



# Influence of Exoskeleton Modeling of an Elbow Exosuit on Simulated Joint Reaction Forces


## Einfluss der Exoskelett-Modellierung eines Ellbogen- Exosuits auf Simulierte Gelenkkontaktkräfte

Scientific Work for Obtaining the Degree

M.Sc. Maschinenwesen

at the School of Engineering and Design of the Technical University of Munich.

**Supervised by** Univ.-Prof. Dr. phil. Klaus Bengler  
M.Sc. Christina Harbauer-Rieß  
Chair of Ergonomics

**Submitted by** Nicolas Niessen  
  
nicolas.niessen@tum.de

**Submitted on** Garching, 09 December 2021



# Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

---

Garching, 09. Dezember 2021, Unterschrift

# Vereinbarung zum Urheberrecht

Hiermit gestatte ich dem Lehrstuhl für Ergonomie diese Studienarbeit bzw. Teile davon nach eigenem Ermessen an Dritte weiterzugeben, zu veröffentlichen oder anderweitig zu nutzen. Mein persönliches Urheberrecht ist über diese Regelung hinaus nicht beeinträchtigt.

Eventuelle Geheimhaltungsvereinbarungen über den Inhalt der Arbeit zwischen mir bzw. dem Lehrstuhl für Ergonomie und Dritten bleiben von dieser Vereinbarung unberührt.

---

Garching, 09. Dezember 2021, Unterschrift

# Lizenzvereinbarung

Der Student/ Die Studentin Nicolas Niessen,

Matrikelnummer [REDACTED] stellt im Rahmen der Studienarbeit

Influence of Exoskeleton Modeling of an Elbow Exosuit on Simulated Joint Reaction Forces

Einfluss der Exoskelett-Modellierung eines Ellbogen- Exosuits auf Simulierte Gelenkkontaktkräfte

entstandenen Quellcode und dazugehörige Dokumentation ("die Software") unter die angekreuzte Lizenz:

**Public Domain**

Eine Textdatei mit dem Lizenztext von [unlicense.org](http://unlicense.org) liegt dem Code bei.

**MIT-Lizenz**

Eine Textdatei mit dem Lizenztext von <http://choosealicense.com/licenses/mit/> liegt der Software bei. (Empfohlen für die meisten Studienarbeiten, wenn keine kommerzielle Nutzung geplant ist.)

**GPLv2**

Eine Textdatei mit dem Lizenztext von <http://choosealicense.com/licenses/gpl-2.0/> liegt der Software bei. (Zwingend für Studienarbeiten, die Bibliotheken nutzen, die unter GPL stehen.)

Der Lehrstuhl für Ergonomie und die Professur für Sportgeräte und -materialien der TU München erhalten die Erlaubnis, die Software uneingeschränkt zu benutzen, inklusive und ohne Ausnahme dem Recht, sie zu verwenden, kopieren, ändern, fusionieren, verlegen, verbreiten, unterlizenzieren und/oder zu verkaufen, und Personen, die diese Software erhalten, diese Rechte zu geben.

Der Urheberrechtsvermerk "© Copyright [2021][Nicolas Niessen]" und folgender Haftungsausschluss sind in allen Kopien oder Teilkopien der Software beizulegen. Die Software wird ohne jede ausdrückliche oder implizierte Garantie bereitgestellt, einschließlich der Garantie zur Benutzung für den vorgesehenen oder einen bestimmten Zweck sowie jeglicher Rechtsverletzung, jedoch nicht darauf beschränkt. In keinem Fall sind die Autoren oder Copyright-Inhaber für jeglichen Schaden oder sonstige Ansprüche haftbar zu machen, ob infolge der Erfüllung eines Vertrages, eines Deliktes oder anders im Zusammenhang mit der Software oder sonstiger Verwendung der Software entstanden.

Unterschrift des Studenten/der Studentin: \_\_\_\_\_



# Abstract

The strain on the body resulting from the use of an exoskeleton can be assessed via musculoskeletal simulations of joint reaction forces (JRFs). JRFs are the forces transferred from one bone to another within a joint. Parameters like the human musculoskeletal model as well as the simulation software and tools can influence the simulated JRFs. Until now, many studies used simple torque actuators to represent the exoskeleton's support in their simulations. The results were then often utilized to optimize the exoskeletons' physical design and controls, which is why a correct computation of the JRFs is important. This thesis investigates the effect that the representation of an exoskeleton has on the JRFs. This is done by implementing and simulating them for different virtual models of the same soft, cable-driven elbow exosuit prototype developed by Harbauer et al. (2020).

