0_5_0 :
r0_cm6_6 := r0_cm6 − r0_6_0 :
J6 := Matrix([[CrossProduct(e1, r0 _cm6_1), CrossProduct(e2, r0_cm6_2) , CrossProduct(e3, r0_cm6_3), CrossProduct(e4, r0_cm6_4) ,
      CrossProduct(e5, r0_cm6_5) , CrossProduct(e6, r0_cm6_6)], [e1, e2, e3, e4, e5, e6]]) :
```

**Inertia matrices expressed in local system:**

```
I1_1 := Matrix([[I_1x, 0, 0], [0, I_1y, 0], [0, 0, I_1z]]) :
I2_2 := Matrix([[I_2x, 0, 0], [0, I_2y, 0], [0, 0, I_2z]]) :
I3_3 := Matrix([[I_3x, 0, 0], [0, I_3y, 0], [0, 0, I_3z]]) :
I4_4 := Matrix([[I_4x, 0, 0], [0, I_4y, 0], [0, 0, I_4z]]) :
I5_5 := Matrix([[I_5x, 0, 0], [0, I_5y, 0], [0, 0, I_5z]]) :
I6_6 := Matrix([[I_6x, 0, 0], [0, I_6y, 0], [0, 0, I_6z]]) :
```

**Inertia matrices expressed in terms of the base frame**
**-Link 1**

```
R1_0 := Transpose(R0_1) :

I1 := R0_1 I1_1 R1_0 :
```

**-Link 2**

```
R0_2 := R0_1 R1_2 :
R2_0 := Transpose(R0_2) :
I2 := R0_2 I2_2 R2_0 :
```

**-Link 3**

```
R0_3 := R0_2 R2_3 :
R3_0 := Transpose(R0_3) :
I3 := R0_3 I3_3 R3_0 :
```

**-Link 4**

```
R0_4 := R0_3 R3_4 :
R4_0 := Transpose(R0_4) :
I4 := R0_4 I4_4 R4_0 :
```

**-Link 5**

```
R0_5 := R0_4 R4_5 :
R5_0 := Transpose(R0_5) :
I5 := R0_5 I5_5 R5_0 :
```

**-Link 6**

```
R0_6 := R0_5R5_6:
R6_0 := Transpose(R0_6) :
I6 := R0_6I6_6 R6_0:
```

**Mass:**

```
M1 := IdentityMatrix(3) m1:
M2 := IdentityMatrix(3)m2:
M3 := IdentityMatrix(3)m3:
M4 := IdentityMatrix(3)m4:
M5 := IdentityMatrix(3)m5:
M6 := IdentityMatrix(3)m6:
```

**Mass matrices:**

```
B1 := Transpose(J1) Matrix([[M1, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), I1]]) J1:
B2 := Transpose(J2) Matrix([[M2, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), I2]]) J2:
B3 := Transpose(J3) Matrix([[M3, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), I3]]) J3:
B4 := Transpose(J4) Matrix([[M4, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), I4]]) J4:
B5 := Transpose(J5) Matrix([[M5, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), I5]]) J5:
B6 := Transpose(J6) Matrix([[M6, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), I6]]) J6:
```

**Gravity vector:**

```
g0 := Vector([0, 0,-g]) :
```

```
B := B1+B2+B3+B4+B5+B6:
dBdq1 := simplify(map(diff, B, q1)) :
dBdq2 := simplify(map(diff, B, q2)) :
dBdq3 := simplify(map(diff, B, q3)) :
dBdq4 := simplify(map(diff, B, q4)) :
dBdq5 := simplify(map(diff, B, q5)) :
dBdq6 := simplify(map(diff, B, q6)) :
dVdq1 := m1·g0%T Jl1[1 ..3, 1] + m2·g0%T Jl2[1 ..3, 1] +m3·g0%T Jl3[1 ..3, 1] +m4·g0%T Jl4[1 ..3, 1] +m5·g0%T Jl5[1 ..3, 1] + m6·g0%T Jl6[1 ..3, 1] :
dVdq2 := m1·g0%T Jl1[1 ..3, 2] +m2·g0%T Jl2[1 ..3, 2] +m3·g0%T Jl3[1 ..3, 2] +m4·g0%T Jl4[1 ..3, 2] +m5·g0%T Jl5[1 ..3, 2] +m6·g0%T Jl6[1 ..3, 2] :
dVdq3 := m1·g0%T Jl1[1 ..3, 3] +m2·g0%T Jl2[1 ..3, 3] +m3·g0%T Jl3[1 ..3, 3] +m4·g0%T Jl4[1 ..3, 3] +m5·g0%T Jl5[1 ..3, 3] +m6·g0%T Jl6[1 ..3, 3] :
dVdq4 := m1·g0%T Jl1[1 ..3, 4] +m2·g0%T Jl2[1 ..3, 4] +m3·g0%T Jl3[1 ..3, 4] +m4·g0%T Jl4[1 ..3, 4] +m5·g0%T Jl5[1 ..3, 4] +m6·g0%T Jl6[1 ..3, 4] :
dVdq5 := m1·g0%T Jl1[1 ..3, 5] +m2·g0%T Jl2[1 ..3, 5] +m3·g0%T Jl3[1 ..3, 5] +m4·g0%T Jl4[1 ..3, 5] +m5·g0%T Jl5[1 ..3, 5] +m6·g0%T Jl6[1 ..3, 5] :
dVdq6 := m1·g0%T Jl1[1 ..3, 6] +m2·g0%T Jl2[1 ..3, 6] +m3·g0%T Jl3[1 ..3, 6] +m4·g0%T Jl4[1 ..3, 6] +m5·g0%T Jl5[1 ..3, 6] +m6·g0%T Jl6[1 ..3, 6] :
```

**Code generation**

```
with(CodeGeneration) :
with(codegen, optimize, makeproc) :
```

**Matrices B and dBdqi are symmetric so code generated onle for one triangle of each**

```
for i from 1 to 6 do for j from i to 6 do  F := makeproc(B[i, j]); C(F, resultname = cat("b", i, j), output = "bij.txt", optimize) :end do end do:

for i from 1 to 6 do for j from i to 6 do  F := makeproc(dBdq1[i, j]); C(F, resultname = cat("dbd1_", i, j), output = "bij.txt", optimize) :end do end
    do:


for i from 1 to 6 do for j from i to 6 do  F := makeproc(dBdq2[i, j]); C(F, resultname = cat("dbd2_", i, j), output = "bij.txt", optimize) :end do end
    do:

for i from 1 to 6 do for j from i to 6 do  F := makeproc(dBdq3[i, j]); C(F, resultname = cat("dbd3_", i, j), output = "bij.txt", optimize) :end do end
    do:

for i from 1 to 6 do for j from i to 6 do  F := makeproc(dBdq4[i, j]); C(F, resultname = cat("dbd4_", i, j), output = "bij.txt", optimize) :end do end
    do:

for i from 1 to 6 do for j from i to 6 do  F := makeproc(dBdq5[i, j]); C(F, resultname = cat("dbd5_", i, j), output = "bij.txt", optimize) :end do end
    do:

for i from 1 to 6 do for j from i to 6 do  F := makeproc(dBdq6[i, j]); C(F, resultname = cat("dbd6_", i, j), output = "bij.txt", optimize) :end do end
    do:
F := makeproc(dVdq1); C(F, resultname = "dv1", output = "bij.txt", optimize) :
F := makeproc(dVdq2); C(F, resultname = "dv2", output = "bij.txt", optimize) :
F := makeproc(dVdq3); C(F, resultname = "dv3", output = "bij.txt", optimize) :
F := makeproc(dVdq4); C(F, resultname = "dv4", output = "bij.txt", optimize) :
F := makeproc(dVdq5); C(F, resultname = "dv5", output = "bij.txt", optimize) :
F := makeproc(dVdq6); C(F, resultname = "dv6", output = "bij.txt", optimize) :
```

```
/*---------------------------------------------------------------
    IC-field code - Titan4 Manipulator Simulator
-----------------------------------------------------------------
By Boerge Rokseth 02.06.2014
---------------------------------------------------------------*/



parameters
  string dll_name = 'Titan4_Dynamics.dll';
  real animScale = 1.5;//parameter for 3Danimation
variables
  //output from external functions
  real B_out[21];
  real db_out1[21];
  real db_out2[21];
  real db_out3[21];
  real db_out4[21];
  real db_out5[21];
  real db_out6[21];
  real dV1_out[1];
  real dV2_out[1];
  real dV3_out[1];
  real dV4_out[1];
  real dV5_out[1];
  real dV6_out[1];
  //states placeholders
  real p[6]; //generalized momentum
  real q[6]; //generalized displacement
  //rate of change generalized displacement
  real q1_dot;
  real q2_dot;
  real q3_dot;
  real q4_dot;
  real q5_dot;
  real q6_dot;
  //rate of change generalized momentum
  real p1_dot;
  real p2_dot;
  real p3_dot;
  real p4_dot;
  real p5_dot;
  real p6_dot;
  //mass matrix
  real global B[6,6];
  real global Binv[6,6];
  real global g[6];
  real global C[6,6];
  real c1[1,6], c2[1,6], c3[1,6], c4[1,6], c5[1,6], c6[1,6];
  //mass matrix differentiated wrt generalized coordinates
  real dBd1[6,6];
  real dBd2[6,6];
  real dBd3[6,6];
  real dBd4[6,6];
  real dBd5[6,6];
  real dBd6[6,6];
  //potential energy rate of change wrt generalized displacements
  real dVd1;
  real dVd2;
  real dVd3;
  real dVd4;
  real dVd5;
  real dVd6;
  //Rotation matrices for animation
  real R01[3,3];
  real R12[3,3];
  real R23[3,3];
  real R34[3,3];
  real R45[3,3];
  real R56[3,3];
  //variables for animation
  real x12, x23, x34, x45, x56, x6e;
  real y12, y23, y34, y45, y56, y6e;
  real z12, z23, z34, z45, z56, z6e;
  real H1,R1, L2, L3, L4, L5, L6, B2, B3, H2, H3;
  real xc_f2, xc_f3, xc_f4, yc_f5, zc_f6;
initialequations
  x12 = 0;
  x23 = 0.851*animScale;
  x34 = 0.483*animScale;
  x45 = 0.133*animScale;
  x56 = 0;
  x6e = 0;
  y12 = 0.121*animScale;
  y23 = 0;
  y34 = 0;
  y45 = 0;
  y56 = 0.107*animScale;
  y6e = 0;
  z12 = 0.195*animScale;
  z23 = 0;
  z34 = 0;
  z45 = 0;
  z56 = 0;
  z6e = (0.336-0.107)*animScale;
  H1 = z12;
  R1 = y12;
  L2 = x23;
  L3 = x34;
  L4 = x45;
  L5 = y56;
  L6 = z6e;
  B2 = 0.2*animScale;
  B3 = 0.18*animScale;
  H2 = 0.15*animScale;
  H3 = 0.1*animScale;
  xc_f2 = x23/2;
  xc_f3 = x34/2;
  xc_f4 = x45/2;
  yc_f5 = y56/2;
  zc_f6 = z6e/2;
equations
  //Rotation matrices for animation
  R01 = [cos(q[1]), -sin(q[1]), 0; sin(q[1]), cos(q[1]), 0; 0, 0, 1];
  R12 = [0, 0, 1; cos(q[2]), -sin(q[2]), 0; sin(q[2]), cos(q[2]), 0];
  R23 = [cos(q[3]), -sin(q[3]), 0; sin(q[3]), cos(q[3]), 0; 0, 0, 1];
  R34 = [cos(q[4]), -sin(q[4]), 0; sin(q[4]), cos(q[4]), 0; 0, 0, 1];
  R45 = [sin(q[5]), cos(q[5]), 0; 0, 0, 1; cos(q[5]), -sin(q[5]), 0];
  R56 = [sin(q[6]), cos(q[6]), 0; 0, 0, 1; cos(q[6]), -sin(q[6]), 0];
  //mass matrix
  B_out = dll(dll_name,'massMatrix',q);
  B = [B_out[1],B_out[2], B_out[3], B_out[4], B_out[5], B_out[6];
      B_out[2],B_out[7], B_out[8], B_out[9], B_out[10],B_out[11];
      B_out[3],B_out[8], B_out[12],B_out[13],B_out[14],B_out[15];
      B_out[4],B_out[9], B_out[13],B_out[16],B_out[17],B_out[18];
      B_out[5],B_out[10],B_out[14],B_out[17],B_out[19],B_out[20];
      B_out[6],B_out[11],B_out[15],B_out[18],B_out[20],B_out[21]];
  //mass matrix derivatives
  db_out1 = dll(dll_name,'divMass_1',q);
  dBd1 = [db_out1[1],db_out1[2], db_out1[3], db_out1[4], db_out1[5], db_out1[6];
          db_out1[2],db_out1[7], db_out1[8], db_out1[9], db_out1[10],db_out1[11];
          db_out1[3],db_out1[8], db_out1[12],db_out1[13],db_out1[14],db_out1[15];
          db_out1[4],db_out1[9], db_out1[13],db_out1[16],db_out1[17],db_out1[18];
          db_out1[5],db_out1[10],db_out1[14],db_out1[17],db_out1[19],db_out1[20];
          db_out1[6],db_out1[11],db_out1[15],db_out1[18],db_out1[20],db_out1[21]];
  db_out2 = dll(dll_name,'divMass_2',q);
  dBd2 = [db_out2[1],db_out2[2], db_out2[3], db_out2[4], db_out2[5], db_out2[6];
          db_out2[2],db_out2[7], db_out2[8], db_out2[9], db_out2[10],db_out2[11];
          db_out2[3],db_out2[8], db_out2[12],db_out2[13],db_out2[14],db_out2[15];
          db_out2[4],db_out2[9], db_out2[13],db_out2[16],db_out2[17],db_out2[18];
          db_out2[5],db_out2[10],db_out2[14],db_out2[17],db_out2[19],db_out2[20];
          db_out2[6],db_out2[11],db_out2[15],db_out2[18],db_out2[20],db_out2[21]];
```

