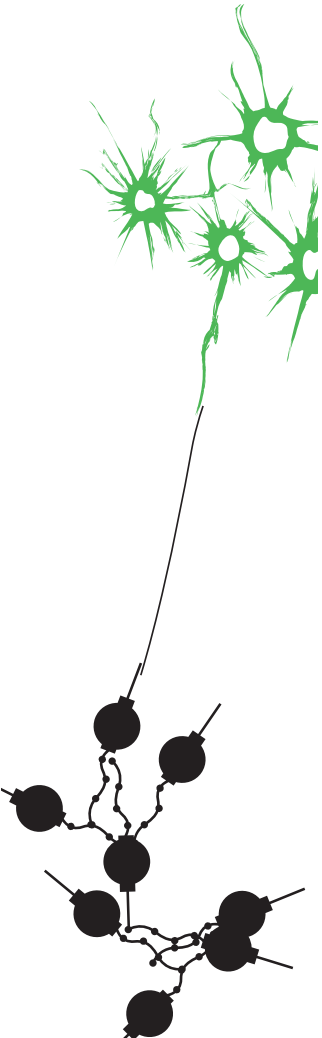


RAM ROBOTICS AND MECHATRONICS

# TDPN BASED CONTROL FOR HAPTIC SETUP

D.A. (Diego) Rooijackers

BSC ASSIGNMENT

**Committee:**
dr. ir. D. Dresscher
L.B.L. Lenders, MSc
dr. ing. G. Englebienne

July, 2024

025RaM2024
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE  UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

# TDPN based Control for Haptic Setup

Diego Rooijackers

Thursday 18th July, 2024

# Summary

Robotic avatars are robotic representations of human operators which are controller by humans, but can be placed and function remotely from them. Avatars.report is a trend watcher website which teaches about the topic. It also offers a DIY @ Home setup. However, this setup suffered from channel activity due to time delays. This thesis proposed the time-domain-power-network (TDPN) based method as a solution.

The TDPN based method is based on the principal that the energy send over the channel should be released on the other side of the channel. If more is released than absorbed, the system is not passive. In order to apply this controller in a discrete time environment, a method was used which calculates the energy based on previous force, and current and previous position output. This would deliver more accurate energy results and a clear method to assign what energies are released and absorbed. The correct assignment of energies, level of energy observed, and the passivity observation were all validated to work as the theory described.

The add dissipation and stop energy release controllers were researched. The former adds some damping to a variable based on observed states, while the latter sets that variable to zero. The variables impacted were the velocity at the robot, and force at the human interface side. The add dissipation method would be used until some maximum level of energy generated, then switch to stop energy release. This would guarantee the most transparency in the system. Due to time constraints, the add dissipation method was not tested. The stop energy release method tests showed the expected behaviour. However, it would cause loss of communication between setups.

# Contents

# 1 Introduction

## 1.1 Context

Robotic avatars are robots which act like a physical embodiment of a human operator (Dresschers, 2024*e*). This means the robot mimics the exact movement and interactions of a human operator. Meanwhile, the human operator experiences the environment the robot is perceiving through feedback. The process of having interaction to an environment with some feedback is called telemanipulation. This allows the robot to be placed in a remote environment from the user and the user to be able to manipulate this environment through the robot whilst getting some visible, audible and/or sensible feedback.

Telemanipulation technology would be to great benefit to us all. Robotic avatars can be used to do tasks in areas too dangerous for humans to operate in, while still having a human, with their skills and knowledge, actually operate.

Avatars.report is a trend-watchers and authority on the evolving field of robotic avatars (Dresschers, 2024*a*). The site provides video lectures on the field such that anybody could understand and learn about robotic avatars. For those more serious about the topic, it offers consultancy. On Avatars.report, an Arduino-based DIY @ home setup is also offered, which someone can use to learn about the aspects of telemanipulation, like time domain passivity control and different architectures. This setup can be seen in fig. 1.1. The DIY @ Home setup uses materials which can be easily acquired, like the Arduino UNO.

The full telemanipulation setup, as can be seen from fig. 1.2, consists of a two 3D-printed robotic arms controlled by an operator and delivering haptic feedback. The two setups are connected through three wires. These form a communication channel between the setups, over which information about the systems can be sent. Position values are sent to the robot side to where it has to move. The robot side sends back force values to deliver haptic feedback.

In telemanipulation setups, the information is sent over a channel. This channel introduces time delays to the system. These delays can cause the system to become unstable (Dresschers, 2024*a*).

## 1.2 Thesis assignment and research questions

This thesis discusses the time-domain-power-network (TDPN) method, as proposed in a paper by J. Artegas et al. Artigas et al. (2010) to use as time domain passivity control in the DIY @ home setup.

The following research questions are used to guide the design.

- How can the TDPN based method be effectively implemented on an Arduino Uno?
- What limitations of working in the discrete time domain influence the implementation of the TDPN based method? And how could those limitations be mitigated?
- How can energy be computed accurately in discrete-time systems?
- In what sense does the communication channel resemble a time -delay-power-network, and what limitations do working with the TDPN based method bring?
- If any were present, how could limitations in the given hardware be worked around?

## 1.3 Methodology

Firstly, a literature study has to be abducted in order to answer some of these research questions or formulate hypotheses. A deep analysis of the existing architecture used in the setup also has to be conducted to answer research questions. Based on these findings, an implementation can be designed for the Arduino Uno. This design then has to be evaluated.

## 1.4 Report layout

This paper first goes over the theory of the issue, previous solutions, and how the TDPN based method works in chapter 2. Chapter 3 will discuss hypothesis on the implementation of the method to the actual system. The results of the applied script are then shown and later on discussed in chapter 4 and chapter 5. Lastly, all is wrapped up in the chapter 6.

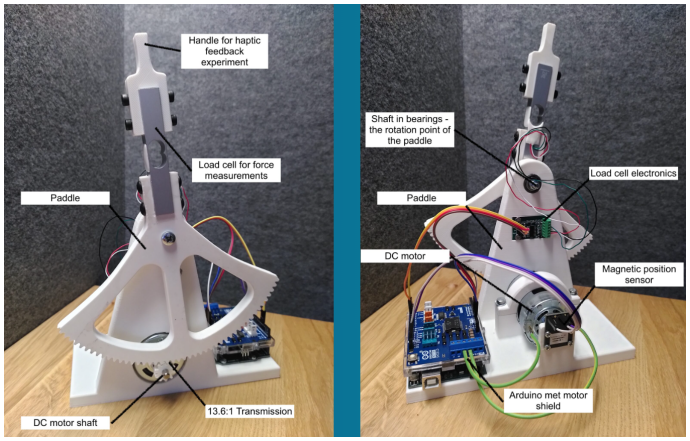



**Figure 1.1:** The DIY @ Home setup from Dresschers (2024*a*).

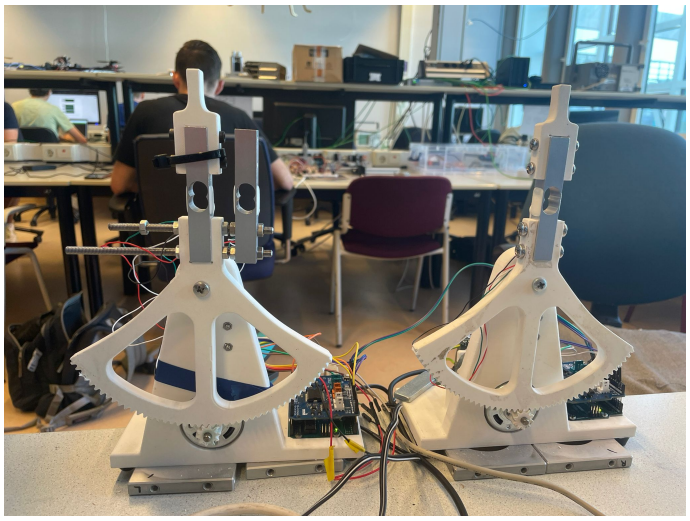


**Figure 1.2:** The DIY @ Home setup experimental setup.

# 2 Background

This section goes over the necessary knowledge required to understand the report. It discusses the relevant concepts of passivity and transparency. Time-domain-power-networks are discussed in detail, as well as the control mechanism of the system.

## 2.1 Passivity

Passivity is a property of a system which describes that a system, or subsystem, is exclusively dissipating energy at any point in time. Mathematically, the following statement must be sufficed:

$$\sum \int P \geq E_0 \tag{2.1}$$

The left-hand side describes the energy absorbed or released by the (sub-)system through every port of the (sub-)system. The right-hand side describes the initially stored energy of the system, usually assumed to be zero. Research (Artigas et al., 2010) concludes that a system must be stable when it is passive. Passivity is a great tool to use when designing a system, as each block of the system can be individually analysed to determine the stability of the whole system (Hannaford and Ryu, 2002). It is noted that passivity is a sufficient statement, but not a necessary one. A system can be non-passive, yet stable.

## 2.2 Transparency

Telemanipulation has two setups, one at the human and at the robot. These two communicate over a communication channel. Transparency is a measure of resemblance between the two setups at any given time. If the states of both sides are the same at any given time, we call this having full transparency (Hannaford and Ryu, 2002).

