Thesis

Current-Based Impedance Control for Mobile Manipulators Without Force/Torque Sensors

MSc Robotics

Jelmer de Wolde

Delft University of Technology

TU Delft

Thesis

Current-Based Impedance Control for Mobile Manipulators Without Force/Torque Sensors

by

Jelmer de Wolde

Primary supervisor:	Dr. Javier Alonso Mora
Daily supervisor:	Luzia Knödler
Student number:	4705041
Year:	2024

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

Mobile manipulators, which combine a mobile platform with a robotic arm, are versatile robots that can be used for a variety of tasks like logistic pick-and-placing, manufacturing or assembly. Compliant control for mobile manipulators could improve the safety of the users sharing their workspace with these robots. The two general methods of compliant control, admittance control and impedance control, require force/torque sensors, which are often not available on low-cost or lightweight robots. This report presents an adaption of impedance control, including a strategy to compensate for joint friction, that can be used on current-controlled robots without the use of force/torque sensors. A calibration method is designed for the arm, that enables estimation of the actuator's current/torque ratios and frictions, used by the adapted impedance controller. Software is developed to use the controller on a combination of the Kinova GEN3 Lite arm and the Clearpath Dingo Omnidirectional base. Real-world experiments with the arm show that the calibration method is consistent and that the designed controller is compliant while also being able to track targets with five-millimeter precision when no interaction is present. Experiments with the complete mobile manipulator showcase two new modes, both suitable for interaction with a human. The first is a guidance mode where the user can control the robot using interaction with the arm. The second is a tracking mode where the mobile manipulator tracks a moving target while still being compliant.

Contents

List of Figures

List of Tables

1

Introduction

The world is currently experiencing the fourth industrial revolution. In the initial revolution, steam-powered machines were introduced, the second revolution added electricity, and the third revolution started automation using computers. The current revolution introduces cyber-physical systems that lead to so-called ”Smart Factories”. Collaborative robots, robots that work together with humans and share the same workspace (Figure 1.1), are a key factor of this fourth industrial revolution [2].

Mobile manipulators, which combine a mobile platform with a robotic arm, are an important type of collaborative robot. Due to their flexibility, mobile manipulators can easily move around to do tasks like logistic pick-and-placing, manufacturing or assembly [3]. However, also outside the industry robots are rising, since professional service robots which are used for tasks like cleaning or inspection are gaining more market share [4].

The increasing use of collaborative robots emphasizes the importance of the safety of these robots. Although many aspects contribute to the safety level of a robot, the control of the robot certainly has a great influence. To achieve safe human-robot interaction, mobile manipulators should be compliant [5], which means that the robot allows deviations from its target pose depending on the external forces applied due to interaction with humans or obstacles. This can be achieved without mechanical adaptions

Figure 1.1: Mobile manipulator sharing its workspace with humans. [1]

by only adapting the control of the robot. This thesis investigates the use of compliant control methods on mobile manipulators. To this end, different compliant control methods are introduced in Section 1.1 and existing research on compliant mobile manipulators using these methods are described in Section 1.2. The existing control methods discussed in those two sections all require one or multiple force/torque sensors. The goal of this thesis is to implement compliant control on a current-controlled mobile manipulator, without the use of force/torque sensors. The contributions of this thesis are described in detail in Section 1.3.

1.1. Compliant control

Control of robotics can be divided into motion control and force control [6]. Motion control aims to let a robot follow a desired trajectory. This works well when the robot can move freely but can cause problems when the robot interacts with its environment. By controlling the force of a robot, instead of the motion, compliance can be achieved. Force control can be used directly or indirectly.

With direct force control, a certain force on the end effector of the robot is desired, usually in a specific cartesian direction, for example, to perform a task like polishing. Direct force control can provide very specific compliance, but cannot guarantee fully compliant behavior [7] [8], and is thus not suited for the goal of whole-body compliance for mobile manipulators.

With indirect force control, one tries to control the relationship between force and motion. This relationship between force and motion is called mechanical impedance and can be written as a mass-spring-damper system reacting to a force. For a one-dimensional case, this is defined as:

$$m\ddot{x} + d\dot{x} + kx = f \tag{1.1}$$

where x is the position and f the force applied to the system. The parameters mass m, damping d and stiffness k define the impedance of the system. The higher the impedance of a system (usually due to large stiffness or damping), the more force is required to achieve a certain motion [9]. The goal of indirect force control is to let a robot behave as a mass-spring-damper system with parameters that lead to the desired compliance.

There are two ways to implement indirect force control on a robot. The first method is called admittance control and can be used on any position or velocity-controlled robot, but requires sensing of the external forces applied to the robot. Generally, these forces are sensed using a force/torque sensor located at the wrist of the robot arm, resulting in only the end-effector being compliant. The force feedback is used to determine what motion the robot should make to match the desired compliance, whereafter the position or velocity controller can perform this motion. For admittance control, a low stiffness can lead to instability, since noise in the external force measurement can already lead to a corresponding motion [10]. Furthermore, the fact that admittance control is only compliant with measured external forces is an important limitation in achieving whole-body compliance.

The second method is called impedance control and can be used on a torque-controlled robot, of which an accurate model is available. Impedance control uses the model of the robot combined with joint position and velocity feedback to constantly estimate the joint torques due to gravity, friction and the motion the robot is currently performing. It uses torque control to compensate for these torques and introduce additional torques that lead to the desired compliance. For impedance control, defining a high stiffness can lead to instability, since noise in the position measurement can already lead to a corresponding force [11]. Moreover, inaccuracies in the dynamic model lead to poor position tracking in free motion. Finally, torque control is generally a closed-loop control, requiring torque sensors in the joints to provide feedback to the controller.

Both controllers can be used passivity-based or as feedback-linearization. With the passivity-based method, the stiffness and damping can be shaped but the natural inertia of the robot is preserved. With feedback linearization, also the resulting inertia of the robot can be shaped, but it always requires force feedback to do this [12]. A comparative study between a linearization-based controller and two passivity-based controllers was presented in [13], where the passivity-based methods showed high robustness against model uncertainties.

Mobile manipulators require control for both the arm and the driving base. Therefore, mobile manipulators can use whole-body admittance control, whole-body impedance control or a combination of both. Existing research on compliant mobile manipulators for all of these three options is presented in the next section.

1.2. Existing compliant mobile manipulators

Both admittance and impedance control have been used for compliant mobile manipulators. The classical use of impedance control on a mobile manipulator requires both the arm and the base to be torque-controlled, a setup that is not very common. A robot that does contain this setup is the DLR Lightweight Rover Unit (Figure 1.2), a mobile manipulator designed for planetary exploration, equipped with four individually steerable wheels that provide torque control and a torque-controllable arm [17]. The individually controllable wheels introduce an additional redundancy and the research focused on solving this by optimizing for the use of power or force, which is important for a planetary rover to be efficient.

Since many mobile manipulators offer no torque control for the base, a combination of impedance control for the arm and admittance control for the base is employed by several robots. This combination is researched extensively on the robot called Rollin'Justin (Figure 1.3). This humanoid on wheels, has a velocity-controlled base and torque-controlled arms, hands and torso with a total of 51 actuated degrees of freedom, resulting in a highly redundant robot [18]. The research focussed on implementing the avoidance of collision, singularity and mechanical limits in a prioritized order, using the high degree of freedom the robot contains. More experimental validation was presented in [19] and asymptotic stability was proved while the performance was increased in [20]. The monograph *Whole-Body Impedance Control of Wheeled Humanoid Robots* [21] by Dietrich bundles the research of implementing impedance control on Rollin'Justin. Research of the combination of impedance and admittance control for other robots was presented in [22] for a Clearpath Husky base with Franka Panda arm and in [23] [24] for a manipulator mobile manipulator called MOCA. Both works were focused on methods to combine different tasks of the robots.

Admittance control can be used on mobile manipulators which are completely velocity-controlled, requiring measurement of the external force to be compliant with it. This results in the research of different methods to sense the external forces applied to the robot. To enable whole-body compliance, a mobile manipulator called TOMM was provided with artificial skin (Figure 1.4) on its links that can measure the location, direction and magnitude of tactile interactions [25]. Compliance was implemented with a propagation strategy, meaning when a link exposed to the external force cannot move, for example, due to joint limits, the corresponding movement should be performed by its parent link or eventually the base. A later publication showed improvements, where the skin data was first merged per link to reduce computations [26]. Other research on whole-body admittance control was focused on the use of a disturbance observer, to estimate external forces based on disturbances of the robot compared to the modeled behavior [27] and the combination of admittance control with model predictive control [28].

To summarize, the existing research can be divided into mobile manipulators using whole-body impedance control, using whole-body admittance control, or using impedance control for the arm and admittance control for the base. There are few examples of mobile manipulators using whole-body impedance control since most mobile bases are not torque-controlled. For the mobile manipulators using whole-body admittance control, most research is focused on alternatives to force/torque sensors, to enable compliance on robots that lack this sensor and to enable sensing for other parts of the robot than

Figure 1.2: Lightweight Rover Unit. [14]

Figure 1.3: Rollin'Justin. [15]

Figure 1.4: TOMM. [16]

only the end-effector. Research that combines impedance and admittance control is mainly focused on methods that handle the redundancy of the robots and the combination of multiple tasks with different priorities.

1.3. Contribution

The goal of this thesis is to implement a whole-body compliant controller on the Clearpath Dingo Omnidirectional base with the Kinova GEN3 Lite arm, as shown in Figure 1.5. Both the base and the arm can be controlled by position or velocity. For the arm, direct control of the actuator currents is also possible. No force or torque sensors are available for the arm or the base. Therefore, the use of admittance control is not possible without the addition of at least one force/torque sensor.

The classical use of impedance control requires torque sensors in the joints, to provide feedback for closed-loop torque control. Lightweight or cheaper robots can lack these sensors, which is also the case for the Kinova arm and Dingo base. Although the robot has no torque sensors, the Kinova arm provides an estimation of the joint torques based on the actuator currents. This led to the goal of this thesis, to implement impedance control without the conventional reliance on torque sensors. Instead, the approach involves leveraging the direct correlation [29] between current and generated torque. An accurate determination of this relation between torque and current allows impedance control via current control, without the use of torque sensors.

The proposed method cannot directly be transferred to the base since the software provided by Clearpath does not support current control. However, the motor drivers used in the Dingo are theoretically able to also be used in the current control mode. Therefore, to apply the current-based impedance control method on the base, the motor driver software has to be adapted. This thesis investigates the adaptations required to implement current control on the Dingo base as well.

This thesis contributes:

- a calibration method that can estimate the actuator current/torque ratio and friction of a joint, described in Chapter 2.
- an adaptation of the classic impedance controller using the calibration values, that can be used on current-controlled robots, described in Chapter 3.
- a software architecture for compliant control of mobile manipulators, suitable to use with the Kinova arm and Dingo base, presented in Chapter 4.
- an experimental validation of the performance of the adapted impedance controller, described in Chapter 5 with the results presented in Chapter 6.

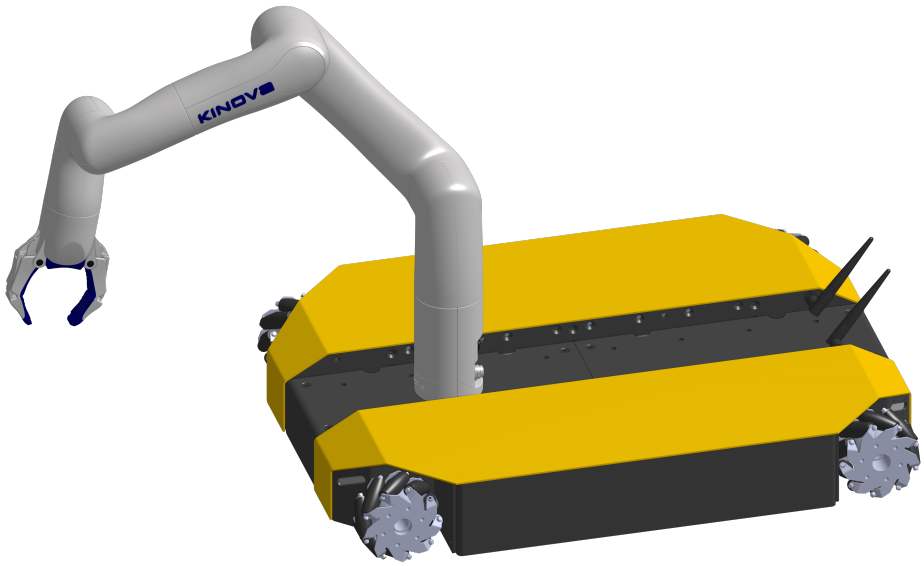


Figure 1.5: Clearpath Dingo Omnidirectional base with Kinova GEN3 Lite arm.

2

Calibration

Impedance control is generally achieved by the use of torque sensors in the joints. Such a torque sensor enables closed-loop torque control where the actuator can constantly adapt its current to achieve the desired torque. The goal of this project is to use impedance control on a robot without torque sensors. To do this, knowing the ratio between actuator torque and current is required. A calibration process is developed that uses the model of the robot and the earth's known gravity to define this ratio, as described in Section 2.1. This calibration process is also able to estimate the friction of a joint, which is further explained in Section 2.2. Problems due to model inaccuracies and a corresponding solution are described in Section 2.3. For some joints, the calibration method cannot be used since they do not experience gravitational effects. Solutions for this problem are described in Section 2.4.

2.1. Current/Torque ratio

To match a joint's actuator current with the actual torque it produces, the earth's gravity constant of $g = 9.81\ \mathrm{m\,s^{-2}}$ can be used in combination with an accurate model of the dimensions and masses of the individual links of the robot. For any pose of the robot, it is possible to calculate the torque resulting from gravity $\tau_{gravity}$ on the joint of the robot. For the rotation of a single joint while assuming zero friction (note that friction is addressed in Section 2.2), the dynamic equation for that single joint can be expressed as the scalar equation:

$$I\alpha = \tau_{gravity} + \tau_{actautor} \tag{2.1}$$

where I is the inertia of the body rotating around the joint, α is the angular acceleration, $\tau_{actuator}$ is the torque provided by the actuator and $\tau_{gravity}$ is the torque due to gravity. If the robot's body rotates

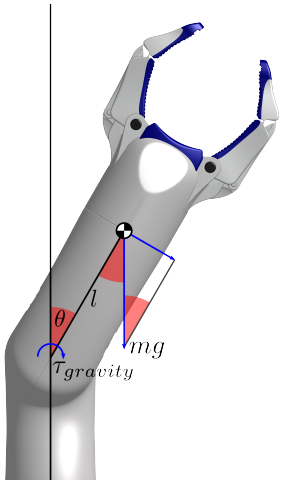


Figure 2.1: Torque acting on a joint due to gravity.

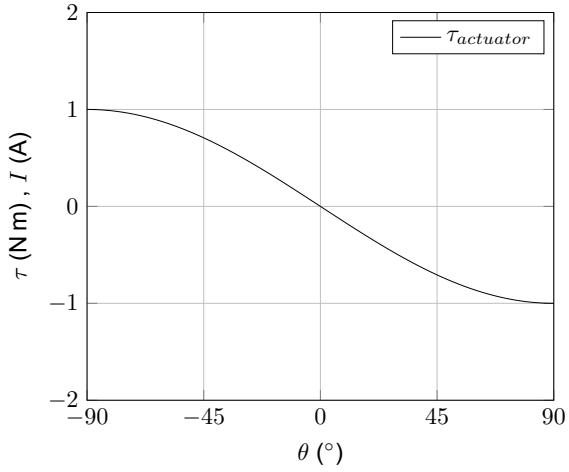


Figure 2.2: Model-based torque for calibration movement.

around the joint with a constant velocity, so that $\alpha = 0$, Equation (2.1) can be rewritten to:

$$\tau_{actuator} = -\boldsymbol{\tau}_{gravity} \tag{2.2}$$

Therefore, for a constant rotational velocity of the robot, the actuator torque $\tau_{actuator}$ is exactly the opposite of the torque the joint experiences from gravity $\tau_{gravity}$. The gravitational torque acting on a joint depends on the mass it is carrying m, the length of the moment arm l, and the angle between the moment arm and the direction of gravity θ, as shown in Figure 2.1. The gravitational torque on a joint is therefore defined as:

$$\tau_{gravity} = mgl \sin(\theta) \tag{2.3}$$

In this equation, mgl is a constant factor. For simplicity, this constant factor is assumed to be $mgl = 1$ during the explanation of the calibration method in this chapter. This results in the gravitational torque being equal to $\tau_{gravity} = \sin(\theta)$. When rotating the joint from $\theta = -90\ °$ to $\theta = 90\ °$ with a constant velocity, the joint actuator torque should then be the opposite of this gravitational torque, and thus $\tau_{actuator} = -\sin(\theta)$ as plotted in Figure 2.2. The high-level controller of the robot can be used to rotate a joint from one side to the other side with a constant velocity. During this movement, the current used by the joint actuator can be recorded. To avoid the acceleration and deceleration at the beginning and the end of the trajectory, guaranteeing a constant speed as desired in Equation (2.2), the movement can be performed from $\theta = -100\ °$ to $\theta = 100\ °$, while only measuring between $\theta = -90\ °$ and $\theta = 90\ °$. This will result in a dataset $\mathcal{D}$ containing N measurements of the actuator current $I_{actuator}$ and the model-based actuator torque $\tau_{actuator}$, defined using the measured values of θ. For readability, $I_{actuator}$ and $\tau_{actuator}$ are written as I and τ with the corresponding index of the sample:

$$\mathcal{D} = \{(I_1, \tau_1), \ldots, (I_N, \tau_N)\} \tag{2.4}$$

Based on such data, it is possible to estimate the ratio between the joint actuator's current and torque. This can be done by solving the following optimization problem:

$$r^* = \underset{r}{\operatorname{argmin}} \sum_{i=1}^{N} (I_i - r\tau_i)^2 \tag{2.5}$$

where I_i and τ_i are respectively the measured current and the model-based torque at sample i and the outcome value r^* is the current/torque ratio for which the difference between the model-based torque and the measured current is minimal. Figure 2.3 shows the model-based torque and expected actuator current for the example where the actuator requires a current of $I = 2$ A to generate a torque $\tau = 1$ N m, which will lead to a value of $r = 2$ after solving to the minimization problem of Equation (2.5).