```
db_out3 = dll(dll_name,'divMass_3',q);
dBd3 = [db_out3[1],db_out3[2], db_out3[3], db_out3[4], db_out3[5], db_out3[6];
        db_out3[2],db_out3[7], db_out3[8], db_out3[9], db_out3[10],db_out3[11];
        db_out3[3],db_out3[8], db_out3[12],db_out3[13],db_out3[14],db_out3[15];
        db_out3[4],db_out3[9], db_out3[13],db_out3[16],db_out3[17],db_out3[18];
        db_out3[5],db_out3[10],db_out3[14],db_out3[17],db_out3[19],db_out3[20];
        db_out3[6],db_out3[11],db_out3[15],db_out3[18],db_out3[20],db_out3[21]];
db_out4 = dll(dll_name,'divMass_4',q);
dBd4 = [db_out4[1],db_out4[2], db_out4[3], db_out4[4], db_out4[5], db_out4[6];
        db_out4[2],db_out4[7], db_out4[8], db_out4[9], db_out4[10],db_out4[11];
        db_out4[3],db_out4[8], db_out4[12],db_out4[13],db_out4[14],db_out4[15];
        db_out4[4],db_out4[9], db_out4[13],db_out4[16],db_out4[17],db_out4[18];
        db_out4[5],db_out4[10],db_out4[14],db_out4[17],db_out4[19],db_out4[20];
        db_out4[6],db_out4[11],db_out4[15],db_out4[18],db_out4[20],db_out4[21]];
db_out5 = dll(dll_name,'divMass_5',q);
dBd5 = [db_out5[1],db_out5[2], db_out5[3], db_out5[4], db_out5[5], db_out5[6];
        db_out5[2],db_out5[7], db_out5[8], db_out5[9], db_out5[10],db_out5[11];
        db_out5[3],db_out5[8], db_out5[12],db_out5[13],db_out5[14],db_out5[15];
        db_out5[4],db_out5[9], db_out5[13],db_out5[16],db_out5[17],db_out5[18];
        db_out5[5],db_out5[10],db_out5[14],db_out5[17],db_out5[19],db_out5[20];
        db_out5[6],db_out5[11],db_out5[15],db_out5[18],db_out5[20],db_out5[21]];
db_out6 = dll(dll_name,'divMass_6',q);
dBd6 = [db_out6[1],db_out6[2], db_out6[3], db_out6[4], db_out6[5], db_out6[6];
        db_out6[2],db_out6[7], db_out6[8], db_out6[9], db_out6[10],db_out6[11];
        db_out6[3],db_out6[8], db_out6[12],db_out6[13],db_out6[14],db_out6[15];
        db_out6[4],db_out6[9], db_out6[13],db_out6[16],db_out6[17],db_out6[18];
        db_out6[5],db_out6[10],db_out6[14],db_out6[17],db_out6[19],db_out6[20];
        db_out6[6],db_out6[11],db_out6[15],db_out6[18],db_out6[20],db_out6[21]];
//Potential energy derivatives
dV1_out = dll(dll_name,'divPot_1',q);
dVd1 = dV1_out;
dV2_out = dll(dll_name,'divPot_2',q);
dVd2 = dV2_out;
dV3_out = dll(dll_name,'divPot_3',q);
dVd3 = dV3_out;
dV4_out = dll(dll_name,'divPot_4',q);
dVd4 = dV4_out;
dV5_out = dll(dll_name,'divPot_5',q);
dVd5 = dV5_out;
dV6_out = dll(dll_name,'divPot_6',q);
dVd6 = dV6_out;

Binv = inverse(B);
p = int ([p1_1.e;p1_2.e;p1_3.e;p1_4.e;p1_5.e;p1_6.e]); // momentum
q = int ([p2_1.f;p2_2.f;p2_3.f;p2_4.f;p2_5.f;p2_6.f]); // displacement
[q1_dot; q2_dot; q3_dot;q4_dot;q5_dot;q6_dot] = Binv*p;
//centrifugal and coriolis matrix
c1 = [q1_dot,q2_dot,q3_dot,q4_dot,q5_dot,q6_dot]*dBd1;
c2 = [q1_dot,q2_dot,q3_dot,q4_dot,q5_dot,q6_dot]*dBd2;
c3 = [q1_dot,q2_dot,q3_dot,q4_dot,q5_dot,q6_dot]*dBd3;
c4 = [q1_dot,q2_dot,q3_dot,q4_dot,q5_dot,q6_dot]*dBd4;
c5 = [q1_dot,q2_dot,q3_dot,q4_dot,q5_dot,q6_dot]*dBd5;
c6 = [q1_dot,q2_dot,q3_dot,q4_dot,q5_dot,q6_dot]*dBd6;
C = 0.5*[c1[1,1],c1[1,2],c1[1,3],c1[1,4],c1[1,5],c1[1,6]  ;
       c2[1,1],c2[1,2],c2[1,3],c2[1,4],c2[1,5],c2[1,6]  ;
       c3[1,1],c3[1,2],c3[1,3],c3[1,4],c3[1,5],c3[1,6]  ;
       c4[1,1],c4[1,2],c4[1,3],c4[1,4],c4[1,5],c4[1,6]  ;
       c5[1,1],c5[1,2],c5[1,3],c5[1,4],c5[1,5],c5[1,6]  ;
       c6[1,1],c6[1,2],c6[1,3],c6[1,4],c6[1,5],c6[1,6]];
//gravity forces vector
g = [dVd1;dVd2;dVd3;dVd4;dVd5;dVd6];
//generalized momentum rate
[p1_dot;p2_dot;p3_dot;p4_dot;p5_dot;p6_dot] = C*[q1_dot;q2_dot;q3_dot;q4_dot;
     q5_dot;q6_dot] - g;
p1_1.f = q1_dot;
p2_1.e = p1_dot;
p1_2.f = q2_dot;
p2_2.e = p2_dot;
p1_3.f = q3_dot;
p2_3.e = p3_dot;
p1_4.f = q4_dot;
```