The connection established over the communication channel is an energetic connection (Dresschers, 2024*e*). This means power-conjugate variables are used to calculate the values of the powers and energies flowing in and out of the channel. This is achieved through software. If the power at the robot port of the channel given any point in time is the same as the power at the human port, we also say there is a direct connection or full transparency (Hannaford and Ryu, 2002).

## 2.3 Time domain passivity control

Time delays in the communication channel introduce non-passive elements to the system, which means energy is generated over the channel, leading to instability. One of the first methods of guaranteeing the system would not be become active was to implement a fixed damper. Its value was based on the worst case of energy generation. This damper would be able to dissipate any generated energy. Even though it showed to guarantee passivity, it caused a lot of energy loss. The method also heavily limits the transparency of the system (Hannaford and Ryu, 2002).

A better solution would be one where the current state of the system would be taken into account to decide when to apply damping and how much damping, which would increase the transparency of the system at the same time. This forms the basis of time domain passivity control (TDPC). Here, passivity is measured through a passivity observer and any generated energy is dissipated through a passivity controller.

Two TDPC methods that can be applied are the energy tank based method and TDPN based method. The energy tank based method keeps track of the energy content of the system by

looking at different power ports of a (sub-)system. The energy content of the system must always be larger or equal than zero (Hannaford and Ryu, 2002) for the system to be passive. Mathematically speaking, this statement must be met.

$$\int_0^t f(\tau)\dot{x}(\tau) \geq 0 \tag{2.2}$$

When this criterion is not sufficed, a passivity controller is activated. Possible passivity controllers will be discussed later.

## 2.4 TDPN based method

### 2.4.1 Time domain power networks

A system interchanging position and force can be modelled like a simple network where these variables flow in opposite directions. A port is defined as a signal going into the channel and its conjugate signal going out of the channel. This conjugated signal arrives with some unknown time delay. This can be seen in fig. 2.1.

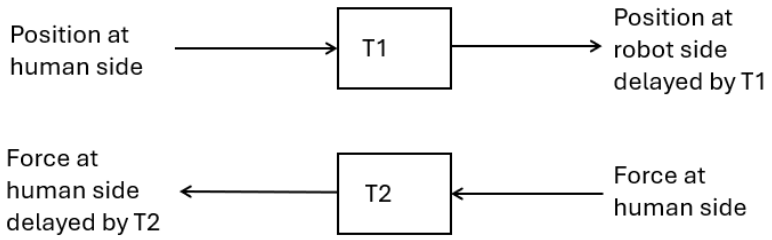



**Figure 2.1:** Two port representation of a position-force architecture

The conjugated position and force variables can be multiplied to form a power value. By integrating these variables over time, the energy flowing into and out of the channel can be defined. Mathematically speaking, these energies come from rewriting eq. (2.1). The power port at the human side is denoted with the superscript $H$, and the robot side with $R$.

$$E_{channel} = \int_0^t P^H(\tau)d\tau + \int_0^t P^R(\tau)d\tau = E^H + E^R \geq 0 \tag{2.3}$$

Artigas et al. (2010) introduces an axiom which states that these energies can be further split up depending on direction of propagation. We define energies going into the channel as absorbed, and energies going out of the channel as released energy. This can be seen here:

$$E_{channel} = E^H + E^R = E^H_{abs} - E^H_{rel} + E^R_{abs} - E^R_{rel} \geq 0 \tag{2.4}$$

We define a positive power port as $P_+$ and a negative power port as $P_-$.

$$P_+(t) = f(t)\dot{x}(t) > 0 \tag{2.5}$$

$$P_-(t) = f(t)\dot{x}(t) < 0 \tag{2.6}$$

We can define that the input, also known as absorbed, energy correspond with $P_+$ and the output, also known as released, energy with power $P_-$. So, the exact definitions of each energy shown in fig. 2.2 are:

$$E^H_{abs}(t) = \int_0^t P^H_+(t)\,dt = \int_0^t f^H(t)\dot{x}^H(t)\,dt, \quad \text{for } f^H(t)\dot{x}^H(t) > 0 \tag{2.7}$$

$$E^H_{rel}(t) = \int_0^t P^H_-(t)\,dt = -\int_0^t f^H(t)\dot{x}^H(t)\,dt, \quad \text{for } f^H(t)\dot{x}^H(t) < 0 \tag{2.8}$$

$$E^R_{abs}(t) = \int_0^t P^R_+(t)\,dt = \int_0^t f^R(t)\dot{x}^R(t)\,dt, \quad \text{for } f^H(t)\dot{x}^H(t) > 0 \tag{2.9}$$

$$E^R_{rel}(t) = \int_0^t P^R_-(t)\,dt = -\int_0^t f^R(t)\dot{x}^R(t)\,dt, \quad \text{for } f^H(t)\dot{x}^H(t) < 0 \tag{2.10}$$

We can see that these energies cannot decrease in magnitude. In other words, the absorbed and released energies are monotonically increasing.

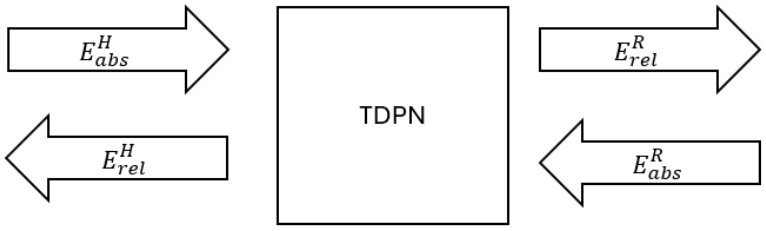



**Figure 2.2:** A two port TDPN. $H$ denotes the human side and $R$ the robot side.

### 2.4.2 Passivity observer

Mathematically, we can define two new energy flows called $E^{H2R}$ and $E^{R2H}$, for the energy flows from human to robot and robot to human, respectively. These are defined as $E^H_{in} - E^R_{out}$ and $E^R_{in} - E^H_{out}$, respectively. We can rewrite eq. (2.4):

$$E_{channel} = E^{H2R} + E^{R2H} \geq 0 \tag{2.11}$$

So, conservatively, as long as both energy flows are positive, the TDPN must be passive. We rewrite our passivity requirement to be as follows:

$$E^{H2R} \geq 0 \tag{2.12}$$

$$E^{R2H} \geq 0 \tag{2.13}$$

This makes intuitive sense. In an ideal situation, the energy absorbed by the system at an input port should be released the same immediately at the respective output port. Due to time delays, this is not the case. If more energy is absorbed by the channel than released, it means that energy is left in the channel, and passivity is kept. If more energy is released than absorbed, it means there is more energy coming out of the system than initially put in. In other words, energy has been generated.

In order to know the input energies at a certain point in time at the output port, the value of the energy absorbed by the channel has to be sent over that same channel. This adds time delay to the observed energy. At best, observed energy can be defined as follows:

$$E^H_{observed}(t) = E^R_{abs}(t-T) - E^H_{rel}(t) \geq 0 \tag{2.14}$$

$$E^R_{observed}(t) = E^H_{abs}(t-T) - E^R_{rel}(t) \geq 0 \tag{2.15}$$

Here, T represents time delay.

The exact energy at any time cannot be observed, which seems to make this method not useful. However, eq. (2.5) and eq. (2.6) show that the energies are monotonically increasing. Meaning the observed absorbed energy, and so the general observed channel energies, must be smaller than the real time energies. Equation (2.14) and eq. (2.15) can be used to determine the passivity of the system.

### 2.4.3 Passivity controller

We have two options for controllers: energy dissipation or stop energy release. Both will be discussed.

**Energy dissipation**

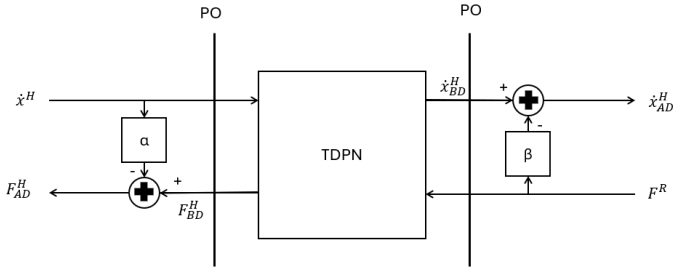



**Figure 2.3:** An example of an energy dissipation controller. At the human side, force is affected. On the robot side, velocity is affected. The lines labelled PO mark where the passivity observes are meant to be placed. Image inspired from Artigas et al. (2010).

Energy dissipation relies on calculating a relevant damping factor based on the energy generated (Hannaford and Ryu, 2002). The dissipated power, $P_D$, and the dissipated energy, $E_D$, are defined as follows:

$$P_D(t) = f_D(t)\dot{x}_D(t) \tag{2.16}$$

$$E_D(t) = \int_0^T P_D(t)\,dt = \int_0^T f_D(t)\dot{x}_D(t)\,dt \tag{2.17}$$

The damper used is a linear damper, of which an example is shown in fig. 2.3. Depending on whether the force or velocity is modified, we define the modification as:

$$f_{AD}(t) = f_{BD}(t) - \alpha(t)\dot{x} \tag{2.18}$$

$$\dot{x}_{AD}(t) = \dot{x}_{BD}(t) - \beta(t)f \tag{2.19}$$

The subscript $BD$ marks that the value is the one before (possible) dissipation or the value out of the TDPN. The modified force and velocity are marked with $AD$. This is visualised in fig. 2.3.