The exosuit is implemented in 12 different models. Each of them goes through a simulation workflow including the computed muscle control (CMC) and joint reaction analysis (JRA) tools of OpenSim to acquire the JRFs. The process is automated using MATLAB. A pure elbow flexion motion of lifting a 5 kg weight is supported by the different models with equivalent support. Three control strategies are implemented for this.

The exoskeleton representation is found to affect the simulated JRFs in different ways. Several relevant parameters in the modeling approach are identified and analyzed. Using a coordinate actuator tends to underestimate the forces compared to path actuators. The bone(s) that a path actuator is attached to can play a role for the JRFs. The level of detail in the representation of the cable path along the forearm also has an influence, but is less relevant when the path points are attached to one bone only. For the selected model, motion and exosuit, the exosuit's cable path going over the user's shoulder toward the motors does not have a relevant effect on the elbow JRFs. The simulation results indicate that using an ulna-attached single- or double-stringed path actuator to represent the LfE exosuit prototype is a reasonable choice for JRF computation with the presented workflow. Finally, the simulated elbow JRF values are compared to the results of other publications to evaluate the quality of the simulation workflow, providing confidence in the results. It is furthermore confirmed that the underlying human model, as well as the exoskeleton's control strategy have a large influence on the produced JRFs.

# Contents

|       |   |    |
|-------|---|----|
| 1     | Introduction .....  | 8  |
| 1.1   | Motivation .....  | 8  |
| 1.2   | Research Question .....   | 9  |
| 2     | State of the Art .....  | 10 |
| 2.1   | Exoskeletons .....  | 10 |
| 2.1.1 | Applications and Types.....                                       | 10 |
| 2.1.2 | Design Challenges .....   | 11 |
| 2.1.3 | Soft, Cable-Driven Exoskeletons .....                             | 11 |
| 2.2   | Joint Reaction Forces (JRFs) .....                                | 12 |
| 2.2.1 | Definition.....   | 12 |
| 2.2.2 | Measurements .....  | 13 |
| 2.2.3 | Musculoskeletal Simulation Tools .....                            | 13 |
| 2.2.4 | Dependency on Musculoskeletal Model Setup .....                   | 16 |
| 2.3   | Musculoskeletal Modeling of Exoskeletons .....                    | 16 |
| 2.3.1 | Goal and Purpose .....  | 16 |
| 2.3.2 | Modeling Examples .....   | 17 |
| 2.4   | Human Elbow Physiology .....                                      | 21 |
| 2.5   | Simulated Elbow JRFs .....  | 22 |
| 2.5.1 | Magnitude .....   | 22 |
| 2.5.2 | Pattern for Elbow Flexion.....                                    | 24 |
| 3     | Methodology .....   | 27 |
| 3.1   | Different Implementations of the Same Exoskeleton and Setup ..... | 27 |
| 3.2   | Comparison of Models via JRFs .....                               | 28 |
| 3.3   | Simulative Workflow .....   | 28 |
| 3.4   | Approach and Generation of Different Models .....                 | 30 |
| 4     | Implementation .....  | 31 |
| 4.1   | OpenSim Tools and Setup .....                                     | 31 |
| 4.1.1 | Musculoskeletal Model .....                                       | 31 |
| 4.1.2 | Motion File.....  | 33 |
| 4.1.3 | Muscle Forces via Computed Muscle Control (CMC) .....             | 34 |
| 4.1.4 | Joint Reaction Forces via Joint Reaction Analysis (JRA) .....     | 35 |
| 4.1.5 | Exoskeleton Moment Arms via Muscle Analysis .....                 | 37 |

