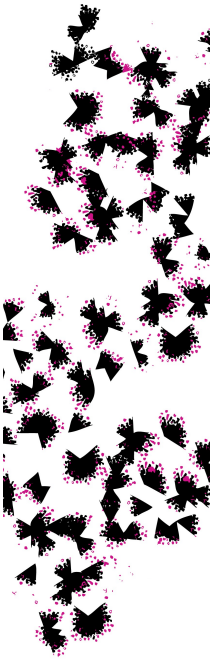

Integration of a passivity layer with a dynamics compensator on a teleoperated KUKA robot arm

C. (Christophe) van der Walt

BSc Report

Committee:
Dr.ir. D. Dresscher
Dr. H.K. Hemmes

July 2018

019RAM2018
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Integration of a passivity layer with a dynamics compensator in a teleoperated KUKA robot arm

Christophe van der Walt

July 2, 2018

Abstract

This thesis focuses on inertia compensation for tele-operated Serial-Chain Manipulators with haptic feedback. It also deals with how a passivity layer needs to be integrated into such a set-up. The idea behind compensating for inertia, here, is to reduce the inertia-effects felt by an operator, as these "fight" any accelerations he or she imparts onto the operator, reducing the operator's ability to carry out dexterity-intensive tasks.

The concept of model inversion was used to design an inertia-compensating controller for such systems. The resulting controller performs as it should, allowing a manipulator to exhibit dynamics of a similar system with arbitrarily reduced mass.

When this control scheme was applied to a haptic tele-operation set-up, the force fed back to the operator was indeed reduced, on average. Also, it was theorised from the results that a passivity layer that works on a similar, uncompensated manipulator can be integrated into the compensated set-up with relatively little changes to its tuning parameters.

However, the results also seem to show that compensating for friction instead of inertia may be more of use if used to operate a KUKA LWR4+, for example.

Contents

1 Introduction

1.1 Context

Humans, as a species, are particularly adept at manipulating the world around them, through a sizeable array of accurate sensing and actuating capabilities. No fully automated machine can handle objects in such a delicate, precise, and versatile manner as humans can. A robot can either be versatile, but then do an imperfect job of opening doors and picking up boxes, or one can make a robot very good at picking up boxes, but then be able to do nothing else. This makes humans still ideally suited to such tasks as disaster relief, for example. In such applications, humans make full use of their dexterity, as well as decision-making capabilities, to save people from under rubble, or dispose of nuclear waste.

Indeed, in some cases, this works very well. However, certain disaster scenarios require certain additional capabilities that humans do not possess, such as increased strength, or the ability to survive indefinitely underwater, or even to subsist in highly radioactive environments. So, with this in mind, the ideal solution would be to create a disaster relief system with the dexterity of human beings, but preserving the potential strength and hardiness of robots.

A proposed solution to this involves controlling the robot using a haptic feedback control set-up. In such a set-up, not only does a controller exert a force on the controlled robot, but so too does the robot exert a force back on the controller, based on how the robot interacts with the environment. This would, for instance, allow for a user to manipulate a robot arm as if it were his own. This set-up allows for both the use of the user's dexterity, as well a robot's robustness and strength.

1.2 Problem Statement

A proposed robot for such applications is being worked on at i-Botics, a group within the Robotics and Mechatronics research group at the University of Twente. It is comprised of a seven degree of freedom robot arm, allowing the gripper on the end to occupy any position and orientation within its workspace, as well as a robust motorised undercarriage.

This assignment focuses on haptic feedback control of the arm specifically. It is controlled via the set-up shown on figure 1.1:

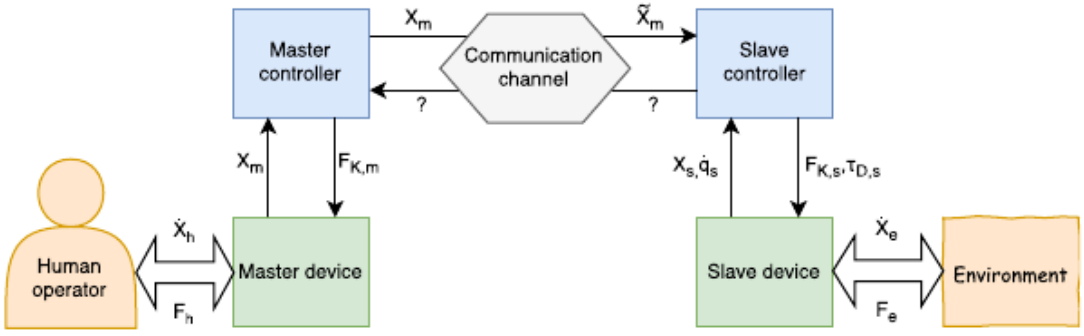

Figure 1.1: Basic Overview of a haptic control system [13]

Figure 1.2: Omega 7 haptic controller ([13])

Here, the master device is an omega 7 haptic controller (see figure 1.2), with control over the full 7 degrees of freedom the robot does. The master controller is what determines what to communicate to the slave based on the omega 7 input, as well as what force to exert on the master device, based on what the slave controller is sending it. The slave controller computes what to send the master controller based on the dynamics of the arm, as well as what movement commands to give the slave device, based on what the master controller is sending. The slave device is the robot arm, that carries out the whims of a human operator. The communication channel, here, is a wireless interface with all the incurring time delays involved in such a system.

Inside the slave controller, it should be noted that between the rest of the slave controller and the slave device, there is an entity called a passivity layer. This essentially monitors all the energy into the system and makes sure nothing more than that is used by the robot, making the whole system passive and thus ensuring its stability.

One of the current limitations of this is that a user can feel the mass of the robot and the friction in its bearings, seeing as the system feeds back the dynamics of the robot to the user. This makes controlling a manipulator in a haptic feedback set-up rather sluggish if the operator tries to change the velocity of the hand quickly. The operator essentially feels the inertia "fighting" this change. The objective of this assignment is to remove these inertia effects. As such, the main problem dealt with in the research is the following:

How would one design an inertia-compensating controller and integrate it with a passivity layer in a haptic teleoperation set-up?

The first logical aspect of this to look into is the matter of what an inertia-compensating controller is, and how one would realise such a controller, practically speaking. This is the topic of a first part that essentially answers the *"How would one design an inertia-compensating controller"* portion of the question. The answer to this is presented both in the most general sense, as well as through examples, and is validated using simulations. Only then can one start looking at integrating this inertia-compensating controller into the teleoperation set-up. This is the subject of a second part. Results are shown both with and without a passivity layer, to show the effects of inertia-compensation on a haptic teleoperation set-up, as well as to show that the passivity layer does indeed perform as expected.

1.3 Related Work

On the subject of inertia compensation, there have been attempts to implement it on robots successfully. [2] and [4] indeed managed to do so. However, the former was for a 1-DOF robot, and the latter does not actually clarify how it is achieved.

There is already some research into and production of robots helpful in disaster relief ([1] & [10]). Some research has also been made into disaster relief robots that are specifically haptic-feedback controlled ([14] uses a haptic feedback controlled bipedal locomotion). However, no robots were found that use the stability and robustness of a wheel or track-based locomotion and combine it with the capacity for large force outputs and fine movements of a haptic-feedback controlled robot arm.

There have, however, been implementations of passivity layers on tele-operated manipulator set-ups. The most important one of these was done by [6] on a 1-DOF system. Nevertheless, implementations of these on higher degree-of-freedom manipulators have been done subsequently, for example by [13]. What seems to be hard to find in literature, additionally, are inertia-compensators within haptic feedback set-ups.

2 Background

This chapter explores some of the concepts necessary to understanding the control of serial robots and teleoperation. A first part deals with the specifics of operating serial manipulators over time-delayed communication channels. Further on, a basis for modelling the dynamics of serial robots is detailed. A final part lists the relevant hardware related to the project.

2.1 Haptic Control of Teleoperated Manipulators

The basics of haptic control over delayed communication channels was presented in the introduction. However, certain technical aspects of haptics and teleoperation are touched upon here, namely transparency, Impedance Control and stability.

2.1.1 Transparency

Transparency is a quality of a haptic teleoperating control system to convey a sense of direct interaction with the environment ([6]). Maximising transparency is then, by definition, the goal of such control systems.

There is no means of quantifying transparency, but it can be used to loosely compare different control systems' effectiveness. It is also used when talking about passivity layers, hence why it is being mentioned here.

2.1.2 Impedance Control

The main difference between impedance control and classical control strategies is that impedance control involves controlling through establishing a dynamic relationship between the two, with a given equilibrium.

Often used in teleoperation is the establishing of a virtual spring between the end-effectors of the master and the slave. This then involves exerting the following wrench on the end-effector of the slave:

$$\begin{bmatrix} \left(m_k^t\right)^T \\ \left(f_k^t\right)^T \end{bmatrix}^T = \begin{bmatrix} K_0 & K_c \\ K_c^T & K_t \end{bmatrix} \begin{bmatrix} \delta\theta_t^d \\ \delta p_t^d \end{bmatrix} \tag{2.1}$$

Here, $\delta\theta$ and δp represent the differences in orientation and position of the master and slave (respectively). In practice, implementing this this involves computing the three co-stiffness matrices associated with K_0, K_c and K_t, given by:

$$G_x = 0.5\,tr\left(K_x\right) I - K_x \qquad , \qquad x = t, 0, c \tag{2.2}$$

Then, it can be proven ([3]) that this wrench can be expressed as:

$$\left(\tilde{m}_k^t\right) = -2as\left(G_0 R_t^d\right) - as\left(G_t R_d^t \tilde{p}_t^d \tilde{p}_t^d R_t^d\right) - 2as\left(G_c \tilde{p}_t^d R_t^d\right) \tag{2.3}$$

$$\left(\tilde{f}_k^t\right) = -R_d^t as\left(G_t \tilde{p}_t^d\right) - as\left(G_t R_d^t \tilde{p}_t^d R_t^d\right) - 2as\left(G_c R_t^d\right) \tag{2.4}$$

Here, R_d^t is the rotation matrix between frames fixed to the master and slave's end effector, and $\tilde{p}_d^t$ is the position difference between these two frames. The $as()$ function is one that returns the antisymmetric part of a square matrix.