We fill these values into eq. (2.17), giving us the equations for the dissipated energy:

$$E_D(t) = \int_0^T \alpha(t)\dot{x}_{BD}^2(t)dt \tag{2.20}$$

$$E_D(t) = \int_0^T \beta(t)f_{BD}^2(t) \tag{2.21}$$

Initially, the damping is set to 0. When the energy observed becomes negative, energy is generated which has to be dissipated by this damper. We can therefore say that the observed energy is the energy dissipated. We assume the velocity and force to be constant between time step $T$, such that the integral can be approximated to that time step $T$. Rewriting our previous equation and adding in these comments, we get a calculation for the value of the damper:

$$\alpha(t) = \frac{E_{observed}(t)}{\dot{x}_{BD}^2(t) * T} \tag{2.22}$$

$$\beta(t) = \frac{E_{observed}(t)}{f_{BD}^2(t) * T} \tag{2.23}$$

These damping values are used in eq. (2.18) and eq. (2.19). The modification of force and velocity produce dissipation. As a result, the total energy of the channel decreases. This should make the channel passive (Artigas et al., 2010).

**Stop energy release**

Stop energy release sets a variable of a port to zero, making the observed power also 0. This forces the channel to dissipate all energy (Dresschers, 2024*c*). Effectively, one device does not observe absorbed or released energy while the other does. The other, passive device can put in more energy to be absorbed by the channel, thus making the active device passive.

# 3 Analysis

This chapter discusses how the theory discussed in the background chapter can be applied to the actual system. Firstly, the system and its architecture are discussed. Secondly, the issue of applying the observer and controller in discrete time are discussed here. Furthermore, issues that may arise from working in discrete time, within the Arduino IDE environment or the general given hardware are also addressed.

## 3.1 The DIY @ home setup

The general setup can be split up in a few subsystems. These can be seen in fig. 3.1.


**Figure 3.1:** The subsystems of the DIY @ Home setup.

Literature (Artigas et al., 2010) tells the human operator, haptic interface, robotic avatar and environment can be assumed passive. The controllers can be designed passive. Only the channel has the possibility to show non-passive behaviour.

The information sent over the communication channel are the power-conjugated values position and force. More specifically, the position-computed force (P-cF) architecture is used to determine the calculated force. A schematic of this architecture can be seen in fig. 3.2. The human operator puts into the haptic interface by what measure of position they want the robot to move. This information is then sent over the communication channel. The difference between the current position of the robot and this input, and an impedance value are then used to calculate the force. This force value is sent over to the robot in order to make it move accordingly, and sent, in order to deliver haptic feedback, to the human operator over the channel.

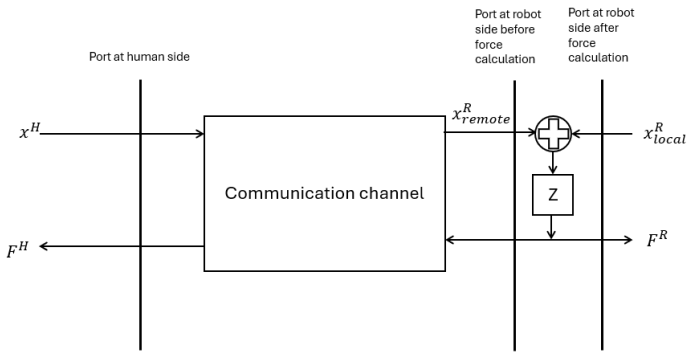



**Figure 3.2:** A simple P-cF architecture with possible ports where passivity can be observed. Figure inspired from Dresschers (2024*d*).

## 3.2 Implementing the TDPN-based method on a P-cF architecture

If no time delays were present over the communication channel, the power absorbed by the channel at an input port must be equal to the power released at the respective output port (Artigas et al., 2010). For passivity, the power absorbed can also be more than the power released.

We can represent the system in fig. 3.2 as a bond graph. Each setup is a moving inertia represented with a I-type element. The computed force architecture works by using a constant impedance to calculate the force, which behaves like a linear spring (Dresschers, 2024*d*). So, we represent this impedance with a C-type element.

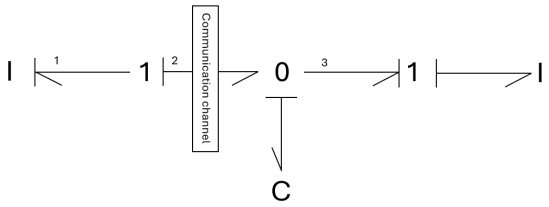



**Figure 3.3:** Bond graph representation of the P-cF architecture. The left hand side of the communication channel represents the human operator side, the right the robot side. 1, 2 and 3 are the relevant power ports.

From fig. 3.3, we can see that we have two power ports at the human interface side. Those are ports 1 and 2. The relations of the force and velocity at the 1-junction connecting these ports, are as follows:

$$f_1 = -f_2 \tag{3.1}$$

$$\dot{x}_1 = \dot{x}_2 \tag{3.2}$$

Equation (3.1) is represented in the script by having a difference between the remote, $f_{remote}$, and local force, $f_{local}$. $f_1$ is the local force and $f_2$ the remote, here. Equation (3.2) shows that the velocity is the same at both power ports. This means that the power at port 1 is the negated power at port 2:

$$P_1 = -P_2 \tag{3.3}$$

So, it is important to know whether to measure the power at port 1 or at port 2, as their powers are opposite. The TDPN-based method looks at the energy flows over the communication channel. The direction of port 2 goes through the channel, while port 1 goes to the haptic interface. So, we analyse the power at port 2 and have to use the remote force to calculate the power.

At the robot side, the power can be analysed at either port 2 or port 3. In different words, the power can be analysed before or after the force is computed. These ports are also visualised in fig. 3.2.

As mentioned before, the TDPN based method works by assuming the energy is monotonically increasing over time. The controller impedance used to compute the force behaves like a spring. If we were to measure at power port 3, we effectively have the situation sketched in fig. 3.4, where the impedance is visualised as a spring. The spring can release its energy towards the human or robot side, or absorb energy.

For this scenario, we assume all initial energy levels to be zero:

$$E^H_{in} = E^H_{out} = E^R_{in} = E^R_{out} = 0 \tag{3.4}$$

We can imagine a situation where energy is absorbed by the spring from the human side. Based on the force calculation, the spring releases energy towards the human side, rather than the robot side (Dresschers, 2024*d*). This situation is visualised in fig. 3.4.

Now, the absorbed energy at the human side is increased, but not at the robot side. From the human to the robot side, passive behaviour is measured, as the absorbed energy is larger than the released energy. However, the absorbed energy at the robot side is zero, while the released energy of the spring is being interpreted at the human side as generated energy. This would give a false positive for a non-passive channel. If we were to measure at port 2, this scenario cannot happen. So, to apply the TDPN based method, we have to place our observer at port 2.

Figure 3.2 shows that power port 2 is between the channel and computed force architecture. Here, it is shown that in order to calculate this power, we have to use the position received from the human interface setup, $x_{remote}$, rather than its locally measured position, $x_{local}$.

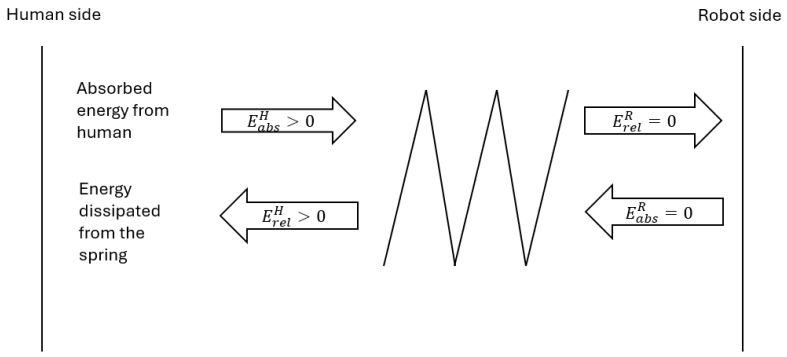



**Figure 3.4:** P-cF architecture represented as a spring

## 3.3 Controller choice

As mentioned in section 2.4.3, we have two types of passivity controllers: stop energy release and add dissipation. Both controllers impact the transparency of the system, because they change the actual values of the force or velocity in such a manner that it does not represent the actual states of the system. Stop energy release temporarily loses complete transparency, as one variable is set to zero.

As it seems now, add dissipation should be the best method, as the least transparency is lost. However, a situation can arrive that the increase of generated energy is too large that the controller cannot keep up (Dresschers, 2024*c*). In this situation, a more drastic measure like the stop of energy release may be necessary. Franken et al. (2011) suggest the use of both controllers. The add dissipation method is used first. If the system generates more than a maximum allowed level of energy, the stop energy release method is used. With this method, passivity is guaranteed while having as little transparency loss as possible.