|       |   |    |
|-------|---|----|
| 4.2   | Automation and Workflow in MATLAB.....  | 37 |
| 4.2.1 | Code and File Structure .....   | 37 |
| 4.2.2 | Generation of Prescribed Force Files .....  | 39 |
| 4.2.3 | Batch Simulation of Models .....  | 39 |
| 4.2.4 | Auxiliary Functions .....   | 41 |
| 4.3   | Exoskeleton Modeling .....  | 42 |
| 4.3.1 | Exoskeleton Actuators and Control .....   | 42 |
| 4.3.2 | Models 0 - 13.....  | 44 |
| 5     | Results .....   | 50 |
| 5.1   | Overview of JRFs (Models 0 – 13) .....  | 50 |
| 5.2   | Comparison of Control Strategies .....  | 52 |
| 5.3   | JRFs for different Models.....  | 53 |
| 5.3.1 | Models without Exoskeleton Support (Models 0 &1).....                                     | 53 |
| 5.3.2 | Coordinate and Path Actuator (Models 2 & 3).....  | 54 |
| 5.3.3 | Single- and Double-Stringed Path Actuators on the Same Body (Models 3 & 7 and 4 & 6)..... | 56 |
| 5.3.4 | Different Attachment Point Placements.....  | 57 |
| 5.3.5 | Path Actuators of Varying Complexity.....   | 61 |
| 5.3.6 | Complex Models with(out) Path over Shoulder (Models 9 & 12 and 7 & 13).....               | 65 |
| 6     | Discussion .....  | 67 |
| 6.1   | Comparison/Evaluation with Literature .....   | 67 |
| 6.2   | Differences in JRFs.....  | 68 |
| 6.2.1 | Actuator Types .....  | 68 |
| 6.2.2 | Single- and Double-Stringed Path Actuators .....  | 69 |
| 6.2.3 | Effect of Attachment Point Placement .....  | 70 |
| 6.2.4 | Effect of Complexity of Path Along Forearm .....  | 71 |
| 6.2.5 | Effect of Actuator Control.....   | 73 |
| 6.3   | Implications for Musculoskeletal Modeling of the Elbow Exosuit.....                       | 74 |
| 6.4   | Alternative Modeling Approaches .....   | 76 |
| 6.5   | Limitations to the Results and their Interpretation.....                                  | 77 |
| 7     | Conclusion .....  | 79 |
| 7.1   | Summary.....  | 79 |
| 7.2   | Key Findings .....  | 80 |
| 7.3   | Further Research Opportunities .....  | 80 |
|       | References .....  | 82 |
|       | Appendix .....  | 94 |

# 1. Introduction

## 1.1. Motivation

Mythological and fictional stories often formulate the dream of superhuman powers. Researchers and engineers have for decades striven towards making weights feel light as feathers becoming reality via exoskeletons. Since the first attempts, the fulfillment of the dream has come closer than ever and wearable robots are now a part of our world.

The global exoskeleton market size was estimated to be USD 218 million for the year 2020 and it is expected to grow by around 20 percent annually until 2028 with the largest market growth in upper extremity exoskeletons (Grand View Research, 2021). The three largest segments in the current market in order are healthcare, military and industry (Grand View Research, 2021). Especially in the latter two, increasing the force and effectivity of a human user, also called human augmentation (Gull, Bai, & Bak, 2020; Grand View Research, 2021; Asbeck, Rossi, Galiana, Ye, & Walsh, 2014) is often the goal. This can mean increasing the user's physical capabilities for the same amount of strain and effort. Besides human augmentation, the exoskeletons act as assistive devices (Grand View Research, 2021), where the goal is mainly reducing strain for a defined task or movement. This way, exoskeletons help make users safer and reduce the burden from tasks (Gull et al., 2020; XO, 2021) by making them more ergonomic through the exoskeleton's support.

This strain can be expressed as e.g. the metabolic cost, individual or total muscle activation or loading on the joints. All these aspects should be considered when designing an exoskeleton, including its mechanical design and the control strategy to operate actuators. Unsurprisingly, some of these metrics are used to optimize these designs in order to enable the highest possible force or power augmentation with the same strain or to minimize the strain resulting from a task.

Joint reaction forces (JRFs) are a common option for this evaluation, because they allow estimating the strain on joint tissue resulting from loading on the body. Due to the invasiveness of direct measurements, these forces are usually calculated using musculoskeletal models and simulation tools. The muscle and external forces during a movement are recorded or calculated. The internal joint forces acting between consecutive bones are then calculated in a multibody dynamical simulation. With some researchers optimizing exoskeleton designs according to metrics like the simulated JRFs it is imperative for them to be correct. Otherwise, the design of an exoskeleton might be optimized for an incorrect criterion.

Different parameters have been found to influence simulated JRFs, like the musculoskeletal model used or the muscle force computation algorithm. So far, no investigation has been conducted yet into how the modeling of an exoskeleton that is involved would influence the simulated JRFs. Depending on the exoskeleton type and simulation purpose, typical modeling approaches range from torque actuators to different levels of path actuators. The latter have a force acting along a path that runs through a set of defined path points. Cable-actuated exoskeletons are usually represented by such actuators.