The impedance controller can then be tuned by adjusting the three stiffness matrices. For a typical 6D Cartesian spring, K_c can be made 0, and K_0 and k_t are diagonal matrices.

2.1.3 Stability and Passivity Layers

For linear systems, it is quite easy to evaluate stability through linear methods. This becomes more of a challenge in systems with non-linear dynamics like robot arms. If one then considers delays in a control system for a robot arm (which are by definition present in teleoperation), computing such metrics as delay margins becomes increasingly difficult. Consequently, so too does designing stabilising controllers for such cases.

When determining stability is difficult, considering a system's passivity becomes a good alternative. A passive system is simply a system that only conserves or dissipates its internal energy, and therefore can not produce any of its own. Passivity guarantees stability in a physical system. Some systems are intrinsically passive, but certain active systems (non-passive systems) can be made passive by artificial means.

One such artificial means is a passivity layer. This essentially monitors the energy put into the system in the time domain. It then guarantees that any potentially unstable components in the system can only draw from the energy already put into the system. This guarantees the passivity of the whole system, and thus its stability as well. The exact working of such a passivity layer, as well as how it transmits energy to various actuators in the system, is described hereafter.

The idea is to divide the system into all of its discrete subsystems, each with its own amount of virtual "internal energy". Here, these subsystems are a master controller, as well as a slave controller. This virtual storage is referred to as an energy tank. The levels of the tanks can only change if they exchange energy with another tank, or exchange energy with the environment. Whenever an exchange with the environment is performed, an energy-sufficiency check is performed, and the exchange is performed or not on the basis of the outcome of the check.

Energy Tanks

Let us now take a look at how these checks are performed.

Assumed here is that, at the interface with the environment, for any actuator, a force τ can be imposed, and a joint angle q can be read out. Since passivity layers are implemented on digital processing devices, all computation is done in the discrete time domain. This means that, knowing an input force and an incurred velocity, that an energy exchange ΔH_I at a time instance k can be expressed as: ([6])

$$\Delta H_I(k) = \int_{(k-1)\Delta T}^{k\Delta T} \tau(k)\,\dot{q}(k)\,dt \tag{2.5}$$

Seeing as imposed torque cannot change during the time step in digital signals, this can be rewritten as:

$$\Delta H_I(k) = \tau(k)\left(q(k) - q(k-1)\right) \tag{2.6}$$

$$\Delta H_I(k) = \tau(k)\,\Delta q(k) \tag{2.7}$$

One can then add this $\Delta H_I(k)$ to the energy level in the tank H_T, and if the result returns a value below 0, the control action can not be carried out. An upper limit is also made by dividing the energy in the tank by the position difference, which saturates the torque so that the tank level never actually goes below 0. This guarantees stability, as an actuator, by construction, can not output any more energy than is put into the system.

The level of these tanks can also change if energy is sent to another tank. If this occurs, the amount of energy to be sent in subtracted from the sender's tank, and is sent through the communication channel to the receiver's tank. Because of the delays in the communication channel, the channel stores this virtual energy, but [6] proves that the channel, too, is passive.

The protocols for transferring energy between tanks are varied. The simplest one involves making the tanks at both the slave and master send a fraction β of their level to the other. It can be proven that, assuming no interaction with the environment for an extended period, this makes both tank-levels converge to the same value. This ensures an equal distribution of energy between the two tanks.

There are certain limitations to this technique. Namely, for high communication delays and high values of β, the energetic capacity of the communication channel becomes high, which is a problem if a controller needs to draw a high enough amount of energy. Solutions to this problem need to be made on a case by case basis, as making this performance better depends on the tasks to be executed by the haptic control system, and the hardware interacting with the environment.

Tank Level Controllers

What remains to be discussed is how energy can enter the system. Of course, the user puts energy into the system by actuating the controller. Yet, it is possible to envision scenarios in which the energy put in during actuation of the master is such that carrying out commands on the slave's end becomes difficult.

A method of sourcing more energy is by using a Tank Level Controller (TLC). The TLC exerts relatively small torques on the actuators of the slave equal to: ([6])

$$\tau_{TLC}(k) = -d(k)\,\dot{q}_m(k) \tag{2.8}$$

, where $\dot{q}_m$ are the velocities of the slave actuators. Here:

$$d(k) = \begin{cases} \alpha\left(H_D - H_M(k+1)\right) & , \quad \text{if } H_M(k+1) < H_D \\ 0 & , \quad \text{otherwise} \end{cases} \tag{2.9}$$

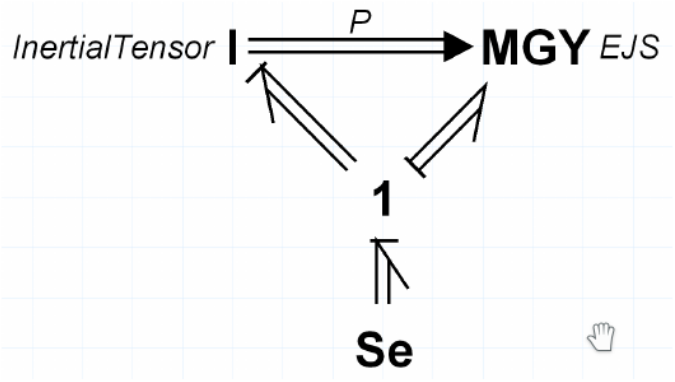

Figure 2.1: Bond graph of a rigid body in 3D space ([5])

, where H_D is a tunable threshold value, α is a tunable gain parameter, and H_M is the tank level of the master controller.

This means that the TLC exerts a modulated viscous damping force on the user. The user then compensates for it by counteracting this with a force approximately equal to the TLC force, putting energy into the master controller's energy tank. Note that this actuation performed by the TLC does not draw energy from the master controller's energy tank. However, this extra energy used by the controller cancels out with the energy added by the user (if τ_{TLC} is small enough). So, in the end the effect this has is to simply add the extra energy the user has to expend into the system.

2.2 Rigid Body Dynamics

To be able to model the motion of a robot arm, you need a dynamic model of the apparatus. There are many ways to achieve this. However, in this paper, a bond-graph model is used as it is quite flexible to changes in the dynamics to be modeled, and power exchanges can be easily identified and monitored, which is important when considering that this needs to be implemented with a passivity layer.

We start here with the equation derived in [12]:

$$I^k \dot{T}_i^{k,0} = \begin{bmatrix} -\tilde{\omega}_k^{k,0} & -\tilde{v}_k^{k,0} \\ 0 & -\tilde{\omega}_k^{k,0} \end{bmatrix} I^k T_k^{k,0} + (W^k)^T \tag{2.10}$$

This can be modelled in bond-graph form by the graph in figure 2.1.

The 1-junction expresses this wrench balance. The effort going into the I-element is of the value $I\dot{T}$. The Se element represents the wrench input to the body $(W^k)^T$ term, and finally, the MGY-element represents the remaining term. Indeed, the constitutive equation of the MGY is (calling the matrix that multiplies the twist QI):

$\vec{e}_{port} = QI * \vec{f}_{port}$

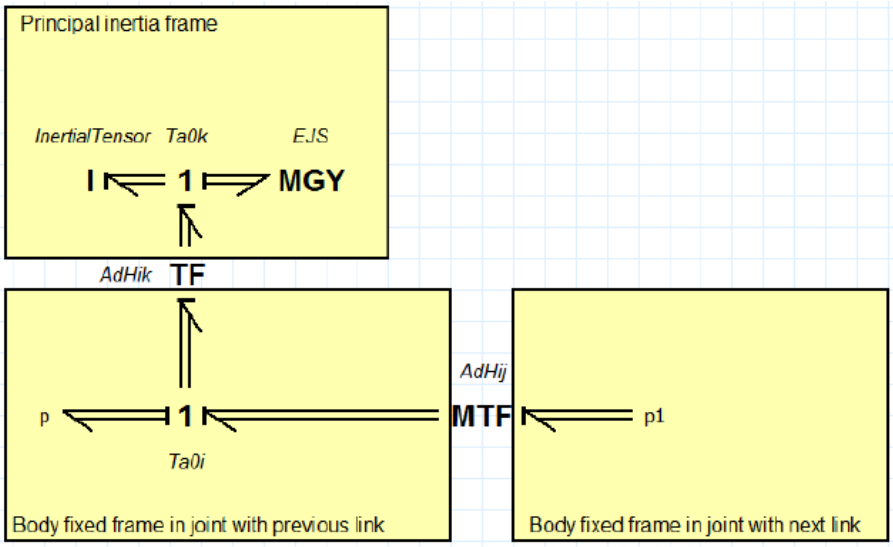

Figure 2.2: Bond graph of a link with transformations into frames i and j ([5])

And here, $\vec{e}_{port}$ is the pseudo-wrench exerted on the body by inertial effects, and $\vec{f}_{port}$ is the twist of the body.

Now, this twist at the 1-junction is expressed in the principal inertia frame (k). However, there are no wrenches being exerted (directly) on the principal inertia frame. Most wrenches considered here are applied to the body at the joints. With this in mind, we need to effectuate coordinate transformations from the principal inertia frame k to a frame i centered at the joint closest to the base, fixed to the current body, and from i to a frame j centered at the other joint on the body, and also fixed to the current body.

To effectuate these transformations, one needs to take the Adjoint of the homogeneous matrix expressing this transformation. This is because all of the power flow is expressed through wrenches and twists, and this is how you express frame conversions of these. Applying these gives the bond graph in figure 2.2 showing the correct power bonds to which to couple external forces:

However, one further aspect of the dynamics of the robot has been ignored up until this point, namely gravity. So, one needs to include an effort source for gravity, and feed it through transformations such that this can be linked to an already defined flow on the graph. The transformation for gravity is modulated here by H_i^0. Figure 2.3 shows what this looks like.