## 3.4 Discrete passivity observer

The observer has to be made discrete. A method of doing so is by using discrete integration methods. An example from Hannaford and Ryu (2002) of such a method is shown below.

$$E_{observed}(n) = \sum_{N}^{k} f(k)\dot{x}(k) \tag{3.5}$$

Discrete integration has a few issues. Firstly, any method of discrete integration is an approximation. Secondly, this method requires discrete differentiation, because we receive position

values, not velocity. This is also an approximation. Therefore, this method cannot reliably deliver accurate results of the current energy.

Another method can be found from literature (Stramigioli et al., 2005), which does take into account that force is calculated based on position. This method looks at the energy difference between the last and current time step and adds that to the total energy count.

$$E(n) = E(n-1) + \Delta E \tag{3.6}$$

Here, $\Delta E$ is defined as follows:

$$\Delta E = E_{n-1}^{n} = \int_{n-1}^{n} f(k)\dot{x}(k)dk \tag{3.7}$$

The force value is only updated once every next time step. Between time steps, the force stays constant (Stramigioli et al., 2005). In the energy equation, this means that the forces at $n$ and $n-1$ are both equal to $f(n-1)$. The force element can be taken out of the integral:

$$\Delta E = f(n-1)\int_{n-1}^{n} \dot{x}(k)dk \tag{3.8}$$

Now, we only integrate over a single velocity value. This translates to the difference between current and previous position. So we get our final calculation:

$$\Delta E = f(n-1)(x(n) - x(n-1)) \tag{3.9}$$

This method only includes variable already known, and does not require the use of estimations. Thus, the energy can be calculated accurately and reliably using this method.

In accordance to the analysis made in section 3.2, eq. (3.9) can be written to comply to the human and robot side specifically. This means we have to use the remote force at the human setup, and the remote position at the robot setup. We rewrite eq. (3.9) for the human operator and robot:

$$\Delta E^{H} = f_{remote}(n-1) * (x(n) - x(n-1)) \tag{3.10}$$

$$\Delta E^{R} = f(n-1) * (x_{remote}(n) - x_{remote}(n-1)) \tag{3.11}$$

In the existing script at the human side, the force and position values are interchanged first. Then, passivity is observed. Afterwards, the received, $f_{remote}(n)$ value is negated to the local force, $f_{local}(n)$, variable. The local force is then applied to the motors to deliver haptic feedback. The script loops, and $f_{remote}(n)$ is updated. When the passivity is calculated, $f_{local}(n)$ is yet to update, and is equal to $-f_{remote}(n-1)$. So, we can replace $f_{remote}(n-1)$ with $-f_{local}(n-1)$, making the equation for the human operator side as follows:

$$\Delta E^{H} = -f_{local}(n-1) * (x(n) - x(n-1)) \tag{3.12}$$

From equations eq. (2.5) and eq. (2.6), we know that positive power is absorbed energy and negative power is released energy. Therefore, we define two observed energies: released and absorbed. These are calculated like eq. (3.6) We can use eq. (3.9) to determine what energy is absorbed or released. If $\Delta E$ is positive, we add it to the absorbed energy. If it is negative, we add it to the released energy. This gives us the equations:

$$E_{abs}(n) = E_{abs}(n-1) + \Delta E(n), \quad \text{if } \Delta E(n) \text{ is positive} \tag{3.13}$$

$$E_{rel}(n) = E_{rel}(n-1) - \Delta E(n), \quad \text{if } \Delta E(n) \text{ is negative} \tag{3.14}$$

If we use eq. (3.11) and eq. (3.12), we can define eq. (2.7), eq. (2.8), eq. (2.9) and eq. (2.10) discretely. We keep in mind that the time delays can vary depending on the direction of the flow of information (Artigas et al., 2010). This makes our passivity criteria:

$$E^H_{observed}(n+1) = E^R_{abs}(n - T^{R2H}) - E^H_{rel}(n) > 0 \tag{3.15}$$

$$E^R_{observed}(n+1) = E^H_{abs}(n - T^{H2R}) - E^R_{rel}(n) > 0 \tag{3.16}$$

## 3.5 Discrete passivity controller

### 3.5.1 Add dissipation

As stated in the previous subsection, the force has to be modified at the human side and the velocity at the robot side. From Dresschers (2024*b*), it can be found that in order to apply an add dissipation controller, the sample time and the force or velocity of the next time step, depending on the controller, have to be known. This is because the passivity controller is applied on this time step.

We can know the force at the next time step, as it is computed before it is applied to the motors and stays constant for the duration of the time step. The velocity, on the other hand, we cannot know as the next position is unknown. The velocity has to be estimated. The sample time can be estimated with the previous sample time.

The equations will be as follows:

$$\alpha(n) = \frac{E_{observed}(n)}{T\dot{x}^2(n)} \tag{3.17}$$

$$\beta(n) = \frac{E_{observed}(n)}{Tf^2(n)} \tag{3.18}$$

This limits the output as follows:

$$f_{modified}(n+1) = f_{new}(n+1) - \alpha(n)\dot{x}(n) \tag{3.19}$$

$$\dot{x}_{modified}(n+1) = \dot{x}_{new}(n+1) - \beta(n)f(n) \tag{3.20}$$

In eq. (3.17), we have a velocity value rather than a position. We estimate the velocity with the position difference between current and last time step, and the sample time $T$.

$$\dot{x} = \frac{x(n) - x(n-1)}{T} \tag{3.21}$$

For eq. (3.20), the velocity is modified, while a position value is communicated. In a time continuous system, we can estimate the position in velocity using:

$$x(t) = \int_0^t \dot{x}(t)\,dt \tag{3.22}$$

The modified velocity, $\dot{x}_{modified}$, also includes a damper value. We rewrite the equation to include this damper:

$$x_{modified}(t) = \int_0^t \dot{x}_{modified}(t)dt = \int_0^t \dot{x}(t)dt - \int_0^t \beta(t)f(n)dt = x(t) - \int_0^t \beta(t)f(n)dt \quad (3.23)$$

To see how this controller behaves, a situation is sketched. We assume the controller is continuously used at all times, but $\beta(n)$ is zero when the system has passive behaviour. We assume the system is activated at time $T_0$, where we assume the position to be 0. Before a time $T_1$, the system is passive. Between $T_1$ and $T_2$, the system is non-passive. Afterwards, the system is again passive. We define a new variable $x_{pass}$. This is the position which guarantees passivity.

Before the time $T_1$, the damping is set to 0, as passivity is kept. The passive position before $T_1$ is calculated as follows:

$$x_{pass}(t)|_{T_0}^{t_1} = \int_{T_0}^{t_1} \dot{x}dt - \int_{T_0}^{t_1} \beta(t)f(t)dt = x(t_1) - 0 = x(t_1) \quad (3.24)$$

In the second block, the controller is activated. The passive position before $T_2$ becomes as follows:

$$x_{pass}(t)|_{T_1}^{t_2} = x_{pass}(t)|_{T_0}^{T_1} - \int_{T_1}^{t_2} \dot{x}dt - \int_{T_1}^{t_2} \beta(t)f(t)dt = x(t_2) - \int_{T_1}^{t_2} \beta(t)f(t)dt \quad (3.25)$$

It can be seen that the damping affects the position. In the third time block, passivity is restored. So, the position after $T_2$ becomes:

$$x_{pass}(t)|_{T_2}^{t_3} = x_{pass}|_{T_0}^{T_1} + x_{pass}|_{T_1}^{T_2} - \int_{T_2}^{t_3} \dot{x}dt - \int_{T_2}^{t_3} \beta(t)f(t)dt = -\int_{T_1}^{T_2} \beta(t)f(t)dt + x(t_3) - 0 \quad (3.26)$$

We can see that the net position modification caused by damper is still present. Furthermore, if we were to continue this block such that between a time $T_3$ and $T_4$, the controller is activated and after $T_4$ is passive once more, we get for the position after $T_4$ as follows. $t_5$ is used to denote the time after $T_4$.

$$x_{pass}(t)|_{T_4}^{t_5} = \int_{T_1}^{T_2} \beta(t)f(t)dt - \int_{T_3}^{T_4} \beta(t)f(t)dt + x(t_5) \quad (3.27)$$

This behaviour is in line with the theory discussed in Dresschers (2024*d*). Every time the velocity is modified, this leads to a mismatch. If the controller is activated again, the new mismatch comes on top of previous ones. So, each impact the controller has on the system is permanent and additive.

The situation above is sketched using time continuous variables. It includes an integral to turn the damper velocity into a position value. This integral can be estimated with a Riemann's sum to make it discrete. $T$ is the sample time.

$$x_D(n) = \sum_{T_0}^{n} \beta(n)f(n)T \quad (3.28)$$

### 3.5.2 Stop energy release

The setup has a P-cF architecture. Position values, rather than velocity, are transmitted over the system. So, while the local force at the human side can be simply set to 0, the velocity at the robot side has to be set to 0 using a different method.