At the Chair of Ergonomics at the Technical University of Munich (LfE), a soft, cable-driven exoskeleton prototype (also called exosuit (Asbeck et al., 2014; Bae, 2013)) supporting the elbow joint was developed and manufactured (Harbauer, Fleischer, Nguyen, Bos, & Bengler, 2020; Nguyen, 2018). Additionally, first

simulations of the elbow JRFs were performed by (Bandmann, 2020) and (Sugiarto, 2020) to optimize and modify its design. In this thesis, the current version of the prototype serves as an example to be modeled in musculoskeletal simulation analysis and thereby answering the open question of its effect on simulated JRFs.

## 1.2. Research Question

Following the motivation, the research question to be answered is as follows:

**How does the modeling strategy in a musculoskeletal simulation of an exosuit affect the resulting computed joint reaction forces?**

There are different ways to implement an exoskeleton in musculoskeletal simulations (see chapters 2.3.2 and 4.3.2). They include different actuator types like torque or path actuators. Path actuators are comprised of two or more path points, which connect to the human musculoskeletal model at different bones. Due to the implementation approaches and the exoskeleton that was selected as an example to model, the research question can be broken down into the following sub problems:

How do the simulated JRFs change for the exosuit prototype by (Harbauer et al., 2020)...

- ... with torque or path actuator type?
- ... when using single-/double-stringed path actuators?
- ... when attaching the path points to different body/bone segments?
- ... with a path of varying complexity along the forearm?
- ... when implementing the cable path going over the shoulder?

## 2. State of the Art

### 2.1. Exoskeletons

Active exoskeletons can be defined as worn assistive systems that mechanically act upon the body (DGUV, 2019). They have also been called “wearable robots” (Pons, 2008) and non-manufacturing robots working in synergy with a human user (Siciliano & Khatib, 2008).

Since first attempts at supporting or enhancing a human user with such a robot like the “Hardiman”, a 680 kg heavy machine developed in 1965 by General Electric (General Electric, 2016; Siciliano & Khatib, 2008) (see figure 1), they have become a lot lighter and more refined. Various designs have been created in the academic world, as well as in the industry.



**Figure 1:** The “Hardiman” exoskeleton by General Electric from 1965 (General Electric, 2016)

#### 2.1.1. Applications and Types

Exoskeletons can be categorized by different principles. For example, their purpose can be defined as either motion assistance or medical rehabilitation (Gull et al., 2020). Their use cases range from weight-carrying exoskeletons for military applications (DARPA, 2021) to stroke-patient rehabilitation through repetitive motion (Islam, Spiewak, Rahman, & Fareh, 2017; Cardona, Solanki, & Cena, 2020).

Other possible categories are exoskeletons with active or passive forces and different actuation principles (Gull et al., 2020). Here are some examples for actuation principles:

- electric actuators (Harbauer et al., 2020)
- pneumatic Artificial Muscles (PAM) (Al-Fahaam, Davis, & Nefti-Meziani, 2018; Sergeyev, Alaraje, Seidel, Carlson, & Breda, 2013)
- shape Memory Alloys (SMA) (Copaci, Cano, Moreno, & Blanco, 2017)

Besides whole-body exoskeletons, there are systems for different regions of the human body, such as lower and upper extremities (Siciliano & Khatib, 2008) or even more specific examples like single-finger support (Agarwal, Kuo, Neptune, & Deshpande, 2013; Agarwal, Fox, Yun, O’Malley, & Deshpande, 2015). The amount of degrees of freedom that are operated varies.

### 2.1.2. Design Challenges

To design and operate an exoskeleton some challenges need to be tackled. The goal “is to replicate kinematics and dynamics of human musculoskeletal structure and support limb movement” (Gull et al., 2020). This can be further broken down into these design challenges (Gull et al., 2020):

- kinematic compatibility
- workspace limitation
- singularity problem of mechanical system
- discomfort and misalignment
- human-robot-interaction
- sensing

