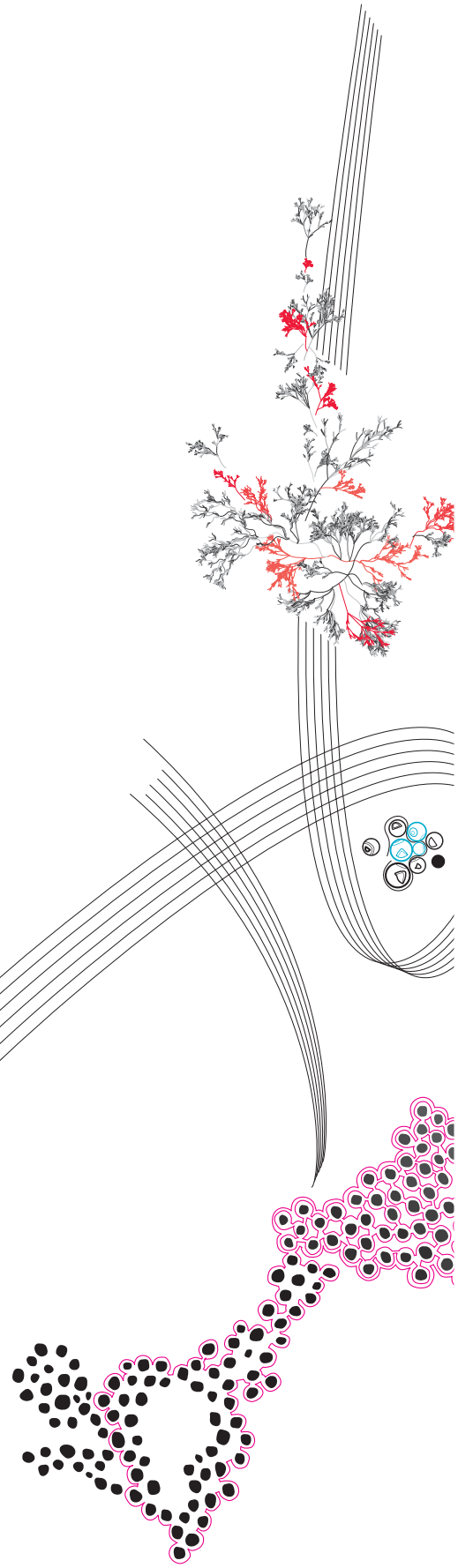

MSc BME Thesis

# Data-Driven Gravity and Stiffness Compensation for a 4-DoF Upper Extremity Robotic Exoskeleton

Thom C.G. Kuenen
s1806718

Graduation Committee:
Chair: Prof. dr. ir. H. van der Kooij
Daily Supervisor: Dr. ir. A.Q.L. Keemink
External Member: Dr. ir. R.G.K.M. Aarts

24 April, 2024

Faculty of Engineering Technology (ET)
Department of Biomechanical Engineering (BE)
Biomedical Engineering - BioRobotics Track

UNIVERSITY OF TWENTE.

**Acknowledgements**

I would like to express my great thankfulness to my daily supervisor: Arvid Keemink. I very much enjoyed and appreciate all our meetings and discussions, and believe that I have learned significantly more during this process because of his supervision. I would also like to thank Suzanne Filius for her unofficial supervision and for the very pleasant collaboration on this project. It certainly provided me with a broader view of the project. Arvid en Suzanne inspired me to work hard and constantly learn, for which I am thankful.

I would also like to thank my parents, brother and sister, my grandparents and the rest of family for their support, especially the last few months. Knowing that I have their support and wanting to make them proud certainly made me get up out of bed on hard mornings.

Furthermore, I would like to thank my friends, both in Enschede and back in de Achterhoek. I would like to thank my old student house, The Royal Pink Palace for their support and all the good times I had which provided me with distraction when I needed it and kept doing so even when I moved. I would like to thank my yearclub HJC Schavuit and new studenthouse SRH for doing the same, but in the last year also providing the required discipline to finish my Master. And I would like to thank my study mates, Sterre Triezenberg and Danique Damen for making my studies a lot more enjoyable and even providing feedback on this thesis.

**Abstract**

Duchenne Muscular Dystrophy (DMD) is a progressive degenerative neuromuscular disorder, causing 1:5000 boys and men to lose functionality in their muscles from a young age. From age 10-13 the upper extremity functionality is affected and patients are unable to function individually. The DAROR-01 is a robotic exoskeleton for the right upper arm, designed to aid these patients during Activities of Daily Living. For this, a gravity and stiffness compensation model is required. This work compares an a priori modelling approach with data driven models. For the data driven models, a white box approach using the kinematic structure and a grey/black box approach using Neural Networks (NN), are designed and validated. The data for the data-driven model is acquired with an identification protocol of 15 minutes in which the system moves through the workspace. The white box model that used the kinematic structure with an extension to also contain joint stiffness proved to be able to learn the gravitational and stiffness compensation model from very few data. This model achieved a Root Mean Square Error (RMSE) of 0.7 Nm, whereas the A Priori model had a RMSE of 1.2 Nm. The grey box model performed better than the black box model, with an RMSE of 2.4 Nm and 2.6 Nm respectively. From this offline analysis can be concluded that a data-driven white box modelling approach thus yields a better compensation model than a an A Priori model.

*Keywords*: Duchenne Muscular Dystrophy (DMD), Gravity compensation, Data driven models, Linear regression, Parameter Linear Form (PLF), Neural Networks (NN), Backpropagation, Conservative vector fields

# Contents

# Chapter 1

# Introduction

Duchenne Muscle Dystrophy (DMD) is a x-linked [1] progressive degenerative neuromuscular disorder. It affects about 1:5000 boys and men [2],[3]. People with DMD tend to be non-ambulant from the age of 10–13 [4] and their upper extremity (UE) functionality is already affected at this point. Being bound to a wheelchair and losing UE functionality from a young age greatly impacts the Quality of Life (QoL) of these people. The loss of UE functionality can be offset by corticosteroid treatment or physical therapy but eventually patients will require some form of UE assistance[4]. Furthermore, there is evidence that suggests that moving —with the aid of assistive devices— might prolong UE functionality [5].

## 1.1 Improving Quality of Life

Aiding DMD patients in Activities of Daily Living (ADL) will greatly improve QoL. In particular improving the UE functionality through assistance will make patients depend less on caretakers and might even offset the further deterioration of their functionality. If a healthy person is considered, in order to move the UE, the generation of force in the relevant joints is required. This in turn requires relevant muscle force. From Newtonian mechanics it is known that these forces must adhere to some equilibrium. During ADL the most amount of human force production is required to move limbs against gravity. This becomes obvious from the equation

$$F_{ADL} = m(g + a), \tag{1.1}$$

where $F_{ADL}$ is the force required for some ADL, $m$ is the mass of the limb performing the ADL, $g$ is the gravitational acceleration of 9.81 $\mathrm{m/s^2}$ and $a$ is the acceleration of the limb during the ADL. During ADL it can be assumed that $g \gg a$. It can therefore be concluded that moving against gravity requires the most force during ADL. Therefore a method of assisting DMD patients against gravity should greatly improve their UE functions.

Furthermore, DMD patients suffer from increased passive joint stiffness [6] that can be up to five times the remaining muscle force (at joint level). This force therefore should also be compensated for if the patient should regain UE functionality.

Compensation for both the gravitational force and the joint stiffness should allow DMD patients to regain upper extremity functionality.

## 1.2 Arm Support Technologies

Many attempts to design arm supports for people with arm disabilities have been made [7]. Commercially available arm supports generally are of two types. They are either passive, i.e. non-motorized support through the use of passive elements such as springs and counterweights. The other type is semi-active, i.e. weight support through the use of passive elements, which are adjustable through actuation. A major limitation of passive [8] or semi-active supports is that they do not provide equal support of the arm over the entire workspace. The passive elements are tuned to work well in a specific range, but outside this range they cannot provide the required support. Furthermore, as the required level of support increases, likely the provided passive support will not match the needs of the user. An active arm supporting device would not have the same challenges, as the level of support would be equal over the entire workspace and could be tuned to needs of the user.

The assistance of people that suffer from movement disability can also be done using an external robotic manipulator, which 'replaces' the human arm, rather than supporting it. These manipulators are active assisting devices. Often such systems are operated by a joystick that controls the end effector of the manipulator. While such devices definitely do improve the QoL of patients —especially in higher disease stages— they do not incorporate any significant physical contribution of the user. It has been shown that use and training of the physical arm slows down the progression of DMD [5]. Therefore the use of an arm supporting device is more desirable.

Given the drawbacks of passive, semi-active arm supports or external robotic manipulators, Filius et al. [7] argued that an active exoskeleton should be developed to provide and aid people with DMD with better arm support technology.

## 1.3 DAROR-01

To realise and test this hypothesis, a robotic exoskeleton for the right upper arm was developed: the DAROR-01. The DAROR-01, shown in Fig. 1.1, is a lightweight 4DoF exoskeleton designed such that it could be mounted on an electric wheelchair. It is development platform to test and research design requirements of an active arm support device. The DAROR-01 is developed as project 7 of the Wearable Robotics program (`https://www.wearablerobotics.nl/`) as a continuation of the Flextension project. The design is based on the Yumen arm design. The actuators and the other hardware are designed and developed by DEMO from the TU Delft. The hardware and low-level software of the exoskeleton were developed before the start of this thesis.

### 1.3.1 Gravity Compensation

As mentioned before, reducing the effects of gravity on the UE improves the QoL of a person with DMD. To compensate for gravity, a gravitational (compensation) model is required. The effect of gravity on a rotatory joint depends on the mass and the horizontal distance of the Centre of Mass (CoM) with respect to the Centre of Rotation (CoR). The CoM will induce a torque in the joints CoR, depending on the horizontal distance and thus for rotatory joints on the joint angle. For a 4 DoF system, the gravitational torque of all the joints thus depends on all joint angles. The gravitational torque thus is configuration dependent.

A gravity compensation model thus consists of a function that maps the joint configurations to joint torques. When the device delivers the gravitational joint torque, these will

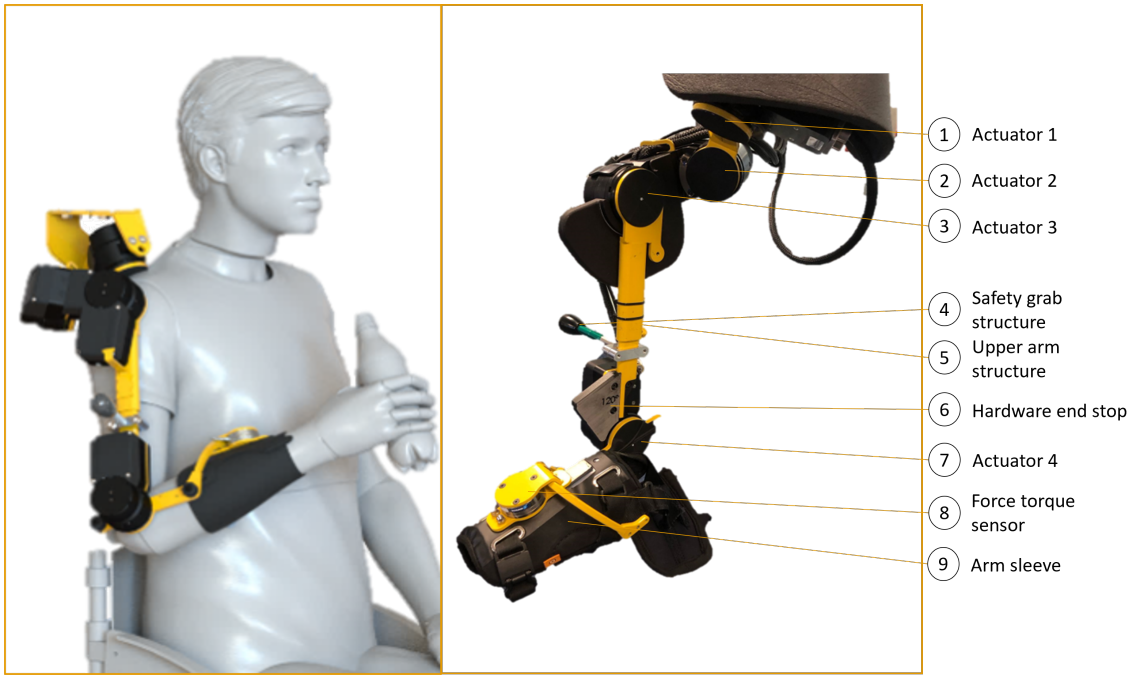


**Figure 1.1:** The DAROR-01: a 4-DoF (right) upper extremity exoskeleton. Left is a CAD generated figure, right is a picture of the actual system.

oppose the effect of gravity. Such a model is thus referred to as a gravity compensation model.

### 1.3.2 Stiffness Compensation

Joint stiffness is also a configuration dependent force. Since DMD patients show increased joint stiffness, a stiffness compensation model would further improve the QoL. The stiffness compensation model is similar to the gravity compensation model in that it also maps joint position to joint torque.

### 1.3.3 Challenges

Developing a gravity and stiffness compensation model for robotic devices has its challenges. Furthermore the addition of a human to the device only makes for a more complex system.

Through inaccuracies during production, the actual mass and CoM differ from those of the initial design. Accurate weighing and measuring after production still depends on the accuracy of the tools used in this process. Furthermore, where weighting and measuring robotic parts is relatively trivial, doing the same for a human upper arm is more complex. Assumptions made during this process —even when supported by literature [9]— can all cause modelling errors. Therefore, this work will explore the use of data-driven models that use actual measurements of the system (device + human) to generate gravitational models.

## 1.4 State of the Art

Models for gravity compensation in orthotics literature can be divided into two categories. The first category consists out of passive gravity compensation and the second proposes active compensation strategies.

Arakelian et al. [10] review many passive compensation strategies. These are based on counter weights or springs to compensate the gravity of the robotic device. These methods are, however, unable to deal with a change in load, given that their passive elements cannot quickly be adapted. Therefore a passive compensation method is does not meet the needs of this project.

Active approaches require some model of the gravitational structure of the system. Ott et al. [11] use a discretized workspace of which each segments has constant, preidentified gravitational compensation torques. Moubarak et al. [12] proposes a gravity compensation method based on the geometric structure of the exoskeleton. Using data they are able to train a compensation model. Liu et al. [13] has a similar approach but instead uses the system dynamics to formulate a model. They finally use linear regression to train a compensation model.

Ugartemendia et al. [14] discuss a similar approach but use this to compare a more novel machine learning approach using decision trees to find the gravitational structure. Lutter et al. [15] do not directly propose a method for gravity compensation in robotics, but instead have developed a method for learning Lagrangian dynamics using a grey-box[1] neural network.

The work of Moubarak et al. and Lui et al. is applicable for the objective of this work. The challenge both authors faced was in the formulation of structure of their device on which a model could be trained. Black-box models as in the work of Ugartemendia et al. and Lutter et al. would circumvent this problem. The work presented in this thesis compares a newly developed gravity compensation neural network, based on the work of Lutter et al., as a modelling method to modelling methods that are require the kinematic structure of the system.

## 1.5 Goal

The goal of this work is to develop several compensation models and compare their performance. A compensation model that requires a priori knowledge is designed, next to two types of data-driven compensation models: an approach based on the kinematic structure of the system and an approach using neural networks. The a priori model will serve as a benchmark for the data driven compensation models. Next to the compensation performance, the ease of use and training time are also considered. Furthermore, this work explores a novel greybox, data-driven gravitational modelling approach.

## 1.6 Thesis Outline

This thesis is outlined as follows: Chapter 2 discusses the kinematic structure and the hardware of the robotic system. Chapter 3 explores the mathematical method used for modelling. In Chapter 4 these methods are applied to the actual systems and the compensation models are derived. This Chapter also discusses the implementation of the models and other required software tools and it introduces how the different modelling

[1] Since this model does contain some preprogrammed knowledge of the data structure it is between a black and white box model

methods will be validated. Chapter 4 discussed the results of the modelling methods. In Chapter 6 the limitations of this work are discussed as well as other factors that affected this work. This is followed by an exposition of future steps in Chapter 7. Finally, in Chapter 8 the conclusion of this work is drawn.

# Chapter 2

# Kinematics, Dynamics and Control

The system used for this thesis is the DAROR-01. The DAROR-01 is a 4 Degree of Freedom (DoF) robotic orthotic device —also referred to as an 'exoskeleton'— containing three DoFs in the shoulder and one DoF at the elbow. The device is connected to the user with a sleeve around the lower arm and support around the elbow. Filius et. al. documented the technical requirements of the system [7]. The following chapter will describe the kinematic structure and relevant hardware and software of the robotic system.

## 2.1 Kinematics

Kinematics describes the motion of system of bodies without considering the accompanying forces. The following section will describe how the movement of the system and/or the pilot[1] is defined in the world frame $\Psi_0$ and how the joints move w.r.t. each other.

### 2.1.1 ISB Angles

In order to describe the movement of the glenohumeral (GH) joint[2] the anatomical expressions are flexion/extenstion, abduction/adduction, internal/external rotation, horizontal abduction/adduction, and horizontal flexion/extension. These, however, overdefine the movements possible in the shoulder. Since the shoulder joint is 3 DoF, only three expressions should be required, Fig. 2.1 shows these expressions. The International Society for Biomechanics (ISB) [16] proposes the following definitions:

- (GH) Elevation: rotation of amount $\gamma_e$ over the global $x$-axis
- (GH) Horizontal Rotation: rotation of amount $\gamma_h$ over the global $y$-axis that is rotated by the elevation
- (GH) Axial Rotation: rotation of amount $\gamma_a$ around the vector pointing from the CoR to the elbow, rotated by both elevation and horizontal rotation.

### 2.1.2 System Kinematics

The kinematic structure of the DAROR-01 is expressed in one world frame and four actuator frames. The world frame ($\Psi_0$ - see Fig. 2.2) is located at Centre of Rotation (CoR) and has a vertical y-axis, pointing up w.r.t. the user.