We use the same situation used in section 3.5.1. For ease, we assume the initial position $x(T_0)$ to be zero. Before $T_1$, the system is passive. We can define the position as follows:

$$x_{pass}(t)|_{T_0}^{t_1} = \int_{T_0}^{t_1} \dot{x}(t)\,dt = x(t_1) \tag{3.29}$$

In the first time block, the position is the same as the current position. For the next block, the stop energy release method is used, so the velocity becomes 0.

$$x_{pass}(t)|_{T_1}^{t_2} = \int_{T_1}^{t_2} \dot{x}(t)\,dt = x_{pass}(t)|_{T_0}^{T_1} + \int_{T_1}^{t_2} \dot{x}(t)\,dt = x(T_1) + \int_{T_1}^{t_2} 0\,dt = x(T_1) \tag{3.30}$$

We can see, in order to have a velocity of 0, the position should deviate from the received position $x(t_2)$, to the last position where the system was passive. In this case, that is $x(T_1)$. This means $x(t_2)$ is still being received, but overwritten by $x(T_1)$.

When the system becomes passive again, the passive position becomes as follows:

$$\begin{aligned} x_{pass}(t)\Big|_{T_2}^{t_3} &= \int_{T_1}^{t_3} \dot{x}(t)\,dt = x_{pass}\Big|_{T_0}^{T_1} + x_{pass}(t)\Big|_{T_1}^{T_2} + \int_{T_2}^{t_3} \dot{x}(t)\,dt \\ &= x(T_1) + 0 + \int_{T_2}^{t_3} \dot{x}(t)\,dt \\ &= x(T_1) + x(t_3) - x(T_2) \end{aligned} \tag{3.31}$$

It can be seen that the velocity values at $T_1$ and $T_2$ are included here as well. This is in line with theory discussed by Dresschers (2024*d*). Modifying the position values leads to a permanent mismatch in position values between the human interface and robot setups. We have to take this mismatch into account when designing our passivity controller. We rewrite the equation as follows:

$$x_{pass}(t)|_{T_2}^{t_3} = x(t_3) - (x(T_2) - x(T_1)) \tag{3.32}$$

We extend the situation such that the controller is used again between a time $T_3$ and $T_4$. After $T_4$, referred to as $t_5$, the position would be:

$$x_{pass}(t)|_{T_4}^{t_5} = x(t_5) - (x(T_2) - x(T_1)) - (x(T_4) - x(T_3)) \tag{3.33}$$

The mismatch, which happened during the first time block when the controller was activated, is still included, similarly to the add dissipation method. This means each difference in position caused by the controller must be included in the equation to correctly represent the permanent impact the controller has on the system (Dresschers, 2024*d*).

We can include both controllers to get the final definition of the applied controller of:

$$x_{pass} = x(n) - \sum (x(n_{nexttimepassive}) - x(n_{lasttimepassive})) - \sum \beta(n) f(n) T \tag{3.34}$$

It is important to note that $x_{pass}$ replaces $x_{remote}$ to compute the force. This means that $x_{pass}$ has to be used to determine the passivity, rather than $x_{remote}$ as described previously by eq. (3.11). The equation of the observed energy at the robot side, $\Delta E^R$, changes to:

$$\Delta E^R = f(n-1)(x_{pass}(n) - x_{pass}(n-1)) \tag{3.35}$$

## 3.6 Hardware limitations

### 3.6.1 Angle sensor issues

The angle sensor is the AS5048. A test was constructed to evaluate the quality of the sensor. The difference between current and previous angle, from now on referred to as angle difference, is plotted. The master setup is turned off and only the slave setup is used. The slave setup is given a reference angle at the centre of the setup. This effectively makes the slave setup a spring, returning to the initial position after a position change.

When the setup is turned on, it is expected the measured angle difference to be 0, as the setup hasn't moved. After the setup has been moved, the angle difference should spike up and then return to 0, as the setup behaves like a spring.

The test was conducted three times per setup.

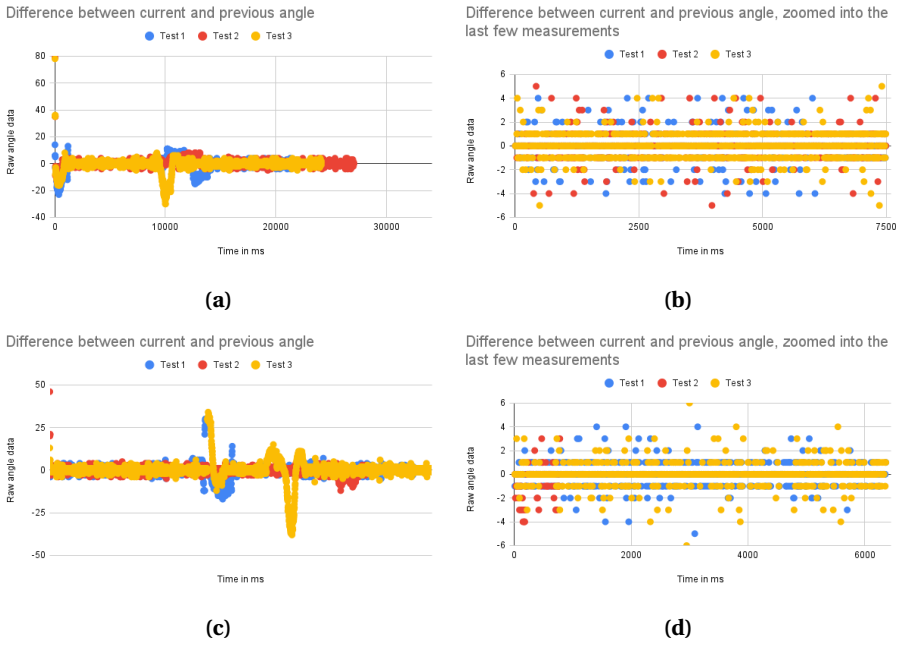



**Figure 3.5:** Angle measurements of both setups. The right graphs show the last few thousand measured samples. The test repeated, such that each setup was the master and slave setup once.

Figure 3.5 shows that the first few angle difference measurements spike up as high as 80 at one setup and 45 at the other, before stabilizing around 0. This shows the initialization issues.

A proposed solution to the initial mismeasurements is to write initialization code in the setup part of the script. It should guarantee the measurements are set around 0 before the channel is used.

From fig. 3.5b and fig. 3.5d, we can see that the sensor at a standstill still measures an angle difference. This is due to noise of the filter. This noise could potentially lead to issues when calculating the energy. The experiment described in the previous section was repeated, but the raw steer was measured instead. For now, we say this steer is effectively our raw force.

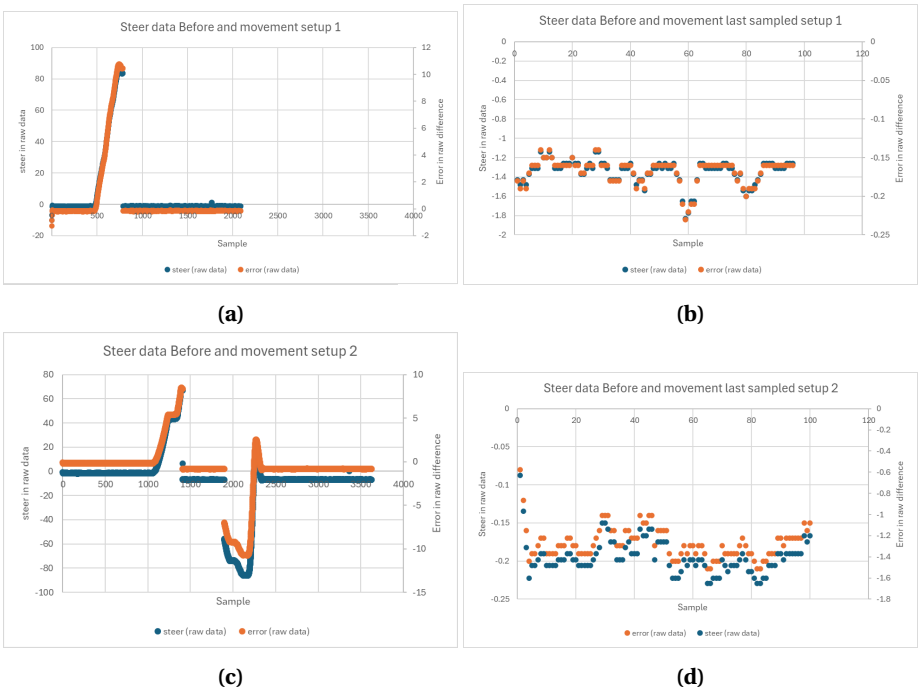



**Figure 3.6:** Steer error after moving in raw data of both setups. The test was repeated for each setup. The data shown at the right-hand side are the last few samples of the graphs on the left-hand side.

From fig. 3.6b and fig. 3.6d, we can see that after movement, the steer never settles at 0. This is due to a difference between remote and local position. In the P-cF architecture, whenever there is a difference between these positions, a force, or in this case steer, value is calculated to drive the motor to match the positions. In this case, however, the force cannot move the system to the desired position, as the computed steer is not large enough to overcome the friction of the system.

This is natural behaviour of the system. However, this behaviour could impact our energy readings in combination with the aforementioned position noise.