Kinematic compatibility refers to the ability of an exoskeleton to adapt to its user’s physiology, such as different body segment lengths (Agarwal et al., 2015; Gull et al., 2020). Regarding the workspace limitation, an ideal exoskeleton should restrict the user’s range of motion as little as possible or necessary for its operation purpose (Gull et al., 2020; Tröster, Schneider, Bauershansl, Rasmussen, & Andersen, 2018). Mechanical singularity arises when e.g. two exoskeleton joints are aligned and can not get out of this singular configuration without infinite torque (Gull et al., 2020). In order not to put stress on misaligned human and robotic joints, mechanisms need to account for joint alignment. This is not always easy due to the human body’s complexity (Lessard et al., 2017). This may be intrinsic to the design or needs to be accounted for with special mechanisms (Tröster et al., 2018; Stienen, Hekman, van der Helm, & van der Kooij, 2009). In order for an exoskeleton to be used properly and comfortably for its user, one needs to be in control of its movements. This is mainly important for motion assistance. Sensing of the current state and position of the exoskeleton and detection of the human intent is required for this. As input from the user different communication methods have been tested, such as monitoring the human-machine interaction force (Yang, Gu, Zhang, & Gui, 2017), integrating joint angle sensors or recording live electromyographic muscle activation data from the user (Moon, Kim, & Hong, 2019).

Even when the intent is known, the exoskeleton force for active actuators can be controlled by implementing different control strategies. These control algorithms have an effect on the metabolic cost and muscle activity of the user for certain movements (Young, Gannon, & Ferris, 2017).

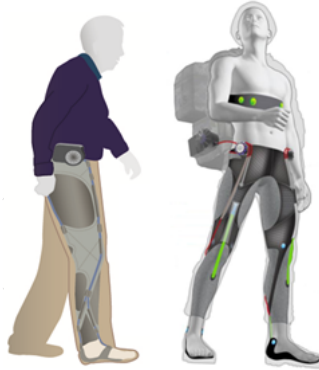
### 2.1.3. Soft, Cable-Driven Exoskeletons

One design approach to overcome some of the described challenges are so-called exosuits. Contrary to a structure of rigid bodies and joints, these are soft and mostly cable-driven exoskeletons (Lessard et al., 2017; EduExo, 2017). They are usually worn like a tight-fitting piece of clothing (Harvard Biodesign Lab, 2021; Asbeck et al., 2014; EduExo, 2017) and are not to be confused with rigid body, cable-driven exoskeletons, as proposed by Mao et al. (Mao & Agrawal, 2010).

Exosuits overcome problems like possible joint misalignment (Harvard Biodesign Lab, 2021; Asbeck et al., 2014; Gull et al., 2020) by avoiding rigid joints and to some extent allow for kinematic compatibility. These “compliant joints and actuators” (Gull et al., 2020) create a safer and more comfortable human-robot-interaction and can often allow for relatively large ranges of motion.

Exoskeletons like these have been developed by different research groups like the Harvard Biodesign

Lab (Harvard Biodesign Lab, 2021; Asbeck et al., 2014) for gait support (see figure 2), as well as Lessard et al. (Lessard et al., 2017) and the Chair of Ergonomics at the Technical University of Munich (LfE) for upper extremity support.



**Figure 2:** Exosuit as used by the Harvard Biodesign Lab (Harvard Biodesign Lab, 2021)

## 2.2. Joint Reaction Forces (JRFs)

### 2.2.1. Definition

“Reaction force refers to Newton’s third law, which states that for any action, there is an equal and opposite reaction. Therefore joint reaction force should represent the force (reaction) equal and opposite to the force (action) that acts on the bones or tissues of which a joint is comprised” (Vigotsky, K. E. Zelik, & Hinrichs, 2019).

JRFs are thus the “forces and moments transferred between consecutive bodies as a result of all loads acting on the model” (OpenSim, 2021e). By this definition they differ from so-called joint net forces, resultant joint forces or sometimes just joint forces (Vigotsky et al., 2019; Troy, 2013; Baltzopoulos & McErlain-Naylor, 2020). These are calculated using simple inverse dynamics, ignoring the internal structure and focusing on the forces a joint will experience overall (Vigotsky et al., 2019; Troy, 2013).

The nomenclature is unfortunately sometimes confused in literature for these two very different forces (Vigotsky et al., 2019). In this thesis, the forces acting from one bone to the adjacent within an actual joint will be called joint reaction force (JRF). They are usually way higher than the joint net forces described previously (Vigotsky et al., 2019; Troy, 2013). JRFs are the forces actually experienced by bones or tissue like cartilage and can be used as a measure of strain in the user’s body.