So, now that this is done, all that remains is to make a dynamic model of the joint. Indeed, all of the robot's input torques are applied at this level, and this allows for the inclusion of dynamics at the joint level (bearing friction, as well as friction and compliance in the constraint). One then obtains the bond-graph in figure 2.4.

Here, the "integrator" block takes the joint velocity from the 1-junction where the torque is applied, computes the joint position from that, and inputs it to a rotational homogeneous matrix that transforms the j frame of the previous link to the i frame of the next link.

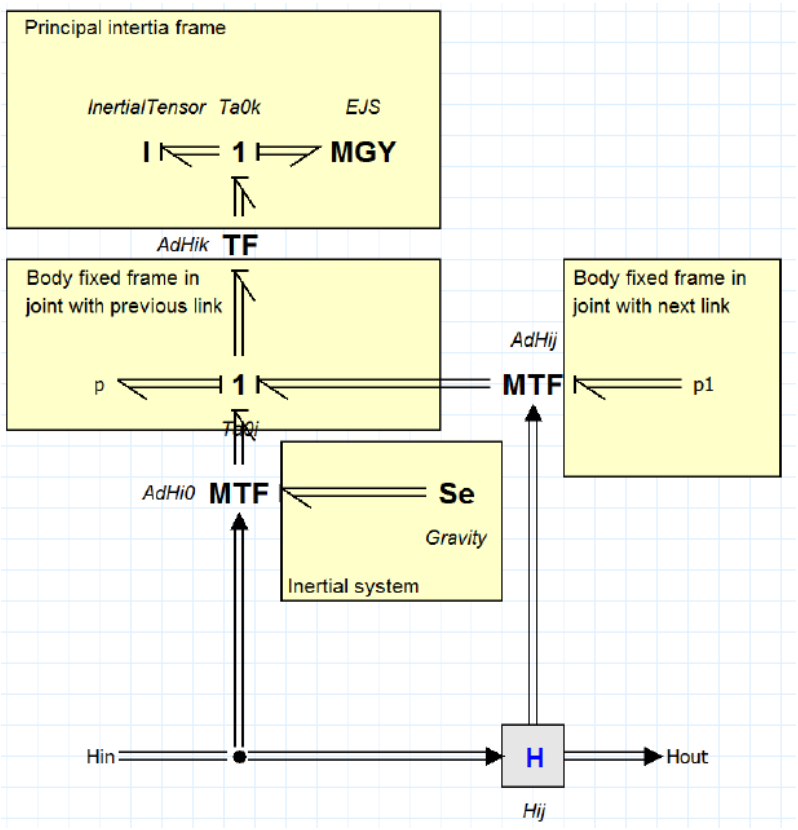

Figure 2.3: Bond graph taking gravity into account ([5])

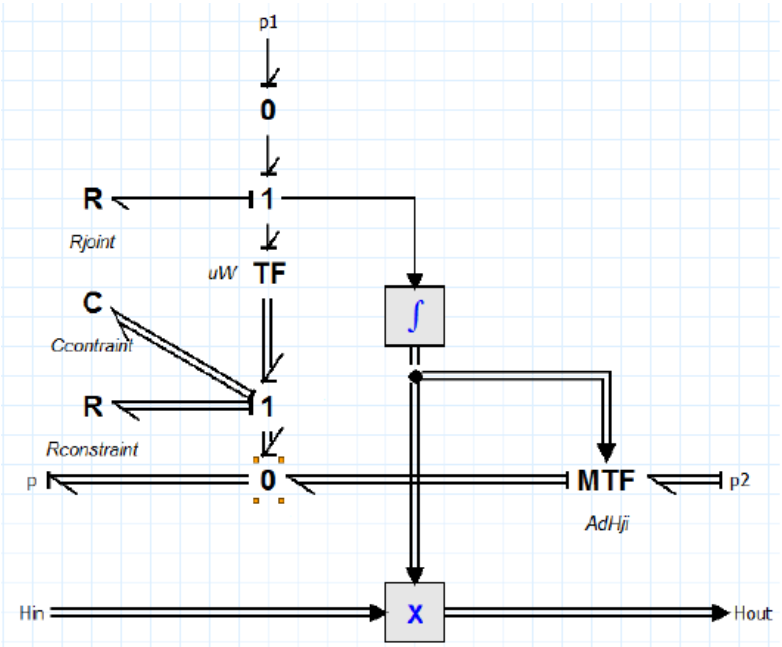

Figure 2.4: Bond graph of a joint ([5])

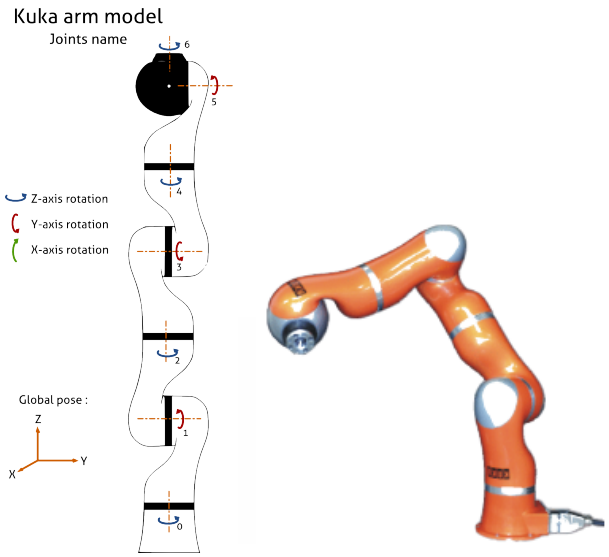

Figure 2.5: The KUKA LWR4+ ([11] & [13])

By chaining all of these together (...-link-joint-link-joint-...), one can make a dynamic model of any serial chain manipulator. This gives us all the resources we need to later make a dynamic model of a robot arm.

2.3 Relevant Hardware

The only piece of hardware worth considering here is the KUKA LWR4+. Essentially, it is a 7-DOF serial manipulator. All of the joints are rotational and have angular encoders on all of them.

Figure 2.5 is a diagram of the joint axes and the aspect of the robot itself.

3 Inertia Compensation for Serial-Chain Manipulators

So, in order to achieve this feeling of massless teleoperation, an inertia-compensating controller is required. This chapter deals with using the concept of model inversion to design an inertia-compensating controller for any plant with inertial dynamics, using a technique called model inversion

This will be introduced by showing what such a controller would look like for a mass-spring-damper system. Showing how inertia-compensation works for this simple system then allows us to justify a general control structure for inertia compensation. This is then applied to a Serial-Chain Manipulator system with an arbitrary amount of links. After this, some comments are made about the passivity properties of this system with inertia-compensated dynamics.

Finally, some experiments are performed in simulation to validate that the proposed control structure does indeed produce the desired dynamic effects.

3.1 Model Inversion

3.1.1 Model Inversion for a mass-spring-damper system

This subsection serves to give somewhat of an intuitive understanding of how inertia compensation through model inversion works through a very thoroughly studied mechanical system: the mass-spring-damper system.

Let us consider a simple mass-spring-damper system (figure 3.1).

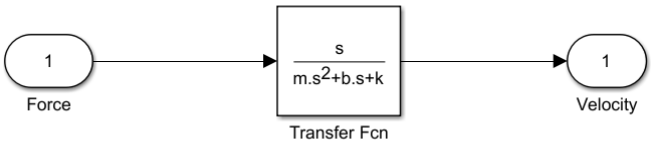

Figure 3.1: Input-output Model of a mass-spring-damper system

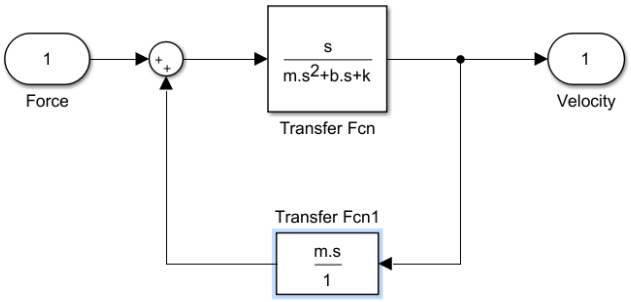

Figure 3.2: Inertia-compensated mass-spring-damper system

If one then takes the output, applies the transfer function ms to it, and adds it to the output (figure 3.2), one gets the following transfer function (using *Mason's Rule* [9]):

$$\frac{V(s)}{F(s)} = \frac{\frac{s}{ms^2+bs+k}}{1-\frac{ms.s}{ms^2+bs+k}} \tag{3.1}$$

$$\frac{V(s)}{F(s)} = \frac{s}{ms^2+bs+k-ms^2} = \frac{s}{bs+k} \tag{3.2}$$

It should be noted that this ms process is an inverted version of newton's second law with respect to how it is used in the actual mass-spring-damper system. It gives a compensation force to compensate for a mass moving with a given acceleration (velocity is differentiated). This is what model inversion is: making a model of a plant, switching the inputs and outputs, and then linking the input of the inverted model to the output of the model, and former's output to the latter's input.

One can see that such a control regime compensates for inertia, since the system now behaves like its mass is 0.

We now take a look at how this works with bond graphs. Our example, in bond graph form, is shown in fig.3.3.

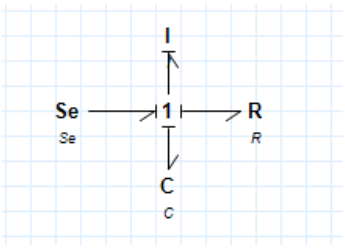

Figure 3.3: Bond graph model of a mass-spring damper-system

To invert this model, one simply needs to invert the causality of the port that interacts with the environment (here, the one connected to the effort-source). As with the example in block diagram form, this will take a model with an effort input and a flow-output, and turn it into a model that has that same flow as an input, and the same effort as an output. To do this, one can just make the effort source a flow source and let the causality propagate through the graph. Doing this yields the bond graph of figure 3.4.