```
p2_4.e = p4_dot;
p1_5.f = q5_dot;
p2_5.e = p5_dot;
p1_6.f = q6_dot;
p2_6.e = p6_dot;
//=====================================================================
```

# Appendix C

# Case Study 2

## C.1 Parameters

This section presents the most important parameters used in the remotely operated vehicle with manipulator model. The parameters are presented in table C.1. A few observations and notes are in place before the parameters are presented.

- The manipulator used consists of the three innermost bodies of the Titan 4 manipulator. Therefore, the manipulator mass, moment of inertia, and geometry is identical to that of the three first bodies in the previous case study.
- All off-diagonal terms in the vehicle inertia tensor is assumed to be zero such that $\boldsymbol{I}_{rov} = \text{diag}(I_x,\, I_y,\, I_z)$.
- The coordinates of the center of buoyancy relative to the origin of the body fixed reference frame is defined as $\boldsymbol{r}^b_{CB/b} = [x_{CB},\, y_{CB},\, z_{CB}]^T$.
- All controller gain matrices and reference model design matrices are diagonal with the following elements
  $\boldsymbol{K}_p^{rov} = \text{diag}(k_{p1}^{rov},\, k_{p2}^{rov},\, ...,\, k_{p6}^{rov})$,
  $\boldsymbol{K}_d^{rov} = \text{diag}(k_{d1}^{rov},\, k_{d2}^{rov},\, ...,\, k_{d6}^{rov})$,
  $\boldsymbol{K}_i^{rov} = \text{diag}(k_{i1}^{rov},\, k_{i2}^{rov},\, ...,\, k_{i6}^{rov})$,
  $\boldsymbol{K}_{pT} = \text{diag}(k_{pT1},\, k_{pT2},\, ...,\, k_{pT6})$,
  $\boldsymbol{K}_{iT} = \text{diag}(k_{iT1},\, k_{iT2},\, ...,\, k_{iT6})$,
  $\boldsymbol{K}_p^{man} = \text{diag}(k_{p1}^{man},\, k_{p2}^{man}, k_{p3}^{man})$,

$\boldsymbol{K}_d^{man} = \mathrm{diag}(k_{d1}^{man},\, k_{d2}^{man},\, k_{d3}^{man}),$
$\boldsymbol{K}_i^{man} = \mathrm{diag}(k_{i1}^{man},\, k_{i2}^{man},\, k_{i3}^{man}),$
$\boldsymbol{M}_{ref}^{rov} = \mathrm{diag}(m_{ref}^{rov},\, m_{ref}^{rov},\, m_{ref}^{rov},\, I_{xref},\, I_{yref}, I_{zref}),$
$\boldsymbol{D}_{ref}^{rov} = \mathrm{diag}(d_{lin},\, d_{lin},\, d_{lin},\, d_{ang},\, d_{ang}, d_{ang}),$
$\boldsymbol{K}_{ref}^{rov} = k_{ref}^{rov}\boldsymbol{I}_{6\times6},$
$\boldsymbol{M}_{ref}^{man} = m_{ref}^{man}\boldsymbol{I}_{3\times3},$
$\boldsymbol{D}_{ref}^{man} = \mathrm{diag}(d_1,\, d_2,\, d_3),$
$\boldsymbol{K}_{ref}^{man} = k_{ref}^{man}\boldsymbol{I}_{6\times6}.$

– All six thrusters on the vehicle are identical.

– The matrices defined in the end effector-external object interface sub model are all diagonal and defined as
$\boldsymbol{D}_g^v = d_g^v\boldsymbol{I}_{3\times3}$
$\boldsymbol{D}_g^\omega = d_g^\omega\boldsymbol{I}_{3\times3}$
$\boldsymbol{K}_g^v = k_g^v\boldsymbol{I}_{3\times3}$
$\boldsymbol{K}_g^\omega = k_g^\omega\boldsymbol{I}_{3\times3}$
$\boldsymbol{M}_o = m_o\boldsymbol{I}_{3\times3}$
$\boldsymbol{I}_o = J_o\boldsymbol{I}_{3\times3}$
$\boldsymbol{D}_o^v = d_o^v\boldsymbol{I}_{3\times3}$
$\boldsymbol{D}_o^\omega = d_o^\omega\boldsymbol{I}_{3\times3}$
$\boldsymbol{K}_o^v = k_o^v\boldsymbol{I}_{3\times3}$
$\boldsymbol{K}_o^\omega = k_o^\omega\boldsymbol{I}_{3\times3}$

Table C.1: Parameters and coefficients used in the bond graph model

| **System** | **Description** | **Value** |
|---|---|---|
| Vehicle | $m_{rov}$, Mass of vehicle. | 4000 $kg$ |
| | $I_x$, Vehicle inertia about x-axis. | 2043 $kgm^2$ |
| | $I_y$, Vehicle inertia about y-axis. | 3576 $kgm^2$ |
| | $I_z$, Vehicle inertia about z-axis. | 3693 $kgm^2$ |
| | $x_{CB}$, x-coordinate of CB. | 0 $m$ |
| | $y_{CB}$, x-coordinate of CB. | 0 $m$ |
| | $z_{CB}$, x-coordinate of CB. | -0.5 $m$ |
| | $\nabla_{rov}$, volume displacement of vehicle. | 3.971 $m^3$ |
| | $X_{\dot{u}}$, Added mass in surge due to surge. | 1600 $kg$ |
| | $Y_{\dot{v}}$, Added mass in sway due to sway. | 1600 $kg$ |
| | $Z_{\dot{w}}$, Added mass in heave due to heave. | 1600 $kg$ |
| | $K_{\dot{p}}$, Added mass in roll due to roll. | 817 $kgm^2$ |
| | $M_{\dot{q}}$, Added mass in pitch due to pitch. | 1430 $kgm^2$ |

| | | |
|---|---|---|
| | $N_{\dot{r}}$, Added mass in yaw due to yaw. | $1477\,kgm^2$ |
| | $d_u$, Skin friction damping coeff. in surge. | $800\,{}^{kg}/\mathrm{s}$ |
| | $d_v$, Skin friction damping coeff. in sway. | $800\,{}^{kg}/\mathrm{s}$ |
| | $d_w$, Skin friction damping coeff. in heave. | $800\,{}^{kg}/\mathrm{s}$ |
| | $d_p$, Skin friction damping coeff. in roll. | $800\,{}^{kgm^2}/\mathrm{s}$ |
| | $d_q$, Skin friction damping coeff. in pitch. | $800\,{}^{kgm^2}/\mathrm{s}$ |
| | $d_r$, Skin friction damping coeff. in yaw. | $800\,{}^{kgm^2}/\mathrm{s}$ |
| | $C_d$, Drag coeff. on vehicle. | $1.1\,-$ |
| Manipulator | $\nabla_1$, Volume displacement body 1. | $1\,m^3$ |
| | $\nabla_2$, Volume displacement body 2. | $1\,m^3$ |
| | $\nabla_3$, Volume displacement body 3. | $1\,m^3$ |
| | $\gamma_{11}$, Added mass scaling in x-dir body 1. | $1.4\,-$ |
| | $\gamma_{12}$, Added mass scaling in y-dir body 1. | $1.4\,-$ |
| | $\gamma_{13}$, Added mass in scaling z-dir body 1. | $1.4\,-$ |
| | $\gamma_{21}$, Added mass in scaling x-dir body 1. | $1.4\,-$ |
| | $\gamma_{22}$, Added mass in scaling y-dir body 1. | $1.4\,-$ |
| | $\gamma_{23}$, Added mass in scaling z-dir body 1. | $1.4\,-$ |
| | $\gamma_{31}$, Added mass in scaling x-dir body 1. | $1.4\,-$ |
| | $\gamma_{32}$, Added mass in scaling y-dir body 1. | $1.4\,-$ |
| | $\gamma_{33}$, Added mass in scaling z-dir body 1. | $1.4\,-$ |
| | $\gamma_{14}$, Added inertia scaling about x-dir body 1. | $1.4\,-$ |
| | $\gamma_{15}$, Added inertia scaling about y-dir body 1. | $1.4\,-$ |
| | $\gamma_{16}$, Added inertia scaling about z-dir body 1. | $1.4\,-$ |
| | $\gamma_{24}$, Added inertia scaling about x-dir body 2. | $1.4\,-$ |
| | $\gamma_{25}$, Added inertia scaling about y-dir body 2. | $1.4\,-$ |
| | $\gamma_{26}$, Added inertia scaling about z-dir body 2. | $1.4\,-$ |
| | $\gamma_{34}$, Added inertia scaling about x-dir body 3. | $1.4\,-$ |
| | $\gamma_{35}$, Added inertia scaling about y-dir body 3. | $1.4\,-$ |
| | $\gamma_{36}$, Added inertia scaling about z-dir body 3. | $1.4\,-$ |
| | $s$, Number of strips to partition bodies in. | $3\,-$ |
| | $d_{sf,1}$, Skin friction damping coeff body 1. | $10\,{}^{\mathrm{kg}}/\mathrm{s}$ |
| | $d_{sf,2}$, Skin friction damping coeff body 2. | $10\,{}^{\mathrm{kg}}/\mathrm{s}$ |
| | $d_{sf,3}$, Skin friction damping coeff body 3. | $10\,{}^{\mathrm{kg}}/\mathrm{s}$ |
| | $C_d$, Drag coeff. manipulator bodies. | $0.7\,-$ |
| Thrusters | $d_1$, Thruster placement parameter. | $1.4\,m$ |
| | $d_2$, Thruster placement parameter. | $0.6\,m$ |
| | $d_3$, Thruster placement parameter. | $1.2\,m$ |
| | $d_4$, Thruster placement parameter. | $0.6\,m$ |

| | | |
|---|---|---|
| | $d_5$, Thruster placement parameter. | $0.9\,m$ |
| | $d_6$, Thruster placement parameter. | $1.2\,m$ |
| | $d_7$, Thruster placement parameter. | $0.3\,m$ |
| | $\alpha$, Thruster placement parameter. | $0.5\,rad$ |
| | $d_p$, Thrusters shaft friction coeff. | $10\,{}^{kgm^2}\!/\mathrm{s}$ |
| | $J_p$, Thrusters moment of inertia. | $5\,kgm^2$ |
| | $p$, Propeller blades pitch. | $0.8\,rad$ |
| | $R$,Propeller radius. | $0.15\,m$ |
| | $\rho$, Density of sea water. | $1025\,{}^{\mathrm{kg}}\!/_{m^3}$ |
| | $C_D$, Drag coefficient on propeller blade. | $2\,-$ |
| | $C_L$, Lift coefficient on propeller blade. | $2\,-$ |
| | $m_w$, Mass of water inside thruster duct. | $29\,kg$ |
| Ctrl.ROV | $k_{p1}^{rov}$, Proportional gain x. | $2000\,{}^{\mathrm{kg}}\!/_{s^2}$ |
| | $k_{p2}^{rov}$, Proportional gain y. | $2000\,{}^{\mathrm{kg}}\!/_{s^2}$ |
| | $k_{p3}^{rov}$, Proportional gain z. | $2000\,{}^{\mathrm{kg}}\!/_{s^2}$ |
| | $k_{p4}^{rov}$, Proportional gain roll. | $2000\,{}^{kgm^2}\!/_{s^2}$ |
| | $k_{p5}^{rov}$, Proportional gain pitch. | $2500\,{}^{kgm^2}\!/_{s^2}$ |
| | $k_{p6}^{rov}$, Proportional gain yaw. | $2000\,{}^{kgm^2}\!/_{s^2}$ |
| | $k_{d1}^{rov}$, Derivative gain x | $1000\,{}^{\mathrm{kg}}\!/_{\mathrm{s}}$ |
| | $k_{d2}^{rov}$, Derivative gain y | $1000\,{}^{\mathrm{kg}}\!/_{\mathrm{s}}$ |
| | $k_{d3}^{rov}$, Derivative gain z | $1800\,{}^{\mathrm{kg}}\!/_{\mathrm{s}}$ |
| | $k_{d4}^{rov}$, Derivative gain roll | $2000\,{}^{\mathrm{kgm}}\!/_{\mathrm{s}}$ |
| | $k_{d5}^{rov}$, Derivative gain pitch | $2000\,{}^{\mathrm{kgm}}\!/_{\mathrm{s}}$ |
| | $k_{d6}^{rov}$, Derivative gain yaw | $1500\,{}^{\mathrm{kgm}}\!/_{\mathrm{s}}$ |
| | $k_{i1}^{rov}$, Integrator gain x | $100\,{}^{\mathrm{kg}}\!/_{s^3}$ |
| | $k_{i2}^{rov}$, Integrator gain y | $100\,{}^{\mathrm{kg}}\!/_{s^3}$ |
| | $k_{i3}^{rov}$, Integrator gain z | $400\,{}^{\mathrm{kg}}\!/_{s^3}$ |
| | $k_{i4}^{rov}$, Integrator gain roll | $1000\,{}^{kgm^2}\!/_{s^3}$ |
| | $k_{i5}^{rov}$, Integrator gain pitch | $1500\,{}^{kgm^2}\!/_{s^3}$ |
| | $k_{i6}^{rov}$, Integrator gain yaw | $100\,{}^{kgm^2}\!/_{s^3}$ |
| | $k_{pT1}$, Proportional gain thruster 1 | $30\,\mathrm{m}$ |
| | $k_{pT1}$, Proportional gain thruster 2 | $30\,\mathrm{m}$ |
| | $k_{pT1}$, Proportional gain thruster 3 | $30\,\mathrm{m}$ |
| | $k_{pT1}$, Proportional gain thruster 4 | $30\,\mathrm{m}$ |
| | $k_{pT1}$, Proportional gain thruster 5 | $30\,\mathrm{m}$ |
| | $k_{pT1}$, Proportional gain thruster 6 | $30\,\mathrm{m}$ |
| | $k_{iT1}$, Integral gain thruster 1 | $1\,{}^{\mathrm{m}}\!/_{\mathrm{s}}$ |

| | | |
|---|---|---|
| | $k_{iT1}$, Integral gain thruster 2 | 1 m/s |
| | $k_{iT1}$, Integral gain thruster 3 | 1 m/s |
| | $k_{iT1}$, Integral gain thruster 4 | 1 m/s |
| | $k_{iT1}$, Integral gain thruster 5 | 1 m/s |
| | $k_{iT1}$, Integral gain thruster 6 | 1 m/s |
| Ctrl.man. | $k_{p1}^{man}$, Joint 1 proportional gain. | 800 Nm |
| | $k_{p2}^{man}$, Joint 2 proportional gain. | 2000 Nm |
| | $k_{p3}^{man}$, Joint 3 proportional gain. | 300 Nm |
| | $k_{d1}^{man}$, Joint 1 derivative gain | 50 Nms |
| | $k_{d2}^{man}$, Joint 2 derivative gain | 300 Nms |
| | $k_{d3}^{man}$, Joint 3 derivative gain | 40 Nms |
| | $k_{i1}^{man}$, Joint 1 integrator gain. | 0 Nm/s |
| | $k_{i2}^{man}$, Joint 2 integrator gain. | 20 Nm/s |
| | $k_{i3}^{man}$, Joint 3 integrator gain. | 70 Nm/s |
| Ref.mod.ROV | $m_{ref}^{rov}$, ref. mod. mass. | 5000 kg |
| | $I_{xref}$, ref. mod. inertia about x. | 3000 kg$m^2$ |
| | $I_{yref}$, ref.mod. inertia about y. | 3000 kg$m^2$ |
| | $I_{zref}$, ref. mod. inertia about z. | 3000 kg$m^2$ |
| | $d_{lin}$, ref. mod. damping lin. motion. | 2500 kg/s |
| | $d_{ang}$, ref.mod. damping ang. motion. | 2000 kg$m^2$/s |
| | $k_{ref}^{rov}$, ref. mod. gain. | 400 kg/$s^2$ |
| Ref.mod.man. | $m_{ref}^{man}$, ref. mod. mass. | 10 kg$m^2$ |
| | $d_1$, ref. damping joint 1 | 10 kg$m^2$/s |
| | $d_2$, ref. damping joint 2 | 10 kg$m^2$/s |
| | $d_3$, ref. damping joint 3 | 10 kg$m^2$/s |
| | $k_{ref}^{man}$, ref. model gain | 10 kg/$s^2$ |
| Object | $m_o$, Mass on object. | 70 kg |
| | $J_o$, Moment of inertia on object. | 4.375 kg |
| | $d_g^v$, Damping coeff. end eff.- obj., linear. | 1e4 kg/s |
| | $d_g^\omega$, Damping coeff. end eff.- obj., angular. | 20 kg$m^2$/s |
| | $k_g^v$, Stiffness coeff. end eff.-obj. - linear. | 5e5 N/m |
| | $k_g^\omega$, Stiffness coeff. end eff.-obj. - angular. | 0.05 Nm |
| | $d_o^v$, Damping coeff. obj.-surface, linear. | 1e3 kg/s |
| | $d_o^\omega$, Damping coeff. obj.-surface, angular. | 1e3 kg$m^2$s |
| | $k_o^v$, Stiffness coeff. obj.-surface, linear. | 5e5 N/m |
| | $k_o^\omega$, Stiffness coeff. obj.-surface, angular. | 3e5 Nm |

## C.2 Codes and Algorithms

This section provides the Maple code and the underlying code for the IC-field for the remotely operated vehicle with manipulator simulation. The purpose of the Maple code is similar to that of the the Titan 4 simulator. In addition, the matrix $\gamma$ must be exported as quasi-coordinates are used. The Maple code is presented forst, before the underlying code of the IC-field in 20-sim.

*restart*; *with*(*VectorCalculus*) : *with* (*Student*[*LinearAlgebra*]) : *with*(*CodeGeneration*) :
*with*(*LinearAlgebra*) : *with*(*ArrayTools*) :
*with*(*CodeGeneration*) : *with*(*codegen*, *optimize*, *makeproc*) :