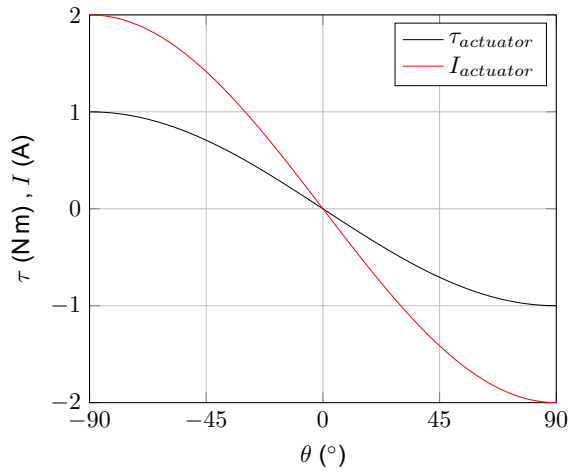


Figure 2.3: Torque and current for a cur/tor ratio of $r = 2$.

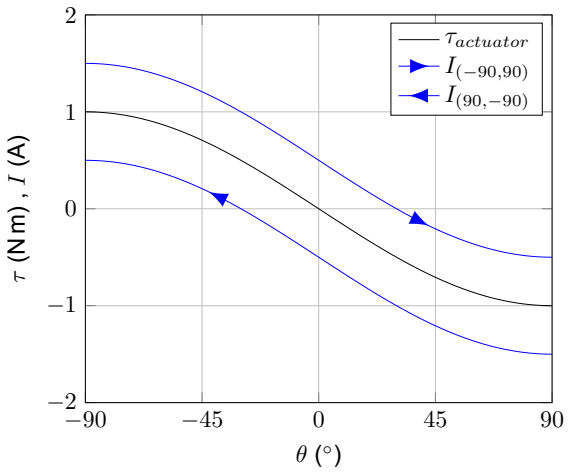


Figure 2.4: Torque and current for a friction of $f = 0.5$ A.

2.2. Friction

The gravity model discussed in Section 2.1 assumes a frictionless system. However, in the real world, there is always friction. Friction always works against the movement of a joint. Figure 2.4 shows the model-based torque and expected actuator current when performing the calibration movement for a joint with a current/torque ratio of 1 and with a current of $\frac{1}{2}$ A being required to exactly deliver the frictional torque. Where the actuator torque for a frictionless joint only depends on joint angle θ, for a joint with friction also the direction of movement is a dependency. This can be seen in Figure 2.4, where the actuator current is shifted up relative to the model for a movement from $\theta = -90\,^\circ$ to $\theta = 90\,^\circ$, while it is shifted down for the opposite movement from $\theta = 90\,^\circ$ to $\theta = -90\,^\circ$. This makes sense since the friction is working against the upward link movement in the first part of both trajectories, resulting in the absolute value of the current being larger than the model torque, while the friction partly prevents acceleration of the joint during the downward movement of the link at the end of both trajectories, resulting in an absolute value of the current being smaller than the model torque. Based on such data, it is possible to estimate the friction (still assuming a current/torque ratio of $r = 1$), by solving the following minimization problem:

$$f^* = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{N} (I_i - (\tau_i + f))^2 \tag{2.6}$$

where I_i and τ_i are the measured current and model-based torque at sample i respectively and the outcome value f^* is the friction value for which the difference between the model-based torque and the measured current is minimal. This friction f is thus expressed as the current that is required to generate the torque required to overcome the friction of the joint. Depending on the direction of movement during the collection of the data points, the friction f that results from the minimization problem can be a positive or a negative value, but since it is friction, the absolute value is of interest.

Section 2.1 described the method to estimate the current/torque ratio for a frictionless joint and this section described the method to estimate the friction of a joint with a current/torque ratio of 1. However, most actuators have both friction and a ratio different than 1. A data plot of such an actuator would therefore look like a combination of Figure 2.3 and Figure 2.4, with the actuator current being a scaled and vertically shifted version of the model torque. To estimate the ratio and friction of such an actuator, one needs to solve the following minimization problem:

$$r^*, f^* = \underset{r,f}{\operatorname{argmin}} \sum_{i=1}^{N} (I_i - (r\tau_i + f))^2 \tag{2.7}$$

which is a combination of the previous two minimization problems from Equation (2.5) and Equation (2.6). This problem can be solved with a minimization solver supporting multi variables, for example with Python's Scipy library [30], where the outcome is the current/torque ratio and the friction of the actuator.

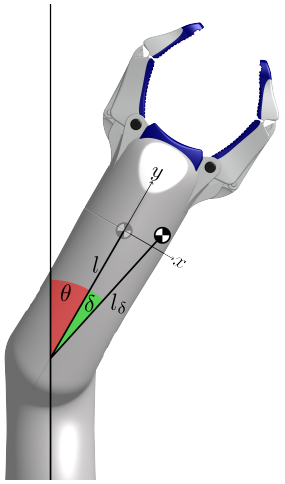


Figure 2.5: Difference between model (l) and actual (r) COM.

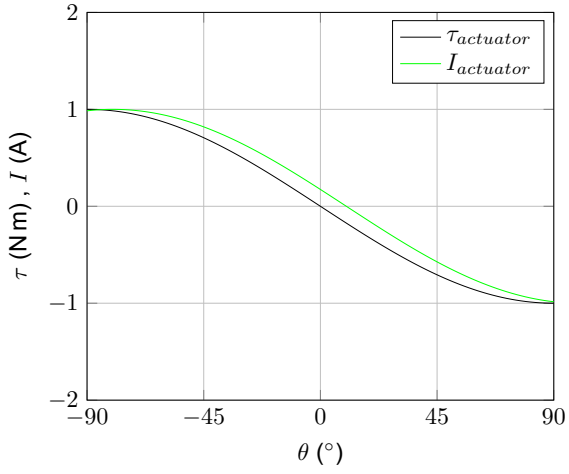


Figure 2.6: Torque and current for a phase shift of $\delta = -10\,^\circ$.

2.3. Phase shift due to model inaccuracies

Section 2.1 and Section 2.2 described how the current/torque ratio r and the friction f of a joint actuator can be estimated. This was done by solving a minimization problem that fits the torque $\tau_{actuator}$ based on the model of the robot onto the actual current $I_{actautor}$, measured during the movement of a joint. Therefore, this method is highly dependent on an accurate model of the robot with the correct center of mass (COM) location and magnitude. However, the model can be inaccurate, as shown in Figure 2.5, where the COM is displaced. A displacement in the z direction would mean a displacement parallel to the axis of rotation of the joint and therefore does not affect the calibration of this joint. Note that such a displacement can influence the calibration of other joints, but this will be discussed later. A displacement only in the y direction would affect the length of the moment arm l, but since this was part of the constant factor mgl, only the current/torque ratio will be slightly affected. A displacement in the x direction leads to major problems, since this results in a reformulation of the gravitational torque from Equation (2.3) to:

$$\tau_{gravity} = mgl_{\delta} \sin(\theta + \delta) \tag{2.8}$$

where l_{δ} is different than the original l and δ is the angle difference between the model and the actual robot. This factor δ results in a phase difference between the model-based torque and the measured current. Figure 2.6 shows an example of the expected actuator current with an angle difference of $\delta = -10\ °$, assuming a current/torque ratio of $r = 1$ and zero friction. If there exists such a phase difference, the estimation of the current/torque ratio and friction as described in Equation (2.7) will be inaccurate.

To solve this problem, the model of the robot needs to be adjusted such that it matches the real robot. Since the COM location used to calculate the gravitational torque on a joint is a combination of the COM locations of all individual links, changes should be made starting at the last link, and working back to the first link. Therefore, the phase of the last joint on the actual robot needs to be estimated first. This can be done by recording the actuator current again for a constant joint rotation, while also saving the corresponding joint angle θ this time, resulting in the new dataset:

$$\mathcal{D} = \{(I_1, \tau_1, \theta_1), \ldots, (I_N, \tau_N, \theta_N)\} \tag{2.9}$$

Note that if the joint is also influenced by friction, a vertical displacement of the current data is caused as was shown in Figure 2.4. By recording the data on the trajectory from $\theta = -90\ °$ to $\theta = 90\ °$ and on the trajectory from $\theta = 90\ °$ to $\theta = -90\ °$ and taking the average on each angle θ, the effect of friction

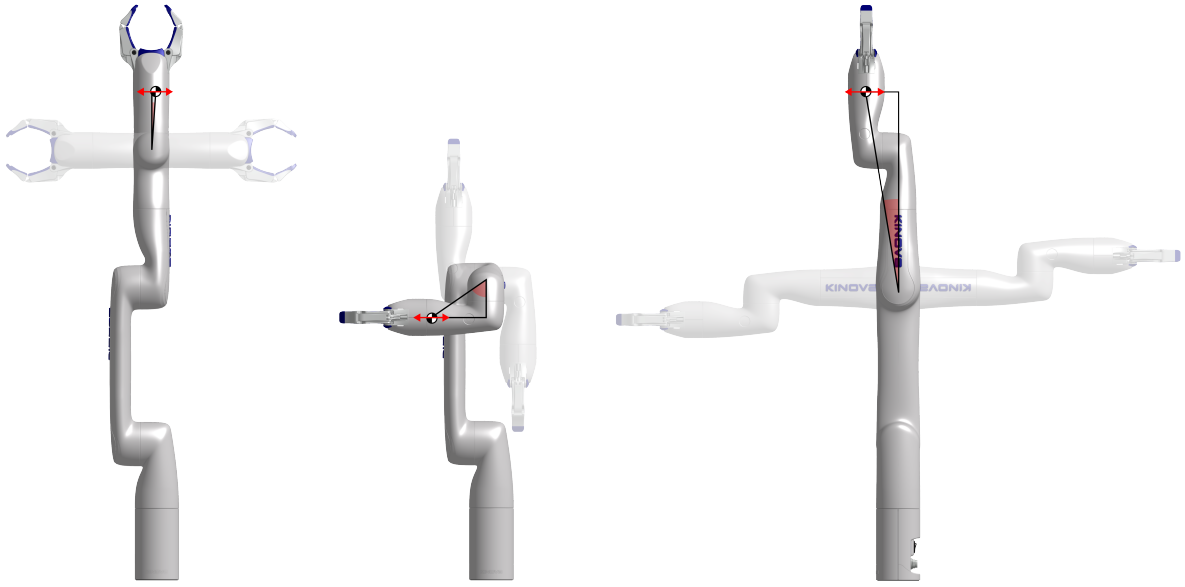


Figure 2.7: Three calibrations, all evaluating another axis of the COM location of the last link. The transparent poses are at $\theta = -90$ and $\theta = 90\ °$,while the solid pose is at $\theta = 0\ °$.

will be canceled out. Next, it is possible to solve the following minimization problem:

$$s^*, \delta^* = \underset{s,\delta}{\text{argmin}} \sum_{i=1}^{N} (I_i - s \sin(\theta_i + \delta))^2 \tag{2.10}$$

where I_i is the measured actuator current again at sample i and where θ_i is the joint rotation at that time step, while $s*$ and $\delta*$ are the scaling factor and the angle difference for which the difference between the model-based torque and the measured current is minimal. The outcome of $\delta*$ can be compared with the model of the robot. If this differs, the model should be adapted so that it matches the real robot. Next, the process can be repeated for the second-last joint and so on for all joints. Note that the outcome value of the scaling factor $s*$ is not used, since after correcting the model using δ, the original method described in Section 2.1 and Section 2.2 is used to estimate the current/torque ratio r and the friction f.

As mentioned before, for the joint shown in Figure 2.5, a displacement of the COM in the x direction causes a phase shift, while displacements in the y and z directions have less effect. However, displacements in the y and z can influence joints lower in the arm much more. Therefore, with other calibration movements, a potential difference in the COM location in other directions can be estimated. Figure 2.7 shows the three different calibration movements that can be used to estimate the COM location of the last link in all directions.

2.4. Joints without gravitational torque.

By first using the method described in Section 2.3 to reduce errors between the actual COM locations and the COM locations of the model, followed by the calibration described in Section 2.1 and Section 2.2, it is possible to estimate the current/torque ratio and the friction of an actuator. However, this is only possible for actuators that experience a gravitational torque during the calibration movement, which is not the case for every joint. For the Kinova arm, the first joint rotates the arm around the gravity vector, which means that this joint does not experience gravity. The last joint of the arm only rotates the gripper, which is exactly symmetric and therefore also not affected by gravity.

To estimate the current/torque ratio of these types of joints, a more complex model is required. To calibrate the first joint, it is possible to place the base link of the arm not flat on the ground, but on a ramp. By doing this, the first link will also experience gravity when rotating, making it possible to use the described calibration method again. It will however also affect the gravitational torques of all other actuators, meaning that a more complex model is required that takes into account the orientation of the base link with respect to the gravity vector. For symmetrical joints, the robot and the model can be adapted to break the symmetry, for example by letting the gripper hold a stick with a known COM that is not symmetric around the axis of rotation. By breaking the symmetry, it is possible to use the described calibration method again to estimate the current/torque ratio.

To avoid the additional complexity of both solutions, during this project the current/torque ratios of the first and the last joint were derived from other similar joints. Kinova uses three different types of actuators, called 'small', 'medium' and 'large'. The first joint contains a 'medium' actuator, similar to the third joint. The last joint contains a 'small' actuator, similar to the last two joints. The assumption is made that the current/torque ratios are equal for actuators of the same type. This allows copying of the current/torque ratios from the calibratable joints to the first and the last joint. With these current/torques ratios, the calibration method can still be used to estimate the friction, by solving the minimization problem from Equation (2.7), but this time with the ratio r already known.

3 Control

With the calibration process described in Chapter 2, it is possible to estimate the current/torque ratios and the frictions of the actuators. With these values, it is possible to design an impedance controller that does not require torque sensors to function. Since impedance control is based on the model of the robot, Section 3.1 introduces the general dynamic model of a robot arm with related information about the transformation into the operational space (Section 3.1.1), the use of a generalized inverse (Section 3.1.2) and nullspace projection (Section 3.1.3). The definition of the impedance controller and the adaptions for the support of current control are presented in Section 3.2, with details about the nullspace controller in Section 3.2.1. Section 3.3 explains the developed strategy to compensate for the joint frictions, which were estimated during calibration. Section 3.4 defines the controller that is used for the Dingo base, to achieve compliance for the arm and base working together as a mobile manipulator.

3.1. Dynamic model

Impedance control is a model-based control technique and therefore an accurate dynamic model of the robot is required. A common way to derive a dynamic model in joint space with joint positions $\boldsymbol{q}$ and joint velocities $\dot{\boldsymbol{q}}$ is using the Lagrange formulation, which is described in Appendix A. To save work and avoid the risks of making errors in the derivation, software like *pinochio* can be used to do the derivation, which is further described in Section 4.3. The result is a dynamic model:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}) + \boldsymbol{\tau}_f = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext} \tag{3.1}$$

where the inertia matrix $\boldsymbol{M}(\boldsymbol{q})$, the Coriolis/centrifugal matrix $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the gravitational vector $\boldsymbol{\tau}_g(\boldsymbol{q})$ all depend on the state of the robot ($\boldsymbol{q}$, $\dot{\boldsymbol{q}}$) and thus describe the dynamics of the robot. The vector $\boldsymbol{\tau}_f$ is a vector of joint friction, which is desired to approach zero for impedance control. The dynamics of the robot equal the torques that are applied on the robot, which are the actuator torques $\boldsymbol{\tau}$ and the external torques $\boldsymbol{\tau}_{ext}$ when there is physical interaction between the robot and its environment.

3.1.1. Operational space

To express the dynamic model in the operational space, a Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$ can be used. The Jacobian matrix contains operational space variables $\boldsymbol{x}$ (for example the position of the end-effector), partially derived with respect to the joint-space variables $\boldsymbol{q}$. With the Jacobian matrix, the following relations can be used between joint space and operational space:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} & \ddot{\boldsymbol{x}} &= \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} & \boldsymbol{f} &= \boldsymbol{J}^{-T}(\boldsymbol{q})\boldsymbol{\tau} \\ \dot{\boldsymbol{q}} &= \boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}} & \ddot{\boldsymbol{q}} &= \boldsymbol{J}^{-1}(\boldsymbol{q})(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}) & \boldsymbol{\tau} &= \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f} \end{aligned} \tag{3.2}$$

By substituting these relations in Equation (3.1), it is possible to express the dynamic model in operational space, as shown in Appendix B. This results to:

$$\boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{f}_g(\boldsymbol{q}) + \boldsymbol{f}_f = \boldsymbol{f} + \boldsymbol{f}_{ext} \tag{3.3}$$

with the inertia matrix in operational space defined as:

$$\boldsymbol{\Lambda}(\boldsymbol{q}) = \boldsymbol{J}^{-T}(\boldsymbol{q})\boldsymbol{M}(\boldsymbol{q})\boldsymbol{J}^{-1}(\boldsymbol{q}) \tag{3.4}$$

and the Coriolis/centrifugal matrix $\boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ in operation space defined as:

$$\boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{J}^{-T}(\boldsymbol{q}) \left(\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{M}(\boldsymbol{q})\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right) \boldsymbol{J}^{-1}(\boldsymbol{q}) \tag{3.5}$$

3.1.2. Generalized inverse

The shape of this Jacobian matrix depends on the task the robot should perform. The task of positioning the end-effector of the Kinova arm at a desired location in space where the orientation does not matter, results in a 3×6 Jacobian matrix since 6 degrees of freedom of the arm are mapped to 3 degrees of freedom of the task. The inverse $\boldsymbol{J}^{-1}(\boldsymbol{q})$ for this example does not exist as the matrix is non-square. To solve this problem, a generalized inverse can be used, which is defined as:

$$\boldsymbol{J}^{W+}(\boldsymbol{q}) = \boldsymbol{W}^{-1}\boldsymbol{J}^{T}(\boldsymbol{q}) \left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{W}^{-1}\boldsymbol{J}^{T}(\boldsymbol{q})\right)^{-1} \tag{3.6}$$

where $\boldsymbol{W}$ is a weight matrix which must be invertible and symmetric so that $\boldsymbol{W}^T = \boldsymbol{W}$. To ensure dynamic consistency for null space projects, the weight matrix is defined as $\boldsymbol{W} = \boldsymbol{M}(\boldsymbol{q})$. More details about the derivation and this dynamic consistency can be found in Appendix C. From now on, when the inverse of the Jacobian $\boldsymbol{J}^{-1}(\boldsymbol{q})$ is used while the normal inverse does not exist, the generalized inverse from Equation (3.6) is used.