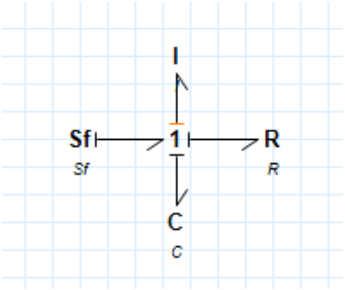

Figure 3.4: Inverted model of the mass-spring-damper system

However, we are only interested in compensating for inertia, in this instance, so we need to remove all of the additional dynamic effects (damping and capacitive). Removing these and

simplifying the graph yield figure 3.5. The bond graph implies that the transfer function of this reduced model is simply the admittance of the I-element: ms, which is exactly the inverted model we added to the input to remove mass effects from the system. So, as with figure 3.2, the flow-output of the plant in figure 3.3 can be fed into the I-element. Then, the effort-output of the I-element can be added to the effort going into the plant.

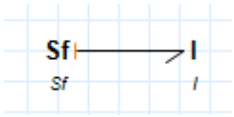

Figure 3.5: Inverted and reduced model of a mass-spring-damper system

3.1.2 Model Inversion as a general concept

This part serves to define, in the general sense, a control structure for model inversion. Consider a plant A and the partially inverted version of its model B. Let us assume, also, that B is partially inverted in such a way that it compensates for some specific dynamic properties of A.

For B to compensate for these dynamics, it needs to be integrated in the control structure shown in figure 3.6.

The first graph shows the first form that was detailed in the mass-spring-damper example. The second graph shows the same structure, but in a manner that makes it clearer how A and B should be interconnected if they are in bond-graph form.

Indeed, flow leaves A, gets fed into B through the 1-junction. B then returns the compensation effort, which then gets added to the input effort to A, which is exactly what the first diagram shows.

3.1.3 Inertia compensation for Serial-Chain Manipulators

Let us now take a look at how one would then compensate for inertia in a serial chain manipulator. Indeed, deriving a LTI-type transfer function and inverting it is impossible here due to the intrinsic non-linearity of the system. However, one most definitely can apply the method of inverting, then reducing a bond graph model.

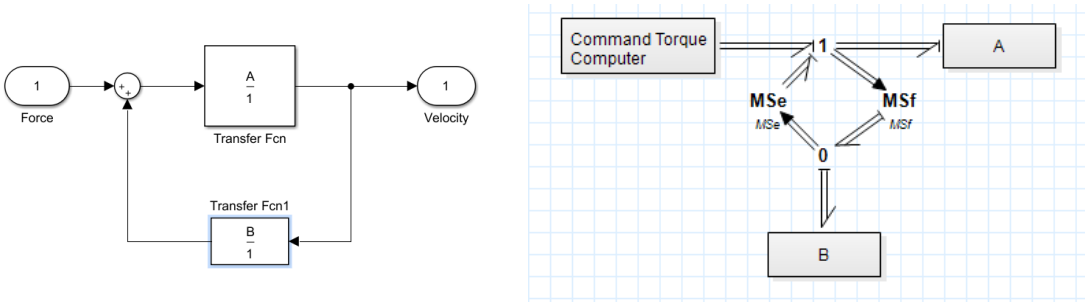

Figure 3.6: The control structure for model inversion

Indeed, a bond graph model was already derived in the background section. Here, we consider a joint and the link attached to it that is closer to the end of the chain, in the context of a larger chain. The bond graph for such a system is shown in figure 3.7.

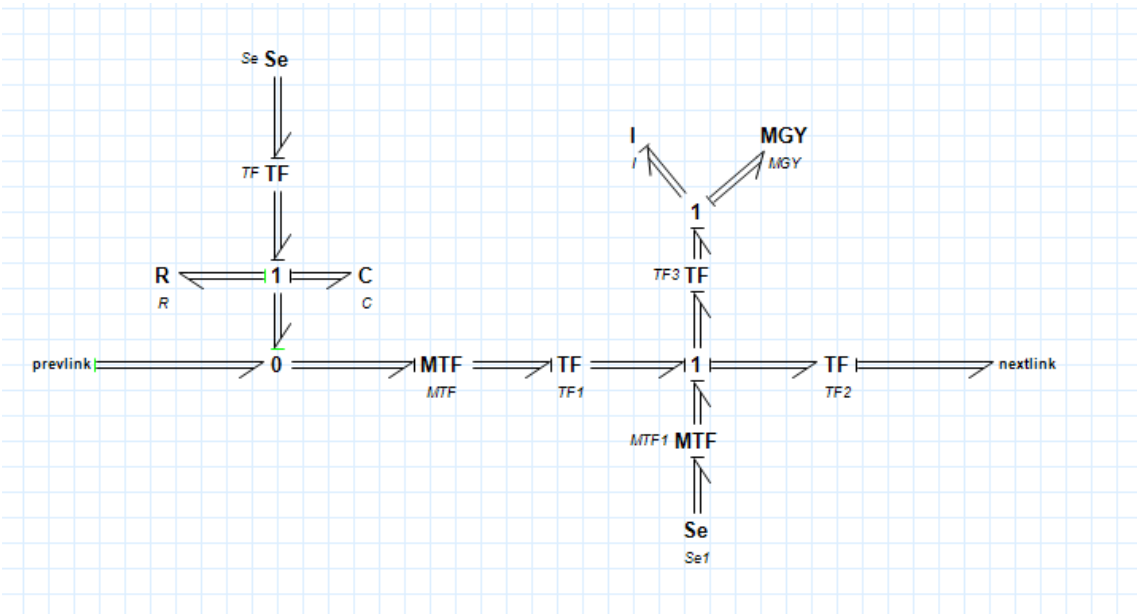

Figure 3.7: Bond graph of a link and a joint of a serial chain manipulator

Inverting, we get figure 3.8.

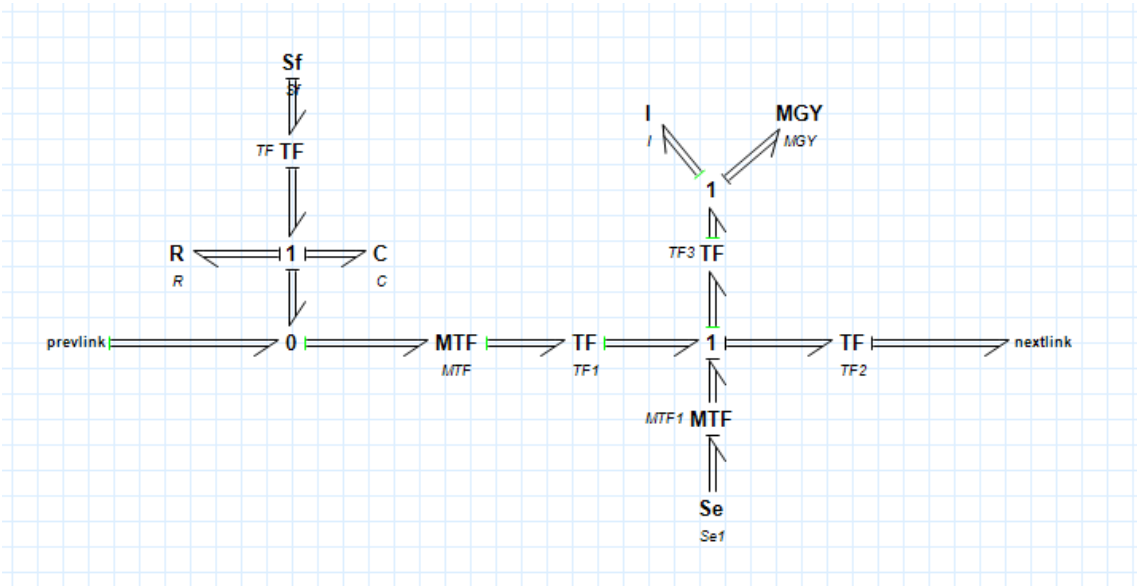

Figure 3.8: Inverted model of the link-joint system

Reducing, we get figure 3.9. The modelling of gyroscopic effects is left in as these too produce inertia-related forces.

So, the compensation torque to be added is the sum of the wrenches coming out of the I-element and the Euler junction of the link-joint system (fed through the appropriate transforms, of course), as well as all of the I-element and Euler junction wrenches from link-joint systems further up the chain. The causality is such that wrenches propagate down the serial

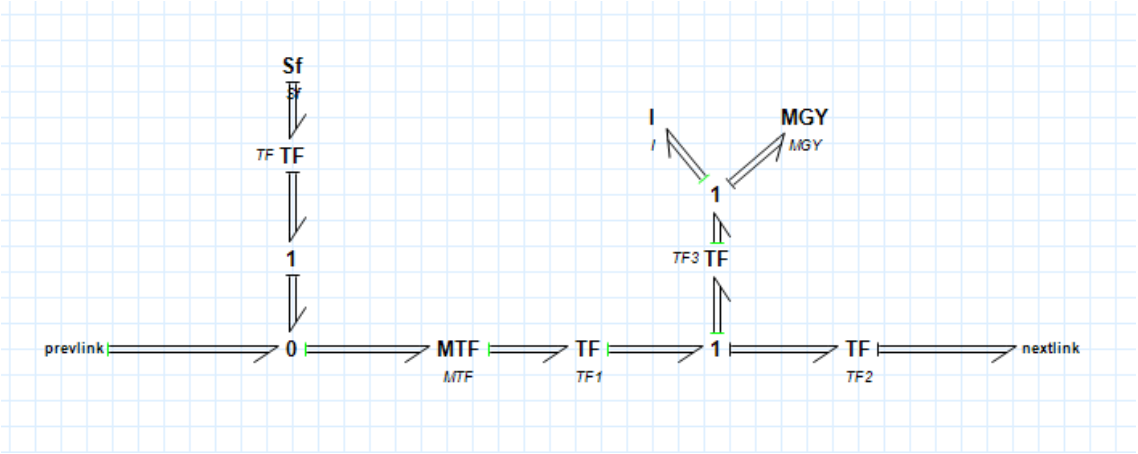

Figure 3.9: Inverted, reduced model of the link-joint system

chain. This is to be expected, intuitively speaking, as if you were to lock all but one joint, and then actuate this joint, an inertia-compensator would need to compensate for all inertias from it to the tip of the chain, not just the one closest to the joint. It is not proven here that the compensator does indeed behave thus, but evidence that this is indeed the case is shown in simulation later on.