[1] The pilot is the individual who is wearing and using the exoskeleton
[2] Commonly known as the shoulder joint

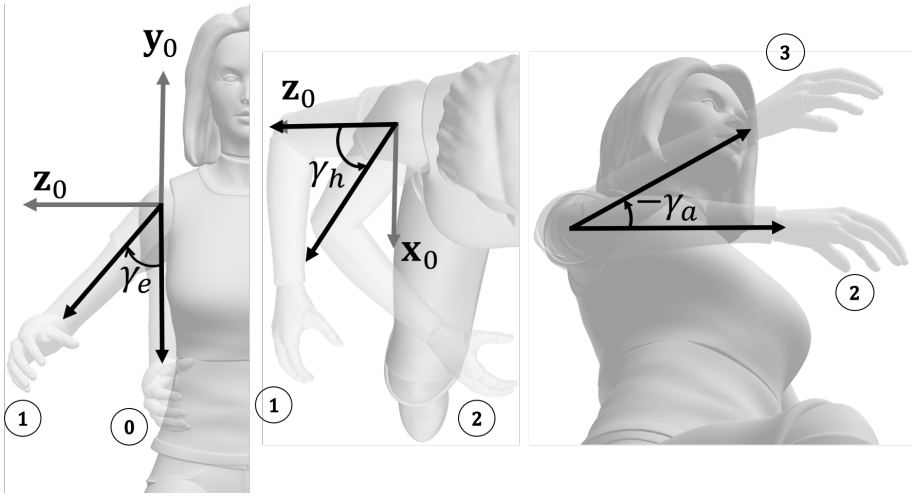


**Figure 2.1:** Definitions of the ISB angles [16]

Following typical choices for robotics, each actuator rotates around its $z$-axis. The Axis of Rotation (AoR) of the first actuator is rotated with respect to the vertical. It is rotated backwards (around $x$) by $-30°$, towards the head of the pilot (around $x$) by $-10°$ and around its own $z$-axis by $-45°$. This gives the following expression for the rotation of the first motors stator frame ${}^{0}\mathbf{R}_{1,s}$

$$\mathbf{R}_a = \mathbf{R}_x(-30°), \tag{2.1}$$
$$\mathbf{R}_b = \mathbf{R}_z(-10°), \tag{2.2}$$
$$v_1 = -\begin{bmatrix}1 & 0 & 0\end{bmatrix}\mathbf{R}_b \times \begin{bmatrix}0 & 0 & 1\end{bmatrix}\mathbf{R}_a, \tag{2.3}$$
$$v_1 = v_1/\|v_1\|, \tag{2.4}$$
$$v_2 = v_1 \times \begin{bmatrix}0 & 0 & 1\end{bmatrix}, \tag{2.5}$$
$$v_2 = v_2/\|v_2\|, \tag{2.6}$$
$$v_3 = v_2 \times v_1, \tag{2.7}$$
$$\bar{\mathbf{R}} = \begin{bmatrix}-1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}, \tag{2.8}$$
$${}^{0}\mathbf{R}_{1,s} = \begin{bmatrix}v_2 & v_1 & v_3\end{bmatrix}\bar{\mathbf{R}}\mathbf{R}_z(-45°), \tag{2.9}$$

where $\mathbf{R}_{x,y,z}$ are canonical rotations around the specified principle axis. Constant rotation matrix $\bar{\mathbf{R}}$ is required since the local $z$-axis needs to be rotated to the global $y$-axis.

In order to determine the kinematics of the system, the robot has been schematically drawn in one plane in Fig. 2.3. In this schematic it is shown that the AoR of the first three actuators intersect in the CoR of the shoulder. Vectors ${}^{i}p_{i,c}$, $i = [1, 2, 3, 4]$ are expressed in the rotor frame of their respective actuator ${}^{i,s}\mathbf{H}_{i,r}$. In order to be able to express any coordinate in the $\Psi_0$-frame, homogeneous transformations are required.

that the the $z$-axis is the AoR, the homogeneous transformation matrix between the

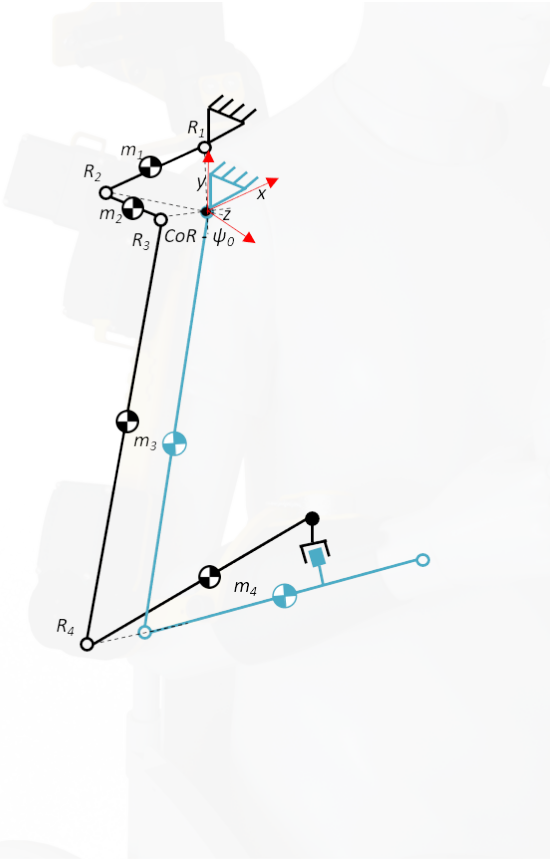


**Figure 2.2:** Free-body diagram of the DAROR-01 and pilot.

$\Psi_0$-frame and the stator of $R_1$ is

$$ {}^{0}\mathbf{H}_{1,s} = \begin{bmatrix} {}^{0}\mathbf{R}_{1,s} & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}, \tag{2.10} $$

$$ {}^{1,s}\mathbf{H}_{1,r}(q_1) = \begin{bmatrix} \mathbf{R}_z(q_1) & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}, \tag{2.11} $$

$$ {}^{0}\mathbf{H}_{1,r}(q_1) = {}^{0}\,\mathbf{H}_{1,s}\ {}^{1,s}\mathbf{H}_{1,r}(q_1). \tag{2.12} $$

The super-/subscript $()_s$ and $()_r$ denotes the frame of the stator (stationary in its frame) and rotor (dynamic) respectively. The expression of Eq. 2.11 hold for all the actuators since all rotate around their $z$-axis

$$ {}^{i,s}\mathbf{H}_{i,r} = \begin{bmatrix} \mathbf{R}_z(q_i) & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}, \ i = 1, 2, 3, 4. \tag{2.13} $$

Furthermore, actuator $R_2$ is rotated 60° with respect to $R_1$ (see Fig. 2.3). The same is true for $R_3$ w.r.t. $R_2$

$$ {}^{1,r}\mathbf{H}_{2,s} = \begin{bmatrix} \mathbf{R}_y(60°) & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}, \quad {}^{2,r}\mathbf{H}_{3,s} = \begin{bmatrix} \mathbf{R}_y(60°) & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}. \tag{2.14} $$

For actuator $R_4$ the AoR does not cross the CoR of the shoulder. It does however rotate over the same axis as $R_3$ but is translated along the upper arm segment of the device

$$ {}^{3,r}\mathbf{H}_{4,s} = \begin{bmatrix} I_{3\times3} & \delta \\ 0_{1\times3} & 1 \end{bmatrix}. \tag{2.15} $$

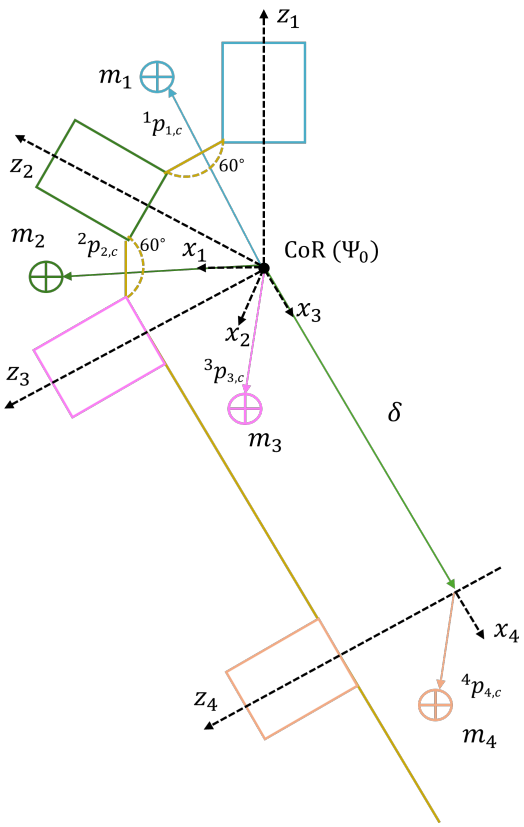


**Figure 2.3:** Schematic of the DAROR-01 in plane. Four actuators that rotate around their own z-axis. Vector ${}^ip_{i,c}$ points to the centre of mass of the link $i$ expressed in the frame of actuator $i$

Now all homogeneous transformation matrices have been defined and coordinates expressed in any of the actuator frames can be expressed in the $\Psi_0$-frame.

## 2.2 Actuation and Control

The hardware and software for the actuation of the DAROR-01 is discussed in the following section.

### 2.2.1 Series Elastic Actuation

For the actuation of the DAROR-01, Series Elastic Actuators (SEAs) are used. In the theoretical design of these actuators, an elastic element (such as a torsion spring) is placed in series between the output of the gearbox and the load. This design enables the measurement of the joint torque of the actuator, measuring the interaction between the actuator and the load. Absolute encoders are placed before and after the spring, essentially measuring the spring deflection. The spring is assumed to be linear

$$\tau = k\Delta q, \tag{2.16}$$

where $\tau$ is the spring torque —and thus the joint torque—, $k(= 289$ Nm/rad) is the spring constant and $\Delta q$ is spring deflection. The spring constant is assumed to be accurate to the specified unit. The precision of the spring deflection measurements is $10^{-4}$ rad.

Since the lowest precision is three digits, three significant digits can be reported for torque measurements.

### 2.2.2 Torque Measurement Error

The SEAs are subject to some phenomenon (or phenomena) that cause an error on the measured joint torque. This is shown in Fig. 2.4. The DAROR-01 executed a trajectory (the training trajectory to be precise, see Sec. 4.3.1) in position control mode (see Sec. 2.2.4). Joint $q_1$, $q_2$ and $q_3$ first run the path forward and thereafter the exact same path but in reverse, which is not true for $q_4$, so it is excluded. The joint torques of the reverse paths are flipped and the waypoints of the paths are synchronised. Given that the system does not change during the reverse path, the torque required for holding a position should be independent of the path. When comparing the measured joint torque (from the SEA) there is a difference between the joint torque of the forward path and the reverse path, as is shown in Fig. 2.4.

This measurement error is attributed to the SEA joint torque measurement system. We are not sure what causes this error. A hypothesis is that the encoder that measures

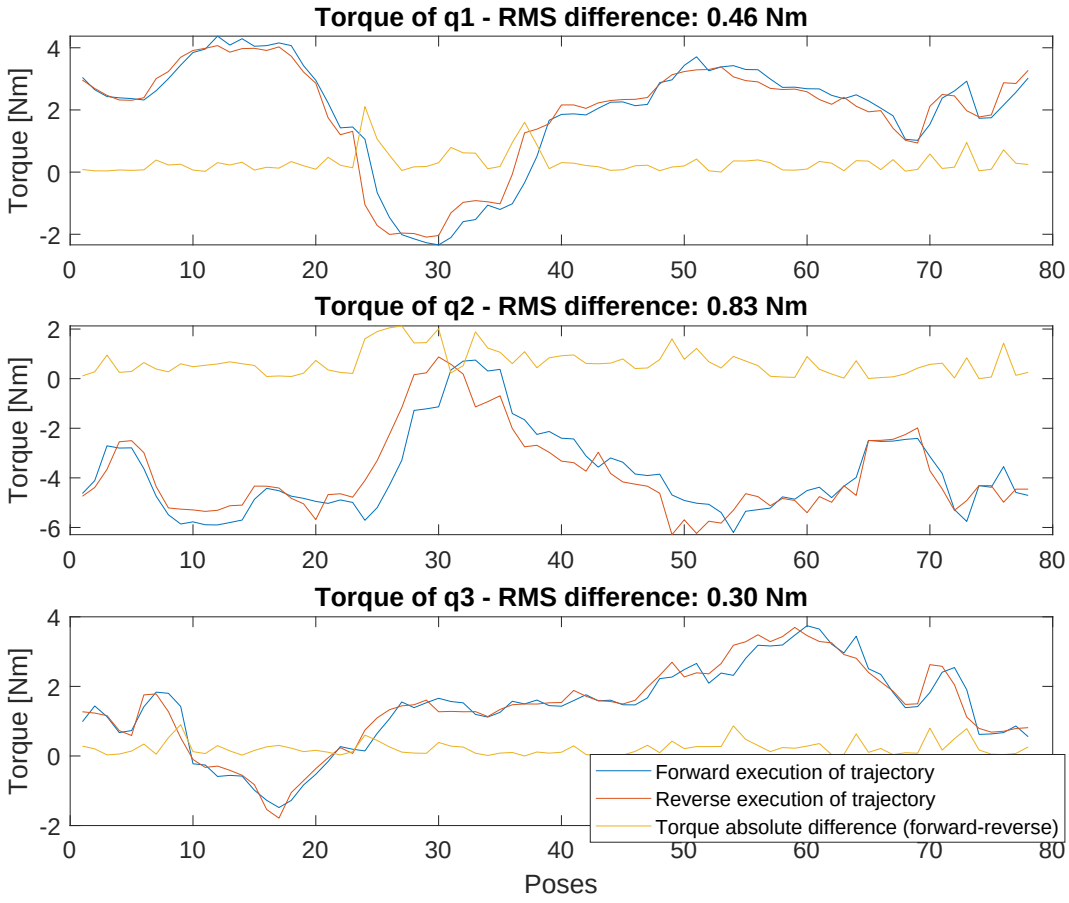


**Figure 2.4:** Representation of the difference between the joint torque measured on the forward path (blue) versus reverse path (red) and the difference (yellow), showing some error in torque measurement. The title of the plots also give the Root Mean Square (RMS) difference

the deflection of the torsion spring moves slightly when the rotor of the actuator rotates. Another hypothesis is that there is static friction between the elements on either side of the torsion spring. This static friction would cause torque measurement errors, since not all force is applied from the actuator to the joint through the spring but through this friction. Both of these hypotheses account for the difference in measurement torque depending on rotation.

From the analysis, actuator 2 (of joint $q_2$) was found to have the largest measurement error. This error is expressed as the Root Mean Square (RMS) difference between the forward and reverse. This RMS difference can increase or decrease when the path changes. but given the design of the trajectory used for the analysis should still be a good indication of the magnitude of the measurement error.

### 2.2.3 Torque Control

Torque control is used to control the joint torque applied to load by the actuator. If the reference torque is set to zero the system is in minimal impedance mode.

In this mode, the actuators are controlled to deliver zero joint torque and thus (theoretically) only external forces act on the joints. Essentially, the controller tries to minimize the spring deflection of the SEA. Thus, intrinsic motor forces are reduced in this mode. Fig. 2.5 shows the control loop.

The torque controller is used for the gravity and stiffness compensation. The reference torque is by the compensation model. This will cause the system to deliver the required compensation torque while still being compliant to external forces. For perfect compensation the net reference is still zero: $(\tau_g + \tau_s) + \tau_c = 0$ Nm (where $\tau_g$ is the gravitational torque, $\tau_s$ the stiffness torque and $\tau_c$ the compensation model torque). Thus the mode will still have all the properties of the initial minimal impedance mode but with the external forces of gravity and joint stiffness removed.

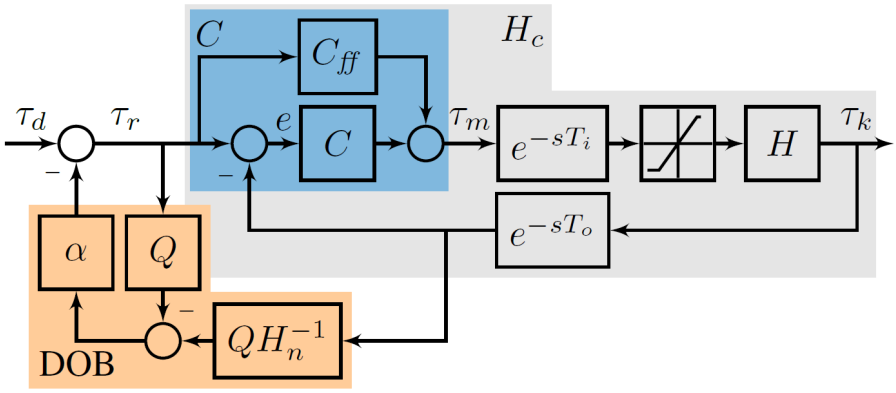


**Figure 2.5:** Torque controller loop for SEAs of Rampeltshammer et al. [17]. The SEA is represented with $H_c$ (grey) containing actuator torque saturation, communication sensor delay $T_o$ and $T_i$. The feedback controller $C$ and torque feedforward controller $C_{ff}$ form the inner control loop $C$(blue). The apparent actuator impedance is shaped by the Disturbance Observer (Based controller) DOB (orange), with nominal model $H_n$, DOB low-pass filter $Q$ and DOB gain $\alpha$ on the outer loop. The load side velocity input $\omega_l$ is omitted.

#### Disturbance Observer

An extension of the torque controller is the Disturbance Observer Based (DOB) controller (see Fig. 2.5), designed by Rampeltshammer et al. [17]. The DOB decreases the apparent impedance by rejecting disturbances to the system. The DOB results in a system that is much more admittant to an external force.

### 2.2.4 Position Control

Position control is required for the identification protocol (see Sec. 4.3.1). A simple, joint level PD controller was designed. The controller is essentially a position and velocity controller, with a reference position and reference velocity as input. Thus, the position and velocity error are both computed. The controller is manually tuned, $K_P = 700$ Nm/rad, $K_D = 175$ Nms/rad are found to have good performance.

The reference position and velocity are determined with a trajectory generator (see Sec. 4.3.1), in which ISB target positions (and velocities) can be inserted and the generator determines the contribution of each of the joints.

# Chapter 3

# Modelling

This work uses several methods for modelling gravity and joint stiffness in a multi-body system. The theoretical foundation of these methods is explained in this chapter.

## 3.1 Dynamic Modelling

A dynamic model of a controlled system describes the behaviour of the system given its states and the external states that act on the system [15]. Essentially, a controlled dynamic model is a function that relates the control and external forces to a change of internal states

$$f(x, \dot{x}, \tau, F_i) = \ddot{x}, \tag{3.1}$$
$$f^{-1}(x, \dot{x}, \ddot{x}) = \tau + J_i^T F_i, \tag{3.2}$$

where $f$ is a dynamic model, $x$ are the states of the system; $\tau$ is the control input; $F_i$ are the interaction wrenches with $J_i$ the geometric interaction Jacobian. A typical 2nd order dynamic model [18] is written as

$$\ddot{q} = M^{-1}(q)(\tau + J_i^T F_i - C(q, \dot{q})\dot{q} - g(q)), \tag{3.3}$$
$$\tau + J_i^T F_i = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q), \tag{3.4}$$

where $q$, $\dot{q}$ and $\ddot{q}$ are the generalized position, velocity and acceleration vectors of a system respectively. $\tau$ is the vector of the generalized forces due to actuators and $J_i$ are the geometric interaction Jacobian and $F_i$ the interaction wrenches acting on the system. The system dynamics are represented by inertia matrix $M(q)$, a vector of force terms that account for centrifugal and Coriolis forces $C(q, \dot{q})$ and the forces due to gravity $g(q)$. The expression of equations 3.1 and 3.2 hold for equations 3.3 and 3.4

Eq. 3.4 is a typical form achieved through solving the Euler-Lagrange equations. When deriving the model using knowledge of the system, such a model can be defined as a parameterized model. Thus the equation is built up of (at least somewhat) interpretable terms that have some physical interpretation.

### 3.1.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are a family of models that are able to fit to both linear and non-linear data. These models are inspired by the operation of biological neurons [19]. ANN are models that are able to do various tasks, such as decision making and

classification, pattern recognition and non-linear mapping [20]. This last ability allows ANN be used to map dynamic systems.

The model uses series of non-linear functions, consisting of many layers in series. Each layer is expressed as

$$z_{i+1} = \alpha_{i+1}(w_{i+1}z_i + b_{i+1}), \tag{3.5}$$

where $z_i$ is the input vector of layer $i+1$, and thus $z_{i+1}$ is the output vector of this layer; Activation function $\alpha_{i+1}$ is a non-linear function, for layer $i+1$; $w_{i+1}$ is the weight matrix between layer $i$ and $i+1$ and $b_{i+1}$ is the bias vector of layer $i+1$. The number of layers, the size of the layer and the activation function are all part of the structure of the network and are so-called hyperparameters (HP). When a ANN consist out of multiple hidden layers it is referred to as a deep ANN.