3.1.3. Nullspace projection

When a robot has more degrees of freedom than required for its task, there is a redundancy. This leads to a non-square Jacobian matrix, as mentioned above. The additional degrees of freedom of the robot can be used to perform a secondary task. Often it is desired to implement such a secondary task when there is a redundancy, to avoid unexpected behavior of the robot. To ensure that the torques of the secondary task $\boldsymbol{\tau}_2$ do not affect the main task, one can use a nullspace projection matrix to project the secondary task onto the nullspace of the main task. In the example of a main task of impedance control, which applies a force $\boldsymbol{f}$ on the end-effector in operational space, the joint torques of the main task $\boldsymbol{\tau}_1$ are, according to Equation (3.2), defined as:

$$\boldsymbol{\tau}_1 = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f} \tag{3.7}$$

Joint torques regarding a secondary task $\boldsymbol{\tau}_2$ can be projected onto the nullspace of the main task using the nullspace projection matrix $\boldsymbol{N}$, defined as:

$$\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{J}^{-T}(\boldsymbol{q}) \tag{3.8}$$

with $\boldsymbol{I}$ as the identity matrix. Since the end-effector force regarding the secondary task $\boldsymbol{f}_2$ is always zero after projection:

$$\boldsymbol{f}_2 = \boldsymbol{J}^{-T}(\boldsymbol{q})\boldsymbol{N}\boldsymbol{\tau}_2 = \boldsymbol{J}^{-T}(\boldsymbol{q}) \left(\boldsymbol{I} - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{J}^{-T}(\boldsymbol{q})\right) \boldsymbol{\tau}_2 = \left(\boldsymbol{J}^{-T}(\boldsymbol{q}) - \boldsymbol{J}^{-T}(\boldsymbol{q})\right) \boldsymbol{\tau}_2 = \boldsymbol{0} \tag{3.9}$$

a secondary task can be added to a main task after projection, resulting in the total control command defined as:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_1 + \boldsymbol{N}\boldsymbol{\tau}_2 \tag{3.10}$$

Note that the nullspace projection matrix is different for a position or velocity-controlled robot, due to the difference in Jacobian relations shown in Equation (3.2).

3.2. Impedance controller

The goal of impedance control is to let the robot behave as a spring-damper system with a desired stiffness and damping. To do this without force/torque sensors, a passivity-based controller can be used. A passivity-based controller cannot shape the inertia of the robot, since this would require force feedback from sensors. A passivity-based controller in the operational space is generally defined as:

$$\boldsymbol{f} = \underbrace{\boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}}_d + \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}}_d + \boldsymbol{f}_g}_{\text{"+" part}} \underbrace{-\boldsymbol{K}_d\boldsymbol{e} - \boldsymbol{D}_d\dot{\boldsymbol{e}}}_{\text{PD part}} \tag{3.11}$$

where the "+" part of the controller compensates for the effect of gravitational torques and precompensates for inertial and Coriolis/centrifugal forces that will be present in the desired trajectory [12]. The "PD" part defines the compliance of the robot with the stiffness $\boldsymbol{K}_d$ and damping $\boldsymbol{D}_d$ related to position and velocity errors of the end-effector $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_d$ and $\dot{\boldsymbol{e}}$. Substituting this controller in the dynamic model from Equation (3.3) results to:

$$\boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{e}} + (\boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{D}_d)\dot{\boldsymbol{e}} + \boldsymbol{K}_d\boldsymbol{e} + \boldsymbol{f}_f = \boldsymbol{f}_{ext} \tag{3.12}$$

which is a mass-spring-damper system as desired. Note that the effect of friction $\boldsymbol{f}_f$ is still present. For a perfect robot with (almost) zero friction, this term can just be neglected. However, usually the joint frictions are significant and ignoring them decreases the performance of the impedance controller. Therefore, a method that is used to compensate for this friction is described in Section 3.3.

The end-effector force command defined in Equation (3.11) can be converted to joint-torque commands using the Jacobian relation from Equation (3.2). The torques can be converted to joint-current commands $\boldsymbol{c}$ using the current/torque ratios $\boldsymbol{r}$ defined using calibration, which defines the current command as:

$$\boldsymbol{c} = \boldsymbol{r} \odot \left(\boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f}\right) \tag{3.13}$$

3.2.1. Nullspace controller

Usage of the described impedance controller on robots that have more degrees of freedom than required for the task results in redundancy. This happens for example when a robot with six degrees of freedom only needs to track a position with its end-effector, which requires three degrees of freedom. As a result, the behavior of the impedance controller can be unpredictable. The controller will always try to push the end-effector to the target, but it can do this in a way where the arm gets tangled up in itself. One can try to avoid this by adding a secondary task, that should not influence the main task. This secondary task $\boldsymbol{\tau}_2$ can be a joint-space controller that tries to push the joints to a desired pose $\boldsymbol{q}_d$ with a stiffness $\boldsymbol{K}_2$ and a damping $\boldsymbol{D}_2$, while projecting it on the nullspace of the main task using the nullspace projection matrix $\boldsymbol{N}$, defined in Equation (3.8), resulting in:

$$\boldsymbol{\tau}_2 = \boldsymbol{N}\left(\boldsymbol{K}_n(\boldsymbol{q} - \boldsymbol{q}_d) - \boldsymbol{D}_n\dot{\boldsymbol{q}}\right) \tag{3.14}$$

This joint-torque command can again be converted to a joint-current command using the defined joint/torque ratios.

3.3. Friction compensation

The performance of impedance control is highly dependent on the friction. If there is zero friction, even the smallest position error $\boldsymbol{e}$ will result in a torque that brings the error back to zero. However, when friction is present, a small error can lead to a torque that is too small to overcome the friction of the joint

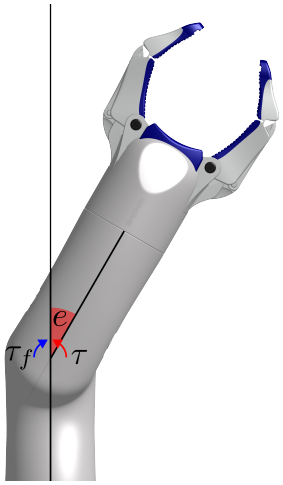


Figure 3.1: Persisting error since actuator torque equals friction.

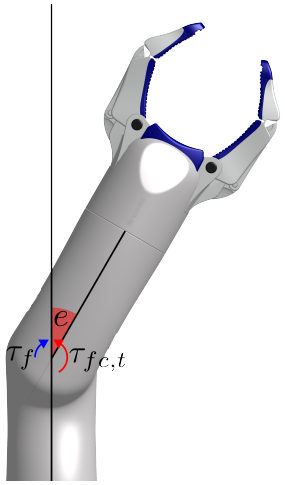


Figure 3.2: Friction compensation in the torque direction.

and the error will remain, as shown in Figure 3.1. A simple solution to this would be to increase the stiffness $\boldsymbol{K}_d$ of the controller, but this will also make the robot less compliant. Moreover, the problem will still exist, only for smaller errors now.

Therefore, to solve the problem, the friction should be compensated. Chapter 2 showed that the friction can be estimated for each joint using calibration. It is known that friction is not a constant, but depends on the velocity of the joint. There is usually also a significant difference between static friction and dynamic friction. However, for simplicity, the joint friction is assumed to be constant and equal to the value that was estimated for each joint during the calibration process. With this assumption, one could define a new controller with friction compensation $\boldsymbol{\tau}_{fc,t}$, based on the original controller $\boldsymbol{\tau}$ from Equation (3.11), as:

$$\boldsymbol{\tau}_{fc,t} = \boldsymbol{\tau} + \text{sign}(\boldsymbol{\tau})\boldsymbol{\tau}_f \tag{3.15}$$

where the friction value is added to the direction of the original torque output. This controller compensates for the joint friction and still provides the original impedance torque so that every position error results in a total torque that is large enough to overcome the friction and bring the error back to zero, as shown in Figure 3.2. However, compensating for the friction using this method also makes the robot less compliant. Pushing away the robot will now require a force large enough to overcome the friction twice, once for the actual friction and once since the friction compensation of the controller is also always pushing the robot back to its target. To avoid this, it is desired to compensate for friction in the direction of moving $\dot{\boldsymbol{q}}$, instead of the direction of the original controller torque $\boldsymbol{\tau}$. This makes sense since friction is always directed in the opposite direction of the velocity. Compensating in the direction of moving leads to the new controller $\boldsymbol{\tau}_{fc,v}$ defined as:

$$\boldsymbol{\tau}_{fc,v} = \boldsymbol{\tau} + \text{sign}(\dot{\boldsymbol{q}})\boldsymbol{\tau}_f \tag{3.16}$$

While the control law from Equation (3.15) always results in a torque in the error-decreasing direction, this control law can result in a torque in the error-increasing direction, as shown in Figure 3.3. This happens when the joint is moving in the error-increasing direction with a small position error, such that the torque of the friction compensation $|\text{sign}(\dot{\boldsymbol{q}})\boldsymbol{\tau}_f|$ is greater than the impedance torque $|\boldsymbol{\tau}|$ trying to reduce the error. This results in more compliance of the robot compared to no friction compensation.

The method of compensating friction in the velocity direction as done in Equation (3.16) is preferred over the method of compensating in the torque direction as done in Equation (3.15) since the velocity-based method leads to a more compliant robot. However, this method does not work when the robot has zero velocity while also having a position error e that is too small to overcome the friction to initialize movement. A solution is to use both methods, where the contribution of each depends on the velocity of the joint. One can define a threshold velocity t, where only the velocity-based compensation is used for an absolute joint velocity above that threshold. Below the threshold, the velocity-based compensation scales down while the torque-based compensation increases for a smaller absolute velocity. This is visually shown in Figure 3.4.

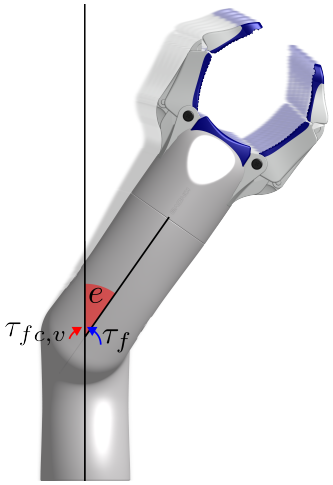


Figure 3.3: Friction compensation in the velocity direction.

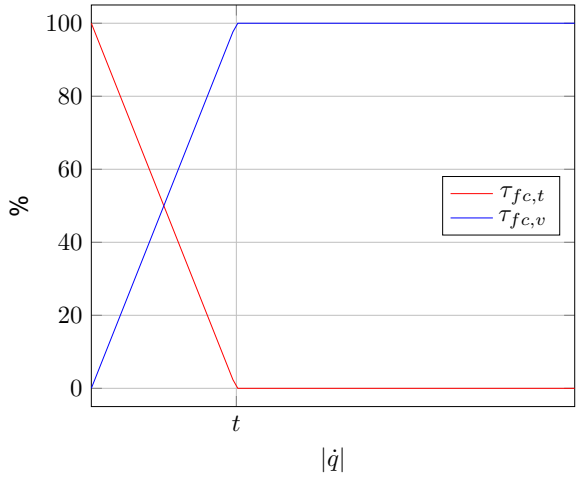


Figure 3.4: Transition of friction compensations with threshold t.

3.4. Base controller

To control a mobile base in a compliant way, a similar calibration to define the current/torque ratios in combination with an impedance controller could be used. The calibration method would require a slightly different setup, for example where the base drives uphill and downhill to compare the current with a modeled gravitational torque. However, for an omnidirectional base, the effect of wheel friction can be major, especially in combination with the relatively short moment arm that the wheels have in comparison with the link of a robot arm. It is hard to push an omnidirectional mobile base in the desired direction, even with the actuators turned off. This is because the base naturally prefers to roll over the almost frictionless rollers of the omnidirectional wheels, while rotation of the wheels required to move into the desired direction it is pushed to, is opposed by the wheel friction. Another challenge in this project is that the Dingo base that is used does not officially support current control. More details about the investigation of the adaption of the driver to enable current control are described in Section 4.1.2.

Because of the limitations, a different control strategy is used for the base. The base can be controlled in such a way that it tries to follow the end-effector. To do this, a desired end-effector position $\boldsymbol{x}_d$ is defined relative to the base. The controller always pushes the base in the direction that brings the end-effector ($\boldsymbol{x}$) to the desired relative position ($\boldsymbol{x}_d$). This happens with a stiffness $\boldsymbol{K}_b$, resulting in the desired base force:

$$\boldsymbol{f}_b = \boldsymbol{K}_b(\boldsymbol{x} - \boldsymbol{x}_d) \tag{3.17}$$

Note that an error in the vertical direction has no effect since the base can only move in the horizontal plane of the ground. However, since the base is omnidirectional, it can move in any direction on the ground. The base force of the controller can be converted into wheel torques $\boldsymbol{\tau}_w$ which corresponds with the desired force $\boldsymbol{f}_b$, using the relation between omnidirectional wheel rotations and resulting base movement. If a current/torque ratio is estimated, the wheel torques $\boldsymbol{\tau}_w$ can be converted to the corresponding wheel currents.

With this controller, the base will receive a force $\boldsymbol{f}_b$ pushing itself in the direction defined by the vector $\boldsymbol{x} - \boldsymbol{x}_d$, as shown in Figure 3.5. Since the end-effector location will change when the base moves, the impedance controller of the arm will correct this so that the end-effector still tries to reach its target. Since the base will always try to keep the end-effector at the desired relative position, the base will follow the end-effector that is pushed away. The stiffness $\boldsymbol{K}_b$ defines how compliant the base will be while following the end-effector. If $\boldsymbol{K}_b$ is small so that the base just overcomes the wheel frictions, the base could be easily stopped by a small external force pushing against the moving direction of the base. With this control strategy, a limited form of whole-body compliant control is possible, where the base is not fully compliant, but indirectly manipulatable since it follows the end-effector.

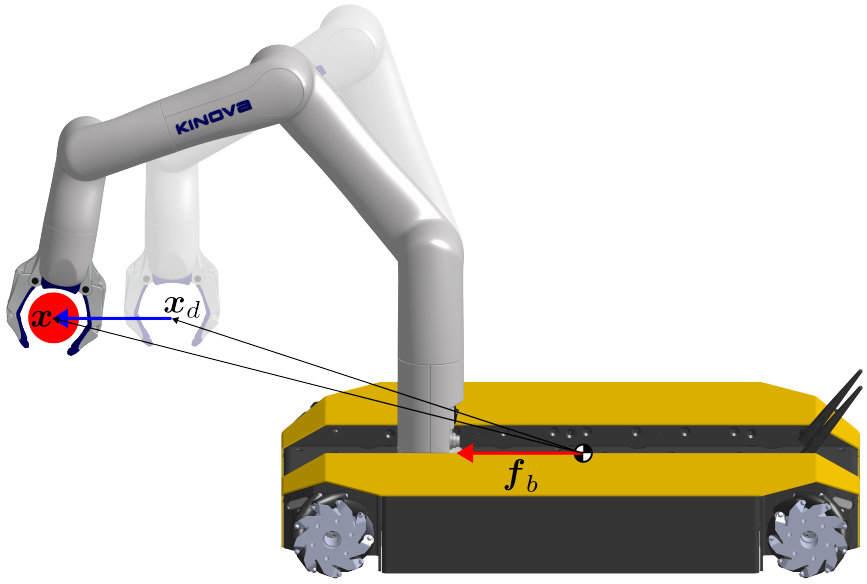


Figure 3.5: Functioning of the base controller. The end-effector tracks the target (red), resulting in an error (blue arrow). The controller creates a force (red arrow), pushing the base forward, to restore the position of the end-effector relative to the base.

4 Software

To apply the theory of the calibration and control described in Chapter 2 and Chapter 3, the development of software is required. The software is made as general as possible, but since the goal of this project is to use compliant control on the mobile manipulator shown in Figure 1.5, consisting of a Kinova arm and a Clearpath Dingo base, part of the software is specifically designed for these two systems.

Figure 4.1 gives an overview of the developed software and its structure. The software is split into two programs (ROS2 nodes), where the `user_interface_node` on the left side is executed on the user's device and the `control_interface_node` can be executed on the robot's computer. ROS2 [31] is used to let the two nodes communicate, such that user commands can be sent to the controller, while information about the state of the robot can be displayed in the user interface. When the simulation is used, both nodes are executed on the user's device. All software is written in Python to allow fast iterations during development, except for the `dingo_driver`, which is partly written in C++, as further explained in Section 4.1.2.

Section 4.1 discusses the low-level control software that was specifically made for the use of the Kinova arm (Section 4.1.1) and the Dingo base (Section 4.1.2) in current control mode. Section 4.2 explains the more general developed simulation and visualization (Section 4.2.1) and the user interface (Section 4.2.2) which provides information about the state of the robot and allows the user to toggle the settings and different modes of the controller. Section 4.3 explains the implementation of high-level control, consisting of the controllers discussed in Chapter 3.