Now, just like in the mass-spring-damper example, one can read the joint torque compensation torques that this inverted model computes, and add them to whatever command torques are at the input to the system, and this will compensate for the inertia of the system. To refer back to the control structure of figure 3.6, this inverted model is B, and the actual manipulator is A.

One can add a *compensation factor* multiplier $0 < \eta < 1$ to the I-elements and Euler-junctions in the compensator to only partially compensate for a manipulator's inertia. This is used in the below-described experiments.

A final comment should be made regarding the effect of η on stability. When inertia represents the dominant part of the dynamics in the system, like in equation 3.3, then values of η close to 1 will result in overly high accelerations, which will cause instabilities in the controller. When friction is higher, however, η can be made 1, provided derivatives in input torque are not too high. High derivatives in input torque, if inertia is too low, would provoke high derivatives in velocity, and so quasi-infinite accelerations.

3.1.4 Passivity

It is clear from the bond-graph form of the passivity layer that the compensator, on its own, is an active element. It is fundamentally a modulated source of effort. However, for any allowable value of η, interconnecting it with the plant effectively results in the same plant with reduced inertia or no inertia at all. So, the compensator-plant system is passive, in the ideal case.

However, uncertainty in model parameters could cause the inertia-compensator to overcompensate, which would result in an active system. Discretising the compensator also creates innacuracies in the compensator's outputs. This would impair the compensator's ability to produce an output that actually compensates for inertia. The compensator, being active on its

own, would then render the whole system active.

This is the essence of why a passivity layer would still be required here. The specifics of this will be discussed later.

3.2 Simulation results

Having shown how one would achieve inertia compensation through model inversion, it should now be shown that this does actually work. To this end, the KUKA LWR4+ model, as well as its corresponding inertia-compensating controller, are put into 20-sim, and some unit tests are performed to ensure that model inversion does indeed work as expected.

Essentially, the same test is performed for different cases. The idea is to strip all dynamics off the plant except for inertia. This means that gravity is removed from the simulation, and joint damping is set to 0. Then, when a joint in the model is given an input force, it should accelerate by the following amount:

$$\alpha = \frac{\tau_{in}}{(1-\eta) * I_{eq}} \tag{3.3}$$

Here, I_{eq} is the inertia of the part of the chain that goes from the joint in question to the tip, about the axis of rotation of the joint, and η is the fraction of the manipulator's mass that is being compensated for in the controller.

The following tests are performed on the fifth and sixth joint of the model, so as to show that the controller does indeed also compensate for inertias further up the chain, as well as to show that this works for both "z-axis" and "y-axis" joints. The rest of the manipulator is left in its reference configuration. Inertia values for this were taken from [8].

The fifth joint is a "z-axis joint", so all relevant inertias are on the axis of rotation already. Therefore, the inertia felt by the joint is the sum of the z-axis inertia of links 5, 6, and 7. The inertia then works out to being $I = 0.0144 kg.m^2$.

Therefore, for $\tau_{in} = 0.1 N.m$, and for $\eta_1 = 0$ and $\eta_2 = 0.5$ respectively:

$$\alpha_1 = 6.93 m/s^2 \qquad \alpha_2 = 13.86 m/s^2$$

Indeed, this is what can be seen from the simulation results also (figure 3.10).

The input torque is a cycloidal square pulse that plateaus at $0.1 N.m$, so the analytically computed acceleration can be found at the corresponding plateau in acceleration.

The sixth joint is a "y-axis joint", so an extra parallel-axis correction term per joint is required too. We get $I = 0.0112377 kg.m^2$.

Using the same other parameters for computing acceleration, the result is:

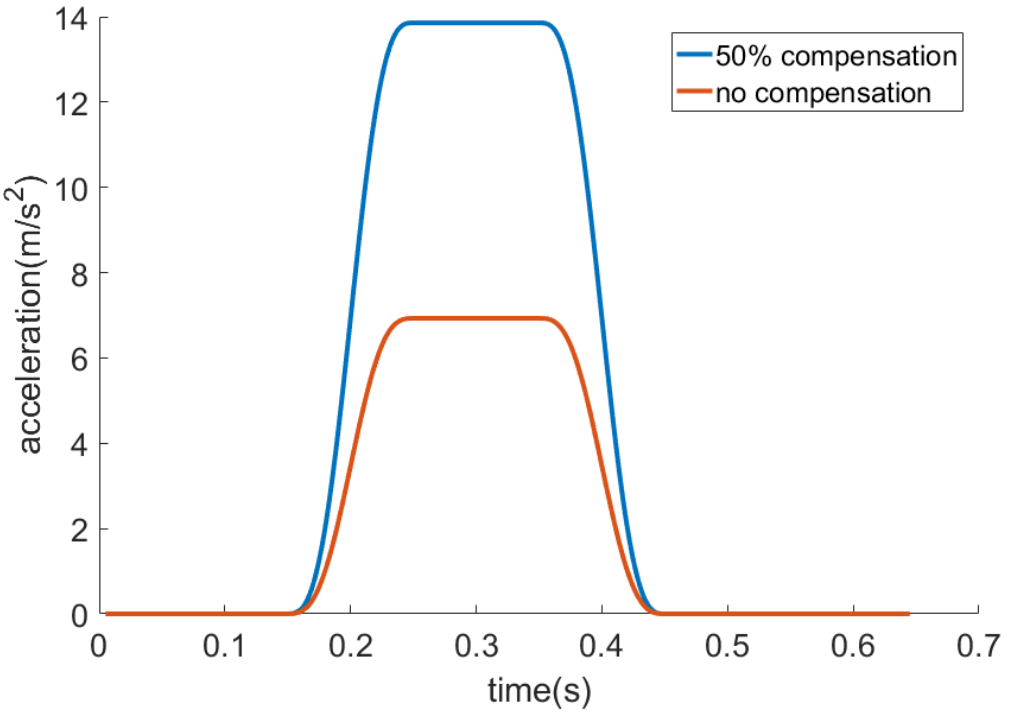

Figure 3.10: Simulation runs for the z-axis joint

$$\alpha_1 = 8.89 m/s^2 \qquad \alpha_2 = 17.80 m/s^2$$

Likewise with this joint, simulation results corroborate this calculation. See figure 3.11.

3.3 Discussion

Simulating the controller's action on the plant in 20-sim is not always accurate, however. Certain parameter value ranges render the simulation unstable.

Indeed, the simulation is unstable for high derivatives in input torque, for a given value of η. The system becomes more tolerant the more this value is lowered. The instability is of different nature depending on which solver is used, so this is most probably a solver-related instability rather than a physical one. This is to be expected, seeing as regardless of high values of friction, if inertia is too low, a step in input torque is a step (approximately) in velocity, which results in overly high values of acceleration for a potential simulator to be able to deal with. Presumably, an explicit integration scheme with a high enough sample frequency would be able to counter this instability, but this was not tested due limitations in computational resources.

Simulator instabilities also come into play when modelling gravity in the arm. This is most likely due to the high accelerations produced by gravity when η is not low enough. It should be noted, however, that this does not impair any ability to draw conclusions as to the inertia-compensator's effectiveness, because the controller should work regardless of what other dynamics are present or not in the plant model.

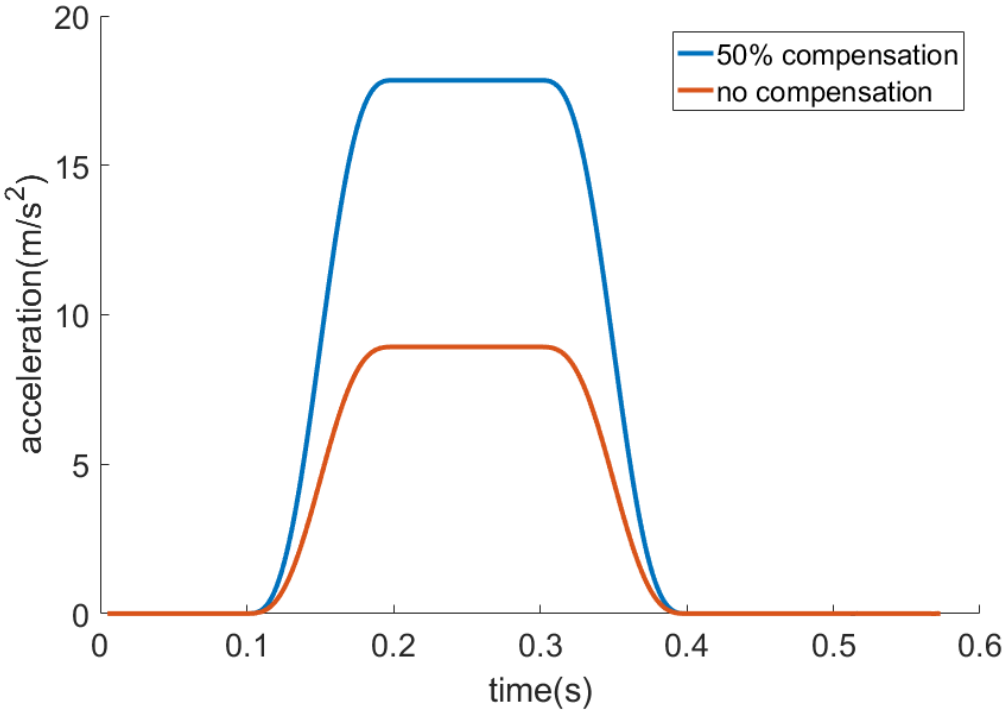

Figure 3.11: Simulation runs for the y-axis joint

4 Inertia-Compensated Tele-Operation of Serial-Chain Manipulators

Now that inertia compensation through model inversion, as a concept, has been proven to work, we can now evaluate whether the expected effects on teleoperation from inertia compensation are indeed present. First, a simple set-up with an impedance controller will be discussed. Next, the effects of discretising the controller, as well as adding a passivity layer will be discussed. All experiments are performed, once again, in 20-sim.