As mentioned, an ANN used the expression of Eq. 3.5 in series. The expression then becomes

$$f_{\mathrm{NN}}(x,\theta) = \alpha_N(w_N(\alpha_{N-1}(w_{N-1}...\alpha_1(w_1x + b_1)... + b_{N-1}) + b_N), \tag{3.6}$$

where $()_N$ denotes the total number of layers of the network. The network output $f_{\mathrm{NN}}(x,\theta)$, which depends on the network parameters $\theta$ and the network input $x$. The expression of Eq. 3.5 substitutes the output of each layer with the symbol $z$, which becomes more clear from this total series expression.

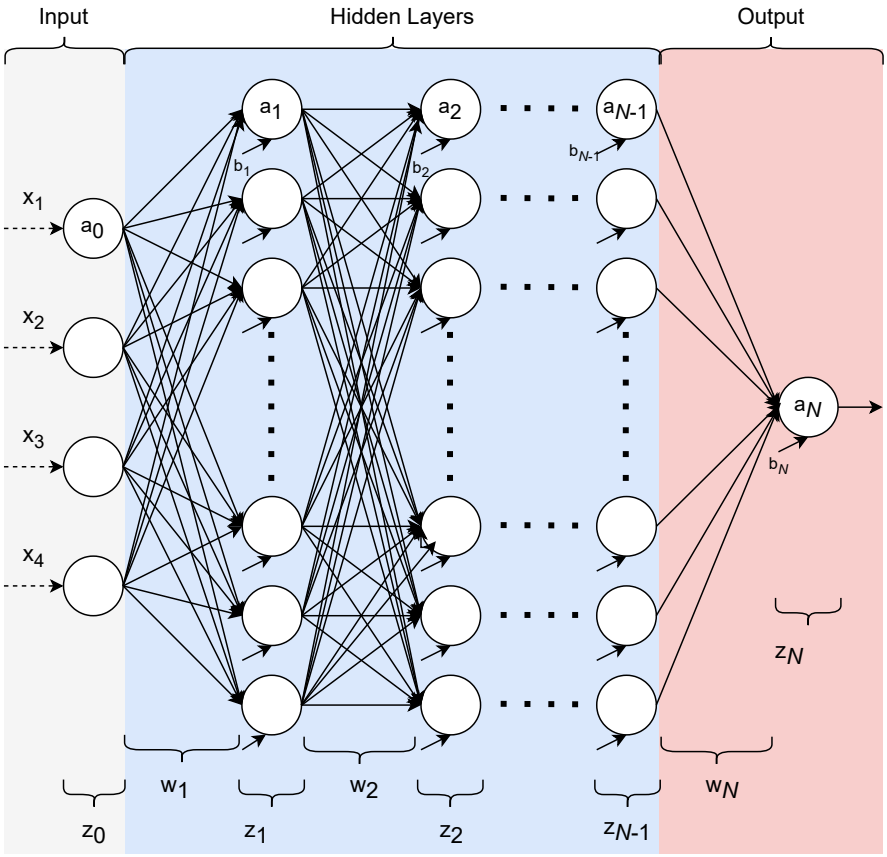


**Figure 3.1:** Schematic of a deep ANN. The notation is only given for the top neuron to make the figure readable; $b$ is a vector. The input $x$ of the model flows from left to right

The network parameters $\theta$ are critical for the performance of the ANN, but do not have any interpretable meaning. This makes designing this vector manually challenging. The vector $\theta$ is therefore chosen such that the output of the network mimics some target data that is labelled to every respective input. The performance of the network is expressed using Loss, also referred to as Cost

$$\ell(f_{\text{NN}}(x,\theta),\tau_t) = \|f_{\text{NN}}(x,\theta) - \tau_t\|_2^2 + \lambda \|\theta\|_1 , \tag{3.7}$$

where $\tau_t$ is the labelled target data; $\lambda$ is the regularization rate and $\|\theta\|_1$ is the $L_1$ norm of the parameter vector. This last term causes the Loss to not only depend on the fit to the data, but also with the smallest parameters value. This reduces overfitting and thus leads to a better generalizing model. $\lambda$ is part of the hyperparameters. Sec. 3.3.2 shows how this Loss is used to find the optimal solution $\theta^*$ is discussed further.

Eq. 3.6 is similar to Eq. 3.1 if the input vector $x$ is defined as the system state and the output of the equation is torque. However, since the ANN and its parameters do not model any physical attributes and the model is data-driven (and thus only learn structures present in the data) there is no method to prove that the resultant model will model the corresponding system in all possible states. In other words, there is no guarantee that the model will obey to the laws of physics. In this work an ANN that uses no system information or predefined structure will referred to as a **Naive Neural Network (NNN)**.

### 3.1.2 Lagrangian Neural Networks

Lutter et al. propose a method for dynamical modelling using neural networks and Lagrangian mechanics [15], [21]. This tackles the previously stated issue regarding the use of ANN to model physical systems,as the network is now forced to learn an energy conservative function.

In the work of Lutter et al., the structure of Lagrangian mechanics is forced on the network. This is achieved by formulating the Loss of the network as

$$\theta^* = \arg\min_{\theta} \ \ell(f^{-1}(x,\dot{x},\ddot{x},\theta),\tau_t), \tag{3.8}$$

where $\theta^*$ are the optimal parameters. Some Loss $\ell$ is chosen that expresses the difference between the model estimation and the targets $\tau_t$. The network input is represented $x$ and its derivatives. Since a physical system is being modelled, $x$ is, for example, the position. The parameters of the network are denoted by $\theta$.

The work done by Lutter et al. [15] is used as inspiration for **Potential Energy Neural Networks**, which is explored in Sec. 3.2.2

### 3.1.3 Conservative Vector Fields

The concern expressed in Sec. 3.1.1 can be generalised better. There is no guarantee that an ANN will mimic a Conservative Vector Field (CVF).

The definition of a CVF is that it is a vector field that is the gradient of some scalar function [22]. A property of a CVF is that its line integral is path independent. That means that the line-integral for vector field v is

$$\int_{P_1} \mathrm{v} \cdot d\mathrm{r} = \int_{P_2} \mathrm{v} \cdot d\mathrm{r} \tag{3.9}$$

$$\oint_{P_C} \mathrm{v} \cdot d\mathrm{r} = 0, \tag{3.10}$$

where $P_1$ and $P_2$ are two different paths between the same endpoints in the vector field and $d$r a step along that path; $P_C$ is a closed path in the vector field. This entails that returning to the initial position in the vector field must yield a line integral equal to zero. A well-known example of a CVF is gravitational force [22], which is the gradient of potential energy (PE), regardless of which coordinates are used to describe this PE. Moving an object first up and then down to its initial location will not yield any net gain or loss in PE of the object. This illustrates that the gravitational vector field is conservative.

## 3.2 Gravitational Modelling

Gravitational modelling is part of dynamic modelling, only taking static effects acting on the system due to gravity into account.

The most well-known method for deriving a gravitational model is Newtons method, where the equation

$$F_g = mg, \tag{3.11}$$

is solved. $F_g$ is the gravitational force, $m$ is the mass of the system and $g$ is the gravitational acceleration. For systems that rotate around some axis usually the gravitational torque is more relevant than the force. The expression can be rewritten to

$$\tau_g = mgl\sin(q), \tag{3.12}$$

where $l$ is the length of the rod, from the origin to the Centre of Mass(CoM), as shown for an example system in Fig. 3.2.

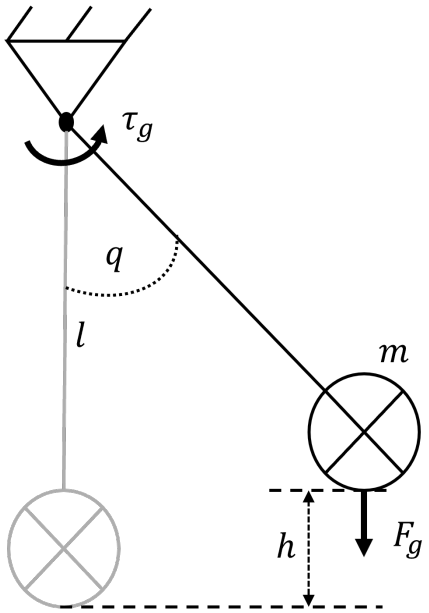


**Figure 3.2:** Single pendulum with rod-length $l$, joint angle $q$, mass $m$, height $h$ gravitational force $F_g$ and torque $\tau_g$

Alternatively, Lagrangian dynamics can be used to derive a gravitational model. In this method, the equations of motion are derived by finding the potential and kinetic energy equations of the system. For the PE of the system in Fig. 3.2,

$$V = mgh = mgl(1 - \cos(q)), \tag{3.13}$$

can be found, where $V$ is the PE of the system, and $q$ is the (joint) angle . The gravitational model can be derived by taking the partial derivative of the PE with respect to the configuration of the system

$$\tau_g = \frac{\partial V}{\partial q} = mgl\sin(q), \tag{3.14}$$

which yield the same model as Eq. 3.11.

By using the partial derivative of the PE, it is ensured that the resultant force or torque must a CVF (see Sec. 3.1.3). However, this method is unable to discern between different types of conservative forces. When the PE of a system is made up of, for example, both gravitational and spring forces the both are encapsulated in system torque $\tau$ (following from Eq. 3.14). In this work, a method for finding distinguishing gravity and spring torque is designed. This is discussed in Sec. 4.2.1.

### 3.2.1 Parameterized Models

Using the kinematics of a system, the PE function and thus a gravitational model can be derived.

#### A Priori

The expression for PE due to gravity is

$$V = mgp_{y,c}(q), \tag{3.15}$$

where $p_{y,c}$ is the vertical distance of the CoM w.r.t. the frame in which the vector $p$ is expressed. In robotic systems, if the CoM belongs to a joint of the device, $p_{y,c}(q)$ depends on the joint angle $q$. Thus, when $q$ is and the mass of the joint are known, the PE is known. Given Eq. 3.14, the required joint torque to compensate gravity can determined. In this work such an approach —where the mass $m$ and $p_{y,c}(q)$ are known and determined beforehand— is referred to as an **A Priori (AP)** gravity compensation model.

#### Parameter Linear Form

Following the expression for the PE derived equation for the gravitational model, Eq. 3.14, the equation can be split into two parts. The first terms contains the configuration dependent terms (referred to as the Regressor-block $\mathbf{K}(q)$) and the other contains the respective parameters

$$\tau = \mathbf{K}(q)\theta, \tag{3.16}$$

where $\tau$ is torque, $\mathbf{K}(q)$ contains configuration dependent term based on the kinematics and $\theta$ are the respective parameters. Rewriting the equation in such a way is referred to as **Parameter Linear Form (PLF)**; the regressor can contain non-linear terms, but the function is linear with its parameters.

### 3.2.2 Potential Energy Neural Network

Derived from the work of Lutter et al. is the so called **Potential Energy Neural Network (PENN)**. This model uses the same principle as the Lagrangian NN. The difference between Lutter et. al. their method and the PENN that only conservative forces can be learned by the network. This is achieved by taking the partial derivative of

the output with respect to the joint configurations $q$. This is achieved by defining the Loss as

$$\ell(V(\theta, q), q, \tau_t) = \left\| \frac{\partial V(\theta, q)}{\partial q} - \tau_t \right\|_2^2, \tag{3.17}$$

where $\tau_t$ are the targets torques for the model. The input of the model is $q$, which represents the joint angles. As discussed in Sec. 3.1.3, by using the partial derivative of the PE w.r.t. the joint angles, the resultant torque must be a conservative vector field. Therefore the model learned by the PENN must be a conservative vector field.

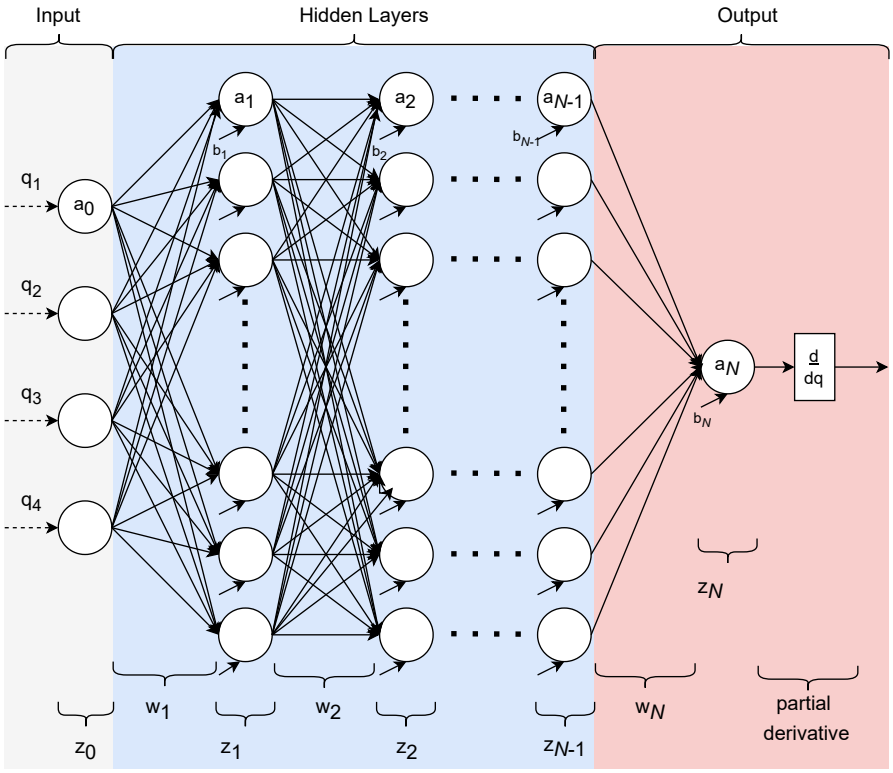


**Figure 3.3:** Schematic of a deep PENN. The notation is only given for the top neuron to make the figure readable; $b$ is a vector

## 3.3 Identification

While a priori knowledge of the system will solve the gravitational model equations, the correctness of the model relies fully on how well the parameters are determined. This can be avoided by using measurement data of the actual physical system and using this to generate the gravitational model.

Both the PLF and the NN models are data-driven. They rely on input data with respective target data and aim to find the structure between input and target. In the following section how these models find this structure is be discussed. In all cases the input data is denoted with $q$ and target data as $\tau_t$.

### 3.3.1 Linear Regression

Eq. 3.16 proposes that if the regressor and the parameters are known, the gravitational torque can be calculated. If there is some estimate of the parameters $\hat{\theta}$, an estimate of the

gravitational torque is found

$$\hat{\tau} = \mathbf{K}(q)\hat{\theta}, \tag{3.18}$$

where $\hat{\cdot}$ implies that this is an estimate of the real system. The best estimation $\theta^*$ can be found by finding the argument of $\hat{\theta}$ that minimizes the squared difference (Least Squares (LS)) between the target $\tau_t$ and the model output $\hat{\tau}$

$$\theta^* = \arg\min_{\hat{\theta}} \left\| \mathbf{K}(q)\hat{\theta} - \tau_t \right\|_2^2. \tag{3.19}$$

Considering the expression of Eq. 3.19, finding the minimizing argument for $\hat{\theta}$ can be done by

$$\arg\min_{\hat{\theta}} ||\mathbf{K}(q)\hat{\theta} - \tau_t||_2^2 \Longleftrightarrow \frac{\partial((\mathbf{K}(q)\hat{\theta} - \tau_t)^T(\mathbf{K}(q)\hat{\theta} - \tau_t)}{\partial\hat{\theta}} = 0. \tag{3.20}$$

If a solution for this equation exists, it must be the parameter vector $\hat{\theta}$ which best fits the data in a least squares sense. The equation above is expanded to

$$X = (\mathbf{K}(q)\hat{\theta} - \tau_t)^T(\mathbf{K}(q)\hat{\theta} - \tau_t) \tag{3.21}$$

$$= \hat{\theta}^T\mathbf{K}^T(q)\mathbf{K}(q)\hat{\theta} - \hat{\theta}^T\mathbf{K}^T(q)\tau_t - \tau_t^T\mathbf{K}(q)\hat{\theta} + \tau_t^T\tau_t \tag{3.22}$$

$$X = \hat{\theta}^T\mathbf{K}^T(q)\mathbf{K}(q)\hat{\theta} - 2\tau_t^T\mathbf{K}(q)\hat{\theta} + \tau_t^T\tau_t. \tag{3.23}$$

Now, the expression of Eq. 3.20 can be inserted

$$\frac{\partial X}{\partial\hat{\theta}} = 2\mathbf{K}^T(q)\mathbf{K}(q)\theta^* - 2\mathbf{K}^T\tau_t = 0 \tag{3.24}$$

$$\mathbf{K}^T(q)\mathbf{K}(q)\theta^* = \mathbf{K}^T(q)\tau_t \tag{3.25}$$

$$\theta^* = (\mathbf{K}^T(q)\mathbf{K}(q))^{-1}\mathbf{K}^T(q)\tau_t. \tag{3.26}$$

The expression $(\mathbf{K}^T(q)\mathbf{K}(q))^{-1}\mathbf{K}^T(q)$ of Eq. 3.26 is often referred to as the pseudo-inverse or the Moore-Penrose inverse [23]. However, $\mathbf{K}^T(q)\mathbf{K}(q)$ is not always invertable and therefore this method for finding the pseudo-inverse is not always possible. There are, however, other methods for determining the pseudo-inverse. With `Matlab` function `pinv()` the pseudo-inverse is determined through a Singular Value Decomposition (SVD) [24]. Important to note is that for rank-deficient (see Sec. 4.2.1 for the more extensive analysis of the Regressor-block) matrices, `pinv()` does not determine the pseudo-inverse as defined in this section. Instead —when $\mathbf{K}$ is rank deficient— it returns

$$\texttt{svd}(\mathbf{K}) = U\Sigma V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}, \tag{3.27}$$

$$\mathbf{K}^\dagger = V_1\Sigma_1^{-1}U_1^T, \tag{3.28}$$

where $U$ are the left singular vectors of $\mathbf{K}$, $\Sigma$ are the singular values of $\mathbf{K}$ and $V$ are the right singular vectors. Note that the bottom right of matrix $\Sigma$ is zero. This is caused by the rank deficiency of $\mathbf{K}$.

This definition of the pseudo-inverse $\mathbf{K}^\dagger$ gives

$$\theta^* = \mathbf{K}^\dagger\tau_t, \tag{3.29}$$

where $\theta^*$ is the minimum norm solution [25].

Having obtained this expression and assuming that the model is linear in the parameters, the optimal linear parameters can be determined. In turn these can be used as a forward gravitational model as:

$$\hat{\tau}_g = \mathbf{K}(q)\theta^*. \tag{3.30}$$

### 3.3.2 Backpropagation

For non-linear systems the method of Sec. 3.3.1 cannot be used since they have no unique solution. However, a function for expressing the performance of the model is still required. To illustrate, the following Loss is defined (see Eq. 3.17)

$$\ell(f_{\text{NN}}(\theta, x), \tau_t) = \|f_{\text{NN}}(\theta, x) - \tau_t\|_2^2, \tag{3.31}$$

where $f_{\text{NN}}(\theta, x)$ is the expression of the neural network, $\theta$ denotes the collection of all network parameters (weights $w_i$ and biases $b_i$) and $x$ is the input of the network. In order to find a minimal solution for the Loss, an algorithm called backpropagation is used [26]. This algorithm computes the gradient of the Loss of the model with respect to the parameters $w$ and $b$. If Eq. 3.5 is considered as one layer in a NN, backpropagation computes the partial derivatives w.r.t. the model parameters

$$\frac{\partial \ell}{\partial w_i}, \frac{\partial \ell}{\partial b_{i+1}}. \tag{3.32}$$

As is shown in Eq. 3.6, a network can be build up of many layers and thus many weights and biases. Computing the gradient requires the chain rule [27]

$$\frac{\partial \ell}{\partial w_{N-2}} = \frac{\partial z_{N-1}}{\partial w_{N-2}} \frac{\partial \alpha_{N-1}}{\partial z_{N-1}} \frac{\partial z_N}{\partial \alpha_{N-1}} \frac{\partial \alpha_N}{\partial z_N} \frac{\partial \ell}{\partial \alpha_N}, \tag{3.33}$$

where the conventions of Eq. 3.5 and 3.6 are used. While the expression above itself does not contain any repeating terms, the same method has to be applied in order to find $\frac{\partial L}{\partial b_{N-2}}$, $\frac{\partial L}{\partial w_1}$ or any of the partial derivatives. When computing all these, a lot of repeating terms occur. Backpropgagation recognises these terms and only computes them once, storing each partial into memory during every iteration.