```
                        INTERCONNECTED  DYNAMIC MODEL OF ROV AND 3-DOF MANIPULATOR
===============================================================================================
===========
The purpose of this document is to calculate and export expressions for the following
variables:
-----------------------------------------------------------------------------------------------
-----------
B            -system mass-inertia matrix. (nxn-matrix)
dB_dqi       -mass-inertia matrix diff.wrt. the generalized coordinates. (n nxn-matrices)
dalphaij_dq  -element (i,j) of transform gen. and quazi diff. wrt gen. coord.. (1,n matrix)
dalpha_dqi   -transform gen. and quazi diff. wrt gen. coord i. (n nxn matrices)
-----------------------------------------------------------------------------------------------
-----------
The expressions are used in the 20-sim implementation of the interconnected rov-manipulator
dynamic model.
The above variables and matrices are exported as C-code, which in turn is compiled to a
dynamically
linked library file (rov_manip_lib.dll).
-----------------------------------------------------------------------------------------------
------------
By: Boerge Rokseth 02.06.2014
===============================================================================================
============


ROTATION MATRICES
======================================================================
Reference frames: -Inertial fram {0} with x-axis pointing north and z down
                  -Body fixed frame {b} fixed to the ROV with the x axis forward and z down
                  -Frame {i} attached to manipulator link i for i = 1,2 and 3.
Rotation matrices: -R0_b transforming the representation of a vector from {b} to {0}
                   -Rb_0 from {0} to {b}
                   -Rb_1 from {1} to {b}
                   -Rb_2 from {2} to {b}
                   -Rb_3 from {3} to {b}
--------------------------------------------------------------------
```

*R_z* := *Matrix*([[cos(psi), -sin(psi), 0], [sin(psi), cos(psi), 0], [0, 0, 1]]) :
*R_y* := *Matrix*([[cos(theta), 0, sin(theta)], [0, 1, 0], [-sin(theta), 0, cos(theta)]]) :
*R_x* := *Matrix*([[1, 0, 0], [0, cos(phi), -sin(phi)], [0, sin(phi), cos(phi)]]) :
*R_x_T* := *Transpose*(*R_x*) :
*R_y_T* := *Transpose*(*R_y*) :
*Rb_1y* := *Matrix*([[cos(Pi), 0, sin( Pi)], [0, 1, 0], [-sin(Pi), 0, cos(Pi)]]) :

$$Rb\_1z := Matrix\left(\left[\left[\cos\left(\frac{\mathrm{Pi}}{2}+q1\right), -\sin\left(\frac{\mathrm{Pi}}{2}+q1\right), 0\right], \left[\sin\left(\frac{\mathrm{Pi}}{2}+q1\right), \cos\left(\frac{\mathrm{Pi}}{2}+q1\right), 0\right], [0, 0, 1]\right]\right):$$

$$R1\_2z := Matrix\left(\left[\left[\cos\left(\frac{\mathrm{Pi}}{2}\right), -\sin\left(\frac{\mathrm{Pi}}{2}\right), 0\right], \left[\sin\left(\frac{\mathrm{Pi}}{2}\right), \cos\left(\frac{\mathrm{Pi}}{2}\right), 0\right], [0, 0, 1]\right]\right):$$

$$R1\_2x := Matrix\left(\left[[1, 0, 0], \left[0, \cos\left(\frac{\mathrm{Pi}}{2}\right), -\sin\left(\frac{\mathrm{Pi}}{2}\right)\right], \left[0, \sin\left(\frac{\mathrm{Pi}}{2}\right), \cos\left(\frac{\pi}{2}\right)\right]\right]\right):$$

*R1_2z2* := *Matrix*([[cos(*q2*), -sin(*q2*), 0], [sin(*q2*), cos(*q2*), 0], [0, 0, 1]]) :
*R1_2* := *R1_2z R1_2x R1_2z2* :
*R2_3* := *Matrix*([[cos(*q3*), -sin(*q3*), 0], [sin(*q3*), cos(*q3*), 0], [0, 0, 1]]) :

*R0_b* := *R_z R_y R_x* :
*Rb_0* := *Transpose*(*R0_b*) :
*Rb_1* := *Rb_1y Rb_1z* :
*R1_b* := *Transpose*(*Rb_1*) :
*Rb_2* := *Rb_1 R1_2* :
*R2_b* := *Transpose*(*Rb_2*) :
*Rb_3* := *Rb_2 R2_3* :
*R3_b* := *Transpose*(*Rb_3*) :

```
======================================================================


TRANSFORMATIONS BETWEEN GENERALIZED AND QUASI-COORDINATES
======================================================================
```

*invTth* := *Matrix*([[1, 0, -sin(theta)], [0, cos(phi), cos(theta)·sin(phi)], [0, -sin(phi), cos(theta)·cos(phi)]]) :

*alpha_T* := *Matrix*([[*Rb_0*, *ZeroMatrix*(3, 3), *ZeroMatrix*(3, 3)], [*ZeroMatrix*(3, 3), *invTth*, *ZeroMatrix*(3, 3)], [*ZeroMatrix*(3, 3), *ZeroMatrix*(3, 3),
*IdentityMatrix*(3)]]) :
alpha := *Transpose*(*alpha_T*) :
beta := *simplify*(*MatrixInverse*(*alpha_T*)) :

```
======================================================================
```

**JACOBIANS TO RELATE JOINT RATES TO LINK CENTER OF MASS VELOCITIES**
**=================================================================**
**First we define relevant geometric vectors**
**----------------------------------------------------**
**Unit normal vectors**

```
i := Vector([1, 0, 0]) : j := Vector([0, 1, 0]) : k := Vector([0, 0, 1]) :
```

**Joints rotation axis in terms of {b}**

```
eb_1 := Rb_1 k :
eb_2 := Rb_1 R1_2 k :
eb_3 := Rb_1 R1_2 R2_3 k :
```

**Vectors from origin of {b} to {1}, {1} to {2}, and {2} to{3}**

```
rb_1_b := Vector([a1_b, b1_b, c1_b]) :
rb_2_1 := Rb_1 Vector([a2_1, b2_1, c2_1]) :
rb_3_2 := Rb_1 R1_2 Vector([a3_2, b3_2, c3_2]) :
```

**Vectors from {1} to center of mass of link 1, {2} to center of mass of link 2, and {3} to center of mass of link 3**

```
rb_cm1_1 := Rb_1 Vector([acm1_1, bcm1_1, ccm1_1]) :
rb_cm2_2 := Rb_1 R1_2 Vector([acm2_2, bcm2_2, ccm2_2]) :
rb_cm3_3 := Rb_1 R1_2 R2_3 Vector([acm3_3, bcm3_3, ccm3_3]) :
```

**---------------------------------------------------**

**Jacobian matrices**
**---------------------------------------------------**
**Link 1**

```
jcm1v_x := i :
jcm1v_y := j :
jcm1v_z := k :
jcm1v_wx := CrossProduct(i, rb_1_b + rb_cm1_1) :
jcm1v_wy := CrossProduct(j, rb_1_b + rb_cm1_1) :
jcm1v_wz := CrossProduct(k, rb_1_b + rb_cm1_1) :
jcm1v_q1 := CrossProduct(eb_1, rb_cm1_1) :
jcm1v_q2 := Vector([0, 0, 0]) :
jcm1v_q3 := Vector([0, 0, 0]) :
jcm1w_x := Vector([0, 0, 0]) :
jcm1w_y := Vector([0, 0, 0]) :
jcm1w_z := Vector([0, 0, 0]) :
jcm1w_wx := i :
jcm1w_wy := j :
jcm1w_wz := k :
jcm1w_q1 := eb_1 :
jcm1w_q2 := Vector([0, 0, 0]) :
jcm1w_q3 := Vector([0, 0, 0]) :
Jcm1 := Matrix([[jcm1v_x, jcm1v_y, jcm1v_z, jcm1v_wx, jcm1v_wy, jcm1v_wz, jcm1v_q1, jcm1v_q2, jcm1v_q3], [jcm1w_x, jcm1w_y, jcm1w_z,
    jcm1w_wx, jcm1w_wy, jcm1w_wz, jcm1w_q1, jcm1w_q2, jcm1w_q3]]) :

Jcm1_T := Transpose(Jcm1) :
```

**Link 2**

```
jcm2v_x := i :
jcm2v_y := j :
jcm2v_z := k :
jcm2v_wx := CrossProduct(i, rb_1_b + rb_2_1 + rb_cm2_2) :
jcm2v_wy := CrossProduct(j, rb_1_b + rb_2_1 + rb_cm2_2) :
jcm2v_wz := CrossProduct(k, rb_1_b + rb_2_1 + rb_cm2_2) :
jcm2v_q1 := CrossProduct(eb_1, rb_2_1 + rb_cm2_2) :
jcm2v_q2 := CrossProduct(eb_2, rb_cm2_2) :
jcm2v_q3 := Vector([0, 0, 0]) :
jcm2w_x := Vector([0, 0, 0]) :
jcm2w_y := Vector([0, 0, 0]) :
jcm2w_z := Vector([0, 0, 0]) :
jcm2w_wx := i :
jcm2w_wy := j :
jcm2w_wz := k :
jcm2w_q1 := eb_1 :
jcm2w_q2 := eb_2 :
jcm2w_q3 := Vector([0, 0, 0]) :
Jcm2 := Matrix([[jcm2v_x, jcm2v_y, jcm2v_z, jcm2v_wx, jcm2v_wy, jcm2v_wz, jcm2v_q1, jcm2v_q2, jcm2v_q3], [jcm2w_x, jcm2w_y, jcm2w_z,
    jcm2w_wx, jcm2w_wy, jcm2w_wz, jcm2w_q1, jcm2w_q2, jcm2w_q3]]) :