4.1 Continuous-time Impedance Control

Initially, we take a look at the most ideal version of impedance-control-based tele-operation: no time-delays, and with a continuous-domain controller.

The control structure for these experiments is shown in figure 4.1. The inertia parameters are kept realistic unless mentioned otherwise, but joint damping is set at $b = 0.5N.m.s$. Gravity is neglected, once again, as it introduces rogue artefacts into the simulations. The impedance controller has the following characteristic matrices ($diag(a)$ returns a times the identity matrix):

$$\begin{cases} K_c = 0 \\ K_0 = diag(250) \\ K_t = diag(1000) \end{cases} \tag{4.1}$$

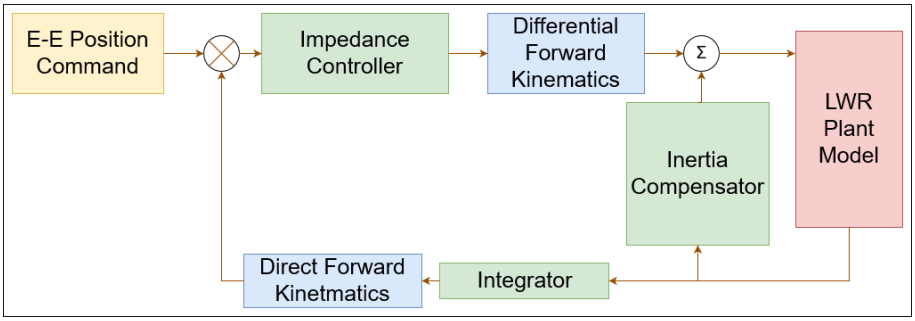

Figure 4.1: Control Structure for Continuous-time experiments

The chosen setpoint is a spatially linear motion characterised by:

$$x(t) = A(1 - cos(\omega t)) \tag{4.2}$$

With $A = 0.1$.

Let us first compare simulation results with $\eta = 0.85$ to a simulation without the compensator, but with a robot that has 85% less inertia. Figures 4.2 and 4.3 show the output wrench of the impedance controller. Here, $\omega = \frac{2\pi}{5}$. The motion of the end-effector is also shown in figure 4.3 to show that the robot is indeed tracking the reference adequately.

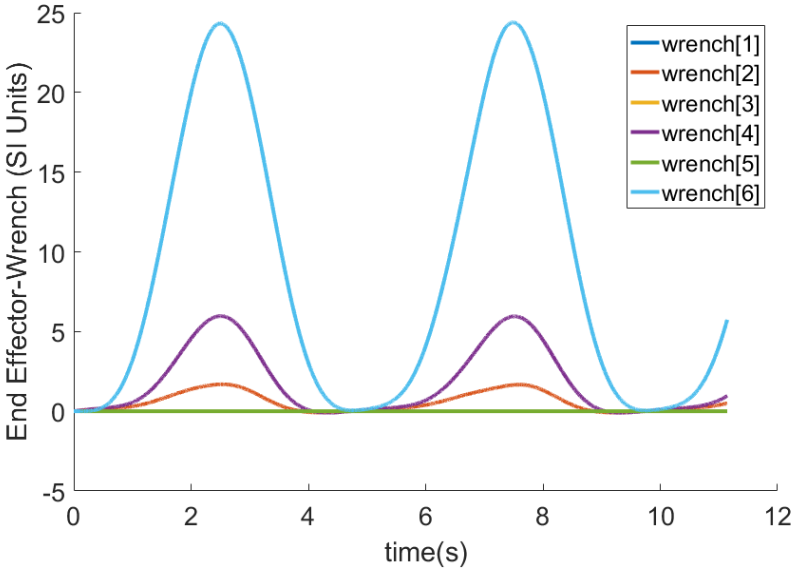

Figure 4.2: Impedance Controller wrench, and End-Effector position for the arm with 85% less inertia

Indeed, these graphs show exactly the same controller performance. This means that the inertia compensator works in this instance too. However, this was to be suspected already. What is more interesting to see is whether an inertia-compensated plant does reduce the wrench output of the impedance controller. This would mean that, in a haptic-feedback set-up, the operator does indeed feel less inertia effects, which is the whole purpose of compensation.

Figures 4.5, 4.6 and 4.7 show the difference between impedance controller output wrench in the 85% compensated state and in the uncompensated state, for different input frequencies.

For clarity's sake, the value plotted in these graphs is:

$$W = W_{nocompensation} - W_{85\%compensation} \tag{4.3}$$

The table in figure 4.4 shows the average values of these wrench differences.

Indeed, one can see that there is a positive difference, on average, between the non-compensated and the compensated output. This becomes more pronounced the more the manipulator accelerates throughout its trajectory, which makes sense. Also showing this is the fact that when one increases the frequency of the set-point, these discrepancies between compensated and non-compensated operation increase.

One of the values in the table is indeed negative, but this value is small enough that it might as well be considered to be zero in this case. Some non-linear effect due to the manipulator's configuration is probably the cause of this, as it is also most likely the cause of these differences becoming negative in the first place.

4.2 Discrete-time Impedance Control

We now take a look at what discretising the controller does to the simulation results. The continuous-time results will be used as a benchmark. For the same input and η value of 0.85

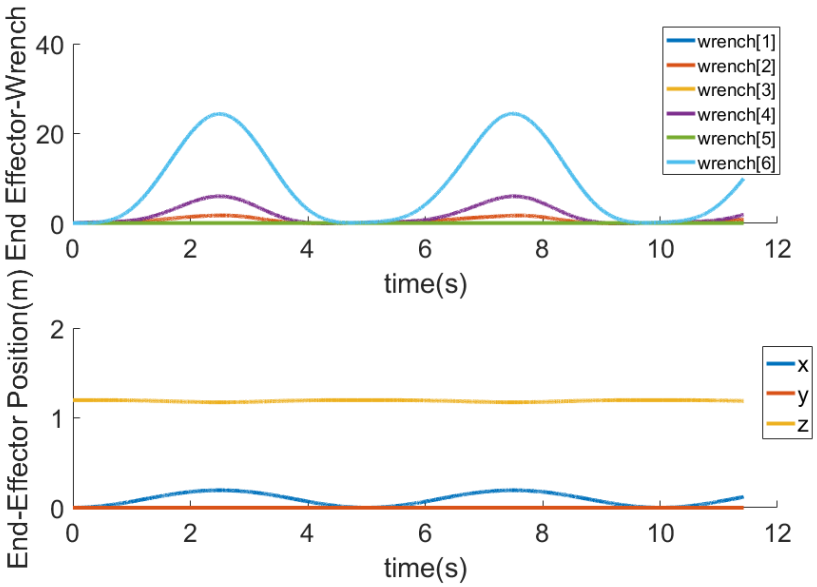

Figure 4.3: Impedance Controller wrench, and End-Effector position with 85% inertia compensation

Frequency	Wrench[1]	Wrench[2]	Wrench[3]	Wrench[4]	Wrench[5]	Wrench[6]
$2\pi/5$	0	0.1387	0	0.6326	0	5.0379
$2\pi/10$	0	0.0196	0	0.084	0	0.8821
$2\pi/20$	0	-0.0021	0	0.0042	0	0.0747

Figure 4.4: Average values of figures 4.5-4.7

as in the continuous-time simulations, one gets the results shown in figure 4.8. The operating frequency is $1000Hz$.

As could be expected, the discrete system's output is much the same as that of the continuous time system.

Given that the system is now discrete, one can also start thinking about how the compensator is going to affect the passivity layer design. Shown in figure 4.9 is the power used by the impedance controller and by the compensator, with a more realistic [7] damping value of $b = 30 \quad N.M.s$. The value is a linearised version of a coulomb-viscous friction model in the range $0-0.6 \quad rad/s$ operating range of the joints. It should be noted that η is still at 0.85, even though the simulation runs without problems for $\eta = 1$, seeing as the dominant dynamics in this scenario are clearly friction effects. The choice of η was more one of convenience.

In the plot, one notices that for most of the time, the power used by the impedance controller is around an order of magnitude higher than that used by the compensator, so it does not affect energy interaction in the short term. Also, one notices that over an oscillation period of the end-effector, the energy used by the compensator is 0. This means that it shows energy conserving behaviour. This is to be expected, seeing as accelerating the manipulator and then decelerating it back to the same velocity should cost the same amount of energy. So, any arbitrary motion that brings the manipulator back to the same velocity should cost no net energy.