Once all the partials have been computed, the network can use these to learn using gradient descent

$$w_N^{t+1} = w_N^t - \eta \frac{\partial \ell}{\partial w_N}, \tag{3.34}$$

where $t$ here denotes the iteration step and $\eta$ is the learning rate. Eq. 3.34 is the most basic implementation of the parameter update step. In this work, a well-known parameter update algorithm, ADAM [28], is used.

## 3.4 Model Validation

To check the implementation of the models, a validation method is proposed. A. Vallinas Prieto developed a Recursive Newton Euler (RNE) model of the DAROR-01 (see Sec. 4.1 for more details). This algorithm can be used as a noise-free feed-forward model of which the gravitational torques can be isolated $\tau_{g,model} = f_{model}(q)$. The data generated by the RNE model is in coherence with the actual physical system. If the data-driven models (Sec. 4.2) are able to find the structure of the RNE model, they are assumed to be correctly implemented.

# Chapter 4

# Methods

The following section outlines the implementation of the models aming to find a relation between the joint angles $q$ and the respective (gravitational) joint torques $\tau$, as described in Sec. 3.1.

A distinction is made between the A Priori method and the data-driven models: Parameter Linear Form (PLF) and Neural Networks (NN).

Subsequently the other functionality developed to facilitate the implementation and performance of these models is discussed.

## 4.1 A Priori

The AP model uses information of the system that is retrieved before the pilot is installed. Information of the design and production of the DAROR-01 provide system parameters for the exoskeleton. When a pilot is installed in the device, this can be seen as a new system, added to the DAROR-01. Given that the PE of a system is additive —therefore so is gravitational torque— the two systems can be evaluated individually.

The gravitational model for the DAROR-01 is derived through different approach than the one described in 3.2.1. Essentially an expression similar to Eq. 3.2 —with $\dot{q} = 0$ rad/s and $\ddot{q} = 0$ rad/s$^2$— is required for determining the gravitational model. This function is derived through the **Recursive Newton-Euler** (RNE) algorithm. This algorithm was implemented into `Matlab` by A. Vallinas Prieto within the department of Biomechanical Engineering and adapted to only determine gravitational torques.

To determine the gravitational model of the pilot the method described in Sec. 3.2.1 is used. In order to determine the PE, the masses and CoMs of the segments of the user are required. For this, Suzanne Filius, derived estimations for these parameters from literature [9], [29]. The literature proposes estimation methods based on total body mass and length and sex of the subject. From this, the vector

$$\theta_h = g \begin{bmatrix} m_1 I_{3\times3} & m_2 I_{3\times3} & m_3 I_{3\times3} & m_4 I_{3\times3} \end{bmatrix} \begin{bmatrix} {}^1p_{1,c} \\ {}^2p_{2,c} \\ {}^3p_{3,c} \\ {}^4p_{4,c} \end{bmatrix}, \tag{4.1}$$

that contains the human parameters ($\theta_h \in \mathbb{R}^{12}$) can be formulated, where ${}^ip_{i,c} = \begin{bmatrix} {}^ip_{i,cx} & {}^ip_{i,cy} & {}^ip_{i,cz} \end{bmatrix}^T$. With this, the PE of the pilot as a function of $q$ can be determined

$$V(q) = \mathbf{K}(q)\theta_h, \tag{4.2}$$

where $\mathbf{K}(q)$ contains the transformations of the vectors ${}^{i}p_{i,c}$ to ${}^{0}p_{i,c}$. How $\mathbf{K}(q)$ is derived will be explored extensively in Sec. 4.2.1.

Now, a vector for the PE $V(q)$ is found, where $V_x(q)$ and $V_z(q)$ should be removed since gravity only acts on vertical displacements

$$V_y(q) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} V(q). \tag{4.3}$$

From here, the gravitational torque is obtained

$$\tau_g(q) = \frac{\partial V_y(q)}{\partial q}. \tag{4.4}$$

The human shoulder is complex in its movement. For the method that was selected —expressing the gravitational torque as a function of $q$, $m_i$ and ${}^{0}p_{i,c}$— the vector ${}^{i}p_{i,c}$ is required and this vector is determined analytically. The assumption was made that there is no mass or CoM in actuator frame 1 and 2. Furthermore, the assumption is made that ${}^{3}p_{3,c}$ only has an $x$-component[1] (see Fig. 2.3). A similar assumption is made for ${}^{4}p_{4,c}$, which also only contains an $x$-component. Finally, from the derivation of Sec. 4.2.1, Eq. 4.12 the segment length of the upper arm is required in order to complete the parameter vector as suggested in Eq. 4.19.

These values have to be directly or indirectly [9], [29] measured. As a result a method for estimation the gravitational torque model of the pilot can be derived a priori.

In this thesis instead of testing with a pilot, only the DA is used for testing the system with load. Sec. 4.4.3 mentions the required parameters for setting up the parameter vector $\theta_h \equiv \theta_{DA}$

$$\theta_{DA} = \begin{bmatrix} 0_{1\times 6} & ({}^{3}p_{3,cx}m_{ua} + m_{la}\delta_{ua})g & 0 & 0 & {}^{4}p_{4,cx}m_{la}g \end{bmatrix}^T, \tag{4.5}$$

$$= \begin{bmatrix} 0_{1\times 6} & 5.0963 & 0 & 0 & 1.1559 & 0 & 0 \end{bmatrix}^T. \tag{4.6}$$

Furthermore, similar to the method described in Sec. 4.2.2 the AP model can be extended to also contain elbow stiffness. Different from the method in Sec. 4.2.2, a 1st order polynomial is used to estimate the joint stiffness. The reason here is that the parameters of a 1st order polynomial are quite easy to interpret and thus could be tweaked to suit the wishes of the pilot. The additional joint stiffness torque is

$$\tau_s = \begin{bmatrix} 0_{3\times 1} & 0_{3\times 1} \\ 1 & q_4 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \end{bmatrix}, \tag{4.7}$$

where $\mu_0 = 1.3$ Nm and $\mu_1 = 1.14$ Nm/rad, resulting from the work of Suzanne Filius.

## 4.2 Data-Driven Models

As discussed in Sec. 3.1 both the PLF and the NNs depend on data from the system. Sections 4.3.1 and 4.3.2 describe how this data was gathered. The input data for these models are the joint angles $q$ at iterations where the DAROR-01 was stationary ($||\dot{q}|| < 0.001$ rad/s) and the target data for these models is $\tau$. The output of the models is an estimation of the gravitational joint torque.

In this work, four data-driven models are explored. Three were already introduced in Sec. 3.1: **PLF**, **PENN** and **NNN** . The fourth model is PLF but extended to also contain

[1] This is based on the assumption that actuators 1-3 and the human shoulder have the same CoR

a PLF stiffness compensation structure. This is achieved by including a specific regressor with the aim to encapsulate joint stiffness. The aim is to separate the two conservative forces (gravitational and spring force) acting on the PE. This model is referred to as **PLF+**

Since these models use data to train, it is important to comment on the process of data gathering and how this might effect the models. Sec. 4.3.2 will go into more depth on how the data was gathered.

For the analysis of the performance of the models, two systems are considered. The first system is the DAROR-01 without any load attached. This explores ability of the model to find the gravitational structure of the system without any perturbations introduced by a pilot (or other load). The second system is the DAROR-01 with the Dummy Arm attached. The performance of the models in this configuration provide a good prediction of the model their performance when used with a human pilot.

### 4.2.1 PLF

Following from the kinematic structure of the DAROR-01 the Parameter Linear Form can be derived (see Eq. 3.16). This equation is derived from 3.14 and thus first the expression for the PE is found.

#### Constructing the PLF

First, an expression for the vector ${}^0P_{i,c}(= \begin{bmatrix} {}^0p_{i,c}^T & 1 \end{bmatrix}^T)$ is required

$$ {}^0P_{1,c}(q_1) = {}^0\mathbf{H}_{1,r}(q_1)\,{}^{1,r}P_{1,c} = {}^0\mathbf{H}_{1,r}(q_1) \begin{bmatrix} {}^1p_{1,c} \\ 1 \end{bmatrix}, \tag{4.8} $$

$$ {}^0P_{2,c}(q_1,q_2) = {}^0\mathbf{H}_{1,r}(q_1)\,{}^{1,r}\mathbf{H}_{2,s}\,{}^{2,s}\mathbf{H}_{2,r}(q_2)\,{}^{2,r}P_{2,c} = {}^0\mathbf{H}_{2,r}(q_1,q_2) \begin{bmatrix} {}^2p_{2,c} \\ 1 \end{bmatrix}, \tag{4.9} $$

$$ {}^0P_{3,c}(q_1,q_2,q_3) = {}^0\mathbf{H}_{2,r}(q_1,q_2)\,{}^{2,r}\mathbf{H}_{3,s}\,{}^{3,s}\mathbf{H}_{3,r}\,{}^{3,r}P_{3,c} = {}^0\mathbf{H}_{3,r}(q_1,q_2,q_3) \begin{bmatrix} {}^3p_{3,c} \\ 1 \end{bmatrix}, \tag{4.10} $$

$$ {}^0P_{4,c}(q)\,{}^0\mathbf{H}_{3,r}(q_1,q_2,q_3)\,{}^{3,r}\mathbf{H}_{4,s}\,{}^{4,s}\mathbf{H}_{4,r}\,{}^{4,r}P_{4,c} = {}^0\mathbf{H}_{4,r}(q) \begin{bmatrix} {}^4p_{4,c} \\ 1 \end{bmatrix}, \tag{4.11} $$

where $P_{i,c}$ is the augmented version of $p_{i,c}$ Eq. 4.11 is the only homogeneous transformation that contains a translation: ${}^{3,r}\mathbf{H}_{4,s}$. This causes the product of the final three terms of the middle equation to become

$$ \begin{bmatrix} I_{3\times3} & \delta \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_z(q_4) & 0 \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} {}^3p_{3,c} \\ 1 \end{bmatrix} = \delta + \mathbf{R}_z(q_4)p_{4,c}. \tag{4.12} $$

For the next part of the derivation, for the ease of notation PE is pretended to exist in $x, y$ and $z$. Eventually only the vertical $V_y$ is used. Now, the PEs for every link can be computed as

$$ V_1(q_1) = gm_1\,{}^0p_{1,c}(q_1) = {}^0\mathbf{R}_{1,r}(q_1)\theta_1, \tag{4.13} $$

$$ V_2(q_1,q_2) = gm_2\,{}^0p_{2,c}(q_1,q_2) = {}^0\mathbf{R}_{2,r}(q_1,q_2)\theta_2, \tag{4.14} $$

$$ V_3(q_1,q_2,q_3) = gm_3\,{}^0p_{3,c}(q_1,q_2,q_3) = {}^0\mathbf{R}_{3,r}(q_1,q_2,q_3)\theta_3, \tag{4.15} $$

$$ V_4(q) = gm_4\,{}^0p_{4,c}(q) = {}^0\mathbf{R}_{3,r}(q_1,q_2,q_3)gm_4\delta + {}^0\mathbf{R}_{4,r}(q)\theta_4 \tag{4.16} $$

$$ = {}^0\mathbf{R}_{3,r}(q_1,q_2,q_3)\theta_\delta + {}^0\mathbf{R}_{4,r}(q)\theta_4. \tag{4.17} $$

And these PEs can be summed to obtain the total PE of the system

$$V = {}^0\mathbf{R}_{1,r}(q_1)\theta_1 + {}^0\mathbf{R}_{2,r}(q_1,q_2)\theta_2 + {}^0\mathbf{R}_{3,r}(q_1,q_2,q_3)(\theta_3+\theta_\delta) + {}^0\mathbf{R}_{4,r}(q)\theta_4, \tag{4.18}$$

$$V = \underbrace{\begin{bmatrix} {}^0\mathbf{R}_{1,r}(q_1) & {}^0\mathbf{R}_{2,r}(q_1,q_2) & {}^0\mathbf{R}_{3,r}(q_1,q_2,q_3) & {}^0\mathbf{R}_{4,r}(q) \end{bmatrix}}_{\bar{\mathbf{K}}(q)} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3+\theta_\delta \\ \theta_4 \end{bmatrix}}_{\theta}. \tag{4.19}$$

From here, only the $V_y$ is considered. This is done through Eq. 4.3. Furthermore, the expression for the parameter vector $\theta$ is found

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3+\theta_\delta \\ \theta_4 \end{bmatrix} = \begin{bmatrix} m_1 g^1 p_{1,c} \\ m_2 g^2 p_{2,c} \\ m_3 g^3 p_{3,c} + m_4 g \delta \\ m_4 g^4 p_{4,c} \end{bmatrix}, \tag{4.20}$$

where ${}^i p_{i,c} \in \mathbb{R}^3$ is the vector pointing to the CoM of element $i$. Thus, the values in $\theta$ have a physical meaning.

The final step is to take the partial derivative to obtain the gravitational torque $\tau_g$.

$$\frac{\partial V_y}{\partial q} = \frac{\partial \bar{\mathbf{K}}_y(q)\theta}{\partial q} = \begin{bmatrix} \frac{\partial V_{y,1}(q_1)}{\partial q_1} & \frac{\partial V_{y,2}(q_1,q_2)}{\partial q_1} & \frac{\partial V_{y,3}(q_1,q_2,q_3)}{\partial q_1} & \frac{\partial V_{y,4}(q)}{\partial q_1} \\ \frac{\partial V_{y,1}(q_1)}{\partial q_2} & \frac{\partial V_{y,2}(q_1,q_2)}{\partial q_2} & \frac{\partial V_{y,3}(q_1,q_2,q_3)}{\partial q_2} & \frac{\partial V_{y,4}(q)}{\partial q_2} \\ \frac{\partial V_{y,1}(q_1)}{\partial q_3} & \frac{\partial V_{y,2}(q_1,q_2)}{\partial q_3} & \frac{\partial V_{y,3}(q_1,q_2,q_3)}{\partial q_3} & \frac{\partial V_{y,4}(q)}{\partial q_3} \\ \frac{\partial V_{y,1}(q_1)}{\partial q_4} & \frac{\partial V_{y,2}(q_1,q_2)}{\partial q_4} & \frac{\partial V_{y,3}(q_1,q_2,q_3)}{\partial q_4} & \frac{\partial V_{y,4}(q)}{\partial q_4} \end{bmatrix} \theta, \tag{4.21}$$

$$\Rightarrow \frac{\partial \bar{\mathbf{K}}_y(q)}{\partial q}\theta = \begin{bmatrix} \frac{\partial V_{y,1}(q_1)}{\partial q_1} & \frac{\partial V_{y,2}(q_1,q_2)}{\partial q_1} & \frac{\partial V_{y,3}(q_1,q_2,q_3)}{\partial q_1} & \frac{\partial V_{y,4}(q)}{\partial q_1} \\ 0 & \frac{\partial V_{y,2}(q_1,q_2)}{\partial q_2} & \frac{\partial V_{y,3}(q_1,q_2,q_3)}{\partial q_2} & \frac{\partial V_{y,4}(q)}{\partial q_2} \\ 0 & 0 & \frac{\partial V_{y,3}(q_1,q_2,q_3)}{\partial q_3} & \frac{\partial V_{y,4}(q)}{\partial q_3} \\ 0 & 0 & 0 & \frac{\partial V_{y,4}(q)}{\partial q_4} \end{bmatrix} \theta = \tilde{\mathbf{K}}(q)\theta. \tag{4.22}$$

Considering the definitions of Sec. 2.1 the chain rule is applied to $\frac{\partial V_{y,j}}{\partial q_i}$. All parts not containing $q_i$ are zero. Therefore only the partial derivative of the rotation matrix ${}^{i,s}\mathbf{R}_{i,r}(q_i) = \mathbf{R}_z(q_i)$ has to be taken. This results in

$$\frac{\partial \mathbf{R}_z(q_i)}{\partial q_i} = \begin{bmatrix} -\sin(q_i) & -\cos(q_i) & 0 \\ \cos(q_i) & -\sin(q_i) & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{R}'_z(q_i), \tag{4.23}$$

and substituting this term in $\bar{\mathbf{K}}_y(q)$ gives $\tilde{\mathbf{K}}(q)$. Therefore all parts of the Eq. 3.30 and the PLF expression, are found

### Algorithm Implementation

The algorithm is implemented in `Matlab`. A function is written that receives (stationary) joint positions and joint torques from the training dataset (see sections 4.3.1 and 4.3.2). With the training data, the algorithm described in Sec. 3.3.1 is employed to find $\theta$.

**Regressor Rank**

The Regressor $\mathbf{K}(q)$ is build up out of (at least) 3 Regressor blocks $\tilde{\mathbf{K}}(q) \in \mathbb{R}^{4\times 12}$. Considering that the Regressor $\mathbf{K}(q) \in \mathbb{R}^{4N\times 12}$ (Where $N \geq 3$ is the number of datapoints), the expected rank of this matrix would be 12. However, `Matlab` function `rank()` returns 8.

Rank is an expression of the dimensionality of the image of a matrix. Given a full-rank matrix of $N$ columns (with more than $N$ rows), that matrix expresses information in $N$ dimensions. When the rank of a matrix is smaller than the number of columns, the matrix is rank-deficient. This means that a subspace of the matrix is mapped to the null-space. Given that $\mathbf{K}(q)$ has rank 8 and 12 columns, a 4D subspace of $\mathbf{K}(q)$ is mapped to the null-space. Thus the parameters corresponding to those dimensions are linear combinations of the eight base parameters. As a consequence there is no unique solution for Eq. 3.26 and the values found for the parameters are not physically explanatory. Since the goal of this method was torque reproduction, this is not an issue.

### 4.2.2 PLF+

As mentioned in Chapter. 1, DMD patients experience high joint stiffness. In Sec. 3.2 it is pointed out that the distinction between PE due to gravity or some other configuration dependent force —such as joint stiffness— cannot be made. This fact is also true for the training data (see Sec. 4.3.2).

If a model for the joint stiffness can be found, the resultant compensation torque can be added to the gravitational compensation torque

$$\tau_{gs} = \tau_g + \tau_s, \tag{4.24}$$

to obtain the gravity + stiffness compensation model $\tau_{g+s}$. From data on the joint stiffness of the elbow, gathered by Suzanne Filius, was concluded that the profile of joint stiffness (of the elbow) can be approximated with a fifth order polynomial. This polynomial can be written in Parameter Linear Form

$$\tau_s = \mathbf{K}_s(q_4)\mu, \tag{4.25}$$

$$\mathbf{K}_s(q_4) = \begin{bmatrix} & & & 0_{3\times 6} & & \\ 1 & q_4 & q_4^2 & q_4^3 & q_4^4 & q_4^5 \end{bmatrix}, \tag{4.26}$$

$$\mu = \begin{bmatrix} \mu_0 & \mu_1 & \mu_2 & \mu_3 & \mu_4 & \mu_5 \end{bmatrix}^T, \tag{4.27}$$

where $\mu_i$ are the parameters that are found by the LS method of Sec. 3.3.1.