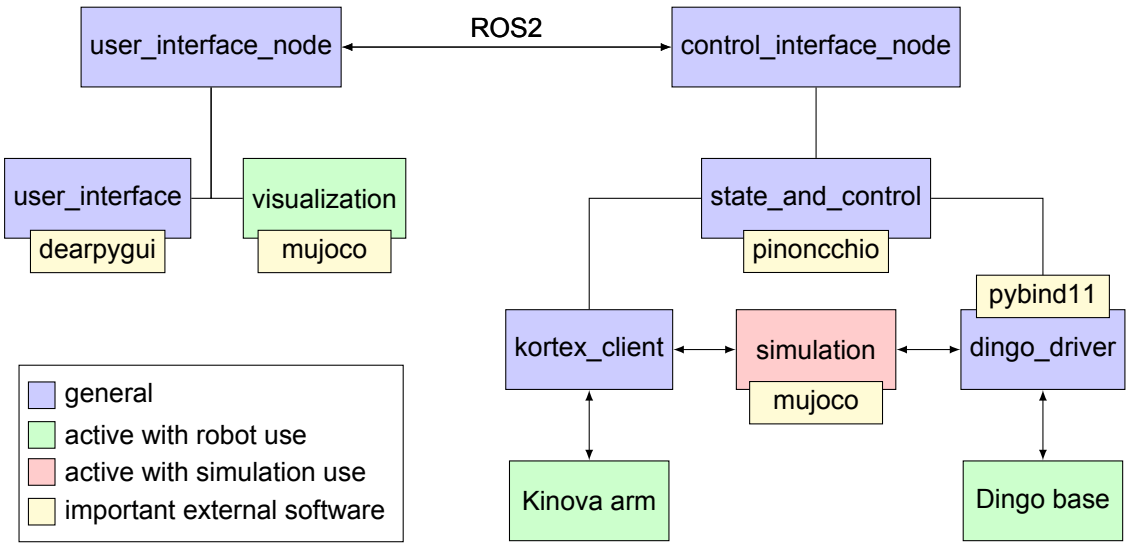


Figure 4.1: Software overview.

4.1. Low level

To use the controller described in Chapter 3, a lower level of software is required that can communicate with the Kinova arm and Dingo base to control them. ROS packages exist for both robots [32] [33] but only support position or velocity commands. Since direct current control of the motors is required to use the impedance controller, these ROS packages could not be used.

4.1.1. Kinova arm

Kinova provides software, called Kortex, which offers Python and C++ APIs to control the robot [34]. This can be done in a simpler high-level mode, which offers protections like singularity management and self-collision avoidance, but again only supports position or velocity commands. Kortex also provides a low-level mode, where it is possible to directly control the current of each actuator at a 1 kHz rate, but lacks the mentioned protections. To control the joint currents of the Kinova arm in low-level mode, the `kortex_client` is developed. This layer of software uses the Kortex Python APIs to constantly read the sensors and send commands to the actuators so that the `state_and_control` higher in the diagram of Figure 4.1 can obtain the state and control the arm.

4.1.2. Dingo base

Where Kinova with its Kortex software provides APIs to directly control the current of each actuator of the arm, Clearpath provides only the ROS package with velocity control for the Dingo base. For the goal of impedance control, it is also desired to control the current of the wheels of the Dingo base. The official driver [35] used in the ROS package only supports velocity control, but the motor controller boards are theoretically also able to be used in other modes. To be able to use the other modes, adaption of the driver is required and since this is officially not supported, no support or documentation is available.

The other modes of interest, theoretically supported by the motor controllers of the Dingo, are voltage mode and current mode. Voltage mode uses pulse-width modulation (PWM) to control the average power of the actuator. This is open-loop control with the control input as a number between 0 and 1, where the voltage is constantly off for 0 and constantly on for 1. Therefore, a larger number leads to more power and can thus generate a larger torque on the wheels. However, since the torque is related to the current, it is more interesting to control the current of the motors. This can be done with the current mode. The current mode is a closed-loop PID controller that uses the voltage mode controller to constantly adapt the pulse width to match the desired current.

Using the current mode results in extreme vibration of the motors. The reason for this is a threshold that is set on the motors themselves, which defines the minimum voltage command required to work. Any voltage command below this threshold is just ignored. Therefore, the motors are constantly switching between on and off when the voltage is around the threshold. To solve the problem in the current mode, adaption of the voltage threshold is necessary, which requires adaption of the software running on the motors itself, which is out of the scope of this project. Therefore, the adapted `dingo_driver` as shown in Figure 4.1 uses the open-loop voltage mode to control the motors. When the controller from the `state_and_control` part higher in the diagram of Figure 4.1 wants to increase the torque on one of the wheels, this developed driver will increase the pulse widths to increase the power of the motor such that the motor can provide a larger torque. However, with this setup, there is no closed-loop control of the current, meaning that there is no guarantee of how much the current and thus the torque changes.

Since the original Dingo driver is only written in C++, the adapted version of the driver is also C++ software. All the other software for this project is developed in Python. Therefore, a Python wrapper is desired, which enables the use of the C++ code in Python as well. This Python wrapper is created with *pybind11*, a library that exposes C++ types in Python and vice versa [36].

4.2. Interfaces

To improve the development and usage of the robot, several interfaces are developed. A simulation environment is made to make testing without the robot possible. A visualization environment can visualize the robot and show its target when the real robot is used. A separate user interface shows information about the robot (real or simulated) and enables the user to easily send commands to the robot.

4.2.1. Simulation and visualization

To enable testing of the controller before using it on the real robot, a simulation environment is developed. This simulation environment uses *Mujoco*, a well-maintained and fast physics engine, that can be used in Python and C++ [37]. Furthermore, it provides interaction with the simulated robot, by clicking somewhere on the surface and dragging the mouse. This generates a spring-like connection between the selected point on the surface and the mouse location, which makes it possible to also test compliance in simulation. Figure 4.2 shows the user interface with on the right side the simulation where the interaction with the robot is shown by the red line.

The simulation can simulate the high-level mode of the Kinova arm, used to calibrate or move to a position like the home position, and the low-level mode, which directly controls the actuator torques of the joints. For the Dingo base, each omnidirectional wheel is completely simulated with all the individual rollers, so that the complete control can be tested. As shown in Figure 4.1, the general software structure is the same when using the simulation. The `kortex_client` and `dingo_driver` only work slightly differently, since they need to communicate with *Mujoco* now instead of the robot's hardware.

When the real robot is used instead of the simulation, *Mujoco* is still used to show a visualization of the robot on the user's device. Instead of simulating the physics of the robot, the visualization uses the sensor data of the robot to constantly render the pose of the robot. In this visualization, there is a red dot representing the target of the robot, which can still be moved by the user. With this setup, one can move a target around in the visualization to test the impedance controller, without requiring a sensor setup to detect a target in the real world.

4.2.2. User interface

The simulation or visualization discussed in Section 4.2.1 shows the state of the robot. Next to this, also a user interface is desired, that can show information about the robot and enable the user to send commands. This interface is created using *dearpygui* [38], a Python wrapper of the popular C++ GUI library *Dear ImGui* [39]. The generated interface is shown in the left part of Figure 4.2. It shows several graphs with live sensor data of the position, velocity and actuator effort (torque or current, depending on the use of the real robot or the simulation) of the joints of the Kinova arm and the wheels of the Dingo base. For the joints of the Kinova arm, it also shows the mode, the current/torque ratio and the friction that is used.

With buttons and checkboxes, it is possible to control the robot. One can enable or disable each joint

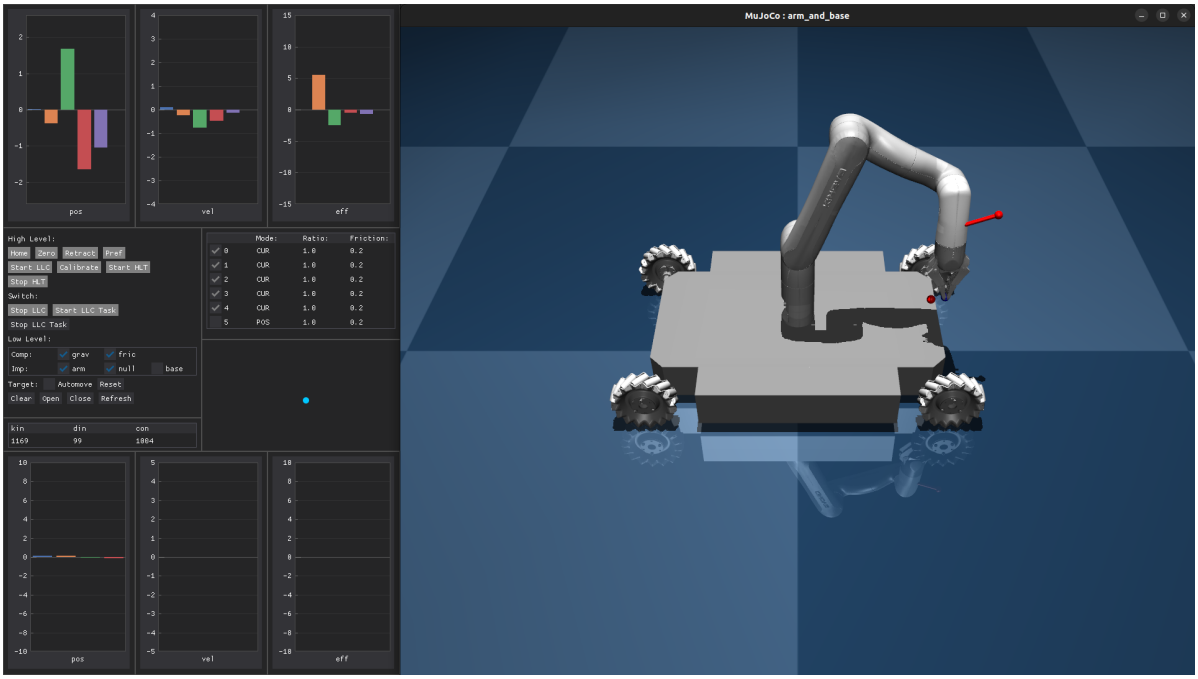


Figure 4.2: User interface and simulation. Interaction is possible and visualized as the red line between the robot and the mouse.

of the Kinova arm. There are buttons to let the Kinova arm move to its predefined positions (home, zero, retract, preferred) when the arm is in high-level control mode. One can switch to low-level control mode, start the low-level control task and enable or disable gravity and friction compensation, impedance of the arm and base and nullspace control. It is possible to open or close the gripper and automatically move the target around. To easily move the base there is a tile with a blue dot that functions as a joystick, to move the base into any direction.

4.3. High level

With the low-level software described in Section 4.1 and the interfaces for the user described in Section 4.2, the last required component is the high-level software, which was defined in Figure 4.1 as `state_and_control`. This software is connected with the low-level `kortex_client` and `dingo_driver` so that it can constantly read the sensor data and send commands to the arm and base. The sensor data contains the joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$ and with these it is possible to calculate the end-effector position $\boldsymbol{x}$ and velocity $\dot{\boldsymbol{x}}$, the variations of the Jacobian $\boldsymbol{J}(\boldsymbol{q})$, $\boldsymbol{J}^{-1}(\boldsymbol{q})$, $\boldsymbol{J}^{T}(\boldsymbol{q})$ and $\boldsymbol{J}^{-T}(\boldsymbol{q})$, the joint torques due to gravity $\boldsymbol{\tau}_g(\boldsymbol{q})$ and the matrics $\boldsymbol{M}(\boldsymbol{q})$ and $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ used by the controller.

Calculations of all these vectors and matrices that are based on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are complex and hard to derive. To do this, the *pinocchio* [40] software is used, which is a library for dynamic computations focusing on robotics. It contains Python bindings and can calculate all the mentioned vectors and matrices based on the URDF (Unified Robot Description Format) of the robot, which is a model that contains the dimensions, masses and inertias. Since these calculations need to be performed every update cycle of the robot (at 1 kHz for the arm), it is important to make the calculations as efficient as possible. Matrix calculations, especially inverses, are heavy tasks that can be hard to perform at such a high rate. The developers of *pinocchio* are working on version 3, which includes the use of *CasADi* [41], a symbolic framework for numeric optimization. It has not been released yet, but a preview version [42] was released in July 2023 [43]. This version is used for this project and can do all the calculations with $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ symbolically and save it as C code, which can later be used by providing the actual values of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. By using this, it was tried to make the calculations as efficient as possible.

With the use of the generated code, the `state_and_control` part of the software runs a continuous loop of defining the state of the robot and defining the actuator currents for the arm and the base using the controller discussed in Chapter 3. It does this with the configuration selected by the user in the user interface discussed in Section 4.2.2. This high-level software therefore forms the connection between the user and the low-level software of the robot.

5

Experimental Methodology

All experiments are performed using the Kinova GEN3 Lite arm and Clearpath Dingo Omnidirectional base, as shown in Figure 5.1 including the definition of joint number and link names. In the first experiment, further described in Section 5.1, the consistency of the calibration method from Chapter 2 is evaluated, by repeating the calibration process. The average of the values obtained during the calibrations is used in the other experiments. In a second experiment, the performance of the current-based impedance controller from Chapter 3 is evaluated for the Kinova arm and compared with a default controller from Kinova, as described in Section 5.2. In a third experiment, both the Kinova arm and the Dingo base are used, to evaluate the performance as a compliant mobile manipulator, described in Section 5.3.

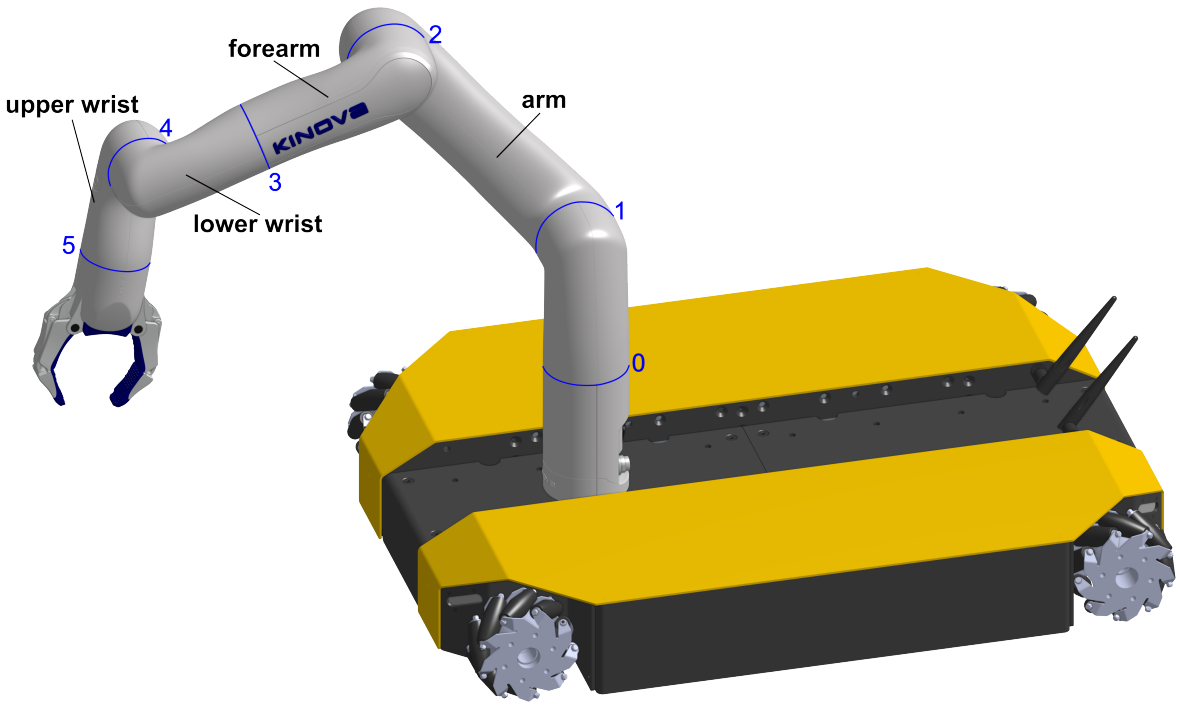


Figure 5.1: Definition of joints and links.

5.1. Experiment 1: Consistency of the calibration method

This experiment validates the consistency of the calibration method that was defined in Chapter 2. The calibration process is divided into three parts. To evaluate the consistency of the calibration method, each part of the calibration is repeated 49 times. In the first part, the phase shift is estimated for the real robot and compared with the model, as described in Section 2.3. In the second part, with the model corrected based on the outcome of the first part, the current/torque ratio and friction are estimated for all joints affected by gravity, as described in Section 2.1 and Section 2.2. In the third part, the friction of the first joint and the last joint, which are not affected by gravity, are estimated with the use of the current/torque ratio from the already calibrated joints, as described in Section 2.4.

Results of this experiment are shown in Section 6.1. The other two experiments, described in Section 5.2 and Section 5.3, use the corrected model and the estimated current/torque ratios and frictions, based on the average of outcome values of the 49 calibrations performed in this experiment.

5.2. Experiment 2: Compliance and performance of the arm

This experiment compares the designed compliant controller with a default velocity controller. To enable comparison between the two controllers, a setup is made as shown in Figure 5.2. A handle is designed for the gripper of the Kinova arm, to which a pulley can be attached. A connector for the Dingo base is made to assemble an aluminum 3030 extrusion profile vertically on top of the Dingo. A second connector is designed which can be used to place another pulley somewhere along the extrusion profile. A spring can be connected between the two pulleys, by tying up two ropes, one on each side of the spring to one of the pulleys. This connection limits the free-moving distance (where the spring is not extended) of the end-effector to the length of the unstretched rope l, as shown in the figure. The restriction of the free-moving distance enables comparison of the two controllers. The task of the robot during this experiment is to follow a predefined trajectory with its end-effector. Some parts of the goal trajectory are out of range l, which means that the spring needs to be stretched when the end-effector tracks the target. The default velocity controller, which is not compliant, should be strong enough to stretch the spring, while the designed compliant controller is compliant enough to be limited by the spring.