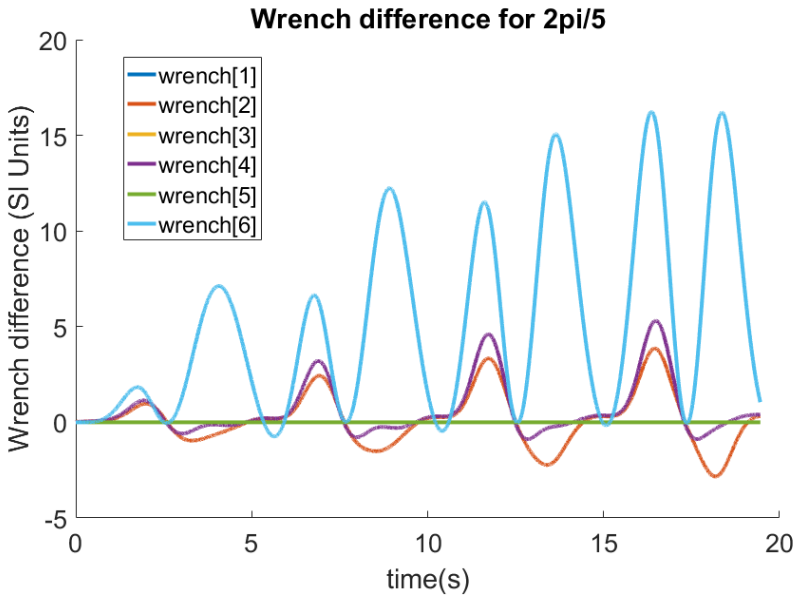

Figure 4.5: Comparison of compensated and non-compensated output for $\omega = \frac{2\pi}{5}$

This means, for passivity layer design, that the initial energy in the tanks just needs to be high enough to cope with the instantaneous accelerations produced by the impedance controller. If one also then adjusts H_d (the activation threshold of the master Tank Level Controller) dynamically by the energy used by the inertia-compensator, the interaction behaviour of the passivity layer should be exactly the same as if the inertia-compensator weren't there. This is not elaborated on further in this thesis, however, since experiments were not performed to support this idea.

4.3 Discrete-time Impedance Control with Passivity Layer

Now, if this controller were in a real teleoperation set-up, time delays would start introducing instabilities into the control set-up. More importantly, the aforementioned active behaviour shown by the inertia compensator also renders the system active. So, it is a good idea to have a passivity layer in the system. The simulation uses the control structure from figure 4.10, and uses the model parameters and input used for the continuous-time simulations. See figure 4.11 and 4.12 for the results.

Indeed, once the energy in the tanks runs out (at around 3.45s), the controller stops actuating the arm, so it stops, so the system can be guaranteed to be passive. It is hard to make any other statements about the design of the passivity layer, seeing as making a dynamic model for the master controller's interaction with the environment is beyond the scope of this thesis.

4.4 Discussion

The first thing to remember when considering the validity of these results is the omission of gravity. The exact values of the controller wrench and output power are not realistic because of this. However, they do achieve the purpose of showing the effect of inertia compensation on teleoperation. What is important is to be able to see is a reduction in the amount of force exerted by the impedance controller when the manipulator accelerates more, which we can discern from the results.

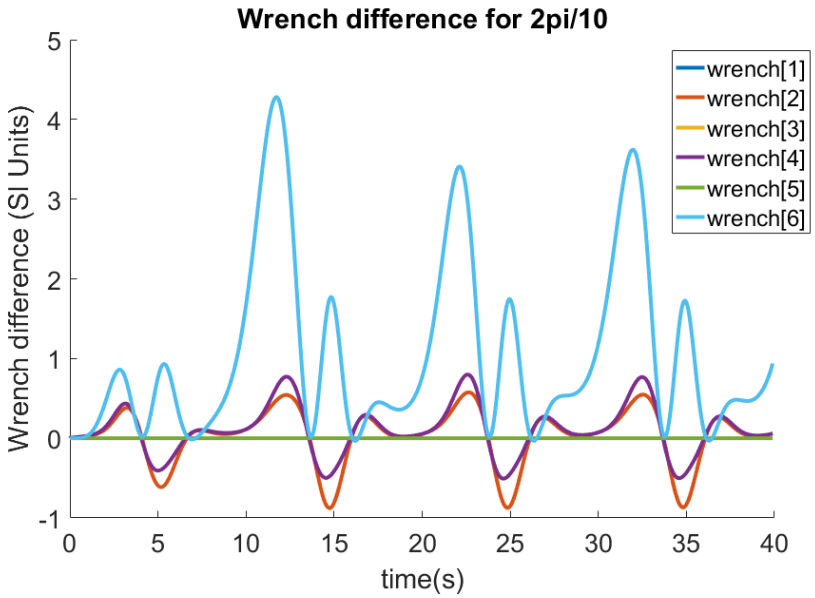

Figure 4.6: Comparison of compensated and non-compensated output for $\omega = \frac{2\pi}{10}$

Also, gravity compensation is a documented topic in robotics ([5]). These results would then be valid if a perfect gravity-compensating controller were also implemented. Of course, the statements made in the previous section about energy in the system will not be as valid, seeing as there would be a new active element in the system. However, the fact that the inertia compensator uses quite a bit less energy than the impedance controller should, in theory, still hold, as the dynamics of the manipulator, the impedance controller, and the inertia-compensator would be much the same. The same goes for the statements regarding the energy-conserving behaviour of the inertia-compensator.

Secondly, the passivity layer simulation unfortunately does not allow one to make any statements regarding the tuning of the various parameters in the passivity layer. The only way to be able to do so would either involve a physical experiment with a human operator, or a realistic dynamic model of such an operator.

The last thing to remark is the discretised controller's sensitivity to operating frequency. For any given value for damping in the joints, there is a value for f_s under which the controller is unstable. This is due to the energy created by an increasingly aliasing discretisation. At some point, this becomes higher than the energy dissipated by damping in the joints, and so the sum of the two powers into the system becomes positive, destabilising the whole system. For realistic damping values for the LWR, however, this operating frequency cut-off is significantly lower than $1kHz$, so this has little bearing on the application stated in this thesis.

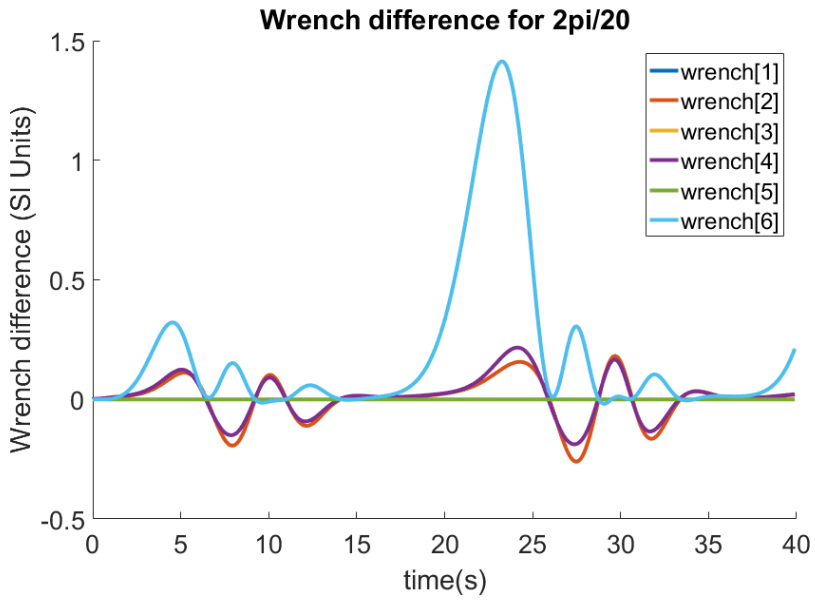

Figure 4.7: Comparison of compensated and non-compensated output for $\omega = \frac{2\pi}{20}$

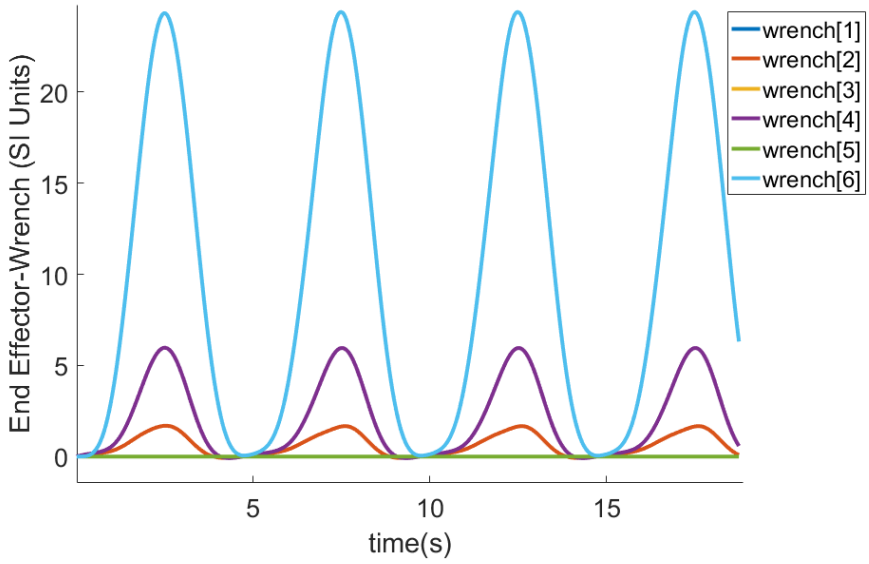

Figure 4.8: Output of the discretised controller for the same input parameters as in fig.4.3