**Algorithm Implementation**

There is essentially no difference between the implementation of the PLF and the PLF+ training algorithm. PLF+ does contain the extension of training the parameter vector $\mu$ for the joint stiffness as well. This is also done using linear regression (Sec. 3.3.1).

### 4.2.3 PENN

The implementation of the PENN (and NNN) models follow quite directly from how they are described in Sec. 3.2.2 (and 3.1.1 respectively).

These HP of the PENN influence the performance of the network. It is, however, quite difficult to predict what choice of HP will give the best performance; Sec. 4.2.5 will go into more depth for some of HP and how they are selected.

Considering the kinematic structure of the system, joint angle $q$ never appears purely in the kinematic equations. Furthermore, $q$ has $2\pi$ rotation invariance. In order to avoid the NN having to learn this invariance, an additional layer is proposed. One layer is added between the input of the network and the first hidden layer

$$z_{\text{SC}} = \begin{bmatrix} \sin^T(q) & \cos^T(q) \end{bmatrix}^T, \tag{4.28}$$

where $z_{\text{SC}}$ is the output of the new layer containing periodic functions. The hypothesis is that this improves the performance, but this has to be validated.

For the batch size, 50% of the dataset was selected (see Sec. 4.3.2 for more information on the dataset). Every 10 batches, the model is tested and the validation loss is returned.

ADAM is used to set the learning rate (see Sec. 3.3.2).

**Algorithm Implementation**

The PENN was written in Python with the help of `JAX` [30]. This is a library that contains auto-differentiation algorithms. This was required since not only the partial derivative of the loss function w.r.t. the network parameters $\theta$ (see Sec. 3.2.2, Eq. 3.17) had to be taken —which is handled by most machine learning libraries— but also w.r.t. the input $q$. Furthermore, `JAX` runs on the GPU (or TPU) of a system, which are more efficient in running the types of computations of `JAX`. Finally, `JAX` only is configured for Linux OS. This all contributed to switching systems to run the PENN and interfacing with this Linux system. The dataset is used for training is sent to the Linux system and once training is complete the Linux system sends back the trained network parameters $\theta$.

For the final implementation, the networks were trained for 35 batches of data, testing every 5 batches.

As Sec. 4.2.5 will point out, random initialization can dictate the final performance of a network. From Table I it becomes clear that the best performing network would not be in the top 10 if it had been for the other randomly initialized network. To circumvent this problem, the algorithm is implemented in such a way that 5 networks are trained in series, and the network parameters $\theta$ with the lowest Minimal Validation Loss (MVL) are selected.

Taking the expression of Eq. 3.17, it should be noted that, since the extra layer was added between the input and the first HL, the partial derivative of this layer also had to be computed

$$\frac{\partial z_{\text{SC}}}{\partial q} = I_{8\times 8} \begin{bmatrix} \cos(q) \\ -\sin(q) \end{bmatrix}, \tag{4.29}$$

as the final step in the backpropagation.

### 4.2.4 NNN

The Naive Neural Networks attempt to find a model using no information of the system except for the data and the input 4.28. The operation of this model follows from how it is described in Sec. 3.1.1. Different from the PENN model, the joint torque $\tau$ is directly computed by the network. The performance of this network logically also depends on the set on HPs. How these are selected is described in Sec. 4.2.5).

#### Algorithm Implementation

Since the environment was already set up for the PENN, NNN was also developed using `JAX`. The main difference is that the partial derivative w.r.t. the input is not computed; the joint torques $\tau$ are the output of the network. The rest of the implementation of the algorithm is the same as the for PENN.

### 4.2.5 Hyperparameter Search

In order to find the best configuration, a brute force search was done. Below is an overview of the HPs that were considered in this search.

- Number of layers: 1, 2, 8, 21
- Neuron per layer: 5, 10, 25
- Activation functions: Hyperbolic Tangent (Tanh), Rectified Linear Unit (ReLU), Sigmoid
- Regularization rate; $10^{-5}$, $10^{-3}$, $10^{-1}$
- Input type: $q$, $SC$ (see Eq. 4.28)

Every configuration was trained twice with random initialization. The dataset classified as training dataset (see Sec. 4.3.2) of the DAROR-01 was used for this search. Per batch, 75% is used as training data, the remaining 25% is validation data. For the selection of the train vs. validation data, k-fold cross validation [31] is used. The performance of the different HP sets is expressed in terms of the minimal validation loss: the lowest value the validation loss is during the training process. The reason this performance indicator is selected is that it provides the best prediction for generalization[2] without requiring a forward computation of the network.

The set of HPs found for PENN and NNN on dataset A (see Sec. 4.3.2) is assumed to also be optimal for dataset B. This assumption is made since both datasets contain identical kinematic structures with only different parameters for mass and CoM. Furthermore, the advantage of using data-driven models for gravity compensation is that the compensation for different pilots can be learned without to much trouble. If a brute force search for the best set of HP is required for every different system this method would be too time consuming.

#### PENN HP Optimization Results

In total, 432 ($= 3 \cdot 4 \cdot 3 \cdot 3 \cdot 2 \cdot 2$) networks were trained with the specifications mentioned above. The selected performance indicator is the MVL, computed for all network configurations. Ten networks are selected based on the performance indicator. Fig. 4.1 shows the progression of the validation loss

The smallest MVL occurs around Epoch 20000. This could indicate that the network has coincidentally overfitted to the validation dataset. However, given the progression of the validation loss of that network and the fact that k-fold cross validation is used, this seems very unlikely.

Table I shows the ten networks with the MVL. The best performing network configuration is a network consisting of two layers with ten neurons per layer. The layers have the Sigmoid activation function, input of Eq. 4.28 and use a $L_1$ regularization rate of 0.1.

It is quite clear from Table I, column MVL that the order and performance of the networks depends greatly on the random initialization of the network. From the tested

[2]Assuming that the validation data contains a good representation of the system. There is no direct proof of this assumption but given the cross validation, overfitting should be very incidental

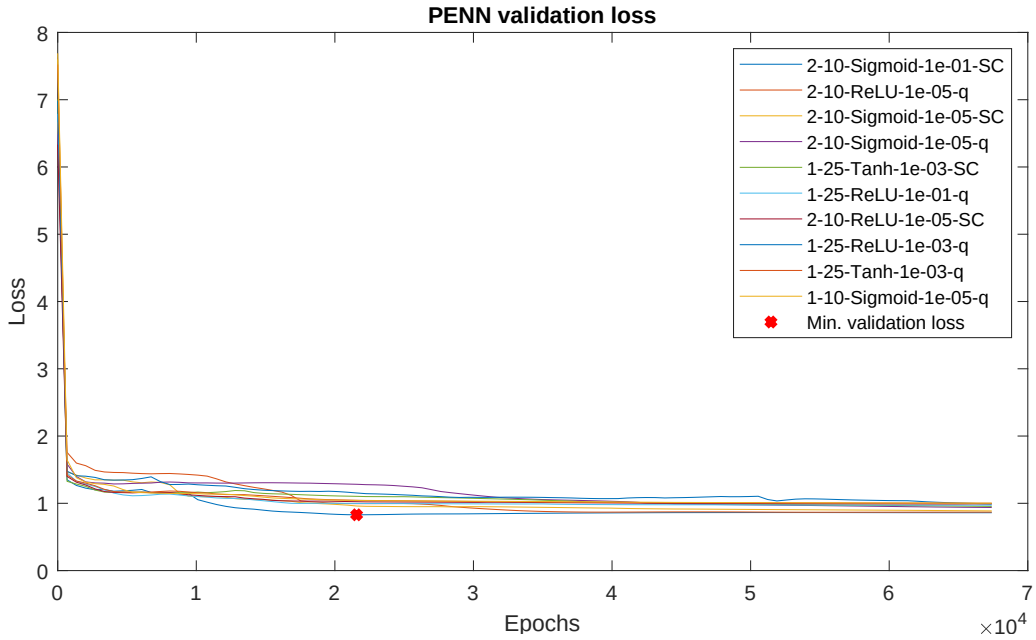


**Figure 4.1:** Training and validation losses of the ten PENNs with the best performance indicator

networks, small networks (referring to few layers and few neurons per layer) perform better than larger networks. No clear conclusion can be drawn for the other HPs. For this, a larger scale global search with more random initialized networks would be required in order to perform proper statistical analysis.

While finding the set of HP with the highest probability of getting the lowest MVL would be interesting, there still would be no guarantee that a newly trained network would have a good performance. Finding the best network would still require training multiple random versions. Since this is also true for the results of this brute force search, this is solved in the algorithm implementation.

**Table I:** PENN - The results of the brute force search over all combinations of the hyper-parameters in their specified range. HL is the number of Hidden Layers, N/L is the number of Neuron per Layer. $\alpha$ denotes the type of activation function. $\lambda$ is the regularization rate and the min. validation loss is the minimum value of the validation loss during the learning process of that network. The green value is the smallest MVL, the other MVL is of the other randomly initialized network.

| HL | N/L | $\alpha$ | $\log_{10}(\lambda)$ | Network input | MVL | |
|---|---|---|---|---|---|---|
| 2 | 10 | Sigmoid | -1 | $SC$ | 1.2106 | 0.8293 |
| 2 | 10 | ReLU | -5 | $q$ | 1.1299 | 0.8671 |
| 2 | 10 | Sigmoid | -5 | $SC$ | 1.3656 | 0.8873 |
| 2 | 10 | Sigmoid | -3 | $q$ | 0.9379 | 1.2093 |
| 1 | 25 | Tanh | -3 | $SC$ | 0.9538 | 1.3053 |
| 1 | 25 | ReLU | -1 | $q$ | 0.9728 | 1.2339 |
| 2 | 10 | ReLU | -5 | $SC$ | 1.2229 | 0.9916 |
| 1 | 25 | ReLU | -3 | q | 0.9905 | 1.0791 |
| 1 | 25 | Tanh | -3 | $q$ | 1.0011 | 1.1448 |
| 1 | 10 | Sigmoid | -5 | q | 1.2469 | 1.0019 |

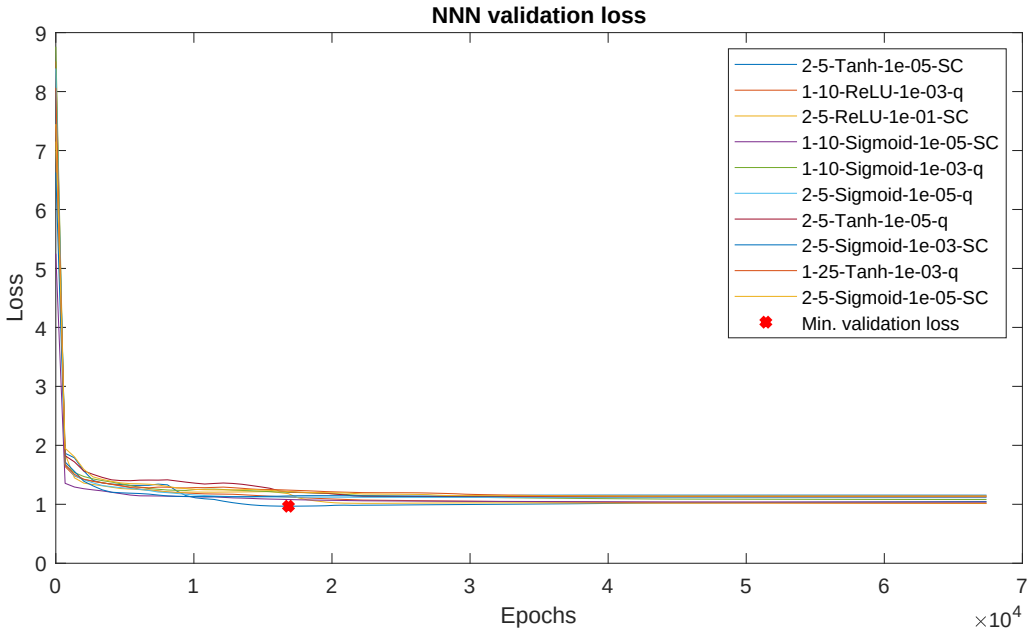


**Figure 4.2:** Training and validation losses of the ten NNNs with the best performance indicator

### NNN HP Optimization Results

Again, the smallest min. validation loss —shown in Fig. 4.2— is not a unique occurrence and therefore the model has not incidentally overfitted to the validation data.

In Table II the results of the search of NNN network configurations is shown. The best performing network has two layers with five neurons, the hyperbolic tangent activation function, a $L_1$ regularization rate of $10^{-5}$ and Eq. 4.28 as the input layer of the network.

**Table II:** NNN - The results of the brute force search over all combinations of the hyperparameters in their specified range. HL is the number of Hidden Layers, N/L is the number of Neuron per Layer. $\alpha$ denotes the type of activation function. $\lambda$ is the regularization rate and the min. validation loss is the minimum value of the validation loss during the learning process of that network. The green value is the smallest MVL, the other MVL is of the other randomly initialized network.

| HL | N/L | $\alpha$ | $\log_{10}(\lambda)$ | Network input | Min. validation loss | |
|---|---|---|---|---|---|---|
| 2 | 5 | Tanh | -5 | $SC$ | 0.9675 | 1.2472 |
| 1 | 10 | ReLU | -3 | $q$ | 1.3807 | 1.0138 |
| 2 | 5 | ReLU | -1 | $SC$ | 1.0161 | 1.1851 |
| 1 | 10 | Sigmoid | -5 | $SC$ | 1.0375 | 1.2667 |
| 1 | 10 | Sigmoid | -3 | $q$ | 1.0807 | 1.1723 |
| 2 | 5 | Sigmoid | -5 | $q$ | 1.2100 | 1.1129 |
| 2 | 5 | Tanh | -5 | $q$ | 1.2512 | 1.1270 |
| 2 | 5 | Sigmoid | -3 | $SC$ | 1.1283 | 1.2170 |
| 1 | 25 | Tanh | -3 | $q$ | 1.1343 | 1.1725 |
| 2 | 5 | Sigmoid | -5 | $SC$ | 1.1400 | 1.2407 |

### 4.2.6 HP Analysis and Conclusion

From the analysis performed in Sec. 4.2.5 can be concluded that small networks are able to best represent the structure within the data. No clear conclusion can be drawn w.r.t. the activation function, regularization rate or network input. This would require more randomly initialized networks per HP set to enable proper statistical analysis. However, as can be seen from both Tables I and II random initialization has a much larger impact than these HPs. The same is true for specific numbers of layers or neurons per layer.

The statistical analysis potentially would result in the 'best' combination of HPs but then still, there is no absolute guarantee that a new and randomly initialized network will perform well. Finding a NN with good performance is more likely to be the result of random initialization with a HP set in the top ten (see Sec. 4.2.4 on how this conclusion is used). Therefore no further action is taken into optimizing the performance of HP sets.

## 4.3 Facilitative Functionality

The setup of the DAROR-01 consist of three systems that correspond using specific protocols, as is shown in Fig. 4.3. The main software —e.g. the torque and position controllers and the compensation models— runs on the Base Station (on TwinCAT 3) and communicates with the motor drives and sensors to actuate the exoskeleton and retrieve data. In turn, the Base Station also communicates, using ADS, with the Operator PC, where the Graphical User Interface (GUI) displays relevant data and allows interaction with the software of the exoskeleton on the Base Station.

Aside from the aforementioned compensation methods, functionality within the system was developed to assist and/or facilitate the implementation of those methods. The following sections will highlight the most important functionalities.


**Figure 4.3:** Schematic representation of the electronic systems used during operation of the exoskeleton, with the corresponding communication protocols.

### 4.3.1 Trajectory Generation

A trajectory generator is designed that used the position control mode. The trajectory generator generates the reference joint position $q$ and joint velocity $\dot{q}$ for the system. The input of the generator are the waypoints of a trajectory, consisting an elevation, horizontal rotation, axial rotation and elbow flexion angle. During the identification protocol, the exoskeleton moves from waypoint $i$ to waypoint $i+1$. The trajectory generator is designed by dr. ir Arvid Keemink.

Using forward kinematics, the pose of a waypoint can be transformed to a rotation matrix of the shoulder and elbow angle. Since the trajectory of the elbow is trivial to determine, it is not regarded in the following analysis. The ISB angles of the shoulder joint are transformed to a rotation matrix. From here, Eulers Rotation Theorem says that there must be a single AoR $\omega$ between the rotation matrix $i$ and rotation matrix $i+1$. The required shoulder rotation velocity from pose $i$ to $i+1$ is constructed as the product $v\omega$, where $v$ is some scalar speed. This ISB angular velocity is then converted to joint velocity

using differential inverse kinematics:

$$\dot{q} = \begin{bmatrix} J^{-1}(q)v\omega \\ \dot{q}_4 \end{bmatrix}, \tag{4.30}$$

where $J(q)$ is the analytic Jacobian of the shoulder joint. The actual implementation of also contains a feedback correction velocity for the shoulder rotation. This is omitted for simplicity.

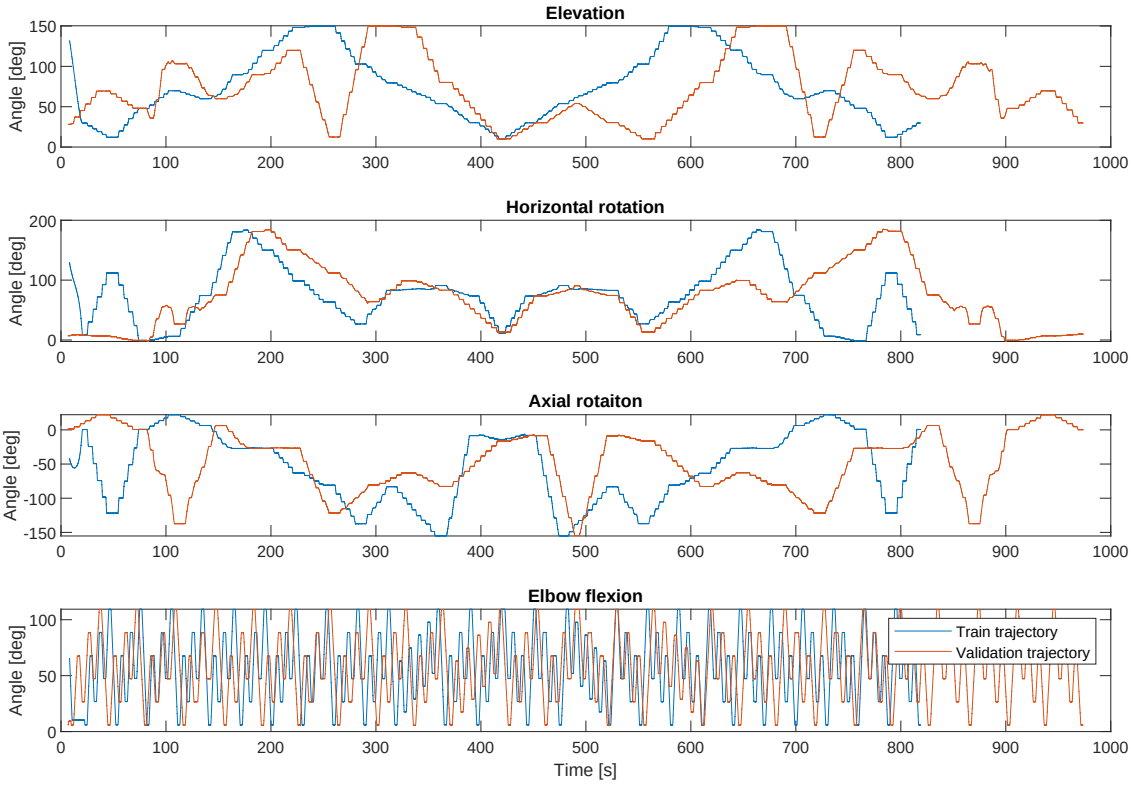


**Figure 4.4:** Training and validation trajectories in ISB space

Two trajectories (see Fig. 4.4) are designed; one used for the training of the data-driven models, the other to validate the performance of the models once trained. The training and validation trajectories contain the same waypoints but in a different order. This causes trajectory to differ while still spanning the same workspace. Furthermore, once the final waypoint is reached, the algorithm runs through the trajectory in reverse. The reason for this is that, if the system contains any forces that depend on the direction of the rotation (as discussed in Sec. 2.2.2), that effect is mitigated.