Jcm2_T := Transpose(Jcm2) :
```

**Link 3**

```
jcm3v_x := i :
jcm3v_y := j :
jcm3v_z := k :
jcm3v_wx := CrossProduct(i, rb_1_b + rb_2_1 + rb_3_2 + rb_cm3_3) :
jcm3v_wy := CrossProduct(j, rb_1_b + rb_2_1 + rb_3_2 + rb_cm3_3) :
jcm3v_wz := CrossProduct(k, rb_1_b + rb_2_1 + rb_3_2 + rb_cm3_3) :
jcm3v_q1 := CrossProduct(eb_1, rb_2_1 + rb_3_2 + rb_cm3_3) :
jcm3v_q2 := CrossProduct(eb_2, rb_3_2 + rb_cm3_3) :
```

```
jcm3v_q3 := CrossProduct(eb_3, rb_cm3_3) :
jcm3w_x := Vector([0, 0, 0]) :
jcm3w_y := Vector([0, 0, 0]) :
jcm3w_z := Vector([0, 0, 0]) :
jcm3w_wx := i :
jcm3w_wy := j :
jcm3w_wz := k :
jcm3w_q1 := eb_1 :
jcm3w_q2 := eb_2 :
jcm3w_q3 := eb_3 :
Jcm3 := Matrix([[jcm3v_x, jcm3v_y, jcm3v_z, jcm3v_wx, jcm3v_wy, jcm3v_wz, jcm3v_q1, jcm3v_q2, jcm3v_q3], [jcm3w_x, jcm3w_y, jcm3w_z,
    jcm3w_wx, jcm3w_wy, jcm3w_wz, jcm3w_q1, jcm3w_q2, jcm3w_q3]]) :

Jcm3_T := Transpose(Jcm3) :
```
==========================================================================

**VARIABLES AND MATRICES RELATED TO KINETIC ENERGY OF ROV**

==========================================================================

**Mass-inertia matrix for ROV**

```
Ig := Matrix([[i_rovx, 0, 0], [0, i_rovy, 0], [0, 0, i_rovz]]) :

B_rov := Matrix([[m_rov·IdentityMatrix(3), ZeroMatrix(3, 3), ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), Ig, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3),
    ZeroMatrix(3, 3), ZeroMatrix(3, 3)]]) :
```

**Mass-inertia matrix for link 1**

```
I1 := Matrix([[gamma14·i_1x, 0, 0], [0, gamma15·i_1y, 0], [0, 0, gamma16·i_1z]]) :
Gamma1 := Matrix([[gamma11, 0, 0], [0, gamma12, 0], [0, 0, gamma13]]) :
Ib_1 := Rb_1 I1 R1_b :
M_1 := Matrix([[m_1·Gamma1, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), Ib_1]]) :
B_1 := Jcm1_T M_1 Jcm1 :
```

**Mass-inertia matrix for link 2**

```
I2 := Matrix([[gamma24·i_2x, 0, 0], [0, gamma25·i_2y, 0], [0, 0, gamma26·i_2z]]) :
Gamma2 := Matrix([[gamma21, 0, 0], [0, gamma22, 0], [0, 0, gamma23]]) :
Ib_2 := Rb_2 I2 R2_b :
M_2 := Matrix([[m_2·Gamma2, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), Ib_2]]) :
B_2 := Jcm2_T M_2 Jcm2 :
```

**Mass-inertia matrix for link 3**

```
I3 := Matrix([[gamma34·i_3x, 0, 0], [0, gamma35·i_3y, 0], [0, 0, gamma36·i_3z]]) :
Gamma3 := Matrix([[gamma31, 0, 0], [0, gamma32, 0], [0, 0, gamma33]]) :
Ib_3 := Rb_3 I3 R3_b :
M_3 := Matrix([[m_3·Gamma3, ZeroMatrix(3, 3)], [ZeroMatrix(3, 3), Ib_3]]) :
B_3 := Jcm3_T M_3 Jcm3 :
```

**System mass-inertia matrix**

```
B := B_rov + B_1 + B_2 + B_3 :
```

**#Differentiate system mass-inertia matrix wrt generalized coordinates**

```
#dB_dX  := map(diff, B, X) = 0
#dB_dY:= map(diff, B, Y)  = 0
#dB_dZ:= map(diff, B, Z)  = 0
#dB_dphi:= map(diff, B, phi)  = 0 :
#dB_dtheta  := map(diff, B, theta)  = 0 :
#dB_dpsi  := map(diff, B, psi)  = 0 :
dB_dq1 := simplify(map(diff, B, q1)) :
dB_dq2 := simplify(map(diff, B, q2)) :
dB_dq3 := simplify(map(diff, B, q3)) :


#Differentiate each element in alpha wrt genearized coordinates  vector (produce a column
    vector)
q := Vector([phi, theta, psi]) :
dalphaij_dq := Matrix(3, 81) :
indexer := Matrix([[1, 2, 3, 4, 5, 6, 7, 8, 9], [10, 11, 12, 13, 14, 15, 16, 17, 18], [19, 20, 21, 22, 23, 24, 25, 26, 27], [28, 29, 30, 31, 32, 33, 34, 35, 36],
    [37, 38, 39, 40, 41, 42, 43, 44, 45], [46, 47, 48, 49, 50, 51, 52, 53, 54], [55, 56, 57, 58, 59, 60, 61, 62, 63], [64, 65, 66, 67, 68, 69, 70, 71, 72], [73,
    74, 75, 76, 77, 78, 79, 80, 81]]) :

for i from 1 to 9 do for j from 1 to 9 do for k  from 1 to 3 do  dalphaij_dq[k, indexer[i, j]] := diff(alpha[i, j], q[k]) :end do end do end do
```

**#Differentiate alpha wrt each generalized coordinate**

**The alpha matrix is only dependet on the eulerangles phi, theta and psi (the rest is zero)**

```
dalpha_dphi := map(diff, alpha, phi) :
dalpha_dtheta := map(diff, alpha, theta) :

dalpha_dpsi := map(diff, alpha, psi) :
```
==========================================================================

**CODE EXPORT**

**========================================================================**

**Export the mass-inertia matrix**

```
for i from 1 to 9 do for j from i to 9 do  F := makeproc(B[i, j]); C(F, resultname = cat("b", i, j), output = "input_file.txt", optimize) :end do end do:
```

**Export the mass-inertia matrix differentiated wrt joint angles**

```
for i from 1 to 9 do for j from i to 9 do  F := makeproc(dB_dq1[i, j]); C(F, resultname = cat("dbdq1_", i, j), output = "input_file.txt", optimize) :end do end
    do:

for i from 1 to 9 do for j from i to 9 do  F := makeproc(dB_dq2[i, j]); C(F, resultname = cat("dbdq2_", i, j), output = "input_file.txt", optimize) :end do end
    do:

for i from 1 to 9 do for j from i to 9 do  F := makeproc(dB_dq3[i, j]); C(F, resultname = cat("dbdq3_", i, j), output = "input_file.txt", optimize) :end do end
    do:
```

**Export the tranformation matrix elementwhisely differentiated wrt. the generalized coordinate vector**

```
for i from 1 to 3 do for j from 1 to 81 do  F := makeproc(dalphaij_dq[i, j]); C(F, resultname = cat("dalphaij_dq_", i, j), output = "input_file.txt", optimize) :
    end do end do:
```

**Export the differentiated transformation matrix wrt to each generalized coordinate (only dependent on phi, theta, psi)**

```
for i from 1 to 9 do for j from 1 to 9 do  F := makeproc(dalpha_dphi[i, j]); C(F, resultname = cat("dalpha_dphi_", i, j), output = "input_file.txt", optimize) :
    end do end do:
for i from 1 to 9 do for j from 1 to 9 do  F := makeproc(dalpha_dtheta[i, j]); C(F, resultname = cat("dalpha_dtheta_", i, j), output = "input_file.txt",
    optimize) :end do end do:
for i from 1 to 9 do for j from 1 to 9 do  F := makeproc(dalpha_dpsi[i, j]); C(F, resultname = cat("dalpha_dpsi_", i, j), output = "input_file.txt", optimize) :
    end do end do:
```

```
/*------------------------------------------------------------
    IC-field code - ROV and Manipulator
--------------------------------------------------------------
By Boerge Rokseth 02.06.2014
------------------------------------------------------------*/

parameters
  string dll_name = 'RovAndManipulatorCalcAM.dll';
  // manipulator added mass coeffs
  real gamma11=1.4,gamma12=1.4,gamma13=1.4,gamma14=1.4m,gamma15=1.4,gamma16=1.4;
  real gamma21=1.4,gamma22=1.4,gamma23=1.4,gamma24=1.4,gamma25=1.4,gamma26=1.4;
  real gamma31=1.4,gamma32 = 1.4,gamma33=1.4,gamma34=1.4,gamma35=1.4,gamma36=1.4;
variables
  // transformation matrix from bodyfixed angular velocity to euler rates
  real T[3,3];
  // transformation from quasi-coordinates to generalized rates
  real beta[9,9];
  real beta_T[9,9];
  // euler angles
  real EulerAng[3];
  real phi, theta, psi;
  // rotation matrix from rov body-fixed to inertial representation
  real R0_b[3,3];
  // external functions argument
  real x[24];
  // quasi-coordinates
  real omega[9];
  // output flow vector
  real vI[9];
  // input effort vector
  real eI[9];
  // output flow vector
  real eC[9];
  // system mass-inertia matrix and its derivatives wrt generalized coordinates
  real B[9,9];
  real dBdX[9,9];
  real dBdY[9,9];
  real dBdZ[9,9];
  real dBdphi[9,9];
  real dBdtheta[9,9];
  real dBdpsi[9,9];
  real dBdq1[9,9];
  real dBdq2[9,9];
  real dBdq3[9,9];
  // derivatives of transformation matrix (quasi to generalized rates) wrt.
      generalized coordinates
  real dalpha_ij[243];
  real dalpha11_dq[9];
  real dalpha12_dq[9];
  real dalpha13_dq[9];
  real dalpha14_dq[9];
  real dalpha15_dq[9];
  real dalpha16_dq[9];
  real dalpha17_dq[9];
  real dalpha18_dq[9];
  real dalpha19_dq[9];

  real dalpha21_dq[9];
  real dalpha22_dq[9];
  real dalpha23_dq[9];
  real dalpha24_dq[9];
  real dalpha25_dq[9];
  real dalpha26_dq[9];
  real dalpha27_dq[9];
  real dalpha28_dq[9];
  real dalpha29_dq[9];

  real dalpha31_dq[9];
  real dalpha32_dq[9];
  real dalpha33_dq[9];
  real dalpha34_dq[9];
real dalpha35_dq[9];
real dalpha36_dq[9];
real dalpha37_dq[9];
real dalpha38_dq[9];
real dalpha39_dq[9];

real dalpha41_dq[9];
real dalpha42_dq[9];
real dalpha43_dq[9];
real dalpha44_dq[9];
real dalpha45_dq[9];
real dalpha46_dq[9];
real dalpha47_dq[9];
real dalpha48_dq[9];
real dalpha49_dq[9];

real dalpha51_dq[9];
real dalpha52_dq[9];
real dalpha53_dq[9];
real dalpha54_dq[9];
real dalpha55_dq[9];
real dalpha56_dq[9];
real dalpha57_dq[9];
real dalpha58_dq[9];
real dalpha59_dq[9];

real dalpha61_dq[9];
real dalpha62_dq[9];
real dalpha63_dq[9];
real dalpha64_dq[9];
real dalpha65_dq[9];
real dalpha66_dq[9];
real dalpha67_dq[9];
real dalpha68_dq[9];
real dalpha69_dq[9];

real dalpha71_dq[9];
real dalpha72_dq[9];
real dalpha73_dq[9];
real dalpha74_dq[9];
real dalpha75_dq[9];
real dalpha76_dq[9];
real dalpha77_dq[9];
real dalpha78_dq[9];
real dalpha79_dq[9];

real dalpha81_dq[9];
real dalpha82_dq[9];
real dalpha83_dq[9];
real dalpha84_dq[9];
real dalpha85_dq[9];
real dalpha86_dq[9];
real dalpha87_dq[9];
real dalpha88_dq[9];
real dalpha89_dq[9];

real dalpha91_dq[9];
real dalpha92_dq[9];
real dalpha93_dq[9];
real dalpha94_dq[9];
real dalpha95_dq[9];
real dalpha96_dq[9];
real dalpha97_dq[9];
real dalpha98_dq[9];
real dalpha99_dq[9];
//gamma matrix
real gamma1[9,9];
real gamma2[9,9];
real gamma[9,9];
// outputs of external functions
real out1_81[81];
```