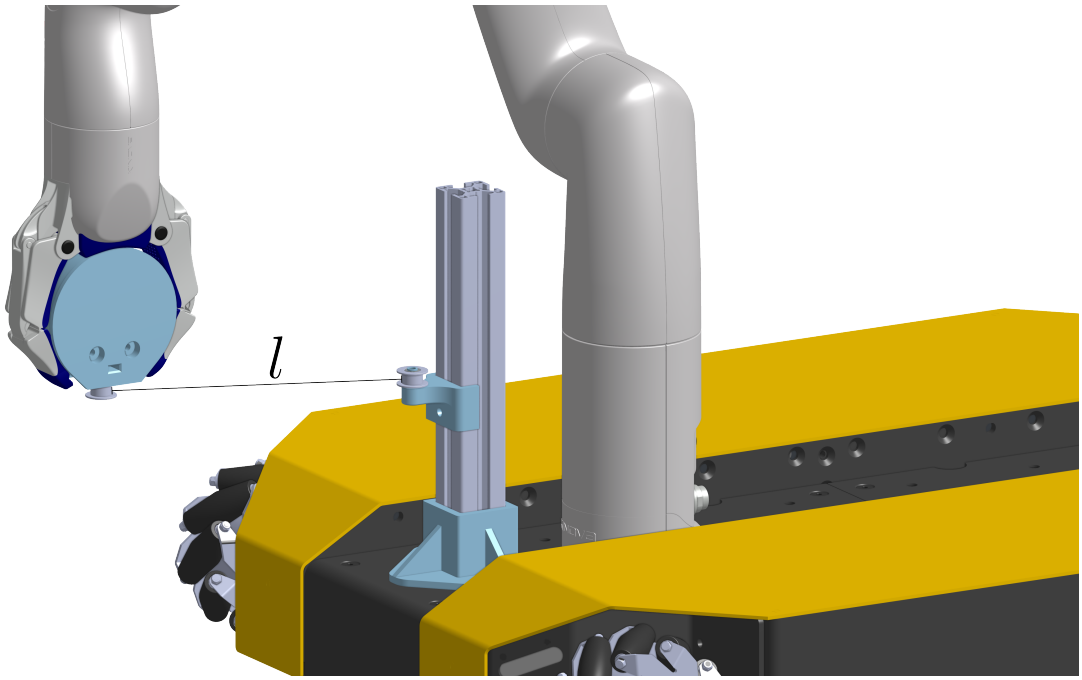


Figure 5.2: Setup of experiment 2. A gripper handle and base connector are designed to limit the free distance l.

5.2.1. Compliant controller

The compliant controller consists of a main task and a secondary nullspace task. The main task is position tracking of a slowly moving target with the end-effector. The impedance controller is defined according to Section 3.2 as:

$$\boldsymbol{f} = \underbrace{\boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}}_d + \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}}_d + \boldsymbol{f}_g}_{\text{"+" part}} \underbrace{-\boldsymbol{K}_d\boldsymbol{e} - \boldsymbol{D}_d\dot{\boldsymbol{e}}}_{\text{PD part}} \tag{3.11}$$

The effects of inertial, Coriolis and centrifugal forces are assumed to be small during this experiment, since the target and thus also the end-effector are moving with a low velocity. Therefore, for simplicity, the controller is used with $\ddot{\boldsymbol{x}}_d = 0$ and $\dot{\boldsymbol{x}}_d = 0$. Since there are no restrictions on the orientation of the end-effector, the robot has more degrees of freedom than required for the task, which results in redundancy. Since the last joint of the robot can only change the orientation of the gripper, but not the location of the tip of the fingers, this joint is locked and not used during the experiment. A null-space task is introduced because of the redundancy, which is defined as tracking the preferred pose defined in Table 5.1 as best, without affecting the main task of tracking. The stiffness and damping values set for the compliant controller are shown in Table 5.2. The controller uses friction compensation, as described in Section 3.3.

Joint 1:	Joint 2:	Joint 3:	Joint 4:	Joint 5:	Joint 6:
0	-20	100	-90	-60	0

Table 5.1: Preferred joint angles used as the null-space task for experiment 2. All are given in degrees.

Cartesian stiffness:	Cartesian damping:	Null-space stiffness:	Null-space damping:
40 $\mathrm{N\,m^{-1}}$	3 $\mathrm{N\,s\,m^{-1}}$	0.2 $\mathrm{N\,m\,rad^{-1}}$	0.1 $\mathrm{N\,m\,s\,rad^{-1}}$

Table 5.2: Configurations of the compliant controller during experiment 2.

5.2.2. Default velocity controller

A default velocity controller is used as a comparison for the compliant controller. Kinova provides a cartesian-space controller, but this can only be used for end-effector poses containing a position and orientation. Since the main task of this experiment is only tracking of position, this controller cannot be used and a joint-space controller is required. A joint-space controller can be used with position or velocity commands, where the velocity mode is used since this is more suitable for tracking a moving target. The velocity command is defined using the Jacobian relation between joint velocities and end-effector velocities, from Equation (3.2), where the end-effector velocity is based on the position error of the end-effector:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}} \qquad\qquad \dot{\boldsymbol{x}} = \boldsymbol{K_v}\boldsymbol{e} \tag{5.1}$$

where the gain $\boldsymbol{K_v}$ determines how aggressively the controller reacts to an error. This velocity controller is also redundant with the experiment task. Therefore, the same null-space task of the desired pose defined in Table 5.1 is added. However, since this is a velocity controller, the null-space matrix $\boldsymbol{N_v}$ is defined as:

$$\boldsymbol{N_v} = \boldsymbol{I} - \boldsymbol{J}^{-1}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q}) \tag{5.2}$$

5.2.3. Target trajectory

To compare both controllers, a target trajectory is required which is partly within the free-moving range of the setup, and partly outside it, as shown in Figure 5.3. Inside the free-moving range, the spring is not stretched which simulates the situation of no physical interaction with the arm. In this range, the tracking performance of the compliant controller can be compared with the default velocity controller. For the part of the trajectory that is outside the free-moving range, the compliance of the arm can be evaluated.

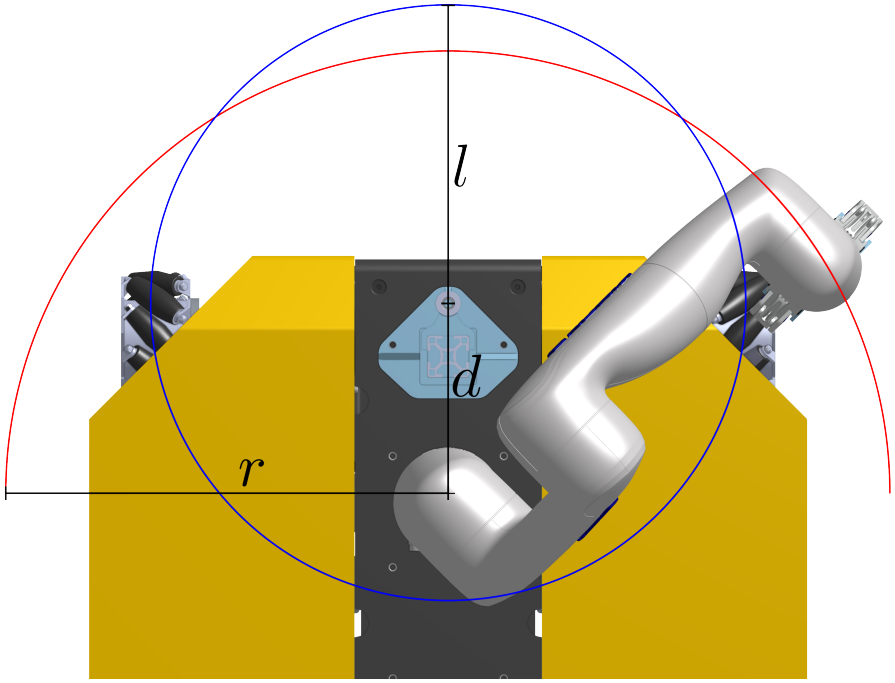


Figure 5.3: A spring limits the free-moving space of the robot. The target trajectory (red) is half a circle with radius r, the free-moving space is within the blue circle with radius l.

The length of the unstretched rope l, the radius of the target trajectory r, the distance between the first pulley and the origin of the arm d and the speed of the target moving along the trajectory during the experiment are as defined in Table 5.3. The target moves over the red line in Figure 5.3. At both ends of the trajectory and in the middle, the target stops moving for two seconds, enabling the end-effector to reach the target. The target moves in 9 seconds (excluding the stops), from one side to the other side, resulting in a speed of about 0.1 m s^{-1}.

rope length (l):	trajectory radius (r):	pulley distance (d):	linear speed (v):
0.260 m	0.315 m	0.145 m	0.101 m s^{-1}

Table 5.3: Setup configurations during experiment 2.

5.3. Experiment 3: Compliant mobile manipulator

This experiment combines the arm and the base, to validate the performance as a compliant mobile manipulator. The arm uses the same impedance controller as used in Section 5.2, but with the error for the end-effector limited to a maximum of 0.1 m to avoid extreme forces when the target is far away from the end-effector. Also, a different preferred pose is used as the null-space task, given in Table 5.4. With this preferred pose, the end-effector is positioned upwards, which simplifies interaction with the robot.

The experiment is performed in two parts, both using the designed controller for the base as described in Section 3.4. In the first part, the target is static relative to the base of the robot, which enables the user to use the robot in a guidance mode, where the user can control the mobile manipulator using interaction with the end-effector. In the second part, the target is defined in world space, which enables the tracking mode of the robot, where the robot tracks a target in the world while still being compliant.

Joint 1:	Joint 2:	Joint 3:	Joint 4:	Joint 5:	Joint 6:
0	20	90	0	0	0

Table 5.4: Preferred joint angles used as the null-space task for experiment 3. All are given in degrees.

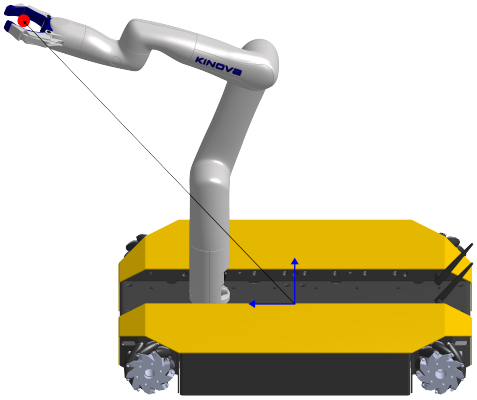


Figure 5.4: Guidance mode. Achieved using a static target specified in the reference frame of the base.

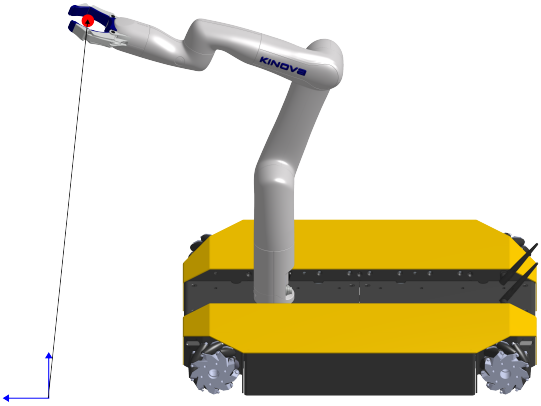


Figure 5.5: Tracking mode. Achieved using a target specified in the reference frame of the world.

5.3.1. Guidance mode

For the first part of the experiment, the target is statically placed at the location of the end-effector, when the arm is in the preferred pose stated in Table 5.1. The target is defined in the reference frame of the base, which means that the target follows the movement of the base, as shown in Figure 5.4. Therefore, the end-effector always tries to reach its original position after a disturbance caused by the interaction of the user.

The arm of the robot is compliant since it uses the designed impedance controller. This enables the user to push or pull the end-effector away from the preferred position. When the end-effector is moved away from the preferred position, the base will start to move as described in Section 3.4, trying to correct it. As long as the user keeps pushing the end-effector away from the preferred pose, the base will keep moving. When the user releases the robot, the end-effector can move to the target, which is still at the same location relative to the base. Therefore, the arm returns to approximately the original preferred pose and the base stops moving. By defining a static target relative to the base, placed at the end-effector location when the arm is in the preferred pose, easy guidance of the robot is possible.

5.3.2. Tracking mode

For the second part of the experiment, the target is placed in the reference frame of the world, as shown in Figure 5.5. Therefore, the target does not follow the movement of the base anymore, in contrast to the guidance mode. The user can push or pull the end-effector away from the target, due to the compliance. The base will also follow this movement, as the base always tries to follow the end-effector. When releasing the robot, the arm tries to move back to the target and the base will again follow. Therefore, by defining the target in the reference frame of the world, the robot will track the target in space when not disturbed by the interaction of the user.

In this mode, the target defined in the reference frame of the world can be moved. When the target moves, the end-effector tries to follow the target, similar to the second experiment described in Section 5.2. When the end-effector moves away from the preferred position, the base starts to follow the end-effector. When an obstacle blocks the end-effector in its attempt to track the target, the base will also come to a stop. Therefore, this mode enables the mobile manipulator to track a target, while still being compliant with interaction from objects or humans.

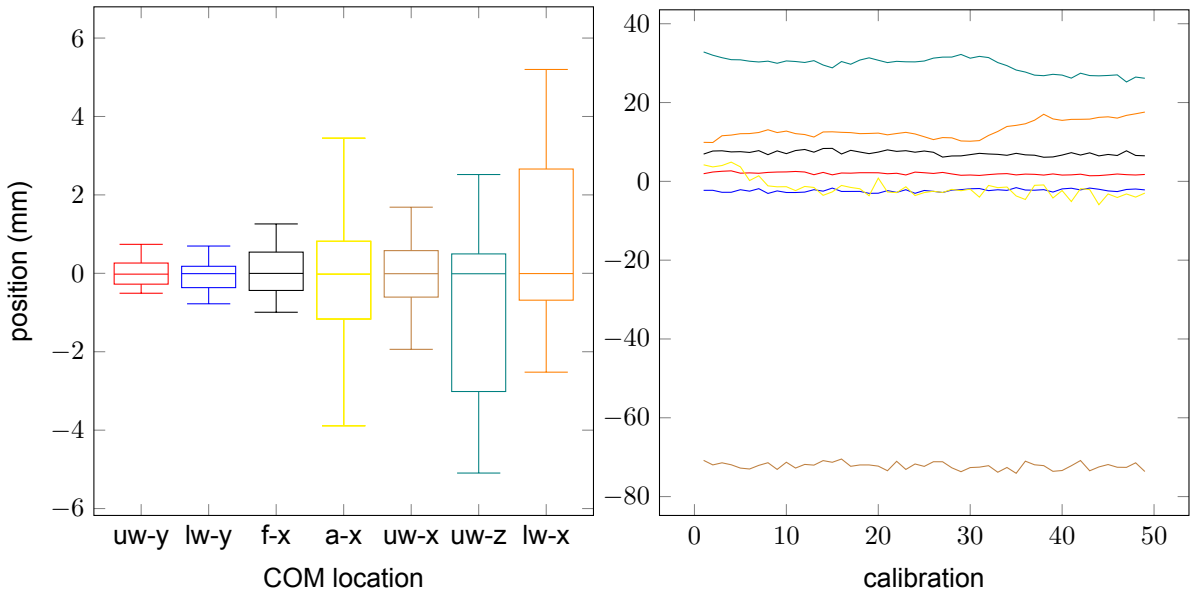
Since the location of the target in tracking mode is not static relative to the base, the robot needs to determine the target location continuously. A common method to track a real-world target, like an object of interest, is the use of a vision sensor like a depth camera. However, to avoid the integration of such a sensor and the related software required to track real-world targets, a virtual target is used for this experiment. This virtual target is a location in world space, but to use this, the robot also needs to know its own location in world space. A motion capture system from Vicon [44] is used to track the location of the robot. With this, the robot constantly knows its own location in world space. Since the location of the virtual target is also given in world space, the robot can now calculate its location relative to the virtual target and track it.

6 Results

This chapter presents the results of the experiments described in Chapter 5. Section 6.1 shows the consistency of the calibration method that is used. Section 6.2 presents the performance of the designed compliant controller and the default non-compliant controller of the arm and compares them. Section 6.3 shows the performance of the arm and base combined as the compliant mobile manipulator while used in the defined guidance mode and tracking mode.

6.1. Experiment 1: Consistency of the calibration method

In the first part of the experiment, 7 COM locations for the links defined in Figure 5.1 are estimated using 49 calibration rounds. Figure 6.1a shows the spread of each COM location using a boxplot with the median normalized around zero. The length of each box being smaller than 4 mm showcases the consistency of the calibration method. The greatest variances are present for the upper wrist z-location (blue) and the lower wrist x-location (orange). Figure 6.1b shows the COM locations per calibration round, where a synchronic change of the two locations is visible between the 30th and 35th calibrations. Calibration of the former (blue) affects the latter (orange), due to the order of calibration. This can explain the simultaneous change in the calibration outcome of these two locations. With the collected data, the average value for each COM location and the corresponding standard deviation are calculated



(a) Boxplot per COM location, normalized by subtracting median.

(b) COM location per calibration.

Figure 6.1: The COM location of the upper wrist (uw), lower wrist (lw), forearm (f), arm (a), in x/y/z direction, for 49 calibrations.

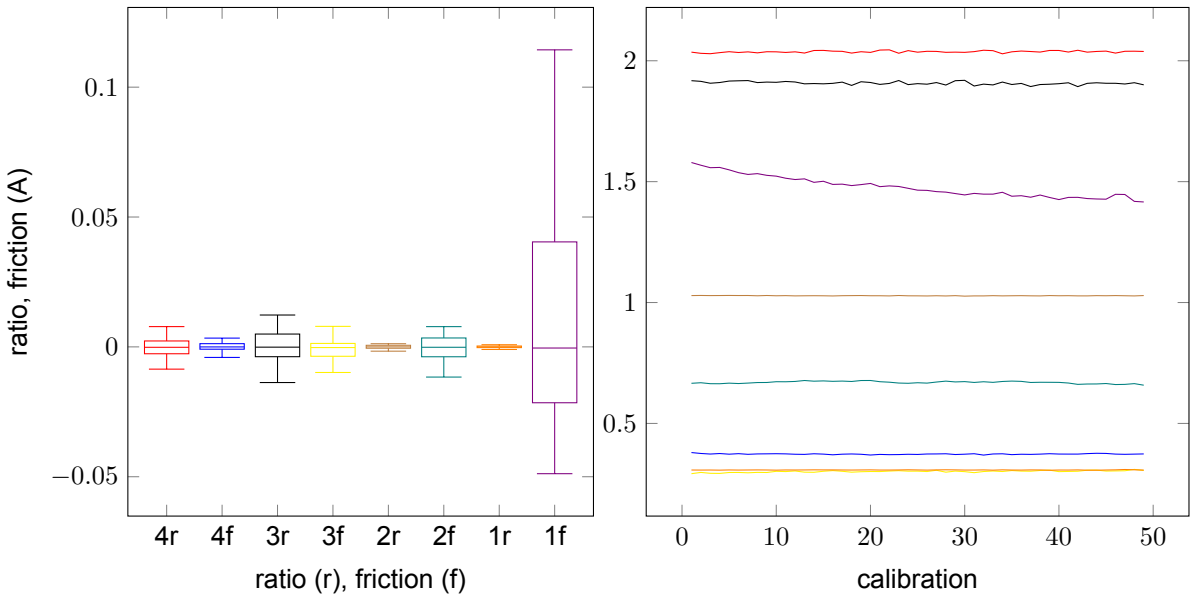
Link:	xyz:	Original:	Calibrated (avg):	Difference:	Std deviation:
upper wrist	y	-9.80	1.96	11.76	0.32
lower wrist	y	-10.00	-2.34	7.66	0.39
forearm	x	-30.16	7.17	37.33	0.58
arm	x	-29.98	-1.69	28.29	2.42
upper wrist	x	-80.57	-72.22	8.34	0.90
upper wrist	z	18.73	29.54	10.81	1.96
lower wrist	x	5.75	13.09	7.34	2.17

Table 6.1: Comparison of the original COM locations and the values obtained using calibration. All in mm.

and shown in Table 6.1, where the obtained values are compared with the original values from the model provided by Kinova. The difference between the calibrated values and the original values differs between 7 and 37 mm, while the standard deviation of the values differs between 0.3 and 2.4 mm.