The results of the conversion from ISB-space to $q$-space is shown in Fig. 4.5.

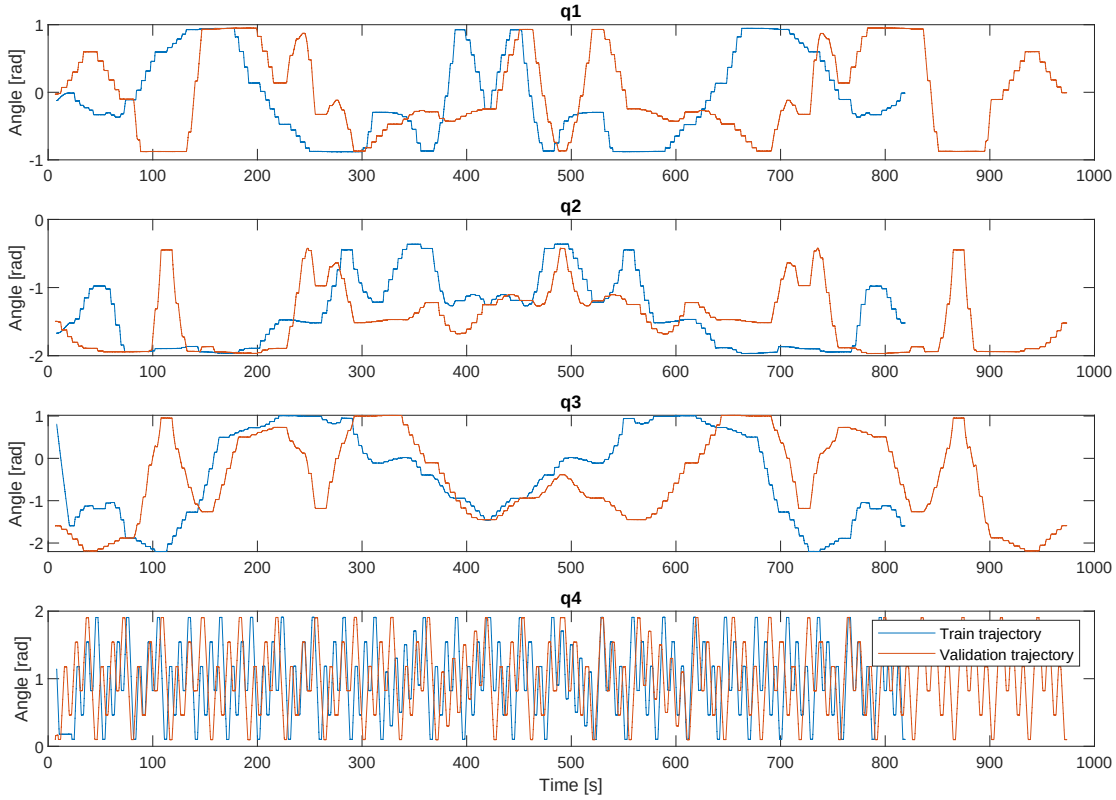


**Figure 4.5:** Training and validation trajectories in $q$-space

### 4.3.2 Data Acquisition

The dataset required for the training of the models contains the configuration of the joints $q$ and the associated joint torque $\tau$. The data should contain only gravitational (or spring) torques. During movement, the SEAs also measure velocity- and acceleration induced torques. Thus, only data at stationary points should be considered. A dataset is built up out of datapoints at which the norm of the velocity vector is near zero.

Using the trajectory generator, the waypoints in ISB space are converted to path in global frame rotation, which is then converted to $q$-space. Given that the models require data at which the system is stationary, the trajectory is build up out of small movements with short breaks between each movement. Sec. 4.3.1 describes the design of the trajectory using waypoints. The path between these waypoints is interpolated and divided into smaller sections, creating sub-waypoints. At every (sub-) waypoint, the system is briefly stationary.

Once the trajectory has finished, the norm of the joint velocities is determined and only data at which $||\dot{q}|| < 0.0002$ rad/s is selected for the dataset. Since the joint velocity data is quite noisy, a lowpass butterworth filter with cut-off frequency $\omega_c = 0.1$ Hz is used to filter the joint velocity data. Both the lower bound for $\dot{q}$ and $\omega_c$ are chosen such that data was stationary enough and the signal looked smooth but still contained all information. From this, both datasets contain about 25000 stationary datapoints. For the training dataset, all data was randomly shuffled before training to ensure that the models would not overfit to substructures in, for example, the first 5% of the dataset. While the large size of the dataset is no immediate problem for the training dataset, for validation of the model such a large dataset can be hard to interpret and display. Therefore, another method is chosen

for the selection of data.

Given the fact that at every waypoint the system is stationary, these indexes are instead selected for constructing the validation dataset; reducing the dataset to only 166 datapoints.

Two different training and validation datasets were recorded, one where the DAROR-01 alone performed the trajectories and a second one where the DA was installed into the system and then the trajectories were run. The first dataset, with only the DAROR-01, is referred to as dataset A and the other is referred to as dataset B.

In order to ensure that the models will note learn substructures in the data, the data is randomly permutated.

### 4.3.3 Finite State Machine

Given all the required functionality contained in the software, a Finite State Machine (FSM) that controls the program and information flow is designed. Also, the Finite State Machine will ensure a safe testing protocol with a pilot installed in the device by requiring numerous safety checks. A schematic of the FSM can be found in App. A

The FSM has three distinct super states, a pre-op(erative) state, an installation state and an operative state. The pre-op superstate is the state which contains the start-up state and all states that include the checks and settings required before a pilot is installed in the device. The DAROR-01 can be conservatively turned on in this state but only to check the disabling and contributing safety functions.

The installation superstate contains the states in which the pilot is installed in the device with the contributing safety checklists. It also contains the setting of the ISB limits of the user and a state in which the user can test and experience the safety mechanisms of the device.

Finally, the operative superstate contains the identification protocol, an idle state in which the different compensation models are trained[3] and finally a state in which the different compensation models can be selected and tuned.

State transition only through input of the device operator. The operator has to actively press an 'advance' button to induce a state transition. Only in some unique cases where a state transition directly causes more user safety such a transition is triggered within the logic of the FSM itself.

### 4.3.4 Graphical User Interface

Operating all the functionality is done through a GUI. This GUI was designed for the operator of the device during experiments. It contains the controls for the Finite State Machine and provides visual feedback for the operator.

The GUI is mainly designed for development and relies on a experienced operator to be used correctly. However, the integration of the FSM in the GUI should inform and prevent an unfamiliar operator to make any large mistakes that could cause harm to the pilot.

The GUI currently contains many functionalities that should be removed when actual tests are performed with human pilots. These features purely exist to ease the development of the GUI and all other software. While these features are still present in the system, they can be easily removed without altering the required functionality of the GUI.

[3]Training is optional and can be skipped if trained models already exist or if only the AP model is required.

## 4.4 Validation methods

Now that the mathematical tools have been applied to the DAROR-01, the resulting models require validation. For this, a performance parameter is required to express the performance of the models. Furthermore, the design and implementation of the models also needs to be checked. Finally, whether the method for checking if the models are a CVF is explained.

### 4.4.1 Performance Validation

In the next Chapter the performance of all models are compared and discussed. For this comparison the validation datasets (see Sec. 4.3.2) are used. For both systems a separate validation dataset was recorded, which is used for the performance analysis. The performance of the compensation models is expressed in terms of the Root Mean Squared Error (RSME)

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\tau_{v,i} - \tau_{m,i})^2}, \tag{4.31}$$

where $\tau_v$ is the validation torque and $tau_m$ is the compensation model torque. RMSE (in Nm) thus is the error between the torques of the validation dataset and the torques predicted (given the attributing joint angles) by the compensation models.

While RMSE is a relatively good measure for expressing how well the compensation model predicts the validation data, there are two main drawbacks. Both are caused by the fact that RMSE is an expression of absolute error.

The first drawback (of this implementation) of RMSE is caused by the fact that not all actuators have to deliver the same range of torque. If the model is able to learn the structure of the actuator that has the highest torque range well and structure of the other actuators poorly, the RMSE could still be quite low. It could be argued that performing well on 'stronger' actuators is more important. However, this makes comparing the performance of the compensation models less fair, since RMSE is biased for performance on high torques.

Secondly, using regular RMSE, the model performance on dataset A cannot be compared to its performance on dataset B. This is because dataset B contains much higher torques (since the system had higher mass and added joint stiffness).

To circumvent this bias in RMSE, it needs to be normalized. This is done by dividing the error between the validation torque and the model torque with the peak validation torque, per individual actuator over $N$ samples

$$\text{nRMSE} = \sqrt{\frac{1}{N}\sum_{j=1}^{4}\sum_{i=1}^{N}\left(\frac{\tau_{v,i,j} - \tau_{m,i,j}}{\max(\tau_{v,j})}\right)^2}, \tag{4.32}$$

where $\tau_v$ is the validation torque of the dataset and $\tau_m$ the prediction of the compensation model. Subscript $j$ represents actuator $1, 2, 3, 4$. Due to this normalization, nRMSE has no unit.

### 4.4.2 Model Validation

The method for validating the implementation of the models described in Sec. 3.4 is executed. The results of the PLF and PLF+ are shown in Fig. 4.6. The training RMSE[4]

[4]Here the training RMSE is considered since the outcome of this test is to find whether the models are able to sufficiently find the structure of the data

is $3.52 \cdot 10^{-15}$ Nm and $8.87 \cdot 10^{-15}$ Nm for the PLF and the PLF+ method respectively. Such a small fitting error validates that the implementation of the model is assumed to be correct.

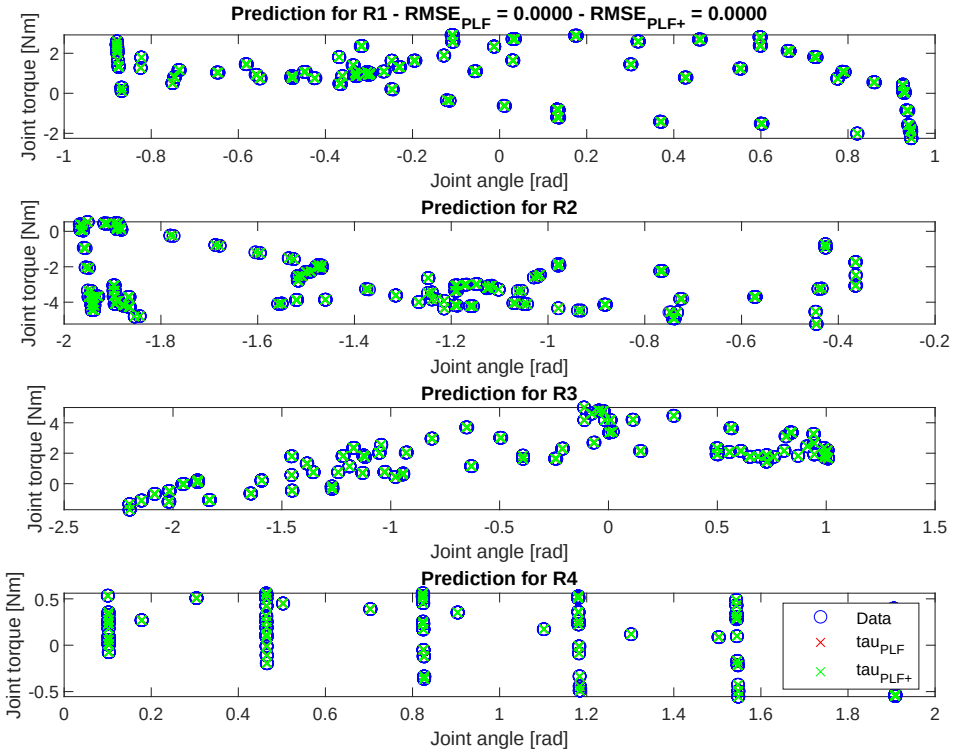


**Figure 4.6:** Validation of the implementation of the PLF models

The same method is employed for the PENN and NNN. Since these models have not been provided the structure of the system, they might not necessarily get as small of a RMSE as the PLF(+) models. The results are shown in Fig. 4.7. Training RSMEs of 0.8962 Nm and 1.0099 Nm are found for the PENN and the NNN respectively.

That these values are this high is most likely caused by the fact that the HP set is selected based on validation loss. As discussed in Sec. 4.2.6 the smallest MVL is found for small networks. It would seem that these small networks lack the ability to find the gravitational structure of the DAROR-01. Neglecting this fact and choosing larger network structures decreases the training RMSE but increases the validation RMSE. Form Fig. 4.7 can be observed that the compensation model torque does follow the trend of the training torque.

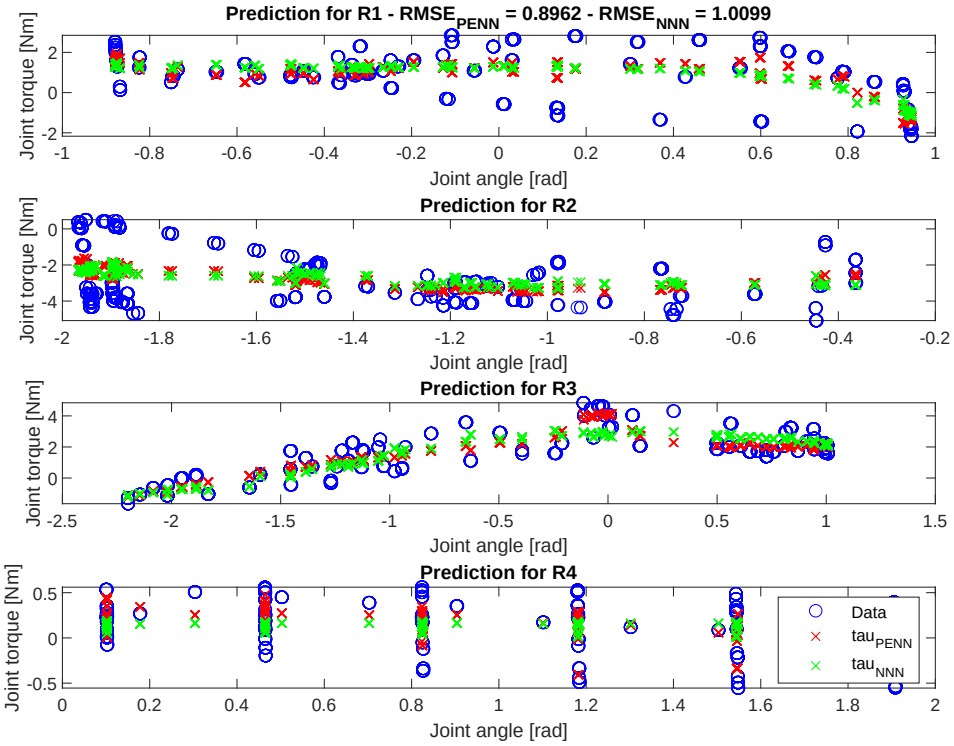


**Figure 4.7:** Validation of the implementation of the NNs

### 4.4.3 Dummy Arm

The compensation models must also be validated under load. A healthy pilot might use some muscular force during the identification protocol and therefore would introduce inaccuracies. Therefore, a passive representation of the upper extremity is developed.

The Dummy Arm (DA) is a model representation of a human (upper and lower) arm, developed by Bas van der Burgh with the goal to be used as testing system in the DAROR-01. The DA has similar DoF, mass, CoM and stiffness to a human arm. The arm can easily be installed in the DAROR–01 and used for testing the system as if a DMD patient is in it. Two springs are attached similarly from the upper arm to the lower arm. One spring resists flexion, like the triceps. The other resists extension, like the biceps. The stiffness of these springs is 2.84 Nm/rad.

**Figure 4.8:** The Dummy Arm

From Fig. 4.8 it can be seen that the DA consist out of two main parts, the upper arm and the lower arm. The DA has the following properties:

- The mass of the upper arm is 1.710 kg
- The mass of the lower arm is 1.053 kg
- The CoM of the upper arm is located at 13.6 cm (w.r.t. the CoR)
- The CoM of the lower arm is 10.9 cm (w.r.t. the elbow joint)
- The distance between the CoR and the elbow joint is 27.5 cm.

### 4.4.4 Energy Conservation

Sec. 3.1.3 introduces the concept of conservative forces. Since gravitational force is a typical conservative force, the compensation model should also be conservative. Furthermore, given the interaction with humans/patients, the compensation model should not gain —and ideally not cost— energy during use.

Determining whether the output of a model is a conservative vector field can be done numerically. As mentioned in Sec. 3.1.3 either two unique paths with the same start and end point should cause the same increase or decrease of (potential) energy, or taking a closed path should result in zero energy gain. The latter method is selected for validating whether the compensation models return a conservative force.

To check whether the models have gained/lost energy when returning to the starting

location of the closed path, Eq. 3.14 can be used. If rewritten to

$$\sum \delta V = \sum \tau_g \delta q, \tag{4.33}$$

the total energy to take the closed path can be computed. Here, $\delta q$ is the size of the step in $q$-space, $\tau_g$ is the gravitational torque computed by the model. The sum of all steps over the path is taken to get $\delta V$; the net PE over the closed loop.

Since this is a numerical integration, computation errors might cause the net energy to not be zero. However, if the stepsize is decreased this should in turn also decrease the PE. In that case, the resultant value of $\delta V$ can be attributed to computation errors.

# Chapter 5

# Results

The performance of all methods is analysed through offline methods. This implies that the performance of the models is tested outside of the actual environment of the DAROR-01. They instead are tested on validation datasets.

## 5.1 Offline Results

The offline performance of the models is analysed through the (n)RMSE of the fit given the validation data. In Fig. 5.1 the validation torque $\tau_v$ (blue) is displayed with all five compensation models when only the DAROR-01 is considered. Fig. 5.2 shows the validation torque and the compensation models when the DA is added to the system. The results for both datasets are shown in Table III.

Figures 5.1 and 5.2 display the validation torque $\tau_v$ measured during the validation trajectory. As mentioned in 4.3.2 for the validation dataset only a single datapoint at every waypoint, at which the system is stationary.

Considering the results from Table III, it is clear that for the dataset A, both PLF and PLF+ compensation models perform the best, followed by the AP model and PENN and NNN have the worst performance.