```
  real out2_81[81];
  real out3_81[81];
  real dalpha_dX[9,9];
  real dalpha_dY[9,9];
  real dalpha_dZ[9,9];
  real dalpha_dphi[9,9];
  real dalpha_dtheta[9,9];
  real dalpha_dpsi[9,9];
  real dalpha_dq1[9,9];
  real dalpha_dq2[9,9];
  real dalpha_dq3[9,9];
  //output arrays for ext.funcs
  real out45[45];
  real db_out1[45];
  real db_out2[45];
  real db_out3[45];
  //coriolis and centrifugal matrix
  real Cq[9,9];
  //for manipulator controller
  real global B_man[3,3];
  real global C_man[3,3];
  //Vehicle added mass and potential damping
  real B_A[9,9];
  real C_A[9,9];
  real A11[3,3],A12[3,3],A21[3,3],A22[3,3];
  real Xu,Yv,Zw,Kp,Mq,Nr;
equations
  //Transformations between generalized rates and quasi-coordinates
  T = [1, sin(theta) * sin(phi) / cos(theta), sin(theta) / cos(theta) * cos(phi);
        0, cos(phi),                          -sin(phi);
        0, sin(phi) / cos(theta),             0.1e1 / cos(theta) * cos(phi)];
  //Calculate euler angles (pIw.f is the input flow wb_b/0)
  EulerAng = int(T*pIw.f);
  phi = EulerAng[1]; theta = EulerAng[2]; psi = EulerAng[3];
  //Rotation matrix inertial-body fixed
  R0_b = [ cos(psi)*cos(theta)  ,  -sin(psi)*cos(phi) + cos(psi)*sin(theta)*sin(
      phi) , sin(psi)*sin(phi) + cos(psi)*cos(phi)*sin(theta)  ;
        sin(psi)*cos(theta)  ,  cos(psi)*cos(phi) + sin(phi)*sin(theta)*sin(psi)
             ,  -cos(psi)*sin(phi) + sin(psi)*cos(phi)*sin(theta)  ;
          -sin(theta)        ,  cos(theta)*sin(phi)                 ,  cos(theta)*
              cos(phi)               ];
  //Transfrom between quasi-coordinates and generalized rates
  beta[1:3,1:3] = R0_b;
  beta[4:9,1:3] = 0;
  beta[1:3,4:9] = 0;
  beta[4:6,4:6] = T;
  beta[7:9,1:6] = 0;
  beta[7:9,7:9] = eye(3);
  beta[3:6,7:9] = 0;
  beta_T = transpose(beta);
  //Putting flows and efforts in vectors
  vI = [pIv.f[1];pIv.f[2];pIv.f[3];pIw.f[1];pIw.f[2];pIw.f[3];pIq1.f;pIq2.f;pIq3.f
      ];
  eI = [pIv.e[1];pIv.e[2];pIv.e[3];pIw.e[1];pIw.e[2];pIw.e[3];pIq1.e;pIq2.e;pIq3.e
      ];
  omega = [pCv.f[1];pCv.f[2];pCv.f[3];pCw.f[1];pCw.f[2];pCw.f[3];pCq1.f;pCq2.f;
      pCq3.f];
  //dll-argument
  x = [phi;theta;psi;int(vI[7]);int(vI[8]);int(vI[9]);gamma11;gamma12;gamma13;
      gamma14;gamma15;gamma16;gamma21;gamma22;gamma23;gamma24;gamma25;gamma26;
      gamma31;gamma32;gamma33;gamma34;gamma35;gamma36];
  // elemetwise differentiation of alpha matrix
  dalpha_ij = dll(dll_name,'divAlpha_dq',x);
  dalpha11_dq = [0;0;0;dalpha_ij[1];dalpha_ij[82];dalpha_ij[163];0;0;0];
  dalpha12_dq = [0;0;0;dalpha_ij[2];dalpha_ij[83];dalpha_ij[164];0;0;0];
  dalpha13_dq = [0;0;0;dalpha_ij[3];dalpha_ij[84];dalpha_ij[165];0;0;0];
  dalpha14_dq = [0;0;0;dalpha_ij[4];dalpha_ij[85];dalpha_ij[166];0;0;0];
  dalpha15_dq = [0;0;0;dalpha_ij[5];dalpha_ij[86];dalpha_ij[167];0;0;0];
  dalpha16_dq = [0;0;0;dalpha_ij[6];dalpha_ij[87];dalpha_ij[168];0;0;0];
  dalpha17_dq = [0;0;0;dalpha_ij[7];dalpha_ij[88];dalpha_ij[169];0;0;0];
  dalpha18_dq = [0;0;0;dalpha_ij[8];dalpha_ij[89];dalpha_ij[170];0;0;0];
```

```
dalpha19_dq = [0;0;0;dalpha_ij[9];dalpha_ij[90];dalpha_ij[171];0;0;0];

dalpha21_dq = [0;0;0;dalpha_ij[10];dalpha_ij[91];dalpha_ij[172];0;0;0];
dalpha22_dq = [0;0;0;dalpha_ij[11];dalpha_ij[92];dalpha_ij[173];0;0;0];
dalpha23_dq = [0;0;0;dalpha_ij[12];dalpha_ij[93];dalpha_ij[174];0;0;0];
dalpha24_dq = [0;0;0;dalpha_ij[13];dalpha_ij[94];dalpha_ij[175];0;0;0];
dalpha25_dq = [0;0;0;dalpha_ij[14];dalpha_ij[95];dalpha_ij[176];0;0;0];
dalpha26_dq = [0;0;0;dalpha_ij[15];dalpha_ij[96];dalpha_ij[177];0;0;0];
dalpha27_dq = [0;0;0;dalpha_ij[16];dalpha_ij[97];dalpha_ij[178];0;0;0];
dalpha28_dq = [0;0;0;dalpha_ij[17];dalpha_ij[98];dalpha_ij[179];0;0;0];
dalpha29_dq = [0;0;0;dalpha_ij[18];dalpha_ij[99];dalpha_ij[180];0;0;0];

dalpha31_dq = [0;0;0;dalpha_ij[19];dalpha_ij[100];dalpha_ij[181];0;0;0];
dalpha32_dq = [0;0;0;dalpha_ij[20];dalpha_ij[101];dalpha_ij[182];0;0;0];
dalpha33_dq = [0;0;0;dalpha_ij[21];dalpha_ij[102];dalpha_ij[183];0;0;0];
dalpha34_dq = [0;0;0;dalpha_ij[22];dalpha_ij[103];dalpha_ij[184];0;0;0];
dalpha35_dq = [0;0;0;dalpha_ij[23];dalpha_ij[104];dalpha_ij[185];0;0;0];
dalpha36_dq = [0;0;0;dalpha_ij[24];dalpha_ij[105];dalpha_ij[186];0;0;0];
dalpha37_dq = [0;0;0;dalpha_ij[25];dalpha_ij[106];dalpha_ij[187];0;0;0];
dalpha38_dq = [0;0;0;dalpha_ij[26];dalpha_ij[107];dalpha_ij[188];0;0;0];
dalpha39_dq = [0;0;0;dalpha_ij[27];dalpha_ij[108];dalpha_ij[189];0;0;0];

dalpha41_dq = [0;0;0;dalpha_ij[28];dalpha_ij[109];dalpha_ij[190];0;0;0];
dalpha42_dq = [0;0;0;dalpha_ij[29];dalpha_ij[110];dalpha_ij[191];0;0;0];
dalpha43_dq = [0;0;0;dalpha_ij[30];dalpha_ij[111];dalpha_ij[192];0;0;0];
dalpha44_dq = [0;0;0;dalpha_ij[31];dalpha_ij[112];dalpha_ij[193];0;0;0];
dalpha45_dq = [0;0;0;dalpha_ij[32];dalpha_ij[113];dalpha_ij[194];0;0;0];
dalpha46_dq = [0;0;0;dalpha_ij[33];dalpha_ij[114];dalpha_ij[195];0;0;0];
dalpha47_dq = [0;0;0;dalpha_ij[34];dalpha_ij[115];dalpha_ij[196];0;0;0];
dalpha48_dq = [0;0;0;dalpha_ij[35];dalpha_ij[116];dalpha_ij[197];0;0;0];
dalpha49_dq = [0;0;0;dalpha_ij[36];dalpha_ij[117];dalpha_ij[198];0;0;0];

dalpha51_dq = [0;0;0;dalpha_ij[37];dalpha_ij[118];dalpha_ij[199];0;0;0];
dalpha52_dq = [0;0;0;dalpha_ij[38];dalpha_ij[119];dalpha_ij[200];0;0;0];
dalpha53_dq = [0;0;0;dalpha_ij[39];dalpha_ij[120];dalpha_ij[201];0;0;0];
dalpha54_dq = [0;0;0;dalpha_ij[40];dalpha_ij[121];dalpha_ij[202];0;0;0];
dalpha55_dq = [0;0;0;dalpha_ij[41];dalpha_ij[122];dalpha_ij[203];0;0;0];
dalpha56_dq = [0;0;0;dalpha_ij[42];dalpha_ij[123];dalpha_ij[204];0;0;0];
dalpha57_dq = [0;0;0;dalpha_ij[43];dalpha_ij[124];dalpha_ij[205];0;0;0];
dalpha58_dq = [0;0;0;dalpha_ij[44];dalpha_ij[125];dalpha_ij[206];0;0;0];
dalpha59_dq = [0;0;0;dalpha_ij[45];dalpha_ij[126];dalpha_ij[207];0;0;0];

dalpha61_dq = [0;0;0;dalpha_ij[46];dalpha_ij[127];dalpha_ij[208];0;0;0];
dalpha62_dq = [0;0;0;dalpha_ij[47];dalpha_ij[128];dalpha_ij[209];0;0;0];
dalpha63_dq = [0;0;0;dalpha_ij[48];dalpha_ij[129];dalpha_ij[210];0;0;0];
dalpha64_dq = [0;0;0;dalpha_ij[49];dalpha_ij[130];dalpha_ij[211];0;0;0];
dalpha65_dq = [0;0;0;dalpha_ij[50];dalpha_ij[131];dalpha_ij[212];0;0;0];
dalpha66_dq = [0;0;0;dalpha_ij[51];dalpha_ij[132];dalpha_ij[213];0;0;0];
dalpha67_dq = [0;0;0;dalpha_ij[52];dalpha_ij[133];dalpha_ij[214];0;0;0];
dalpha68_dq = [0;0;0;dalpha_ij[53];dalpha_ij[134];dalpha_ij[215];0;0;0];
dalpha69_dq = [0;0;0;dalpha_ij[54];dalpha_ij[135];dalpha_ij[216];0;0;0];