The second part of the experiment is performed with the model of the robot adapted to the average COM locations in Table 6.1. In this second part, the current/torque ratio and the friction are estimated for joints 1 up to and including 4, which all experience the effect of gravity. Figure 6.2a provides the results as a boxplot with the median normalized around zero. All eight values show a small box, demonstrating the consistency of the calibration method, except for the friction of joint 1 (purple). Looking at Figure 6.2b, where the values are plotted per calibration round, explains that a continuous decrease in the friction of joint 1 is the reason for the large variation. Joint 1 is the only joint containing a large actuator and needs to deliver the highest torques since it is the first joint of the arm carrying all other joints and links. A possible explanation for the decreasing friction might be the temperature changing over time during the 49 calibrations, but no temperature data is collected and therefore further research is required to validate this theory. With the collected data, the average ratio and friction and their corresponding standard deviations are calculated and shown in Table 6.2.

Finally, in the third part of the experiment, the friction values of the two joints which are not affected by gravity (0 and 5) are estimated, by using the ratios of the other joints. Joint 0 is a medium-sized joint similar to joint 2, thus a ratio of $r = 1.0282$ is used. Joint 5 is a small joint similar to joints 3 and 4, so the average ratio of these two joints of $r = 1.9724$ is used. Table 6.3 shows the obtained frictions and the corresponding standard deviations.



(a) Boxplot per joint ratio/friction, normalized by subtracting median.

(b) Joint ratio/friction values per calibration.

Figure 6.2: The current/torque ratio (r) and friction (f) of the four joints for 49 calibrations.

Joint:	Ratio (avg):	Std deviation:	Friction (avg) (A)	Std deviation:
4	2.037300	0.004042	0.373231	0.001850
3	1.907429	0.006504	0.301255	0.003771
2	1.028151	0.000658	0.669460	0.000023
1	0.307416	0.000528	1.477590	0.043611

Table 6.2: The current/torque ratios and frictions obtained by calibration.

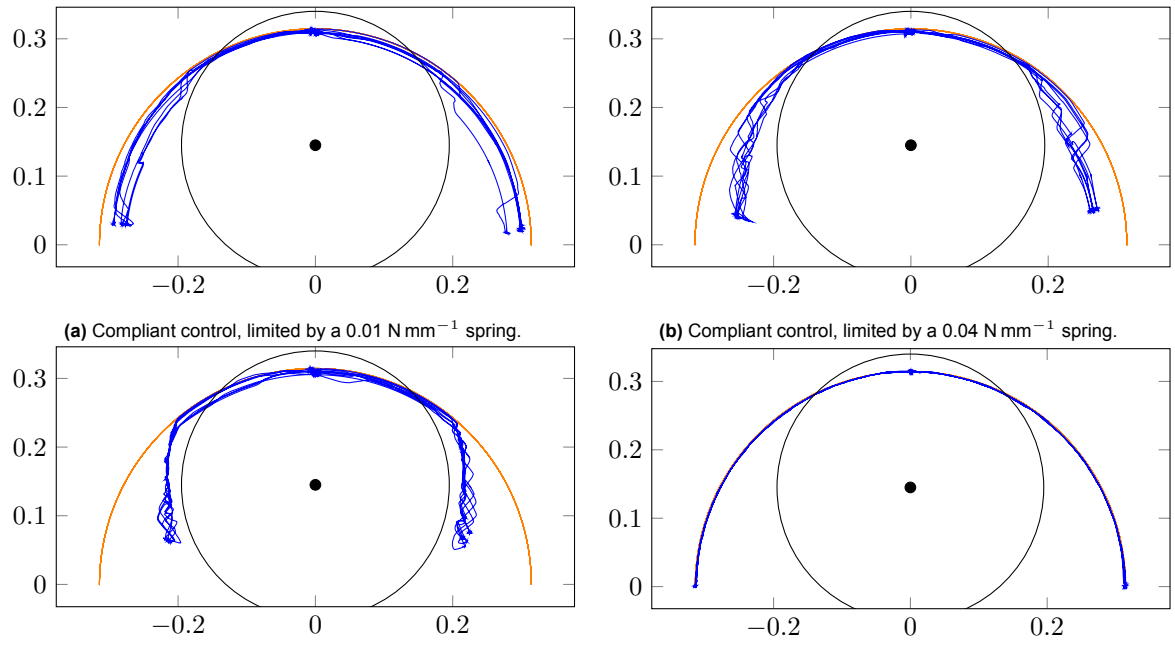
Joint:	Ratio:	Friction (avg) (A)	Std deviation:
0	1.028151	0.531819	0.003202
5	1.972365	0.592300	0.011891

Table 6.3: The frictions obtained by calibration using current/torque ratios based on similar joints.

6.2. Experiment 2: Compliance and performance of the arm

The second experiment is designed to validate the performance of the current-based impedance controller that is developed for the arm. The values for the COM locations in the robot model, the current/torque ratios and the joint frictions obtained in Section 6.1 are used. Data from the arm is recorded while the end-effector is tracking a target moving around half a circle four times. The experiment is performed three times, each time with a different spring, limiting part of the movement of the end-effector. Figure 6.3 shows the target trajectory in orange and the actual trajectory of the end-effector in blue, while the black circle shows the boundary at which the spring starts to pull the end-effector. The end-effector can freely move without physical interaction within the black circle. The further outside the circle, the more the spring is stretched and thus the larger the force that pulls the end-effector back into the direction of the center of the circle.

Figure 6.3a, Figure 6.3b and Figure 6.3c show the tracking performance of the impedance controller, with springs of stiffnesses 0.01, 0.04 and 0.12 N mm^{-1} respectively, limiting the movement outside the black circle. The impedance controller is set up such that the stiffness of the end-effector should be



(a) Compliant control, limited by a 0.01 N mm^{-1} spring.

(b) Compliant control, limited by a 0.04 N mm^{-1} spring.

(c) Compliant control, limited by a 0.12 N mm^{-1} spring.

(d) Velocity control, not affected by any of the springs.

Figure 6.3: Performance of the compliant controller compared with the default velocity controller, limited by different springs. Target trajectories are drawn in orange, end-effector trajectories in blue and the springs are not stretched within the black circles.

0.04 N mm^{-1}, similar to the stiffness of the limiting spring in Figure 6.3b. The end-effector follows the orange target trajectory better when the stiffness of the spring is smaller since in these situations the spring is weaker and thus more compliant than the arm (Figure 6.3a). In Figure 6.3b, the stiffness of the end-effector should be equal to the stiffness of the spring, which seems to be correct at first glance since the blue lines are about centered between the orange line and the black circle, meaning that the end-effector and the spring are about equally strong. In Figure 6.3c, the stiffness of the end-effector is smaller than the stiffness of the spring, resulting in the end-effector staying closer to the black circle, since it is relatively more compliant than the spring.

To compare the designed compliant controller with a default controller, tracking of the target with the end-effector limited by each of the springs is also performed while using a velocity controller. Since a velocity controller is not compliant, regardless of the stiffness of the spring, the result is equal to Figure 6.3d, where the stiffest spring of 0.12 N mm^{-1} is used. For the designed compliant controller, the tracking performance inside the black circle (with no external forces) is desired to be as close as possible to the velocity controller. Table 6.4 shows the average error of the end-effector for interactionless tracking, which corresponds to the average distance between the blue and the orange lines in Figure 6.3 within the black circles. The velocity controller is able to track the target trajectory with a precision of less than a millimeter, while the compliant controller achieves a precision of about five millimeters.

Compliant (1):	Compliant (2):	Compliant (3):	Velocity (1):	Velocity (2):	Velocity (3):
4.96	4.41	5.07	0.55	0.56	0.57

Table 6.4: The average errors for interactionless target tracking. All errors are given in mm, for the compliant and the velocity controller, for all 3 experiments.

Finally, the data from the experiment shown in Figure 6.3b, where the stiffness of the spring and the end-effector should be equal, is processed to estimate the actually achieved compliance. Each data sample of the recorded data contains the target position and the actual position of the end-effector. This enables the calculation of the position error and the extension of the spring for each sample. Multiplying the extension of the spring with the known spring stiffness gives the external force experienced by the end-effector. Figure 6.4 shows a scatter plot of the external force on the end-effector with the corresponding error, for all the data samples where the spring is extended. The data is quite scattered, which means that the relation between the position error and the external force is not very consistent. However, a data trend is visible and a linear fit resulted in the orange line with a slope of 0.04191 N mm^{-1}. This slope represents the measured cartesian stiffness of the controlled end-effector, which is

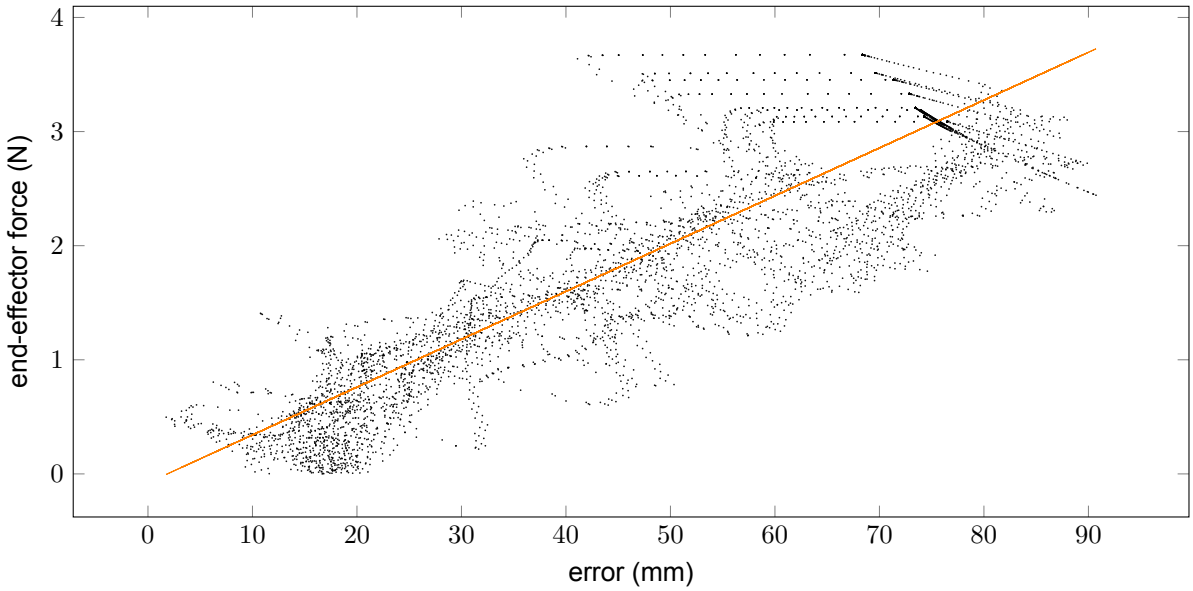


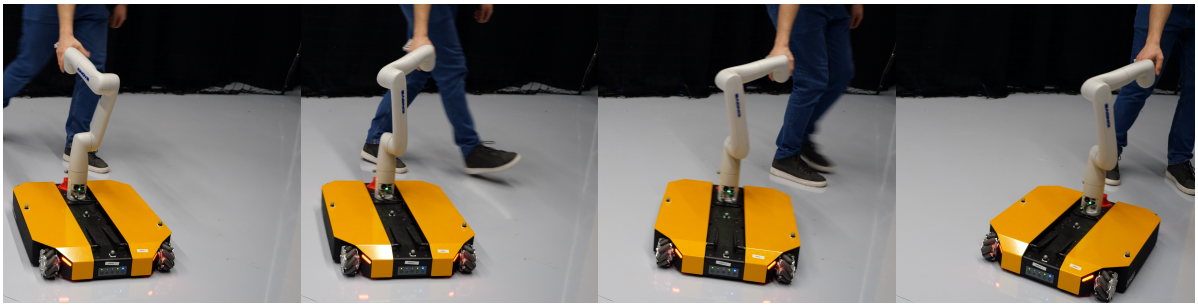
Figure 6.4: End-effector force plotted against the position error, using compliant control. Limited by a 0.04 N mm^{-1} spring.

desired to be 0.04 $\mathrm{N\,mm^{-1}}$, as was defined in Table 5.2. Based on these results, the current-based impedance controller seems to achieve a stiffness close to the desired stiffness, however with a very large variation.

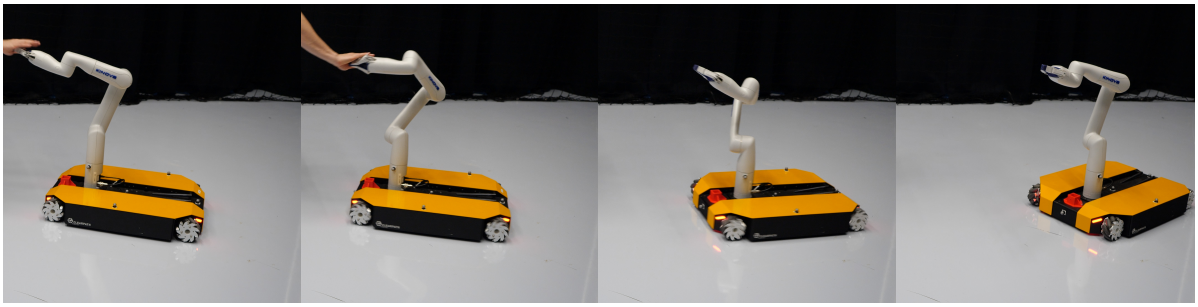
There can be several reasons for the variation in Figure 6.4. First, both the arm and the target are moving in the experiment. Therefore, the force of the end-effector created by the impedance controller is not directly defined by the impedance stiffness, but also affected by the impedance damping. Since the end-effector is moving, also inertia and Coriolis or centrifugal effects can affect the net force of the end-effector. Moreover, the friction compensation can affect the behavior of the end-effector since the friction compensation is applied in the joint space per joint actuator, which can affect the resulting end-effector force in the cartesian space. Furthermore, the measurement of the force is now based on the calculation of the extension of the spring and the known spring stiffness. Inaccuracies of the rope length with zero spring extension or stiffness will therefore also lead to inaccuracies of the measured force. Direct measurements of the force using a force sensor could reduce these uncertainties.

6.3. Experiment 3: Compliant mobile manipulator

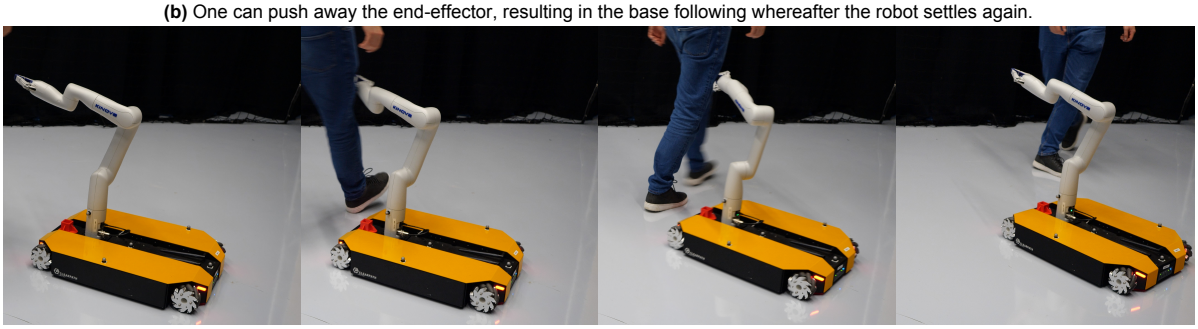
The third experiment validates the performance of the current-based impedance controller for the arm combined with the controller developed for the base. Usage of both controllers provides a form of whole-body compliance for interactions applied to the arm, as described in Section 3.4. In the first part of the experiment, the end-effector target is defined statically in the reference frame of the base. This results in the end-effector always trying to move back to the original position relative to the base so that



(a) One can pull the end-effector of the robot into the desired direction and the robot follows.



(b) One can push away the end-effector, resulting in the base following whereafter the robot settles again.



(c) The robot is compliant with bumps against the arm, for example from a human passing by.

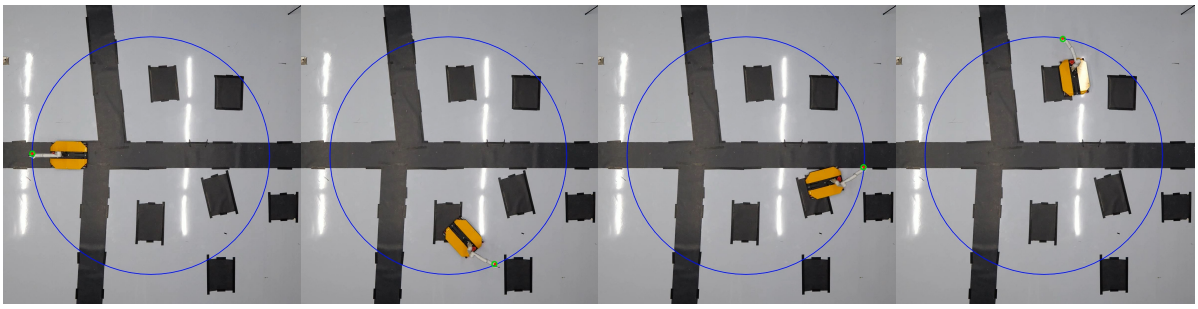
Figure 6.5: The robot operating in guidance mode. Because of the compliance, the user can manipulate the robot using the arm.

guidance of the robot is possible, as described in Section 5.3.1. A complete video of the experiment with the robot in guidance mode was published [45], but Figure 6.5 shows parts of the video as image sequences.