### 3.6.2 Steer to force transformation

The system uses a steer value to drive the motors, rather than an actual force. This steer value is a PWM value. For our method, it is required to have a force in Newton.

Bergsma (2024) researched a method to convert the PWM value to a force in Newton. This was achieved by measuring the force at different PWM duty cycles. The researched showed a linear relation between a PWM duty cycle of 125. A constant was found which relates the force to the steer, which is 0.0714.

$$force = 0.0714 * steer \tag{3.36}$$

### 3.6.3 Rotational domain

It can be noted that the DIY @ home setup rotates, instead of moving linearly. The system can therefore also be analysed in the rotational domain. For the implementation, it was chosen to use torque and angle values rather than force and position.

The angle is already being measured in the system. The torque is related to the measured force as follows:

$$torque = 0.008636 * force \tag{3.37}$$

The exact calculation of this constant can be found in appendix A.

# 4 Design and Realisation

This section will go over the implementation of the TDPN based method to the existing architecture as discussed in chapter 3. It will go over how the energy information will be sent to the setups and the implementation of the passivity observer and controller. To further show the working of the code, pseudocode is provided.

## 4.1 I2C communication

The absorbed energy information has to be sent over a communication channel. In the original script, I2C communication was already established to send steer and angle values over using the $Wire.h$ library. This existing architecture can be expanded to send out send the energy data as well.

The energy data has to be sent as bytes for this communication method. The variables $E\_abs\_send$ and $E\_abs\_rec$ variables are defined to save the transmitted and received energy information, respectively. Both are defined as unions, consisting of float and byte[4] type variables. Floats consists of 4 bytes in the Arduino IDE environment. This way, we can save the float energy data and are able to send this data as a type byte.

For the transmitted information, a simple $Wire.write(E\_abs\_send.\_byte)$ was added after the first write statement. This statement is used to send steer and angle data. This way, a standard has been set up where the first four bytes are the individual variable data, and the last four consists of the energy data, as can be seen in the table below.

| | First transmitted variable | amount of bytes | Second transmitted variable | Amount of bytes |
|---|---|---|---|---|
| Human interface side | angle_raw._long | 4 | E_abs_send._float | 4 |
| Robot side | steer._float | 4 | E_abs_send._float | 4 |

Due to the standard set, at the receiver end, the first four bytes have to be interpreted as steer or angle data, and the last four as receiver energy. In the table below, this is shown. Two while loops are added to the code. The first loop saves the first 4 bytes as the variable data. The second loop saves the last four to the variable $E\_abs\_rec.\_byte$. This is done using the $Wire.read()$ function.

| | First received variable | amount of bytes | Second received variable | Amount of bytes |
|---|---|---|---|---|
| Human interface side | steer_remote._byte | 4 | E_abs_rec._float | 4 |
| Robot side | angle_raw_remote-._long | 4 | E_abs_rec._float | 4 |

The pseudocode for the human interface and robot side, respectively, are shown below. It is noted that even though the human interface is the I2C slave and the robot is the I2C master.

```
void I2C_requestEvent{
   WRITE angle data
```

```
        WRITE absorbed energy data
    }

    void I2C_receiveEvent{
        READ first 4 bytes AS steer data
        READ last 4 bytes AS absorbed energy data
    }
```

```
    BEGIN transmission
    WRITE steer data
    WRITE absorbed energy data
    STOP transmission

    REQUEST data transmission from I2C slave
    READ first 4 bytes AS angle data
    READ last 4 bytes AS absorbed energy data
```

### 4.2 Angle initialization code

Section 3.6.1 describes the issues of the angle sensor. For the initialization code, a while loop can be instated, which makes sure the angle difference is 0 in the setup part of the code. It measures the current angle and saves its last angle. If both are the same, the angle difference is 0, and the sensor is correctly calibrated.

```
    while(check < 3){
        READ current angle
        IF(current angle == previous angle){
            check++
        } else {
            check = 0
        }
        previous angle = current angle
    }
```

### 4.3 Passivity observer

In order to calculate the energy, first the force and angle difference have to be calculated. We can use the built-in function $AS5048\_RawToRad()$ to convert the raw long type angle data of the sensor to a float type angle variable in radians.

To convert raw steer data to a torque, a constant is defined, called $tor\_con$. This constant is the multiplication of the constant used to go from raw steer value to force, as described in section 3.6.2, and the constant used to go from force to torque, as described in section 3.6.3. Multiplying this constant with the steer gives the torque in Nm.

At the human side, eq. (3.12) has to be implemented. The local force is used to calculate the torque. The power is negated as described by the equation.

As mentioned in section 3.5.2, we have to implement eq. (3.35), rather than eq. (3.11), to calculate the energy at the robot side. This equation uses the passive angle variable rather than the remote angle to calculate the energy.

An if-statement is placed for the implementation of eq. (3.13) and eq. (3.14). It determines whether energy is absorbed or released. Note that the absorbed energy is put into the variable $E\_abs\_send.\_float$, because this variable is sent to the other device.

Lastly, eq. (3.15) or eq. (3.16), depending on the script, are implemented as well. This line calculates the observed channel energy to determine passivity based on the released energy of the channel measured at this setup $E\_rel$ and the absorbed energy received from the other setup $E\_abs\_rec.\_float$.

Below, we see the pseudocodes of the implementations for the human interface and robot, respectively.

```
angle difference = current angle - previous angle

torque = tor_con*steer_local

E_add = -angle difference*torque

if(E_add > 0){
    E_abs_send += E_add
} else {
    E_rel -= E_add
}

E_obs = E_abs_rec - E_rel
```

```
angle difference = current passive angle - previous
↪ passive angle

torque = tor_con*steer

E_add = angle difference*torque

if(E_add > 0){
    E_abs_send += E_add
} else {
    E_rel -= E_add
}

E_obs = E_abs_rec - E_rel
```

### 4.4 Passivity controller

The passivity controller is applied after the passivity is observed. An if-statement is placed which determines whether passivity is met. If not, it applies a passivity controller.

At the human operator side, the damper is applied to the local steer. Equation (3.17) and eq. (3.18) are applied here. The velocity is estimated using eq. (3.21). The sample time T is estimated using the loop time, divided by 1000.0000. This is done to correctly convert the long type variable loop time to a float type variable in seconds.

The stop energy release method is applied by setting the steer to 0. It can be seen that the stop energy release method is activated only if a maximum amount of energy is generated or if the angle difference is 0. The latter statement was added to guarantee no unlimited damping could be calculated.

If the system is passive, the damper is set to 0, and the steer is set to the negative of the received steer value, as the system was before.

The pseudocode of the passivity controller at the human side can be seen below.

```
if(E_obs < 0){
    if(E_obs > minimum generated energy AND angle
↪ difference != 0){
        T = (float)loopTime/1000.0000
        Damping = E_obs/((angle difference/T)^2*T)
        steer += Damping*(angle difference/T)/tor_con
    } else{
        local steer = 0
    }
} else {
    damping = 0
    local steer = -remote steer
}
```

At the robot side, the add dissipation method modifies the angle. We can see eq. (3.18) being applied here for the add dissipation method. The sample time is again determined by the loop time. The calculated modification to the angle is saved to a new variable $delta\_x$, which is used to represent the loss in transparency due to the modification of the angle, as described in section 3.5.1. The damping is applied to the passive angle variable.

To apply the stop energy release method as described in section 3.5.2, we use the passive angle variable. When activated, the last passive angle variable is used. The angle difference between the passive angles before and after the controller has been activated is calculated. This value is added to $delta\_x$, to correctly represent the mismatch caused by this controller.

It can be seen that the stop energy release method is only applied after a maximum amount of energy is generated or if the torque is 0. The latter statement was added to guarantee no unlimited damping could be calculated.

If neither controller is applied, the damping is set to 0. The passive angle, which includes the mismatches caused by the controllers and the current remote angle, is used.

The pseudocode of the passivity controller at the robot side can be seen below.

```
PASSIVITY OBSERVER

if(E_obs < 0){
    if(E_obs > minimum generated energy AND torque != 0){
        T = (float)loopTime/1000.0000
        Damping = E_obs/((torque)^2*T)
        delta_x += damping*torque*T
        passive angle = remote angle - delta_x
    } else{
        passive angle = last passive angle
        delta_x += next passive angle angle - last
↪ passive angle
    }
} else {
    damping = 0;
```

```
        passive angle = remote angle - delta_x
    }

    READ current angle

    error = passive angle - current local angle

    steer = error*kp \\ existing P-cF architecture
```

# 5 Evaluation

This section evaluates the implementation discussed in chapter 4. It validates the pieces of code discussed there. Due to time constraints, the add dissipation method could not be tested.