dalpha71_dq = [0;0;0;dalpha_ij[55];dalpha_ij[136];dalpha_ij[217];0;0;0];
dalpha72_dq = [0;0;0;dalpha_ij[56];dalpha_ij[137];dalpha_ij[218];0;0;0];
dalpha73_dq = [0;0;0;dalpha_ij[57];dalpha_ij[138];dalpha_ij[219];0;0;0];
dalpha74_dq = [0;0;0;dalpha_ij[58];dalpha_ij[139];dalpha_ij[220];0;0;0];
dalpha75_dq = [0;0;0;dalpha_ij[59];dalpha_ij[140];dalpha_ij[221];0;0;0];
dalpha76_dq = [0;0;0;dalpha_ij[60];dalpha_ij[141];dalpha_ij[222];0;0;0];
dalpha77_dq = [0;0;0;dalpha_ij[61];dalpha_ij[142];dalpha_ij[223];0;0;0];
dalpha78_dq = [0;0;0;dalpha_ij[62];dalpha_ij[143];dalpha_ij[224];0;0;0];
dalpha79_dq = [0;0;0;dalpha_ij[63];dalpha_ij[144];dalpha_ij[225];0;0;0];

dalpha81_dq = [0;0;0;dalpha_ij[64];dalpha_ij[145];dalpha_ij[226];0;0;0];
dalpha82_dq = [0;0;0;dalpha_ij[65];dalpha_ij[146];dalpha_ij[227];0;0;0];
dalpha83_dq = [0;0;0;dalpha_ij[66];dalpha_ij[147];dalpha_ij[228];0;0;0];
dalpha84_dq = [0;0;0;dalpha_ij[67];dalpha_ij[148];dalpha_ij[229];0;0;0];
dalpha85_dq = [0;0;0;dalpha_ij[68];dalpha_ij[149];dalpha_ij[230];0;0;0];
dalpha86_dq = [0;0;0;dalpha_ij[69];dalpha_ij[150];dalpha_ij[231];0;0;0];
dalpha87_dq = [0;0;0;dalpha_ij[70];dalpha_ij[151];dalpha_ij[232];0;0;0];
dalpha88_dq = [0;0;0;dalpha_ij[71];dalpha_ij[152];dalpha_ij[233];0;0;0];
dalpha89_dq = [0;0;0;dalpha_ij[72];dalpha_ij[153];dalpha_ij[234];0;0;0];
```

```
dalpha91_dq = [0;0;0;dalpha_ij[73];dalpha_ij[154];dalpha_ij[235];0;0;0];
dalpha92_dq = [0;0;0;dalpha_ij[74];dalpha_ij[155];dalpha_ij[236];0;0;0];
dalpha93_dq = [0;0;0;dalpha_ij[75];dalpha_ij[156];dalpha_ij[237];0;0;0];
dalpha94_dq = [0;0;0;dalpha_ij[76];dalpha_ij[157];dalpha_ij[238];0;0;0];
dalpha95_dq = [0;0;0;dalpha_ij[77];dalpha_ij[158];dalpha_ij[239];0;0;0];
dalpha96_dq = [0;0;0;dalpha_ij[78];dalpha_ij[159];dalpha_ij[240];0;0;0];
dalpha97_dq = [0;0;0;dalpha_ij[79];dalpha_ij[160];dalpha_ij[241];0;0;0];
dalpha98_dq = [0;0;0;dalpha_ij[80];dalpha_ij[161];dalpha_ij[242];0;0;0];
dalpha99_dq = [0;0;0;dalpha_ij[81];dalpha_ij[162];dalpha_ij[243];0;0;0];
// compose first term in gamma matrix
gamma1[1,1:9] = [transpose(omega)*beta_T*dalpha11_dq,  transpose(omega)*beta_T*
    dalpha12_dq,  transpose(omega)*beta_T*dalpha13_dq,
          transpose(omega)*beta_T*dalpha14_dq,  transpose(omega)*beta_T*
              dalpha15_dq,  transpose(omega)*beta_T*dalpha16_dq,
          transpose(omega)*beta_T*dalpha17_dq,  transpose(omega)*beta_T*
              dalpha18_dq,  transpose(omega)*beta_T*dalpha19_dq];

gamma1[2,1:9] = [transpose(omega)*beta_T*dalpha21_dq,  transpose(omega)*beta_T*
    dalpha22_dq,  transpose(omega)*beta_T*dalpha23_dq,
          transpose(omega)*beta_T*dalpha24_dq,  transpose(omega)*beta_T*
              dalpha25_dq,  transpose(omega)*beta_T*dalpha26_dq,
          transpose(omega)*beta_T*dalpha27_dq,  transpose(omega)*beta_T*
              dalpha28_dq,  transpose(omega)*beta_T*dalpha29_dq];

gamma1[3,1:9] = [transpose(omega)*beta_T*dalpha31_dq,  transpose(omega)*beta_T*
    dalpha32_dq,  transpose(omega)*beta_T*dalpha33_dq,
          transpose(omega)*beta_T*dalpha34_dq,  transpose(omega)*beta_T*
              dalpha35_dq,  transpose(omega)*beta_T*dalpha36_dq,
          transpose(omega)*beta_T*dalpha37_dq,  transpose(omega)*beta_T*
              dalpha38_dq,  transpose(omega)*beta_T*dalpha39_dq];


gamma1[4,1:9] = [transpose(omega)*beta_T*dalpha41_dq,  transpose(omega)*beta_T*
    dalpha42_dq,  transpose(omega)*beta_T*dalpha43_dq,
          transpose(omega)*beta_T*dalpha44_dq,  transpose(omega)*beta_T*
              dalpha45_dq,  transpose(omega)*beta_T*dalpha46_dq,
          transpose(omega)*beta_T*dalpha47_dq,  transpose(omega)*beta_T*
              dalpha48_dq,  transpose(omega)*beta_T*dalpha49_dq];

gamma1[5,1:9] = [transpose(omega)*beta_T*dalpha51_dq,  transpose(omega)*beta_T*
    dalpha52_dq,  transpose(omega)*beta_T*dalpha53_dq,
          transpose(omega)*beta_T*dalpha54_dq,  transpose(omega)*beta_T*
              dalpha55_dq,  transpose(omega)*beta_T*dalpha56_dq,
          transpose(omega)*beta_T*dalpha57_dq,  transpose(omega)*beta_T*
              dalpha58_dq,  transpose(omega)*beta_T*dalpha59_dq];

gamma1[6,1:9] = [transpose(omega)*beta_T*dalpha61_dq,  transpose(omega)*beta_T*
    dalpha62_dq,  transpose(omega)*beta_T*dalpha63_dq,
          transpose(omega)*beta_T*dalpha64_dq,  transpose(omega)*beta_T*
              dalpha65_dq,  transpose(omega)*beta_T*dalpha66_dq,
          transpose(omega)*beta_T*dalpha67_dq,  transpose(omega)*beta_T*
              dalpha68_dq,  transpose(omega)*beta_T*dalpha69_dq];

gamma1[7,1:9] = [transpose(omega)*beta_T*dalpha71_dq,  transpose(omega)*beta_T*
    dalpha72_dq,  transpose(omega)*beta_T*dalpha73_dq,
          transpose(omega)*beta_T*dalpha74_dq,  transpose(omega)*beta_T*
              dalpha75_dq,  transpose(omega)*beta_T*dalpha76_dq,
          transpose(omega)*beta_T*dalpha77_dq,  transpose(omega)*beta_T*
              dalpha78_dq,  transpose(omega)*beta_T*dalpha79_dq];

gamma1[8,1:9] = [transpose(omega)*beta_T*dalpha81_dq,  transpose(omega)*beta_T*
    dalpha82_dq,  transpose(omega)*beta_T*dalpha83_dq,
          transpose(omega)*beta_T*dalpha84_dq,  transpose(omega)*beta_T*
              dalpha85_dq,  transpose(omega)*beta_T*dalpha86_dq,
          transpose(omega)*beta_T*dalpha87_dq,  transpose(omega)*beta_T*
              dalpha88_dq,  transpose(omega)*beta_T*dalpha89_dq];

gamma1[9,1:9] = [transpose(omega)*beta_T*dalpha91_dq,  transpose(omega)*beta_T*
    dalpha92_dq,  transpose(omega)*beta_T*dalpha93_dq,
                transpose(omega)*beta_T*dalpha94_dq,  transpose(omega)*beta_T*
                     dalpha95_dq,  transpose(omega)*beta_T*dalpha96_dq,
                transpose(omega)*beta_T*dalpha97_dq,  transpose(omega)*beta_T*
                     dalpha98_dq,  transpose(omega)*beta_T*dalpha99_dq];
//alphamatric differentiated wrt. generalized coordiantes
dalpha_dX[:,:] = 0;
dalpha_dY[:,:] = 0;
dalpha_dZ[:,:] = 0;
out1_81 = dll(dll_name,'divAlpha_dphi',x);
dalpha_dphi = [out1_81[1], out1_81[2], out1_81[3], out1_81[4], out1_81[5],
     out1_81[6], out1_81[7], out1_81[8], out1_81[9];
           out1_81[10],out1_81[11],out1_81[12],out1_81[13],out1_81[14],out1_81
                [15],out1_81[16],out1_81[17],out1_81[18];
           out1_81[19],out1_81[20],out1_81[21],out1_81[22],out1_81[23],out1_81
                [24],out1_81[25],out1_81[26],out1_81[27];
           out1_81[28],out1_81[29],out1_81[30],out1_81[31],out1_81[32],out1_81
                [33],out1_81[34],out1_81[35],out1_81[36];
           out1_81[37],out1_81[38],out1_81[39],out1_81[40],out1_81[41],out1_81
                [42],out1_81[43],out1_81[44],out1_81[45];
           out1_81[46],out1_81[47],out1_81[48],out1_81[49],out1_81[50],out1_81
                [51],out1_81[52],out1_81[53],out1_81[54];
           out1_81[55],out1_81[56],out1_81[57],out1_81[58],out1_81[59],out1_81
                [60],out1_81[61],out1_81[62],out1_81[63];
           out1_81[64],out1_81[65],out1_81[66],out1_81[67],out1_81[68],out1_81
                [69],out1_81[70],out1_81[71],out1_81[72];
           out1_81[73],out1_81[74],out1_81[75],out1_81[76],out1_81[77],out1_81
                [78],out1_81[79],out1_81[80],out1_81[81]];