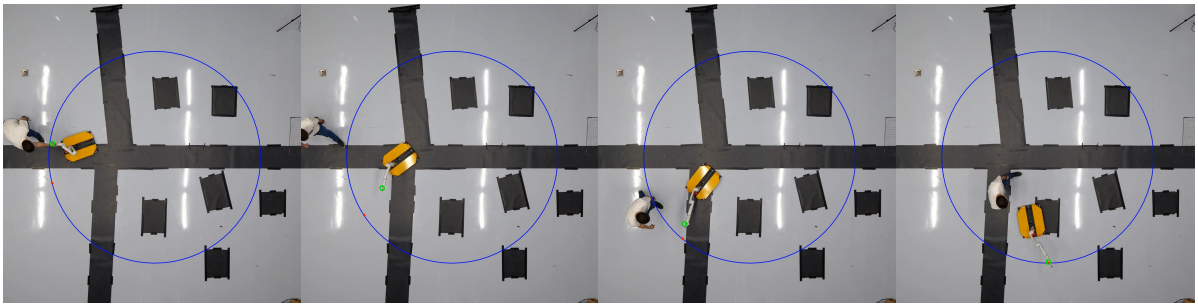
Figure 6.5a shows that the end-effector can be moved away from the preferred position, by pulling at one or multiple links of the arm. The base follows the end-effector as soon as it is moved away from its original position, enabling guidance of the whole robot using the arm. In Figure 6.5b, a push is given against the arm of the robot so that the end-effector moves backward relative to the original position. Since the base follows the end-effector, the base also moves back, whereafter the arm can settle back to its original position and the base stops moving. Figure 6.5c shows the behavior of the robot when someone bumps against the arm while passing by. The leg pushes the arm sidewards and as a result the base also makes a small sidewards movement. Then the arm can bounce back to its original position and the robot settles.

In the second part of the experiment, the target for the end-effector is not defined as a static position relative to the base, but as a moving target in the room. In this tracking mode, the impedance controller of the arm always tries to push the end-effector towards the target. The base still follows the end-effector, enabling the end-effector to move through the room tracking the target. A complete video of the experiment with the robot in tracking mode, captured from a top-down and close-up perspective, was published [45]. Figure 6.6 shows parts of the top-down video as image sequences.

In Figure 6.6a, the target (marked with red) starts to move and the end-effector follows it. As soon as the end-effector moves sidewards, the base starts to follow the end-effector, enabling the robot to follow the target through the room. Figure 6.6b shows the robot during interaction with the arm. The robot is compliant and therefore the user can hold the robot even though the target moves away. As soon as the arm is released, the end-effector is pushed into the direction of the target and the base starts to follow. The robot can catch up with the target and track it again.



(a) Without interaction, the impedance controller moves the end-effector to the moving target while the base follows the end-effector.



(b) The robot is compliant with interactions of the arm, but once released the end-effector tries to move back to the target.

Figure 6.6: The robot operating in tracking mode. The end-effector of the robot (green marker) follows the target (red marker) that is moving over the blue circle. The robot is compliant with interactions of the arm.

7

Summary and Conclusion

Compliant control on mobile manipulators is not a common control strategy. Yet there are many applications where compliance of robots could improve the safety of the robot and the user, especially when robots and humans share the same workspace. A main limitation of compliant control is the requirement of expensive force/torque sensors. Admittance control is only compliant with measurable forces, which are generally only the forces applied to the end-effector and measured using a force/torque sensor located in the wrist of the robot. Impedance control offers compliance but requires torque sensors in every joint since torque control is used.

This thesis focussed on the adaption of impedance control so that it can be used on current-controlled robots without torque sensors. In Chapter 2, a calibration method was designed that enables the robot to estimate the current/torque ratio and friction for each joint. With the method, it is also possible to correct errors in the center of mass (COM) locations for the links in the model of the robot. This is important for the performance since impedance control is completely model-dependent. Section 6.1 showed the consistency of the method through repetition of the calibration, where the COM locations are identified with a standard deviation of less than 2.5 mm, the ratios with a standard deviation of less than 0.4 % of their average value and the frictions with a standard deviation of less than 3 % of their average value.

Chapter 3 presented a current-based impedance controller, that uses the current/torque ratios achieved using the calibration. Since the performance of impedance control is highly dependent on the friction of the joints, a friction compensation strategy was designed and presented. Section 6.2 showed the performance of the impedance controller used on the Kinova arm, where the arm was compliant with external forces, while the same controller was able to track a target with a precision of about 5 mm when not disrupted by external forces. The outcome of the experiment also suggests a close similarity between the desired compliance and the achieved compliance, but further research with better measuring tools is required to conclude this with more certainty.

Limitations of the base complicate the implementation of compliant control on it. The omnidirectional wheels of the base experience significant friction and only velocity control is officially supported. Adaption of the driver software enabled the usage of voltage control, which is a first step towards current control. However, further adaptions required on the motor boards of the base were out of the scope of this project. Due to these limitations, the strategy of current-based impedance control using a calibration method as used for the arm was not possible. Instead, a controller for the base was developed that follows the movement of the end-effector of the arm, so that a limited form of whole-body compliance is still achieved.

Section 6.3 showed the results of the experiment where the compliant controller was used on the mobile manipulator. The robot was evaluated in guidance mode, where the user can manipulate the complete robot using interaction with the arm, and in tracking mode, where the robot tracks a moving target while being compliant with interactions. Videos show that both modes work well, although the robot requires some time to recover from a disturbance while in tracking mode. This could be improved by adapting the compliant parameters of the robot or by adding a high-level planner that defines an optimized path for the end-effector when it is far away from the target.

Overall, the results show that compliant control on a mobile manipulator without force/torque sensors is possible with decent performance. A calibration method like the one designed in this thesis can consistently estimate the ratios between the actuator's currents and the produced torques so that impedance control can be used on a current-controlled robot without torque feedback. It is unlikely that the performance is equally good as torque-based impedance control, simply due to the lack of feedback, but a comparison of the two methods on the same robot is required to exactly define the performance difference.

Due to the mentioned limitations of the base, impedance control was only implemented on the arm, where the base is following the end-effector. However, if the base would also feature current control, a similar calibration method could be designed for the base to estimate the current/torque ratios of the wheels. Therefore, the designed current-based impedance controller is expected to work on the bases of mobile manipulators as well, especially for bases having wheels with low friction. However, further research is required to confirm this theory.

References

[1] "RB-KAIROS+: Mobile robot to expand UR e-Series applications." (), [Online]. Available: https://robotnik.eu/kairos-mobile-robot/ (visited on 06/16/2023).

[2] F. Sherwani, M. M. Asad, and B. Ibrahim, "Collaborative Robots and Industrial Revolution 4.0 (IR 4.0)," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, Mar. 2020, pp. 1–5. DOI: 10.1109/ICETST49965.2020.9080724.

[3] M. Yang, E. Yang, R. C. Zante, M. Post, and X. Liu, "Collaborative mobile industrial manipulator: A review of system architecture and applications," in *2019 25th International Conference on Automation and Computing (ICAC)*, Sep. 2019, pp. 1–6. DOI: 10.23919/IConAC.2019.8895183.

[4] Z.-E. Chebab, J.-C. Fauroux, N. Bouton, Y. Mezouar, and L. Sabourin, "Autonomous Collaborative Mobile Manipulators: State of the Art," 2015.

[5] A. Zacharaki, I. Kostavelis, A. Gasteratos, and I. Dokas, "Safety bounds in human robot interaction: A survey," *Safety Science*, vol. 127, p. 104 667, Jul. 2020, ISSN: 09257535. DOI: 10.1016/j.ssci.2020.104667. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925753520300643 (visited on 06/05/2023).

[6] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics* (Springer Handbooks). Cham: Springer International Publishing, 2016, ISBN: 978-3-319-32550-7 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1. [Online]. Available: https://link.springer.com/10.1007/978-3-319-32552-1 (visited on 02/15/2023).

[7] T. Zhang, Q. Du, G. Yang, C.-y. Chen, C. Wang, and Z. Fang, "A Review of Compliant Control for Collaborative Robots," in *2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA)*, Aug. 2021, pp. 1103–1108. DOI: 10.1109/ICIEA51954.2021.9516193.

[8] A. Calanca, R. Muradore, and P. Fiorini, "A Review of Algorithms for Compliant Control of Stiff and Fixed-Compliance Robots," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 2, pp. 613–624, Apr. 2016, ISSN: 1941-014X. DOI: 10.1109/TMECH.2015.2465849.

[9] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge, UK: Cambridge University Press, 2017, 528 pp., ISBN: 978-1-107-15630-2 978-1-316-60984-2.

[10] M. Schumacher, J. Wojtusch, P. Beckerle, and O. Von Stryk, "An introductory review of active compliant control," *Robotics and Autonomous Systems*, vol. 119, pp. 185–200, Sep. 2019, ISSN: 09218890. DOI: 10.1016/j.robot.2019.06.009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889018307772 (visited on 04/25/2023).

[11] C. Ott, R. Mukherjee, and Y. Nakamura, "A Hybrid System Framework for Unified Impedance and Admittance Control," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 3, pp. 359–375, Jun. 1, 2015, ISSN: 1573-0409. DOI: 10.1007/s10846-014-0082-1. [Online]. Available: https://doi.org/10.1007/s10846-014-0082-1 (visited on 05/08/2023).

[12] A. Dietrich, X. Wu, K. Bussmann, *et al.*, "Practical consequences of inertia shaping for interaction and tracking in robot control," *Control Engineering Practice*, vol. 114, p. 104 875, Sep. 2021, ISSN: 09670661. DOI: 10.1016/j.conengprac.2021.104875. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0967066121001520 (visited on 04/14/2023).

[13] X. Wu, C. Ott, and A. Dietrich, "A Comparative Experimental Study of Multi-Tasking Tracking and Interaction Control on a Torque-Controlled Humanoid Robot," in *2022 American Control Conference (ACC)*, Jun. 2022, pp. 1933–1940. DOI: 10.23919/ACC53348.2022.9867784.

[14] "DLR - Institute of Robotics and Mechatronics - LRU." (), [Online]. Available: https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-11431#gallery/32387 (visited on 06/06/2023).

[15] "DLR - Institute of Robotics and Mechatronics - Rollin' Justin." (), [Online]. Available: https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-11427/#gallery/35411 (visited on 06/06/2023).

[16] "Robot TOMM." (), [Online]. Available: https://www.ce.cit.tum.de/en/ics/research/platforms/robot-tomm/ (visited on 06/06/2023).

[17] K. Bussmann, A. Dietrich, and C. Ott, "Whole-Body Impedance Control for a Planetary Rover with Robotic Arm: Theory, Control Design, and Experimental Validation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 910–917. DOI: 10.1109/ICRA.2018.8460533.

[18] A. Dietrich, T. Wimböck, and A. Albu-Schäffer, "Dynamic whole-body mobile manipulation with a torque controlled humanoid robot via impedance control laws," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 3199–3206. DOI: 10.1109/IROS.2011.6094445.

[19] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, "Reactive Whole-Body Control for Dynamic Mobile Manipulation,"

[20] A. Dietrich, K. Bussmann, F. Petit, *et al.*, "Whole-body impedance control of wheeled mobile manipulators," *Autonomous Robots*, vol. 40, no. 3, pp. 505–517, Mar. 1, 2016, ISSN: 1573-7527. DOI: 10.1007/s10514-015-9438-z. [Online]. Available: https://doi.org/10.1007/s10514-015-9438-z (visited on 04/25/2023).

[21] A. Dietrich, *Whole-Body Impedance Control of Wheeled Humanoid Robots* (Springer Tracts in Advanced Robotics). Cham: Springer International Publishing, 2016, vol. 116, ISBN: 978-3-319-40556-8 978-3-319-40557-5. DOI: 10.1007/978-3-319-40557-5. [Online]. Available: http://link.springer.com/10.1007/978-3-319-40557-5 (visited on 02/28/2023).

[22] S. Kim, K. Jang, S. Park, Y. Lee, S. Y. Lee, and J. Park, "Whole-body Control of Non-holonomic Mobile Manipulator Based on Hierarchical Quadratic Programming and Continuous Task Transition," in *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, Jul. 2019, pp. 414–419. DOI: 10.1109/ICARM.2019.8834269.

[23] W. Kim, P. Balatti, E. Lamon, and A. Ajoudani, "MOCA-MAN: A MObile and reconfigurable Collaborative Robot Assistant for conjoined huMAN-robot actions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 10 191–10 197. DOI: 10.1109/ICRA40945.2020.9197115.

[24] E. Lamon, M. Leonori, W. Kim, and A. Ajoudani, "Towards an Intelligent Collaborative Robotic System for Mixed Case Palletizing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 9128–9134. DOI: 10.1109/ICRA40945.2020.9196850.

[25] Q. Leboutet, E. Dean-León, and G. Cheng, "Tactile-based compliance with hierarchical force propagation for omnidirectional mobile manipulators," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov. 2016, pp. 926–931. DOI: 10.1109/HUMANOIDS.2016.7803383.

[26] Q. Leboutet, E. Dean-Leon, F. Bergner, and G. Cheng, "Tactile-Based Whole-Body Compliance With Force Propagation for Mobile Manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 330–342, Apr. 2019, ISSN: 1941-0468. DOI: 10.1109/TRO.2018.2889261.

[27] H. Xing, A. Torabi, L. Ding, *et al.*, "An admittance-controlled wheeled mobile manipulator for mobility assistance: Human–robot interaction estimation and redundancy resolution for enhanced force exertion ability," *Mechatronics*, vol. 74, p. 102 497, Apr. 2021, ISSN: 09574158. DOI: 10.1016/j.mechatronics.2021.102497. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957415821000076 (visited on 05/10/2023).

[28] J. Pankert and M. Hutter, "Perceptive Model Predictive Control for Continuous Mobile Manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6177–6184, Oct. 2020, ISSN: 2377-3766. DOI: 10.1109/LRA.2020.3010721.

[29] "ELEC 435- Textbook." (), [Online]. Available: https://www.ece.rice.edu/~jdw/435/435_html/textbook.html (visited on 01/31/2024).

[30] "Scipy.optimize.minimize — SciPy v1.12.0 Manual." (), [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html (visited on 02/22/2024).

[31] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, eabm6074, 2022. DOI: 10.1126/scirobotics.abm6074. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[32] *Kinovarobotics/ros_kortex*, Kinova Robotics, Dec. 7, 2023. [Online]. Available: https://github.com/Kinovarobotics/ros_kortex (visited on 12/08/2023).

[33] "Dingo-cpr/dingo." (), [Online]. Available: https://github.com/dingo-cpr/dingo (visited on 12/08/2023).

[34] *Kinovarobotics/kortex*, Kinova Robotics, Nov. 16, 2023. [Online]. Available: https://github.com/Kinovarobotics/kortex (visited on 12/08/2023).

[35] *Clearpathrobotics/puma_motor_driver*, clearpathrobotics, Aug. 9, 2023. [Online]. Available: https://github.com/clearpathrobotics/puma_motor_driver (visited on 12/08/2023).

[36] W. Jakob, J. Rhinelander, and D. Moldovan, *Pybind11 – seamless operability between c++11 and python*, https://github.com/pybind/pybind11, 2017.

[37] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[38] J. Hoffstadt, *Hoffstadt/DearPyGui*, Dec. 8, 2023. [Online]. Available: https://github.com/hoffstadt/DearPyGui (visited on 12/08/2023).

[39] "Ocornut/imgui: Dear ImGui: Bloat-free Graphical User interface for C++ with minimal dependencies." (), [Online]. Available: https://github.com/ocornut/imgui (visited on 12/08/2023).

[40] J. Carpentier, F. Valenza, N. Mansard, *et al.*, *Pinocchio: Fast forward and inverse dynamics for poly-articulated systems*, https://stack-of-tasks.github.io/pinocchio, 2015–2021.

[41] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi - A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, no. 11, pp. 1–36, Mar. 2019. DOI: 10.1007/s12532-018-0139-4.

[42] *Pin3x-jnrh2023: A fast and flexible implementation of Rigid Body Dynamics algorithms and their analytical derivatives - TEMPORARY PACKAGE*, version 2.9.2. [Online]. Available: https://pypi.org/project/pin3x-jnrh2023/ (visited on 12/08/2023).

[43] "Journées Nationales de la Robotique Humanoïde - Sciencesconf.org." (), [Online]. Available: https://jnrh2023.sciencesconf.org/ (visited on 12/08/2023).

[44] "Vicon | Award Winning Motion Capture Systems," Vicon. (), [Online]. Available: https://www.vicon.com/ (visited on 01/24/2024).

[45] J. de Wolde, *Videos of the experiments with the robot.* 2024. [Online]. Available: https://drive.google.com/drive/folders/1QnneiJkTgHDs0_Cm5SYScZbtXXeFjVvJ.

[46] "Robotics: Modelling, Planning and Control," in *Robotics: Modelling, Planning and Control*, ser. Advanced Textbooks in Control and Signal Processing, B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Eds., London: Springer, 2009, pp. 363–405, ISBN: 978-1-84628-642-1. DOI: 10.1007/978-1-84628-642-1_9. [Online]. Available: https://doi.org/10.1007/978-1-84628-642-1_9 (visited on 02/20/2023).

[47] *Cartesian Impedance Control of Redundant and Flexible-Joint Robots* (Springer Tracts in Advanced Robotics). Berlin, Heidelberg: Springer, 2008, vol. 49, ISBN: 978-3-540-69253-9 978-3-540-69255-3. DOI: 10.1007/978-3-540-69255-3. [Online]. Available: http://link.springer.com/10.1007/978-3-540-69255-3 (visited on 02/28/2023).