## 5.1 Passivity observer

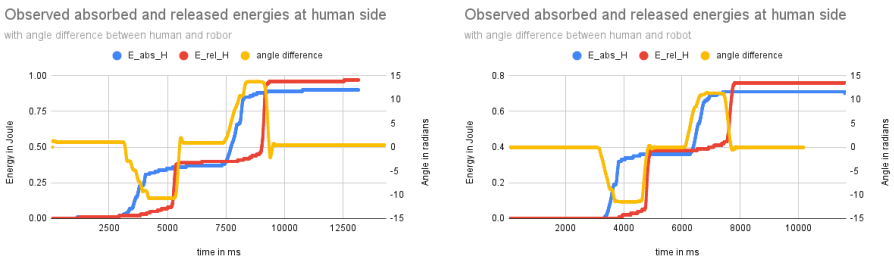
The passivity observer was tested with two tests. Firstly, it was tested whether the absorbed and released energies are correctly and accurately measured. Secondly, it was tested whether passivity was correctly measured.

### 5.1.1 Assignment of absorbed and released energies

The observer should correctly assign the observed energies as absorbed or released. It is also expected that it accurately measures each energy.

The test constructed to determine these criteria was to turn off the motors of the robot setup. This setup was not moved, and leaving it at middle position. The angle of the robot is kept constant this way, but the setup calculates a steer whenever the human interface setup moves. If the human interface setup is moved, a force is calculated to return the setup to the middle position. This behaviour is similar to a spring, as both return to an initial position after a movement.

When the human setup is moved, the system absorbs energy from the human operator. When the system is returns to the initial position, this energy is released. We expect that the absorbed energy first increases, as the setup is moved away from its initial position. The released energy value should increase when the setup returns. As the system returns to its initial position, we expect all energy absorbed by the system to be released. So both energies should be the same after movement.




**(a)** **(b)**

**Figure 5.1:** Assignment of the observed energy to released or absorbed energy. The test was repeated, such that each setup was the human interface and robot setup once

From fig. 5.1 we can see that the absorbed energy increases as the setup is moved away from the initial position. When returning to the initial position, we can see that the released energy increases. This is the expected behaviour.

An unexpected behaviour is seen when the setup is moved away from its initial position, and kept there. The position is seen not to move, thus having a velocity of 0. Yet, the observed absorbed and released energies increase. It is expected that when the velocity is 0, neither of the energies should increase.

This behaviour is explained by section 3.6.1, specifically fig. 3.5. We can see that at a standstill, a difference in angle can still be measured. The force at standstill, seen in fig. 3.6, is almost at

all time present at standstill, but rather small. Therefore, at standstill, a small levels of energy can be measured. This is also seen in fig. 5.1a towards the end of the measurement.

The test causes the force to become rather large, as big differences in angles are used. This means the standard error in position is being amplified way more at a large difference, than when the angle difference is small around the middle point. So, the increase in both absorbed and released energy can be due to the relatively large forces applied.

The last notable observation from this test is that in both figures that the released energy becomes larger than the absorbed energy after movement. It was expected that the energies stayed the same. This implies the system is unstable. In reality, the system is stable, as is seen from the angle difference measurements.

As mentioned before, passivity is a sufficient statement for stability, but not a necessary one. A system can show non-passive readings, yet be stable. Because the angle and steer values are still sent over the communication channel, time delays are still present in the system. This explains the apparent non-passive observation seen from the figures.

### 5.1.2 Passivity

The passivity observers should correctly measure whenever a system becomes non-passive. It should also correctly show when the system is passive.

Firstly, a test was set up to see whether the observer correctly measures passive states of the system. No extra time delays were added. This guaranteed the frequency of the system was set between 300 and 500 Hz, which should guarantee stability. We expect the observed energy from human to robot and back to be 0, as this is the ideal case, where all energy absorbed is released.

Secondly, a test was set up to see whether the observer correctly observes non-passive behaviour. It was found that the original architecture is unstable when 8ms time delays were added. This delay is added to the system. It is expected that both observed energies from robot to human and back become below 0, and decrease over time, as the system becomes more unstable. Each test was done twice, with either setup being the slave and master setup once.

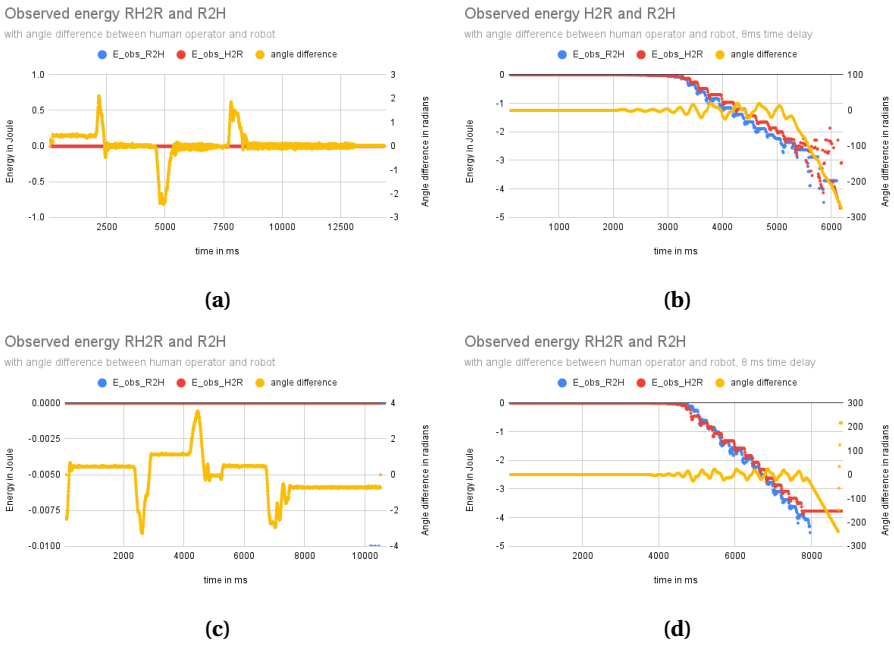



**Figure 5.2:** Observed energies from robot to human and back. The plots on the left side show the observed energy when the system is passive, the right show the observed energies when the system is non-passive. The test repeated, such that each setup was the master and slave setup once.

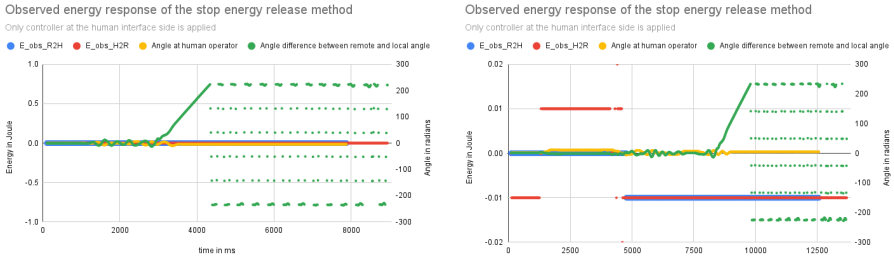
Figure 5.2a and fig. 5.2c show the system in a passive state, while fig. 5.2b and fig. 5.2d show the system in a non-passive state. In the passive case, we can see that the observed energies stays 0. In fig. 5.2c, we see that towards the end, the energy from robot to human becomes -0.01, implying non-passive behaviour. This is due to the noisy behaviour of the system, as discussed in section 3.6. All in all, the observer measures passivity correctly.

## 5.2 Stop energy release

We expect the stop energy release method to guarantee the passivity of the system by setting the torque or angular velocity to 0. From Franken et al. (2011), we find that communication between setups is completely lost, when both controllers are activated at the same time. This phenomenon is called black out. Towards the final measurements in fig. 5.2a, we can see that energy lower than 0 could still be measured, while the system was passive. This implies black-out mode can easily be reached from passive states. So, both controllers are also tested separately, as to not have the issue of black out.

## 5.3 Human interface setup

To test the controller at the human operator, we apply the 8 ms time delay, guaranteeing non-passive behaviour. Because the robot side does not activate its passivity controller, we expect it to show active behaviour. The human side should show no signs of active behaviour with an activated passivity controller. The observed energies, the current position of the human interface and the angle difference between the human interface and robot are plotted.




**(a)** **(b)**

**Figure 5.3:** Observed energies when only the stop energy release method is applied to the human interface side. The position at the human side and the position difference between remote and local angle at the robot side are plotted. The test repeated, such that each setup was the master and slave setup once.

It seems from fig. 5.3a that the energy measured at the robot side never becomes below 0, yet the controller seems to activate. This is due to the method of measuring. The measured values are displayed with 2 decimal precision, while the controller can have values smaller than -0.01. This means, in rounding, we get values of -0.00. We can still see whenever non-passive behaviour happens from the angle data. Such measurements will also appear in fig. 5.4a and fig. 5.4b.

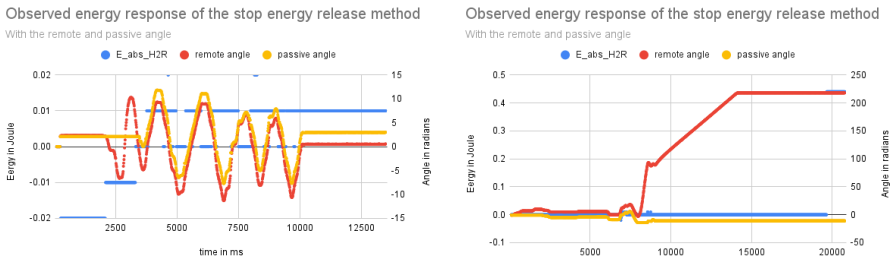
By comparing the "angle difference between remote and local angle" and "angle at human operator" lines in fig. 5.3, we can see that the robot side eventually becomes unstable. The measured angle at the human side shows that the human interface setup stays stable. The human operator side clearly does not follow the unstable, non-passive behaviour shown by the robot side. This proves the controller guarantees stability at the human operator side.