The performance of the compensation models on dataset B should be the most representative for how the models will perform when a pilot is using the DAROR-01. Clearly the PLF+ model has the best performance. This model is able to learn both the gravitational model of the DAROR-01 and the DA but also is able to find a compensation model for the added joint stiffness. The AP and PLF model have very similar performances when considering RMSE on this dataset. The AP model contained a compensation function for the joint stiffness (see Sec. 4.1). If this stiffness compensation is set to zero, the RMSE of the AP model increases to 1.39 Nm, which is quite a lot higher then the RMSE of PLF (= 1.24 Nm). Thus, the PLF algorithm is better at learning the gravitational structure of dataset B, but when a priori knowlegde of the joint stiffness is used in the AP model, this stiffness compensation makes up for the worse gravity compensation. This is also clear when comparing the performance of AP with PLF+, two models that both contain stiffness compensation modules, where PLF+ has a much lower RMSE than AP. The PENN and NNN perform the worst. The PENN model outperforms the NNN on both datasets with a small margin. This shows that adding Eq. 3.17 slightly improves the overall performance of a NN model. The difference in performance is larger for dataset B.

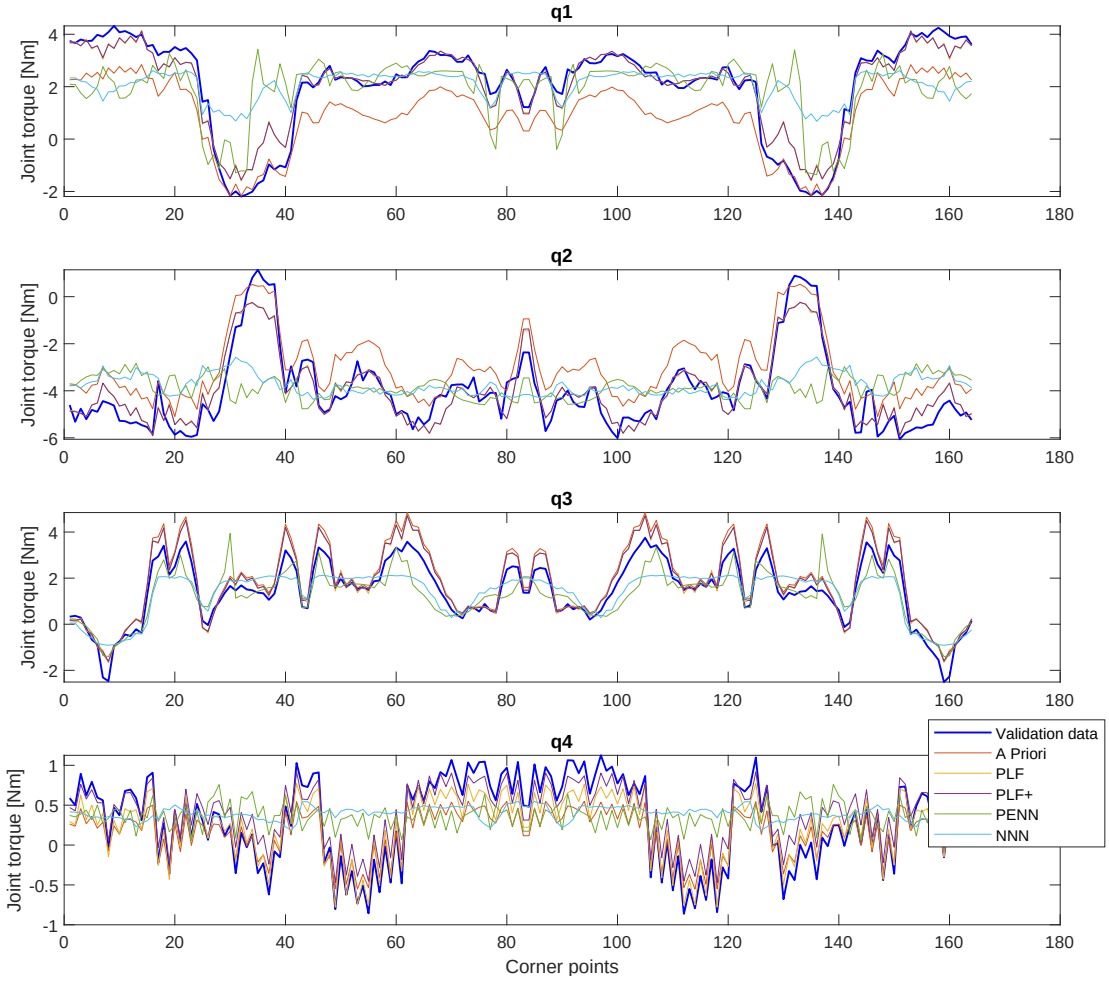


**Figure 5.1:** Validation data and the compensation models on the dataset of only the DAROR-01

**Table III:** Results of the compensation models w.r.t. the validation dataset (A/B) expressed in (normalized) Root Mean Squared Error (RMSE) and nRMSE.

| Model | RMSE A (Nm) | nRMSE A () | RMSE B (Nm) | nRMSE B () |
|---|---|---|---|---|
| A Priori | 0.909 | 0.231 | 1.21 | 0.176 |
| PLF | 0.476 | 0.134 | 1.24 | 0.190 |
| PLF+ | 0.475 | 0.140 | 0.742 | $9.32 \cdot 10^{-2}$ |
| PENN | 1.18 | 0.327 | 2.38 | 0.315 |
| NNN | 1.21 | 0.338 | 2.59 | 0.345 |

The nRMSEs of the models between dataset A and B should be similar if the proposed normalization was fair. This is the case for all values except for the PLF model, of which the nRMSE increases, and thus the performance decreases. This is to be expected since dataset B introduces joint stiffness on the fourth joint, where this model is unable to learn that structure.

Finally, the performance of the NNs is relatively constant for dataset A and B. This is expected as these models contain no extra terms or knowledge on how the system of A differs from B. However, it is noteworthy that the nRMSE of NNN increases when the DA is added to the system, whereas the nRMSE of PENN decreases.

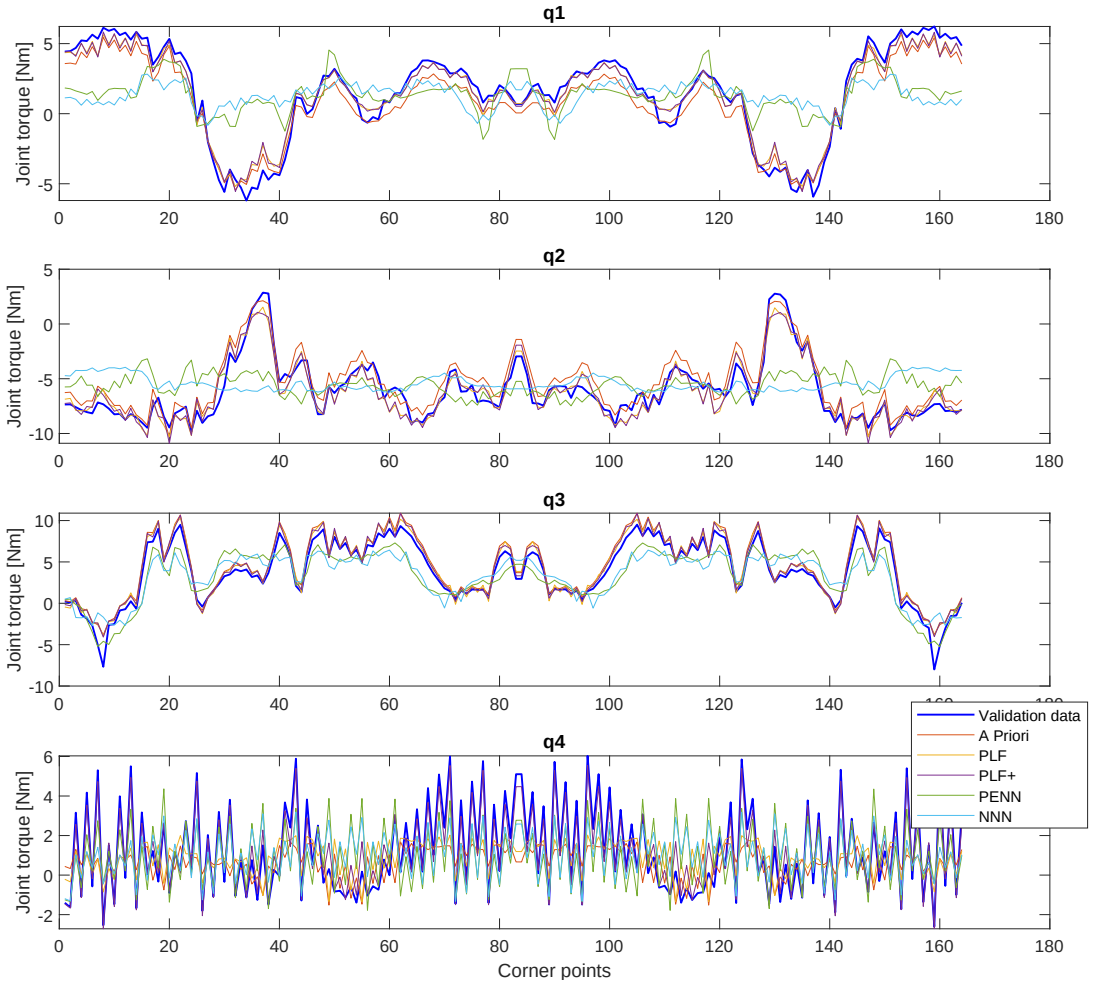


**Figure 5.2:** Validation data and the compensation models on the dataset of the DAROR-01 and the Dummy Arm

## 5.2 Conservative Force Analysis

The path through $q$-space is shown in Fig. 5.3. It is a four-dimensional path, stepping (first positive, then negative) $\pi/4$ in every $q$.

This $q$-space is used as input for the PLF, PLF+, PENN and NNN model. The results of these tests is shown in Table IV.

For models PLF, PLF+ and PENN the value for $\delta V$ decreases with $\delta q$. Thus the remaining value for $\delta V$ can be attributed to computational inaccuracies and $\delta V = 0$ can be concluded. Thus, the output of these models is a conservative vector field.

For NNN this is not the case. $\delta V$ has decreases slightly over the closed path, indepen-

**Table IV:** The results from the cvf test of the data-driven models

| **Model** | $\delta V$ $(\delta q = \frac{pi}{400})$ | $\delta V$ $(\delta q = \frac{pi}{4000})$ | $\delta V$ $(\delta q = \frac{pi}{40000})$ |
|---|---|---|---|
| PLF | 0.014 | 0.0014 | $1.35{\cdot}10^{-4}$ |
| PLF+ | 0.013 | 0.0013 | $1.29{\cdot}10^{-4}$ |
| PENN | 0.0178 | 0.0018 | $1.76{\cdot}10^{-4}$ |
| NNN | -0.0223 | -0.0298 | -0.0305 |

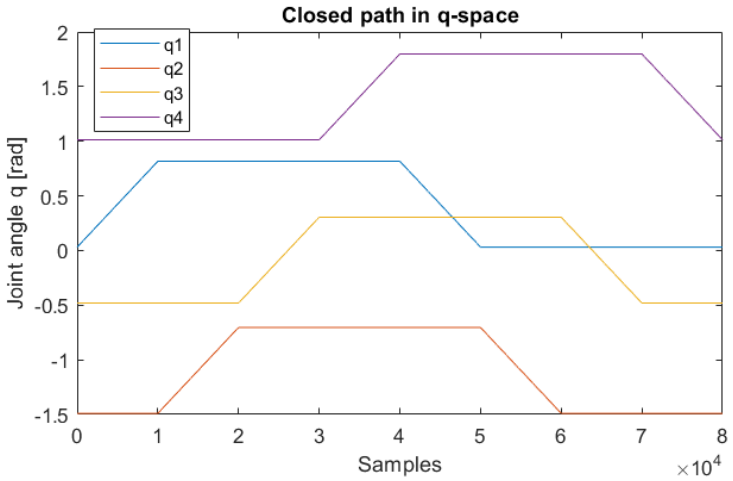


**Figure 5.3:** Closed path in $q$-space

dent of the stepsize $\delta q$ and therefore it cannot be a CVF. This decrease means that PE is lost relative to the start position and therefore the model delivers energy to the system. Thus, when a pilot is moving with this model the system would deliver energy and thus make it easier to move. However, this is unpredictable and undesirable.

## 5.3 Ease of Use

The online performance of the different models is hard during examine. Furthermore, most likely the offline analysis is the best indication for determining the best compensation model. Still, the practical (online) use of the compensation models can be explored. The main obstacle of the data-driven models is the identification protocol, consisting of the trajectory for the data acquisition and the training of the models.

The models all require some input of either a priori knowledge of the human body or data acquired during the identification protocol. Furthermore all data-driven models require some duration of both the identification protocol and thereafter computation of the gravitational models. On the other hand the AP model requires taking measurements of the pilot before installation. The durations of these processes can be compared with respect to the offline performance.

Table V shows how long it takes to train a single instance of the data-driven models. Clearly, training PENN requires a lot of time, which can be contributed to taking the partial derivatives with respect to the inputs. More precisely, the computation of the partial derivative of Eq. 4.29 is quite slow, since the auto-differentation is not really being exploited here.

Furthermore, Sec. 4.2.6 concludes that PENN and NNN require several trained net-

**Table V:** Training duration of one model

| Model | Training time (s) |
|---|---|
| PLF | 4.5 |
| PLF+ | 4.5 |
| PENN | 36.7 |
| NNN | 15.3 |

works to have a higher probability for finding a good performing network. This will scale the training time with however many random initialized networks are chosen to be trained.

Next to training time, the identification protocol also requires quite some time to be executed. Given that this requires between 5-15 minutes (see Sec. 4.3.1), this is the time determining step for the data driven models. However, as Sec. 5.3.1 mentions, the identification protocol can be reduced, thus also decreasing training time.

For the generation of the AP model, the minimal measurements of the human pilot that are required are body mass, body length and sex. Using only these three measurements for the estimation of the human parameters logically requires more assumptions, which introduce uncertainty into the system since they most likely do not hold for the specific pilot. Still, only taking these measurements takes several minutes, given that the pilot has DMD.

### 5.3.1 Trajectory Optimization

From the analysis performed in Chapter 5 PLF+ (and PLF for the case with no joint stiffness) are the best performing compensation models. However, these models require quite some time to be trained, especially if the long identification protocol is considered to be part of the 'training' time. This raises the question of how much data is actually required for a good model. With this information, a shorter and more optimal identification trajectory could be designed. This would increase the ease of use of the compensation model.

In order to determiner how much training data is required, a PLF+ model is trained using only $n$ datapoints. These datapoints are randomly selected from the training dataset. Per $n$ datapoints, 100 models are trained, to avoid effects of overfitting. The four models are validated and the RMSE is stored. Thereafter, $n$ is increased. This is repeated until $n = N$, the total training dataset.

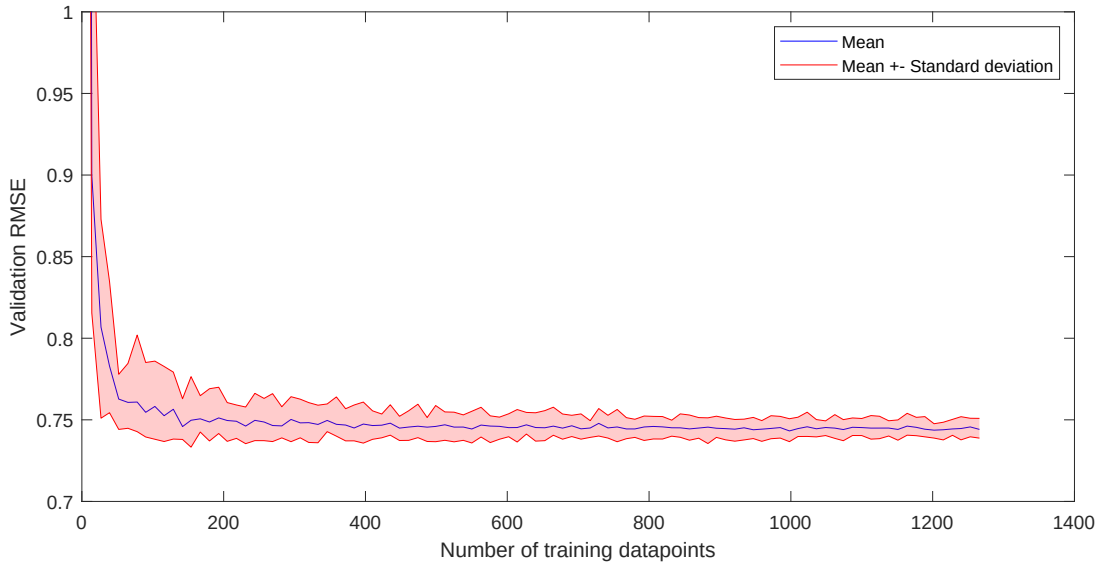


**Figure 5.4:** Performance (expressed in validation RMSE) of PLF+ trained on increasing training dataset sizes

During this process, it became clear that using all 25000 datapoints was not required. Training a model with 100% of the dataset yields a mean validation RMSE of 0.743 Nm,

where the mean validation RMSE of 5% of the dataset is 0.744 Nm. Therefore, the choice was made to run the analysis up to this (0.05N) point. Fig. 5.4 shows the results of the analysis for 1266 (5% of N) datapoints. The figure shows that instead of using 1266 training datapoints only 200-300 ($\simeq$ 1% of N) are required. After this point the mean validation RMSE (blue line) is nearly constant. This means that the identification protocol could be reduced by a large margin without seeing a difference in performance of the gravity and stiffness compensation. Important to note is that for a larger randomly sampled training dataset, the standard deviation decreases. This does point to an upside of a larger training dataset if only a single model is trained.

This conclusion was obtained after all datasets had been gathered. Given that the performance is similar[1] for larger datasets, the choice was made to use these datasets for the analysis and results of this work. However, to speed up training, the datasets were downsampled to approximately 1250 datapoints[2].

[1]Larger datasets yield a slightly better RMSE: ($\text{RMSE}_{n=320}$ = 0.748 Nm, $\text{RMSE}_{n=1266}$ = 0.744 Nm, $\text{RMSE}_{n=N}$ = 0.743 Nm)

[2]This choice was made before obtaining this fact, but is now validated

# Chapter 6

# Discussion

The goal of this work was to design and compare gravity and stiffness compensation. The eventual goal of the larger research team is to see whether a well performing gravity and stiffness compensation model in this active exoskeleton aids people with DMD.

This work thus focused on modelling strategies with data-driven models at its core. From literature we know that finding the structure for such a data-driven model is a challenge. Also, non-linearities introduced by incorrect modelling assumptions could reduce the performance of parameterized models. This let us to attempt using NNs that automatically find some structure from data and are able to map non-linearity. The models in literature that did use a predefined structure followed the same method as we did in this work; separating the EoM into PLF.

## 6.1 Limitations

This work focuses on the different model types and how their performances compare. During the design of the models the main goal was to find a good gravity and stiffness compensation model. The 3 main model types (AP, PLF and NN) were the chosen at the start of the design phase. However, when either of the model types started working well, that type got more attention then the others. This meant that there is a bias in the comparison towards the best performing model types; the PLF models.

The PLF model is constructed from the kinematic structure of the DAROR-01. Within the defined structure, several constant rotation matrices are used. While the production of the DAROR-01 was surely done with care, there is no direct guarantee that these actual static rotations exactly match the theoretical design. Especially since we deal with gravity, a slight deviation in angle could misalign the models direction of gravity with the actual direction.

Other limitations of this work is that it focuses fully on the DAROR-01. While Chapter 3 described the generic forms of the models, these were specifically selected with this system in mind.

The rest of this Chapter will discuss more specific parts of the work where maybe some other choices could have been made.

## 6.2 Model Performance Resolution

In Chapter 5 the RMSE and nRMSE are shown per compensation model. While the conclusion drawn from Table III is quite clear, it should be noted that the analysis did not take into account the performance resolution.