[48] A. Dietrich, C. Ott, and A. Albu-Schäffer, "An overview of null space projections for redundant, torque-controlled robots," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1385–1400, Sep. 1, 2015, ISSN: 0278-3649. DOI: 10.1177/0278364914566516. [Online]. Available: https://doi.org/10.1177/0278364914566516 (visited on 04/20/2023).

A

Lagrange Formulation

This appendix shows the derivation of a dynamic model of a robot using the Lagrange formulation, based on the book “Robotics: Modelling, Planning and Control” [46]. The Lagrange equation is defined as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = \boldsymbol{\xi} \tag{A.1}$$

where the Lagrangian $\mathcal{L} = \mathcal{T} - \mathcal{U}$ is defined as the difference between the kinetic energy $\mathcal{T}$ and the potential energy $\mathcal{U}$. $\boldsymbol{q}$ is a vector of generalized coordinates and $\boldsymbol{\xi}$ is a vector of generalized forces on the system. Substituting the kinetic and potential energy in (A.1) results in:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{T}}{\partial \dot{\boldsymbol{q}}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{U}}{\partial \dot{\boldsymbol{q}}} - \left(\frac{\partial \mathcal{T}}{\partial \boldsymbol{q}} - \frac{\partial \mathcal{U}}{\partial \boldsymbol{q}}\right) = \boldsymbol{\xi} \tag{A.2}$$

A body with mass m and velocity v has a kinetic energy of $\frac{1}{2}mv^2$ while a body with inertia I rotating around its center of mass with angular velocity ω has a kinetic energy of $\frac{1}{2}I\omega^2$. The total kinetic energy of a complete system can be expressed as:

$$\mathcal{T} = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{A.3}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix of the system. Using this definition for $\mathcal{T}$ while calculating the first term of (A.2) gives:

$$\frac{\partial \mathcal{T}}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} \qquad\qquad \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{T}}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{M}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{A.4}$$

Since the potential energy $\mathcal{U}$ only depends on the configuration of the system $\boldsymbol{q}$ and not on the velocities $\dot{\boldsymbol{q}}$, the second term of (A.2) will be zero. Substituting this and the outcome of (A.4) in (A.2) gives:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{M}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \left(\frac{\partial}{\partial \boldsymbol{q}}(\frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}) - \frac{\partial \mathcal{U}}{\partial \boldsymbol{q}}\right) = \boldsymbol{\xi} \tag{A.5}$$

This can be simplified to:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}) = \boldsymbol{\xi} \tag{A.6}$$

where $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is a vector of Coriolis and centrifugal torques and $\boldsymbol{\tau}_g(\boldsymbol{q})$ is a vector of torques due to gravity:

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \dot{\boldsymbol{M}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{q}}(\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}) \qquad\qquad \boldsymbol{\tau}_g(\boldsymbol{q}) = \frac{\partial \mathcal{U}}{\partial \boldsymbol{q}} \tag{A.7}$$

A.1. Derivation of the inertia matrix

To derive $M(q)$ so that (A.3) holds, the COM locations of the links and corresponding masses and inertias are required. If p is the vector of all the COM positions, M_m is a diagonal matrix of all the corresponding masses and $v = \dot{p}$ is the vector of all the translational and rotational velocities of the system, the kinetic energy of the system can be expressed as:

$$\mathcal{T} = \frac{1}{2} v^T M_m v \tag{A.8}$$

One can derive a Jacobian matrix $J(q)$ which contains the COM positions, partially derived with respect to q:

$$J(q) = \frac{\partial p}{\partial q}$$

The relation between the cartesian velocities v and the joint velocities $\dot{q}$ is as:

$$v = J(q)\dot{q}$$

Substituting this in (A.8) gives:

$$\mathcal{T} = \frac{1}{2} \dot{q}^T J^T(q) M_m J(q) \dot{q} \tag{A.9}$$

Comparing (A.9) with (A.3) gives:

$$M(q) = J^T(q) M_m J(q) \tag{A.10}$$

Example: derivation of dynamics for a two-link manipulator in 2D.

Part 1: inertia matrix

A two-link manipulator is defined with angles q_1 and q_2, links of length l, COM locations at the center of the link and with the diagonal matrix M_m containing the masses and inertias that relate to the cartesian accelerations, for both links :

$$M_m = \text{diag}(m) \qquad m = \begin{bmatrix} m_1 & m_1 & I_1 & m_2 & m_2 & I_2 \end{bmatrix}^T$$

To derive the inertia matrix $M(q)$ of the dynamic model, the COM locations in cartesian space are required and defined as:

$$x_1 = \cos(q_1) l/2 \qquad y_1 = \sin(q_1) l/2$$
$$x_2 = \cos(q_1) l + \cos(q_1 + q_2) l/2 \qquad y_2 = \sin(q_1) l + \sin(q_1 + q_2) l/2$$

The Jacobian matrix that maps between all the possible COM velocities in cartesian space v and the joint velocities $\dot{q}$ is defined as:

$$J(q) = \begin{bmatrix} -\frac{l \sin(q_1)}{2} & 0 \\ \frac{l \cos(q_1)}{2} & 0 \\ 1 & 0 \\ -\frac{l \sin(q_1+q_2)}{2} - l \sin(q_1) & -\frac{l \sin(q_1+q_2)}{2} \\ \frac{l \cos(q_1+q_2)}{2} + l \cos(q_1) & \frac{l \cos(q_1+q_2)}{2} \\ 0 & 1 \end{bmatrix} \qquad v = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{q}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{q}_2 \end{bmatrix} = J(q)\dot{q}$$

The inertia matrix $M(q)$ can be defined using the defintion from (A.10):

$$M(q) = J^T(q) M_m J(q) = \begin{bmatrix} I_1 + \frac{l^2 m_1}{4} + l^2 m_2 \cos(q_2) + \frac{5 l^2 m_2}{4} & \frac{l^2 m_2 \cdot (2\cos(q_2)+1)}{4} \\ \frac{l^2 m_2 \cdot (2\cos(q_2)+1)}{4} & I_2 + \frac{l^2 m_2}{4} \end{bmatrix}$$

A.2. Derivation of the centrifugal/Coriolis matrix

The product of the matrix $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the generalized velocities $\dot{\boldsymbol{q}}$ was defined in (A.7) as:

$$\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \dot{\boldsymbol{M}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{q}}(\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}})$$

Due to the velocity cross terms, there is no unique matrix $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ that fulfills this product. The partial derivative of $\boldsymbol{M}(\boldsymbol{q})$ with respect to $\boldsymbol{q}$ in the second term of the product results in a vector of matrices. The most common way to factor $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is based on tensor analysis, which results in an expression for the individual elements of $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ as:

$$c_{ij} = \sum_{k=1}^{n} c_{ijk}\dot{q}_k \qquad c_{ijk} = \frac{1}{2}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i}\right) \tag{A.11}$$

where c_{ijk} are called Christoffel symbols and M_{ij}, M_{ik}, M_{jk} are individual elements of $\boldsymbol{M}(\boldsymbol{q})$.

Example: derivation of dynamics for a two-link manipulator in 2D.

Part 2: centrifugal/Coriolis matrix

The matrix of Coriolis and centrifugal torques $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be derived using the Christoffel symbols as defined in (A.11):

$$c_{ij} = \sum_{k=1}^{2} \frac{1}{2}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i}\right)\dot{q}_k \quad \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{bmatrix} -\frac{L^2 m_2 \sin(q_2)\dot{q}_2}{2} & \frac{L^2 m_2(-\dot{q}_1 - \dot{q}_2)\sin(q_2)}{2} \\ \frac{L^2 m_2 \sin(q_2)\dot{q}_1}{2} & 0 \end{bmatrix}$$

Example: derivation of dynamics for a two-link manipulator in 2D.

Part 3: gravitational vector

The potential energy of the system is the product of the height vector $\boldsymbol{h}$ containing the heights of all bodies multiplied with the corresponding masses defined in $\boldsymbol{m_p}$ multiplied with the gravitational constant g:

$$\boldsymbol{h} = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T \qquad \boldsymbol{m_p} = \begin{bmatrix} m_1 & m_2 \end{bmatrix}^T \qquad \mathcal{U} = \boldsymbol{m_p}^T g \boldsymbol{h} = g(m_1 y_1 + m_2 y_2)$$

Finally, it is possible to define the vector containing gravitational torques $\boldsymbol{\tau}_g(\boldsymbol{q})$ as defined in (A.7):

$$\boldsymbol{\tau}_g(\boldsymbol{q}) = \frac{\partial \mathcal{U}}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{lg(m_1 \cos(q_1) + m_2(\cos(q_1 + q_2) + 2\cos(q_1)))}{2} \\ \frac{lgm_2 \cos(q_1 + q_2)}{2} \end{bmatrix}$$

B

Operational Space Formulation

These derivations of the operational space formulation are based on the book *Cartesian Impedance Control of Redundant and Flexible-Joint Robots* [47]. Recall that the dynamics of a system expressed in joint space are:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}) + \boldsymbol{\tau}_f = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext} \tag{3.1}$$

and that the Jacobian relations were defined as:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} & \ddot{\boldsymbol{x}} &= \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} & \boldsymbol{f} &= \boldsymbol{J}^{-T}(\boldsymbol{q})\boldsymbol{\tau} \\ \dot{\boldsymbol{q}} &= \boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}} & \ddot{\boldsymbol{q}} &= \boldsymbol{J}^{-1}(\boldsymbol{q})(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}) & \boldsymbol{\tau} &= \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f} \end{aligned} \tag{3.2}$$

Substituting the equations for $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ and $\boldsymbol{\tau}$ from (3.2) in Equation (3.1) gives:

$$\boldsymbol{M}(\boldsymbol{q})\left(\boldsymbol{J}^{-1}(\boldsymbol{q})(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}})\right) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\left(\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}}\right) + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_g + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_f = \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f} + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_{ext}$$

Writing the matrices without their dependent variables to increase readability, rearranging and pre-multiplying with $\boldsymbol{J}^{-T}(\boldsymbol{q})$ gives:

$$\begin{aligned} \boldsymbol{M}\left(\boldsymbol{J}^{-1}(\boldsymbol{q})(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}})\right) + \boldsymbol{C}\left(\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}}\right) + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_g + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_f &= \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f} + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_{ext} \\ \boldsymbol{M}\boldsymbol{J}^{-1}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \left(\boldsymbol{C} - \boldsymbol{M}\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{J}}\right)\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}} + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_g + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_f &= \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f} + \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f}_{ext} \\ \underbrace{\boldsymbol{J}^{-T}\boldsymbol{M}\boldsymbol{J}^{-1}(\boldsymbol{q})}_{\boldsymbol{\Lambda}}\ddot{\boldsymbol{x}} + \underbrace{\boldsymbol{J}^{-T}\left(\boldsymbol{C} - \boldsymbol{M}\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{J}}\right)\boldsymbol{J}^{-1}(\boldsymbol{q})}_{\boldsymbol{\mu}}\dot{\boldsymbol{x}} + \boldsymbol{f}_g + \boldsymbol{f}_f &= \boldsymbol{f} + \boldsymbol{f}_{ext} \end{aligned}$$

This can be written, now again with the dependent variables for the matrices, as:

$$\boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{\mu}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{f}_g(\boldsymbol{q}) + \boldsymbol{f}_f = \boldsymbol{f} + \boldsymbol{f}_{ext} \tag{3.3}$$

with the inertia matrix in operational space defined as:

$$\boldsymbol{\Lambda}(\boldsymbol{q}) = \boldsymbol{J}^{-T}(\boldsymbol{q})\boldsymbol{M}(\boldsymbol{q})\boldsymbol{J}^{-1}(\boldsymbol{q}) \tag{3.4}$$

and the Coriolis/centrifugal matrix $\boldsymbol{\mu}(\boldsymbol{q},\dot{\boldsymbol{q}})$ in operation space defined as:

$$\boldsymbol{\mu}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{J}^{-T}(\boldsymbol{q})\left(\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{M}(\boldsymbol{q})\boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\right)\boldsymbol{J}^{-1}(\boldsymbol{q}) \tag{3.5}$$

C

Nullspace projection

These derivations are based on “An Overview of Null Space Projections for Redundant, Torque-Controlled Robots” [48]. Recall the definition of the generalized inverse which can be used as a solution for matrices that are not invertible because they are non-square:

$$\boldsymbol{J}^{W+}(\boldsymbol{q}) = \boldsymbol{W}^{-1}\boldsymbol{J}^{T}(\boldsymbol{q})\left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{W}^{-1}\boldsymbol{J}^{T}(\boldsymbol{q})\right)^{-1} \tag{3.6}$$

For better readability, $\boldsymbol{J}^{-1}(\boldsymbol{q})$ will still be used as the symbol of the inverse of the Jacobian $\boldsymbol{J}(\boldsymbol{q})$, but note that this is actually the generalized inverse defined in Equation (3.6) for non-square matrices. It is possible to define a nullspace matrix $\boldsymbol{N}_i(\boldsymbol{q})$ for a task i as:

$$\boldsymbol{N}_i = \boldsymbol{N}_{i-1}\left(\boldsymbol{I} - \boldsymbol{J}_{i-1}^{T}\boldsymbol{J}_{i-1}^{-T}\right) \qquad \boldsymbol{N}_1 = \boldsymbol{I} \tag{C.1}$$

where $\boldsymbol{I}$ is the identity matrix.

C.1. Static consistency

A projection with this nullspace matrix ensures static consistency for any task k, with respect to any higher order task $j < k$. The effective force $\boldsymbol{f}_j$ as result of the torques $\boldsymbol{\tau_k}$ from task k after projecting with nullspace matrix $\boldsymbol{N}_k$ is:

$$\begin{aligned}\boldsymbol{f}_j &= \boldsymbol{J}_j^{-T}\boldsymbol{N}_k\boldsymbol{\tau}_k \\ &= \boldsymbol{J}_j^{-T}\boldsymbol{N}_{k-1}\left(\boldsymbol{I} - \boldsymbol{J}_{k-1}^{T}\boldsymbol{J}_{k-1}^{-T}\right)\boldsymbol{\tau}_k \\ &= \boldsymbol{J}_j^{-T}\boldsymbol{N}_{k-2}\left(\boldsymbol{I} - \boldsymbol{J}_{k-2}^{T}\boldsymbol{J}_{k-2}^{-T}\right)\left(\boldsymbol{I} - \boldsymbol{J}_{k-1}^{T}\boldsymbol{J}_{k-1}^{-T}\right)\boldsymbol{\tau}_k \\ &\;\;\vdots\end{aligned} \tag{C.2}$$

At some point, the first nullspace matrix in the equation will reach the level of j, giving:

$$\boldsymbol{f}_j = \boldsymbol{J}_j^{-T}\boldsymbol{I}\left(\boldsymbol{I} - \boldsymbol{J}_j^{T}\boldsymbol{J}_j^{-T}\right)\cdots\boldsymbol{\tau}_k = \left(\boldsymbol{J}_j^{-T} - \boldsymbol{J}_j^{-T}\right)\cdots\boldsymbol{\tau}_k = \boldsymbol{0} \tag{C.3}$$

C.2. Dynamic consistency

Appendix C.1 shows that lower-order tasks do not influence higher-order tasks by using the defined nullspace projection, but so far this only holds in equilibrium. Recall the definition of the dynamics in operational space:

$$\boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{\mu}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{f}_g(\boldsymbol{q}) + \boldsymbol{f}_f = \boldsymbol{f} + \boldsymbol{f}_{ext} \tag{3.3}$$

which can be rewritten to:

$$\begin{aligned}\ddot{\boldsymbol{x}} &= \boldsymbol{\Lambda}^{-1}(\boldsymbol{q})\left(\boldsymbol{f} + \boldsymbol{f}_{ext} - \boldsymbol{\mu}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{f}_g(\boldsymbol{q}) + \boldsymbol{f}_f\right) \\ &= \left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{J}^{T}(\boldsymbol{q})\right)\left(\boldsymbol{f} + \boldsymbol{f}_{ext} - \boldsymbol{\mu}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{f}_g(\boldsymbol{q}) + \boldsymbol{f}_f\right)\end{aligned} \tag{C.4}$$

The effect of $\boldsymbol{f}$ on $\ddot{\boldsymbol{x}}$ is:

$$\boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{\tau} \tag{C.5}$$

If $\boldsymbol{\tau}$ contains a secondary task i, the effect of the torques $\boldsymbol{\tau}_i$ after projecting should be zero, to ensure dynamic consistency:

$$\boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{N}_i\boldsymbol{\tau}_i = \boldsymbol{0} \tag{C.6}$$

By substituting the nullspace matrices for all levels, eventually the first-level nullspace matrix definition from Equation (C.1) will be reached, which will give:

$$\boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\left(\boldsymbol{I} - \boldsymbol{J}^{T}(\boldsymbol{q})\boldsymbol{J}^{-T}(\boldsymbol{q})\right) = \boldsymbol{0} \tag{C.7}$$

Remember that $\boldsymbol{J}^{-T}(\boldsymbol{q})$ is actually the generalized inverse defined in Equation (3.6). Substituting this in Equation (C.7) without the depending variables to increase readability gives:

$$\begin{aligned}
&\boldsymbol{J}\boldsymbol{M}^{-1}\left(\boldsymbol{I} - \boldsymbol{J}^{T}\left(\boldsymbol{W}^{-1}\boldsymbol{J}^{T}\left(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^{T}\right)^{-1}\right)^{T}\right) = \boldsymbol{0} \\
&\boldsymbol{J}\boldsymbol{M}^{-1} - \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^{T}\left(\boldsymbol{W}^{-1}\boldsymbol{J}^{T}\left(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^{T}\right)^{-1}\right)^{T} = \boldsymbol{0} \\
&\boldsymbol{J}\boldsymbol{M}^{-1} - \underbrace{\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^{T}\left(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^{T}\right)^{-1}}_{\boldsymbol{I} \text{ if } \boldsymbol{W}=\boldsymbol{M}}\boldsymbol{J}\boldsymbol{W}^{-1} = \boldsymbol{0}
\end{aligned} \tag{C.8}$$

which holds when $\boldsymbol{W} = \boldsymbol{M}(\boldsymbol{q})$. Therefore, when a generalized inverse is required, the generalized inverse defined in Equation (3.6) is used with $\boldsymbol{W} = \boldsymbol{M}(\boldsymbol{q})$ to ensure both static and dynamic consistency.