out2_81 = dll(dll_name,'divAlpha_dtheta',x);
dalpha_dtheta =  [out2_81[1], out2_81[2], out2_81[3], out2_81[4], out2_81[5],
     out2_81[6], out2_81[7], out2_81[8], out2_81[9];
              out2_81[10],out2_81[11],out2_81[12],out2_81[13],out2_81[14],out2_81
                   [15],out2_81[16],out2_81[17],out2_81[18];
              out2_81[19],out2_81[20],out2_81[21],out2_81[22],out2_81[23],out2_81
                   [24],out2_81[25],out2_81[26],out2_81[27];
             out2_81[28],out2_81[29],out2_81[30],out2_81[31],out2_81[32],out2_81
                  [33],out2_81[34],out2_81[35],out2_81[36];
             out2_81[37],out2_81[38],out2_81[39],out2_81[40],out2_81[41],out2_81
                  [42],out2_81[43],out2_81[44],out2_81[45];
             out2_81[46],out2_81[47],out2_81[48],out2_81[49],out2_81[50],out2_81
                  [51],out2_81[52],out2_81[53],out2_81[54];
             out2_81[55],out2_81[56],out2_81[57],out2_81[58],out2_81[59],out2_81
                  [60],out2_81[61],out2_81[62],out2_81[63];
             out2_81[64],out2_81[65],out2_81[66],out2_81[67],out2_81[68],out2_81
                  [69],out2_81[70],out2_81[71],out2_81[72];
             out2_81[73],out2_81[74],out2_81[75],out2_81[76],out2_81[77],out2_81
                  [78],out2_81[79],out2_81[80],out2_81[81]];
out3_81 = dll(dll_name,'divAlpha_dpsi',x);
dalpha_dpsi = [out3_81[1], out3_81[2], out3_81[3], out3_81[4], out3_81[5],
     out3_81[6], out3_81[7], out3_81[8], out3_81[9];
           out3_81[10],out3_81[11],out3_81[12],out3_81[13],out3_81[14],out3_81
                [15],out3_81[16],out3_81[17],out3_81[18];
           out3_81[19],out3_81[20],out3_81[21],out3_81[22],out3_81[23],out3_81
                [24],out3_81[25],out3_81[26],out3_81[27];
           out3_81[28],out3_81[29],out3_81[30],out3_81[31],out3_81[32],out3_81
                [33],out3_81[34],out3_81[35],out3_81[36];
           out3_81[37],out3_81[38],out3_81[39],out3_81[40],out3_81[41],out3_81
                [42],out3_81[43],out3_81[44],out3_81[45];
           out3_81[46],out3_81[47],out3_81[48],out3_81[49],out3_81[50],out3_81
                [51],out3_81[52],out3_81[53],out3_81[54];
           out3_81[55],out3_81[56],out3_81[57],out3_81[58],out3_81[59],out3_81
                [60],out3_81[61],out3_81[62],out3_81[63];
           out3_81[64],out3_81[65],out3_81[66],out3_81[67],out3_81[68],out3_81
                [69],out3_81[70],out3_81[71],out3_81[72];
           out3_81[73],out3_81[74],out3_81[75],out3_81[76],out3_81[77],out3_81
                [78],out3_81[79],out3_81[80],out3_81[81]];
dalpha_dq1[:,:] = 0;
dalpha_dq2[:,:] = 0;
dalpha_dq3[:,:] = 0;
//compose second term in gama matrix
gamma2[1,1:9] = transpose(omega)*beta_T*dalpha_dX;
```

```
gamma2[2,1:9] = transpose(omega)*beta_T*dalpha_dY;
gamma2[3,1:9] = transpose(omega)*beta_T*dalpha_dZ;
gamma2[4,1:9] = transpose(omega)*beta_T*dalpha_dphi;
gamma2[5,1:9] = transpose(omega)*beta_T*dalpha_dtheta;
gamma2[6,1:9] = transpose(omega)*beta_T*dalpha_dpsi;
gamma2[7,1:9] = transpose(omega)*beta_T*dalpha_dq1;
gamma2[8,1:9] = transpose(omega)*beta_T*dalpha_dq2;
gamma2[9,1:9] = transpose(omega)*beta_T*dalpha_dq3;
//gamma matrix
gamma = gamma1 - gamma2;
// System mass-inertia matrix and derivatives wrt. gen. coords.
out45 = dll(dll_name,'massMatrix',x);
B = [out45[1],out45[2], out45[3], out45[4], out45[5], out45[6], out45[7], out45
     [8], out45[9] ;
    out45[2],out45[10],out45[11],out45[12],out45[13],out45[14],out45[15],out45
         [16],out45[17];
    out45[3],out45[11],out45[18],out45[19],out45[20],out45[21],out45[22],out45
         [23],out45[24];
    out45[4],out45[12],out45[19],out45[25],out45[26],out45[27],out45[28],out45
         [29],out45[30];
    out45[5],out45[13],out45[20],out45[26],out45[31],out45[32],out45[33],out45
         [34],out45[35];
    out45[6],out45[14],out45[21],out45[27],out45[32],out45[36],out45[37],out45
         [38],out45[39];
    out45[7],out45[15],out45[22],out45[28],out45[33],out45[37],out45[40],out45
         [41],out45[42];
    out45[8],out45[16],out45[23],out45[29],out45[34],out45[38],out45[41],out45
         [43],out45[44];
    out45[9],out45[17],out45[24],out45[30],out45[35],out45[39],out45[42],out45
         [44],out45[45]];
dBdX[:,:] = 0;
dBdY[:,:] = 0;
dBdZ[:,:] = 0;
dBdphi[:,:] = 0;
dBdtheta[:,:] = 0;
dBdpsi[:,:] = 0;
db_out1 = dll(dll_name,'divMass_q1',x);
dBdq1 = [db_out1[1],db_out1[2], db_out1[3], db_out1[4], db_out1[5], db_out1[6],
     db_out1[7], db_out1[8], db_out1[9] ;
        db_out1[2],db_out1[10],db_out1[11],db_out1[12],db_out1[13],db_out1[14],
             db_out1[15],db_out1[16],db_out1[17];
      db_out1[3],db_out1[11],db_out1[18],db_out1[19],db_out1[20],db_out1[21],
           db_out1[22],db_out1[23],db_out1[24];
      db_out1[4],db_out1[12],db_out1[19],db_out1[25],db_out1[26],db_out1[27],
           db_out1[28],db_out1[29],db_out1[30];
        db_out1[5],db_out1[13],db_out1[20],db_out1[26],db_out1[31],db_out1[32],
             db_out1[33],db_out1[34],db_out1[35];
        db_out1[6],db_out1[14],db_out1[21],db_out1[27],db_out1[32],db_out1[36],
             db_out1[37],db_out1[38],db_out1[39];
        db_out1[7],db_out1[15],db_out1[22],db_out1[28],db_out1[33],db_out1[37],
             db_out1[40],db_out1[41],db_out1[42];
        db_out1[8],db_out1[16],db_out1[23],db_out1[29],db_out1[34],db_out1[38],
             db_out1[41],db_out1[43],db_out1[44];
        db_out1[9],db_out1[17],db_out1[24],db_out1[30],db_out1[35],db_out1[39],
             db_out1[42],db_out1[44],db_out1[45]];
db_out2 = dll(dll_name,'divMass_q2',x);
dBdq2 = [db_out2[1],db_out2[2], db_out2[3], db_out2[4], db_out2[5], db_out2[6],
     db_out2[7], db_out2[8], db_out2[9] ;
        db_out2[2],db_out2[10],db_out2[11],db_out2[12],db_out2[13],db_out2[14],
             db_out2[15],db_out2[16],db_out2[17];
        db_out2[3],db_out2[11],db_out2[18],db_out2[19],db_out2[20],db_out2[21],
             db_out2[22],db_out2[23],db_out2[24];
        db_out2[4],db_out2[12],db_out2[19],db_out2[25],db_out2[26],db_out2[27],
             db_out2[28],db_out2[29],db_out2[30];
        db_out2[5],db_out2[13],db_out2[20],db_out2[26],db_out2[31],db_out2[32],
             db_out2[33],db_out2[34],db_out2[35];
        db_out2[6],db_out2[14],db_out2[21],db_out2[27],db_out2[32],db_out2[36],
             db_out2[37],db_out2[38],db_out2[39];
        db_out2[7],db_out2[15],db_out2[22],db_out2[28],db_out2[33],db_out2[37],
             db_out2[40],db_out2[41],db_out2[42];
```

```
          db_out2[8],db_out2[16],db_out2[23],db_out2[29],db_out2[34],db_out2[38],
               db_out2[41],db_out2[43],db_out2[44];
          db_out2[9],db_out2[17],db_out2[24],db_out2[30],db_out2[35],db_out2[39],
               db_out2[42],db_out2[44],db_out2[45]];

  db_out3 = dll(dll_name,'divMass_q3',x);
  dBdq3 = [db_out3[1],db_out3[2], db_out3[3], db_out3[4], db_out3[5], db_out3[6],
       db_out3[7], db_out3[8], db_out3[9] ;
          db_out3[2],db_out3[10],db_out3[11],db_out3[12],db_out3[13],db_out3[14],
               db_out3[15],db_out3[16],db_out3[17];
          db_out3[3],db_out3[11],db_out3[18],db_out3[19],db_out3[20],db_out3[21],
               db_out3[22],db_out3[23],db_out3[24];
          db_out3[4],db_out3[12],db_out3[19],db_out3[25],db_out3[26],db_out3[27],
               db_out3[28],db_out3[29],db_out3[30];
          db_out3[5],db_out3[13],db_out3[20],db_out3[26],db_out3[31],db_out3[32],
               db_out3[33],db_out3[34],db_out3[35];
          db_out3[6],db_out3[14],db_out3[21],db_out3[27],db_out3[32],db_out3[36],
               db_out3[37],db_out3[38],db_out3[39];
          db_out3[7],db_out3[15],db_out3[22],db_out3[28],db_out3[33],db_out3[37],
               db_out3[40],db_out3[41],db_out3[42];
          db_out3[8],db_out3[16],db_out3[23],db_out3[29],db_out3[34],db_out3[38],
               db_out3[41],db_out3[43],db_out3[44];
          db_out3[9],db_out3[17],db_out3[24],db_out3[30],db_out3[35],db_out3[39],
               db_out3[42],db_out3[44],db_out3[45]];
  //Added mass for vehicle
  Xu = 0.4*B[1,1];
  Yv = 0.4*B[2,2];
  Zw = 0.4*B[3,3];
  Kp = 0.4*B[4,4];
  Mq = 0.4*B[5,5];
  Nr = 0.4*B[6,6];
  B_A = diag([Xu;Yv;Zw;Kp;Mq;Nr;0;0;0]);
  //Calculate output velocity
  vI = inverse(B + B_A)*int(eI);
  //Coriolis and centripetal forces
  Cq[1,1:9] = transpose(omega)*dBdX;
  Cq[2,1:9] = transpose(omega)*dBdY;
  Cq[3,1:9] = transpose(omega)*dBdZ;
  Cq[4,1:9] = transpose(omega)*dBdphi;
  Cq[5,1:9] = transpose(omega)*dBdtheta;
  Cq[6,1:9] = transpose(omega)*dBdpsi;
  Cq[7,1:9] = transpose(omega)*dBdq1;
  Cq[8,1:9] = transpose(omega)*dBdq2;
  Cq[9,1:9] = transpose(omega)*dBdq3;
  //Coriolis and centripetal forces due to added mass
  C_A = 0;
  A11 = B_A[1:3,1:3];
  A12 = B_A[1:3,4:6];
  A21 = B_A[4:6,1:3];
  A22 = B_A[4:6,4:6];
  C_A[1:3,4:6] = -skew(A11*pCv.f+A12*pCw.f);
  C_A[4:6,1:3] = -skew(A11*pCv.f+A12*pCw.f);
  C_A[4:6,4:6] = -skew(A21*pCv.f+A22*pCw.f);
  //For manipulator controller
  B_man = B[7:9,7:9];
  C_man = Cq[7:9,7:9];
  //arrange ic-field outputs
  eC = 0.5*beta_T*Cq*omega - beta_T*gamma*B*omega + C_A*omega;
  pCv.e[1] = eC[1];
  pCv.e[2] = eC[2];
  pCv.e[3] = eC[3];
  pCw.e[1] = eC[4];
  pCw.e[2] = eC[5];
  pCw.e[3] = eC[6];
  pCq1.e = eC[7];
  pCq2.e = eC[8];
  pCq3.e = eC[9];
//================================================================================
```