Here, with performance resolution, the range in which the model performance can be, is meant. This depends on the random initialization of the training data. In Sec. 4.3.2 it is mentioned that before training, the data is randomly shuffled. This shuffling occurs before the downsampling in the algorithm. These two factors together cause every training dataset to be (slightly) different. This causes the data-driven models to find slightly different solutions during training and in turn the performance on the training data is different every time a model is trained (with randomly selected training data).

To avoid this problem, we propose training multiple models and see what the performance (RMSE) range is. For the PLF+ model, Sec. 5.3.1 this process has been done. From this, we see that the range is quite small for the downsampling rate that was selected for the results of Table III. Furthermore, during the design and implementation process (of the models but also of the facilitative functions), many models were trained and their performance was compared. The performance ranking of the models has always been as is concluded from Chapter 5. If we were to estimate a performance resolution from the experience of the aforementioned phases, it would be three significant digits, which is the same as the resolution chosen for the torque measurements.

It certainly would have been better to train multiple randomly initialized models to be able to report the performance resolution. However, the above arguments are valid and point to no significant changes in the conclusion drawn from this work.

## 6.3 Torque Measurement Error

Critical to data-driven models is good quality data. As Sec. 2.2.2 discusses, the torque measurements contain some error. What causes this error is not clear, but we expect either static friction on between the elements on either side of the torsion spring, or movement of the encoder that measures the spring deflection. Either way, the data used for training the (data-driven) models thus contains quite a large measurement error. Most likely this phenomenon (or multiple phenomena) are non-linear and therefore decrease the performance of the PLF(+) models. The NN models should suffer less from this non-linearity but given their worse performance it is difficult to say whether this is actually true.

Decreasing this measurement error is not trivial. First, more research is required to the cause of the error, which most likely requires taking the actuator apart. Suggestions on how this error might be reduced on the model-side are given in Sec. 6.6.

## 6.4 Brute Force Search over HP

In Sec. 4.2.5 the brute force search to find the best performing HP set was explored. However, such an approach does come with several assumptions.

First of all, only 2 random initialized networks per HP set were trained. This was done to speed up the computation, since training 432 networks for a long enough time to have proper results already lasted several days. Introducing more randomly initialized networks , i.e. 4 instead of 2, would double the total computation time and give no guarantee for a more 'optimal' HP set. Furthermore, it is not clear if a more extensive search over a larger range of HP would have yielded better results. It might be possible that a more extensive search would only find worse HP sets for example.

Once the results from the HP optimization were found, they could be interpreted in several ways. One would be to say that the set best performing HP could be taken as the overall best performing set. Another method would be to consider every HP individually and take some weighted average of the top 10. However, this method does not consider

interaction between HPs and therefore was not selected. The current approach —selecting the best performing set— is very prone to change, given that the random initialization could be very optimal for that HP set during the brute force search. Due to the random initialization, the change that such an optimal initialization occurs again is small. This problem could be avoided by selecting the best average MVL but doing so might introduce the same random occurrence of the worst random initialization for that set of HP and then the average would still not account for much. Therefore, the method now used selects for network that has the highest potential of being the 'best' and that why it is selected.

## 6.5 Implementation and Design of the NNs

While the implementation of both NNs was extensively tested and checked, no proper explanation is found for the bad performance. Eventually, the implementation and design was deemed to be correct and we drew the conclusion that the structure of the data was too complex for such simple networks. However, given the power of NNs in other areas of science, it would seem strange that the structure of the data of this system would be 'too complex'. A more likely explanation would be some implementation or design flaw that was not found during the research.

Given that the PENN was proven to model a CVF, this hints towards proper implementation of the network. Also, this fact is promising in that a random, non-linear function can be forced to learn a function that adheres to Lagrangian dynamics. More research into the design of a PENN (or LNN) could possibly yield better results and be very useful. The upside of using PENN remains that it would be able to also learn non-linearity present in the system.

Furthermore, in Sec. 5.1 the nRMSE of PENN is observed to decrease when the DA is added to the system, while the nRMSE of NNN increases. This difference could be a good first step into figuring out what part of the implementation is incorrect. Quite some time was invested in tuning the type and size of the training dataset but it could be possible that something was overlooked.

While further work on PENN could potentially yield better results, it seems likely that a lot of data will be required for training. When working with DMD patients the duration of the identification protocol must be considered. The protocol used for the generation of the datasets of this work was deemed too long for a DMD patient to carry out. This, in combination with the longer training duration decrease the usability of this compensation model, even if the results are improved.

## 6.6 Residual RMSE

While the performance of the PLF+ compensation model is good, some error remains between the actual validation and the prediction torque: the RMSE is not zero. Three explanations for this residual error are given.

The first is that it is caused by sensor noise or sensor inaccuracies. Both the joint position and the joint torque depend on the absolute encoders of the SEA, which are subject to noise. Furthermore, if the spring constant used for the computation of the joint torque is slightly different, the sensor data no longer complies with the actual torque experienced by the system. For the PLF+ model this effect should be reduced in the elbow due to the stiffness compensation. However, this is not observed from the nRMSE of dataset A. These uncertainties remain challenging to determine or avoid. Furthermore,

the measurement error of Sec. 2.2.2 falls in this category, greatly reducing the quality of the torque measurement data.

The second explanation is that the assumptions made in the model on the physical configuration of the system might not be correct. Sec. 2.1 discusses the kinematic structure of the system and introduces some rotation matrices with constant values based on the design of the system. However, production and construction can only be done with limited accuracy. Even validation of these rotations after the construction of the system is finished is limited. Errors in these values essentially introduce non-linearity into the compensation model.

Finally, the actual system contains more complexity than the PLF+ system is designed to handle. Non-linearity caused by, for example, the joint torque measurement error or static friction could influence the joint torque, while PLF+ is unable to recognise and compensate for such effects.

## 6.7 External Challenges

An external challenge influenced the process of this thesis quite significantly. In January 2024 it was discovered that the file that contained all variable names contained an error. This caused six variable names per actuator to be shifted w.r.t . the actual variable in the datastruct. Sadly, the two variables (per actuator) used for the data driven models were among these six variables. This meant that instead of finding a relation between joint angle and joint torque, a relation between gearbox angle and motor torque was attempted to be found. Since the unit of the signal was similar, the models were able to find some structure but they were off by quite a large margin. This was deemed to be caused by measurement error of the joint torque (see Sec. 2.2.2). A lot of work was poured into trying to solve or circumvent this problem. This was all for nothing once the mistake was discovered and the performance of the models improved. If this error had not been there maybe more work could have been done (see Chapter 7).

# Chapter 7

# Future work

## 7.1 Trajectory Optimization

Sec. 5.3.1 touches on the fact that the current identification protocol could potentially be reduced by a factor 100 without significantly decreasing the performance. Something that was not touched on in that section is that more unique the input training data ($q$) of the regressor is causes the model to better generalise. These two factors raise the question whether an optimized trajectory exists which contains optimal configurations. Potentially, using such training data could reduce the standard deviation observed in figure 5.4, thus requiring only very few data while maintaining the compensation model performance. Further research into this subject has led to expressing the quality of waypoints in terms of how well they decrease the estimated parameter covariance of the model fit.

Given the good performance of the data driven PLF+ model, optimizing the identification protocol is a final step towards making the software actually applicable in the lives of DMD patients.

## 7.2 Online Validation

While (normalized) RMSE is a good performance indicator, it does not take user experience into account. How these models perform when used by a pilot remains speculation in this work, as does the subjective experience of the pilot. Possibly, some objective performance indicators could be developed for online analysis and validation of the compensation models. Finally, some method for objectively scoring the compensation models based on user experience would be good to develop.

Initial informal blind comparative tests show that RMSE is a good indicator for how 'good' a compensation methods feels for a healthy pilot. Furthermore, the PLF+ model provided a healthy pilot with good gravity and stiffness compensation, although stiffness compensation in healthy subjects is much less of a factor than gravity.

## 7.3 Compensation Model Combinations

In chapter 6, the residual error of the PLF+ system is discussed. Whatever the specific cause of the residual error is, the effect is most likely non-linear. Further reduction of the error could be achieved by isolating the residual error and trying to find a compensation model for it. This might be possible using another parameter linear form where another structure and/or another input-output pair is considered. This would require knowledge

about the cause of the effect. Another, easier, method would be use a (PE)NN and see if it is able to map the non-linear structure. For PENN to work, the non-linear behaviour would have to adhere to Lagrangian mechanics.

## 7.4 Testing with People with DMD

For now, only either offline tests with the DAROR-01 and DA, or online tests with a healthy pilot have been done. The goal of the larger research project of the DAROR-01 is to discover whether such an exoskeleton —with the proposed compensation models— would increase the QoL of people with DMD and could maybe even postpone the diseases progression. This all remains a question for now and it would be very interesting to find out how much this system can aid these people.

For now, a proof of concept for the gravity and stiffness compensation models is developed. Further tests in healthy subjects and initial tests in subjects with DMD are required to see whether the models and the corresponding protocols are viable.

# Chapter 8

# Conclusion

Data-driven approaches for gravity and joint stiffness compensation models can yield better results than generating a model using only a priori knowledge of the system. For this, the kinematic structure of the system is required as a structure for the model. The Parameter Linear Form + joint stiffness compensation method (PLF+) is able to learn the gravity and stiffness compensation model from a dataset containing only stationary data.

More (grey and) black-box type models, such as the proposed PENN and NNN are able to learn the gravity and stiffness models to some extent but are not able to find the complete complex structure of the data. The PENN model was proven to be energy conservative and it outperformed NNN.

Given the current identification protocol and training duration, in some cases an AP model might still be preferred. However, the identification protocol can be optimized by a large margin, possibly reducing the identification and training time to only several minutes.

Data-driven modelling methods are proven to be a viable concept for the compensation models required for active arm support devices for people with DMD.

# Bibliography


[1] J. K. Mah, L. Korngut, K. M. Fiest, J. Dykeman, L. J. Day, T. Pringsheim, and N. Jette, "A Systematic Review and Meta-analysis on the Epidemiology of the Muscular Dystrophies," *Canadian Journal of Neurological Sciences*, vol. 43, no. 1, pp. 163–177, 2015.

[2] J. R. Mendell and M. Lloyd-Puryear, "Report Of MDA Muscle Disease Symposium On Newborn Screening For Duchenne Muscular Dystrophy," *Muscle and Nerve*, vol. 48, no. 1, pp. 21–26, 2013.

[3] M. M. Janssen, J. Harlaar, B. Koopman, and I. J. de Groot, "Unraveling upper extremity performance in Duchenne muscular dystrophy: A biophysical model," *Neuromuscular Disorders*, vol. 29, no. 5, pp. 368–375, 2019.

[4] M. M. Janssen, J. Harlaar, B. Koopman, and I. J. De Groot, "Dynamic arm study: Quantitative description of upper extremity function and activity of boys and men with duchenne muscular dystrophy," *Journal of NeuroEngineering and Rehabilitation*, vol. 14, no. 1, 2017.

[5] M. Jansen, N. Van Alfen, A. C. Geurts, and I. J. De Groot, "Assisted bicycle training delays functional deterioration in boys with Duchenne muscular dystrophy: The randomized controlled trial "no use is disuse"," *Neurorehabilitation and Neural Repair*, vol. 27, no. 9, pp. 816–827, 2013.

[6] S. Filius, M. Janssen, H. van der Kooij, and J. Harlaar, "Comparison of Lower Arm Weight and Passive Elbow Joint Impedance Compensation Strategies in Non-Disabled Participants," *IEEE ... International Conference on Rehabilitation Robotics : [proceedings]*, vol. 2023, pp. 1–6, 2023.

[7] S. J. Filius, J. Harlaar, L. Alberts, S. H.-v. Opstal, H. V. D. Kooij, and M. M. H. P. Janssen, "Design requirements of upper extremity supports for daily use in Duchenne muscular dystrophy with severe muscle weakness," vol. 11, pp. 1–18, 2024.

[8] V. Longatelli, A. Antonietti, E. Biffi, E. Diella, M. G. D'Angelo, M. Rossini, F. Molteni, M. Bocciolone, A. Pedrocchi, and M. Gandolla, "User-centred assistive SystEm for arm Functions in neUromuscuLar subjects (USEFUL): a randomized controlled study," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, no. 1, pp. 1–16, 2021.

[9] S. Plagenhoef, F. Gaynor Evans, and T. Abdelnour, "Anatomical Data for Analyzing Human Motion," *Research Quarterly for Exercise and Sport*, vol. 54, no. 2, pp. 169–178, 1983.

[10] V. Arakelian, "Gravity compensation in robotics," *Advanced Robotics*, vol. 30, no. 2, pp. 79–96, 2016.

[11] R. Ott, M. Gutierrez, D. Thalmann, and F. Vexo, "Improving user comfort in haptic virtual environments through gravity compensation," *Proceedings - 1st Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems; World Haptics Conference, WHC 2005*, pp. 401–409, 2005.

[12] S. Moubarak, M. T. Pham, R. Moreau, and T. Redarce, "Gravity compensation of an upper extremity exoskeleton," *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, pp. 4489–4493, 2010.

[13] M. Liu and N. H. Quach, "Estimation and compensation of gravity and friction forces for robot arms: Theory and experiments," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 31, no. 4, pp. 339–354, 2001.

[14] A. Ugartemendia, D. Rosquete, J. J. Gil, I. Diaz, and D. Borro, "Machine learning for active gravity compensation in robotics: Application to neurological rehabilitation systems," *IEEE Robotics and Automation Magazine*, vol. 27, no. 2, pp. 78–86, 2020.

[15] M. Lutter, C. Ritter, J. Peters, C. Science, T. Universit, D. Hochschulstr, D. L. Networks, L. Mechanics, and L. Mechanics, "Published as a conference paper at ICLR 2019," pp. 1–17, 2019.

[16] A. H. Stienen and A. Q. Keemink, "Visualization of shoulder range of motion for clinical diagnostics and device development," *IEEE International Conference on Rehabilitation Robotics*, vol. 2015-September, pp. 816–821, 2015.

[17] W. F. Rampeltshammer, A. Q. Keemink, and H. Van Der Kooij, "An Improved Force Controller with Low and Passive Apparent Impedance for Series Elastic Actuators," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 3, pp. 1220–1230, 2020.

[18] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[19] Wikipedia, "Neural networks."

[20] Elastico, "What is a neural network?."

[21] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, and D. Spergel, "Lagrangian neural metworks," pp. 1–9.

[22] G. B. Thomas, M. B. Weir, and J. R. Hass, "Thomas' Calculus Early Trenscendentals," in *Thomas' Calculus Early Transcendentals*, ch. 16.3, pp. 938–939, Pearson Eduction Limited, 12 ed., 2014.

[23] J. C. A. Barata and M. S. Hussein, "The Moore-Penrose Pseudoinverse: A Tutorial Review of the Theory," *Brazilian Journal of Physics*, vol. 42, no. 1-2, pp. 146–165, 2012.

[24] R. Persico, "The Singular Value Decomposition," *Introduction to Ground Penetrating Radar: Inverse Scattering and Data Processing*, vol. 9781118305003, pp. 229–241, 2014.

[25] G. Chen, "Lecture 6: Generalized inverse and pseudoinverse,"

[26] G. Hinton, "Boltzmann Machines," *Encyclopedia of Machine Learning and Data Mining*, pp. 164–168, 2017.

[27] G. von Leibniz, J. Child, and C. Gerhardt, *The Early Mathematical Manuscripts of Leibniz: Translated from the Latin Texts Published by Carl Immanuel Gerhardt with Critical and Historical Notes*. Open court publishing Company, 1920.

[28] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[29] *Anthropometry*, ch. 4, pp. 82–106. John Wiley Sons, Ltd, 2009.

[30] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[31] Wikipedia, "Cross-validation (statistics)."

# Appendix A

# Finite State Machine Diagram

Fig. A.1 is a schematic of the Finite State Machine (FSM) designed for the DAROR-01. The SM was designed for the experimental use of the DAROR-01 and therefore is carefully built up, gradually allowing the system more functionality when more safety precautions are met.

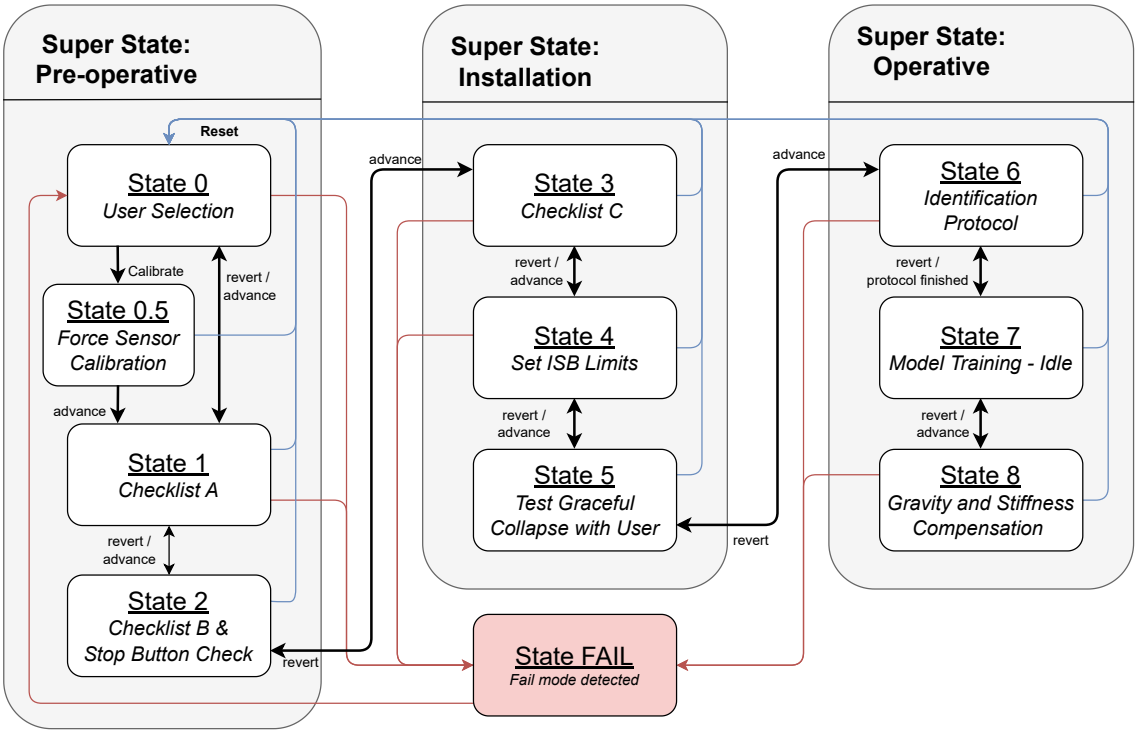


**Figure A.1:** Diagram of the Finite State Machine designed for the development and testing of the DAROR-01

The 'advance' state transition requires at least two conditions. The first is that the Advance button in the GUI has to be pressed by the operator. This ensures that the operator is always attending the system and the pilot when a state transition occurs. The other condition is state dependent and can be read from the *italics* in Fig. A.1. These actions have to be executed by the operator, the system and/or the pilot. Only when this condition is met, can the operator press the Advance button.

The 'revert' state transition is always possible but can only step back one state. The 'Reset' state transition is always possible from any state and requires a button press by

the operator. From most states (not 2, 5 and 7) the FAIL state can be entered when either the system, the operator or the pilot detects a fail condition. This can either be an internal error of the system, or the press of the emergency button by the pilot or operator. The states where the FAIL state cannot be entered the emergency buttons, and their corresponding safety features, are tested and therefore should not trigger a state transition.