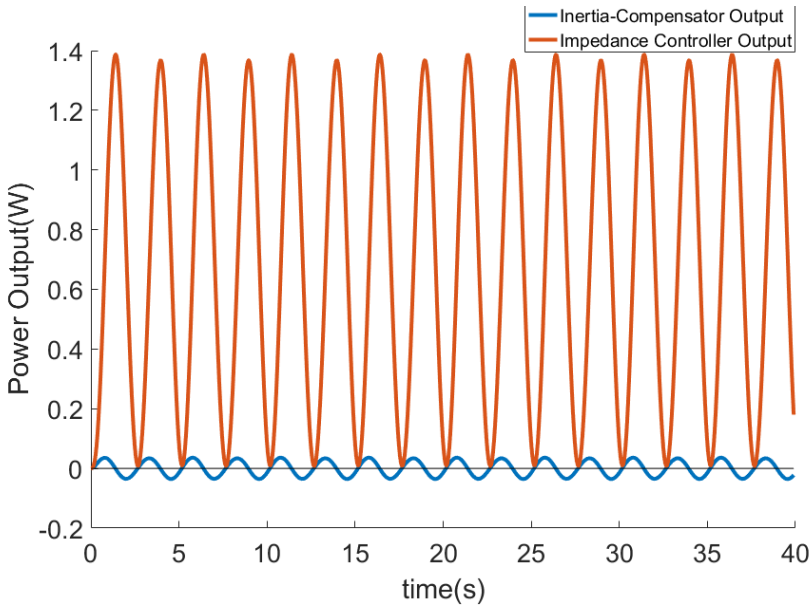

Figure 4.9: Power output of the two controllers

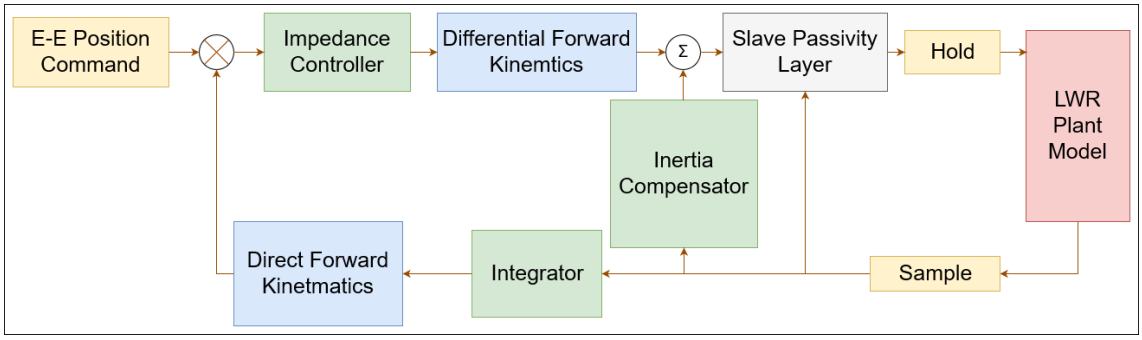

Figure 4.10: Control structure for the passivity layer experiment

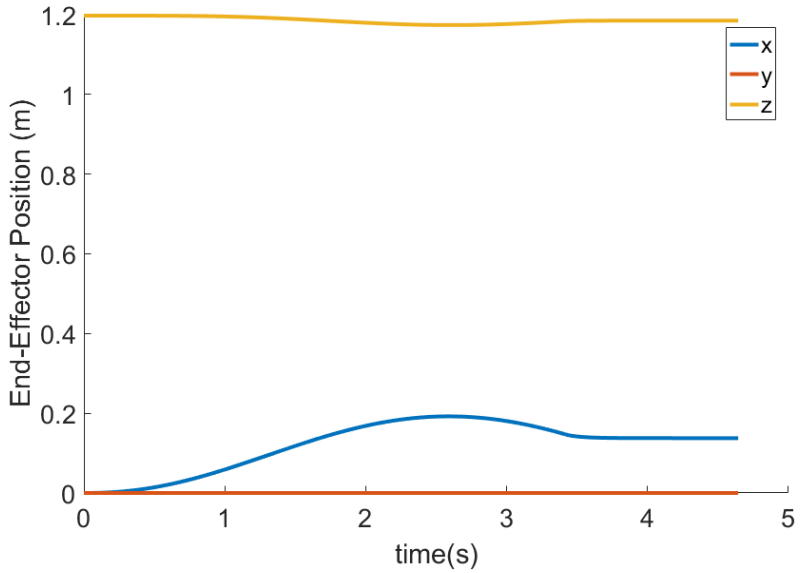

Figure 4.11: End-Effector position with the passivity layer in the control structure

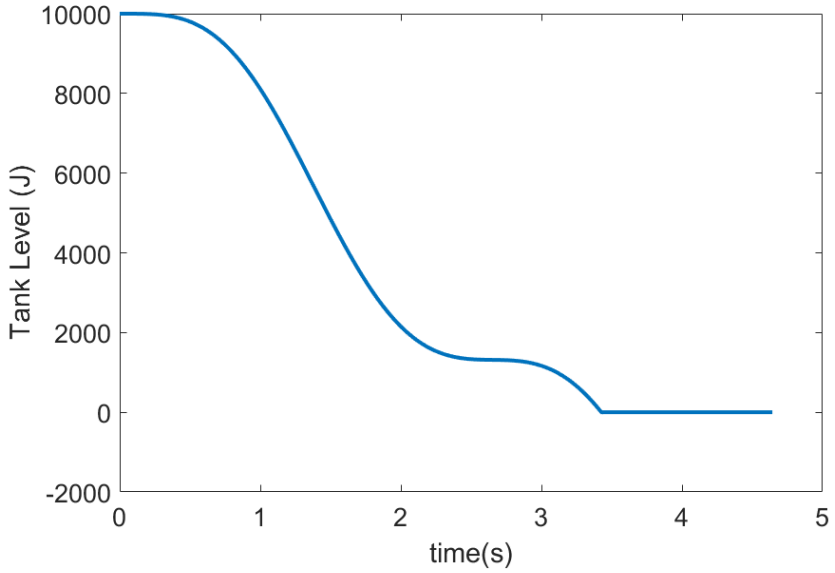

Figure 4.12: Slave passivity layer tank level

5 Conclusions and Recommendations

5.1 Conclusions

Before any conclusions are actually drawn, let us consider the actual goal of the project again. This was to make an inertia-compensating controller for a KUKA LWR4+ and integrate it with a bilateral teleoperation set-up including a passivity layer.

The inertia-compensator behaves as it should. It allows the manipulator it controls to mimic the dynamics of a similar manipulator with arbitrarily less inertia, in simulation. However, analysing how the actual LWR manipulator's behaves with the inertia compensator is difficult given certain eccentricities of the solvers used. Modelling gravity, as well as higher complexity friction models could not be achieved in 20-sim without simulation artefacts that rendered the results inconclusive.

Integrating this into a teleoperation set-up, one can notice an average reduction in force that is fed back to the operator on the master. The larger part of this happens when the manipulator's end-effector is given a high acceleration command, so the compensator is working predictably. The controller does exhibit instabilities below an operating frequency that depends on damping in the joints.

When adding a passivity layer into the teleoperation set-up, only limited conclusions can be drawn. Indeed, when energy in a tank reaches zero, the controller does indeed stop actuating the manipulator. However, making any further statements on how this passivity layer affects transparency, for example, cannot be made, since without a realistic dynamic model for an operator, the energy put into the system on the master side can not be adequately modelled either.

5.2 Recommendations

A very important next step would be to actually perform a physical experiment on a real KUKA LWR4+. This would nullify some of the caveats in the conclusions drawn from simulations, notably when it comes to gravity and friction in the joints. Failing this, finding some way of getting simulation results modelling gravity and a more complicated friction regime would be useful where this is concerned as well.

Also, given the simulations for the most realistic model parameters, it would seem that overall, the impedance controller puts significantly more effort into overcoming friction than it does into overcoming inertial effects. So, compensating for friction might prove to be a lot more valuable for transparent teleoperation than compensating for inertia.

Bibliography

[1] Disaster relief robot and operation controller therefor. 8 1991.

[2] Gabriel Aguirre-Ollinger, J Edward Colgate, Michael A Peshkin, and Ambarish Goswami. Design of an active one-degree-of-freedom lower-limb exoskeleton with inertia compensation. *The International Journal of Robotics Research*, 30(4):486–499.

[3] Carlos Cardenas. Development of a Safety-Aware Intrinsically Passive Controller for Multi-DOF Manipulator. 2017.

[4] Karthik Chandrasekaran, Sakthivel Sivaraman, and Asokan Thondiyath. Static Balancing and Inertia Compensation of a Master Manipulator for Tele-operated Surgical Robot Application. *ACM International Conference Proceeding Series*, 2015.

[5] Douwe Dresscher. Robust autonomy for the youBot. 2010.

[6] Michel Franken, Stefano Stramigioli, Sarthak Misra, Cristian Secchi, and Alessandro MacChelli. Bilateral telemanipulation with time delays: A two-layer approach combining passivity and transparency. *IEEE Transactions on Robotics*, 2011.

[7] A. Jubien, M. Gautier, and A. Janot. Dynamic identification of the Kuka LWR robot using motor torques and joint torque sensors data. *IFAC Proceedings Volumes*, 47(3):8391–8396, 2014.

[8] Es C Luper Report Drir D Dresscher Drir JF Broenink Drir RJ Wiegerink. Component-Based Modelling and Simulation of a KUKA LWR+4 Robotic arm. 2017.

[9] Sam Mason. Further Properties of Signal Flow Graphs*. 1956.

[10] L Matthies, Y Xiong, R Hogg, D Zhu, A Rankin, B Kennedy, M Hebert, R Maclachlan, C Won, T Frost, G Sukhatme, M Mchenry, and S Goldberg. A portable, autonomous, urban reconnaissance robot. *Robotics and Autonomous Systems*, 40:163–172, 2002.

[11] openrobots.org. KUKA LWR arm actuator, 2009.

[12] Stefano Stramigioli and Herman Bruyninckx. Geometry of dynamic and higher-order kinematic screws. *Proceedings - IEEE International Conference on Robotics and Automation*, 2001.

[13] Kees van Teeffelen, Jan Broenink, Douwe Dresscher, A Mader, and A van Dijk. Intuitive Impedance Modulation in Haptic Control using Electromyography. 2018.

[14] Luige Vladareanu, Octavian Melinte, Adrian Bruja, Hongbo Wang, Xiaojie Wang, Shuang Cang, Hongnian Yu, Zeng Guang Hou, and Xiao Liang Xie. Haptic interfaces for the rescue walking robots motion in the disaster areas. In *2014 UKACC International Conference on Control, CONTROL 2014 - Proceedings*, 2014.