Even though the passivity controller is applied, the observed energy $E^{R2H}$ stays negative in fig. 5.3b, implying non-passive behaviour. This is because passivity is a sufficient statement for stability, not a necessary one. A system can be stable, yet non-passive.

## 5.4 Robot setup

To test the controller at the robot side, we apply the 8 ms time delay, guaranteeing non-passive behaviour. The passivity controller should guarantee stable behaviour at the robot side.

Rather than the remote angle, the passive angle is used to compute the force. This passive angle takes the mismatch in angle as a result from the use of modifying the position into account, as mentioned before in section 3.5.2. When both remote and passive angles are plotted, we expect an offset between the values to appear when the passivity controller is activated. When the controller is not used, we expect the passive angle to follow the remote angle.




**(a)** **(b)**

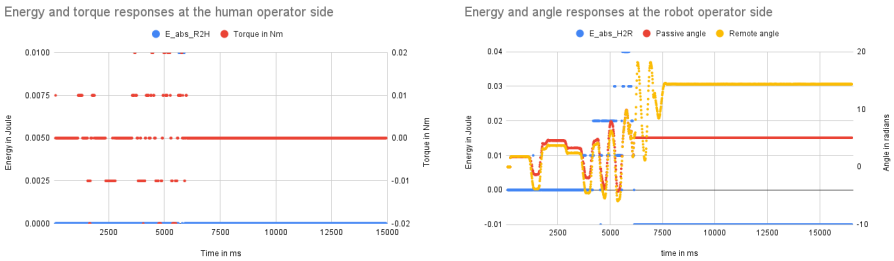
**Figure 5.4:** Observed energy, remote and passive angle at the robot side when only the stop energy release method is applied to the robot side. The test repeated, such that each setup was the master and slave setup once.

Figure 5.4a has an initial observed energy lower than 0. The controller is activated. We can see that the passive angle is set to 0, even though the received, remote angle value differs. Figure 5.4b shows that the observed energy is less than 0 around 9000 ms. Afterwards, the passive angle does not follow the remote angle and stays constant to the position before the system became non-passive. This is in line with the expected behaviour.

Figure 5.4a shows that the system became stable after 3000 ms. The passive angle starts following the remote angle, but with an offset. This offset represents the mismatch in angle as a result from the use of the controller. This was also the expected behaviour. It can be concluded that the passivity controller at the robot side shows the expected behaviour.

## 5.5 Both setups

In order to test both systems, a time delay of 8 ms was added. Due to the low levels of energy observed, it is expected that the system goes into black out quickly.




**(a)** **(b)**

**Figure 5.5:** On the left, the observed energy and applied torque value at the human side. On the right, the observed energy, passive and remote angles at the robot side.

Figure 5.5 shows that until a time of around 6000 ms, the system is mostly passive. We can see that the torque at a torque value is applied at the human side. At the robot side, it sometimes senses non-passive behaviour. We can see that as a result, a difference appears between remote and passive angle.

After 6000 ms, both sides sense non-passive behaviour and activate the respective controllers. We see that the torque becomes zero at the human operator side and the passive position stays constant. At this point in time, no energy can be calculated. As a result, no absorbed energies are sent to make the other device passive.

These results show when both controllers are applied, the system stays passive. Black out is easily reached when both observed energies are less than 0 at the same time, activating the controllers at both sides.

# 6 Conclusions and Recommendations

## 6.1 Conclusions

The goal of this thesis was to apply the time domain passivity control based method as a way of time domain passivity control to the DIY @ Home Setup of Avatars.report. This would solve the issue of energy generative behaviour that arises with time delays in telemanipulation setups.

The main challenges for applying the TDPN based method were the implementation to the P-cF architecture making the theory applicable to the discrete time Arduino environment. The hardware setup was also analysed. Besides initialization issues, it was determined the limitations of the hardware would not be an issue for achieving the goal.

The passivity observer had to be made discrete. First, discrete integration methods were discussed, but they could not deliver accurate results. Another method, as found in Stramigioli et al. (2005), was used instead. It was tested whether the method delivered accurate energy measurements, which was achieved by having the human interface setup effectively act like a spring. Some unexpected results came about, but these were explained by hardware limitations and the existing time delays in the system. In the end, the passivity observer correctly measured the energy.

The TDPN based method was implemented to the P-cF architecture by placing the passivity observer at the robot side before the force was computed. When tested, the observer observed unstable behaviour correctly.

The stop energy release method had to be adapted to the P-cF architecture. While the force could be set to 0 at the human interface side, the robot side had to use a method which uses positions, rather than velocities. A method was argued which uses the last position where the system was passive. This position was used to compute the force, rather than the position received from the human interface design when the system became non-passive. This position would not change while the observer observed non-passive behaviour, setting the velocity to 0. The results from the test showed that the stop energy release method guaranteed passive behaviour in the system at both human interface side and robot side.

In the end, the delivered implementation can be used to guarantee stability to the DIY @ Home setup. However, it suffers from black out, the loss of communication between both devices when both sides have the stop energy release method applied simultaneously. It is advisable to implement the add dissipation method as well, as this could prevent black out, while guaranteeing passivity.

## 6.2 Recommendations

The add dissipation method could not be applied due to time constraints. As a result, the full controller, also including the stop energy release method, implemented as described in section 3.3 and section 4.4.

It was recommended that the add dissipation method should be used up until some maximum allowed level of energy generation, from fig. 5.2a and fig. 5.2c, we can see when the system is just becoming active, the energy measurements show a deflection, before showing seemingly proportional energy generation. From both graphs, the proportional energy generation starts at an observed energy level of -0.3. So, it would be interesting to see whether this is a good threshold to distinguish when to use each controller.

# A Force to torque constant

The setup, as can be seen from fig. 1.1, makes use of two gears. One small gear is connected to the motor and the other gear is the moved arm. We can calculate the torque as follows:

$$\tau = f * r \tag{A.1}$$

The radius $r$ of the setups is therefore important to know. It was measured that the small gear has a radius, $r_{motor}$, of 6.23 mm and the big gear, $r_{arm}$, 93.25 mm. These can also be used to determine the torque ratio $R$ between gears.

$$\tau_{arm} = R * \tau_{motor} \tag{A.2}$$

Here, the ratio $R$ can found as follows:

$$R = \frac{r_{arm}}{r_{motor}} \tag{A.3}$$

This gives a torque ratio of 0.068.

The force can be measured at the arm handle, in fig. 1.1 called "Handle for haptic feedback experiment". This was done by Bergsma (2024). This length is different from the $r_{arm}$. This length, $r_{handle}$, is around 12.7 mm. This length is can be used to determine the torque of the arm gear using eq. (A.1). Using the torque ratio, we can know the torque on the motor gear. The final equation becomes:

$$\tau_{motor} = R * r_{handle} * f \tag{A.4}$$

From this equation, the constant which relates the torque to the measured force is found to be 0.008636.

# Bibliography

Artigas J, Ryu J H, Preusche C, October 2010 "Time domain passivity control for position-position teleoperation architectures" *Presence (Camb.)* **19**, pp. 482–497

Bergsma V, 2024 "Control and force sensing for diy @ home telemanipulation setup"

Dresschers D, 2024*a* "About &x2013; Avatars.Report — avatars.report" https://avatars.report/about/ [Accessed 13-05-2024]

Dresschers D, 2024*b* "PC &x2013; Add extra dissipation &x2013; Avatars.Report — avatars.report" https://avatars.report/pc-add-extra-dissipation/ [Accessed 16-05-2024]

Dresschers D, 2024*c* "PC &x2013; Stop energy release &x2013; Avatars.Report — avatars.report" https://avatars.report/pc-stop-energy-release/ [Accessed 16-05-2024]

Dresschers D, 2024*d* "Tdpc p-cf architecture" URL https://avatars.report/tdpc-p-cf-architecture/ Last accessed on 13 May 2024

Dresschers D, 2024*e* "What is a robotic avatar? &x2013; Avatars.Report — avatars.report" https://avatars.report/what-is-a-robotic-avatar/ [Accessed 16-05-2024]

Franken M, Stramigioli S, Misra S, Secchi C, Macchelli A, August 2011 "Bilateral telemanipulation with time delays: A two-layer approach combining passivity and transparency" *IEEE Trans. Robot.* **27**, pp. 741–756

Hannaford B, Ryu J H, 2002 "Time-domain passivity control of haptic interfaces" *IEEE Trans. Rob. Autom.* **18**, pp. 1–10

Stramigioli S, Secchi C, van der Schaft A J, Fantuzzi C, August 2005 "Sampled data systems passivity and discrete port-hamiltonian systems" *IEEE Trans. Robot.* **21**, pp. 574–587