The internal forces that make up the big difference between joint forces and JRFs are forces from muscles, ligaments and other tissue acting on and over the joint of interest (Vigotsky et al., 2019; Amis, Dowson, & Wright, 1980). Precise knowledge of muscle forces is required in order to have a proper estimation of JRFs (Amis et al., 1980).

As joint reaction forces are the forces experienced by tissue, they are also the ones relevant for damage and injury from e.g. repetitive forces and thus mechanical fatigue (Vigotsky et al., 2019; Gallagher & Schall Jr., 2016; Edwards, 2018). One example that shows the possible implications of high JRFs are baseball pitchers. These pitchers experience high JRFs repetitively whilst throwing and have a high incidence in



upper extremity and elbow injuries (Werner, Fleisig, Dillman, & Andrews, 1993; DiGiovine, Jobe, Pink, & Perry, 1992).

### **2.2.2. Measurements**

As JRFs originate from internal forces inside the body, they need to be either measured invasively or estimated via musculoskeletal simulations (Vigotsky et al., 2019).

One invasive method to determine JRFs are cadaver studies, where force and moment sensors are placed “in situ” within the joint of a human upper limb. This was performed for e.g. the glenohumeral (shoulder) joint by emulating some muscles using 6 hydraulic cylinders. (Apreleva, Parsons, Warner, Fu, & Woo, 2000)

Another invasive approach is to utilize instrumented implants. This was done for the glenohumeral joint as well and allowed for “in vivo” measurements of JRFs (Westerhoff, Graichen, Bender, Rohlmann, & Bergmann, 2009; Bergmann et al., 2011).

There are other methods, where tendon stress and strain is measured “in vivo” to provide data of muscle-tendon-unit forces. From these, resulting JRFs can be computed (Baltzopoulos & McErlain-Naylor, 2020). This presents a hybrid approach between direct measurement and resulting estimation.

All of these methods provide data that is collected directly within the joints and thus gives a good insight on the present forces. However, their invasiveness prevents studies of large scales and takes a lot of preparation, as well as extensive safety measures and preliminary testing in the case of “in vivo” measurements (Bergmann et al., 2011).

Biomechanical, musculoskeletal simulations offer a way of estimating JRFs without this invasiveness. However, without these direct inputs, a simulative workflow for JRF estimation can create and propagate errors and results should thus rather be discussed in terms of trends, rather than absolute values (Blache & Begon, 2018).

### **2.2.3. Musculoskeletal Simulation Tools**

When looking at the body in different scales, musculoskeletal modeling tools for muscle force and JRF computation operate on the organ or bone level to investigate “displacements, stresses, and strains induced by loads acting on the skeleton” (Viceconti, 2012).

For this, rigid multibody dynamic simulations are used, for which different algorithms have emerged and been refined since the 1970s. These usually fall into one of these three categories: (Fregly et al., 2011)

- optimisation methods using minimization of a cost function
- electromyography (EMG)-driven simulations that use muscle activations
- reduction methods which condense the complex muscle structures into fewer groups

All these methods attempt to determine the forces of individual muscles or muscle groups that enable a certain movement. The models contain joint kinematics and muscle paths and attachment points. From these, moment arms of muscles on different joints can be calculated, which allows for computation of the JRFs.

Different programs like OpenSim (Delp et al., 2007), AnyBody Modeling System (AMS) (AnyBody Tech-

nology A/S, 2021; Gull et al., 2020), Biomechanics of Bodies (BoB) (BoB Biomechanics Ltd, 2021) or MB Dyn (Masarati, Morandini, & Mantegazza, 2014; Gull et al., 2020) are available for this task.

## **Muscle Forces Computation**

The calculation of muscle forces is a necessary step for deriving the JRFs (Amis et al., 1980). As OpenSim is used in this thesis, it is selected to describe two methods for this. For recorded or predefined movements, muscle forces can be computed via e.g. Static Optimisation (SO) (Wesseling et al., 2014; Pandy, 2001; OpenSim, 2021d) or Computed Muscle Control (CMC) (Wesseling et al., 2014), two algorithms falling into the optimization method category from above.

Depending on the chosen algorithm, the simulated muscle forces and activations can differ (Roelker et al., 2020). One study found muscle force data to be higher with CMC compared to SO, as it includes “passive muscle forces, muscle activation-contraction dynamics” (Roelker et al., 2020). Another study comparing simulated JRF results to experimental data for the hip joint found SO to be the best match (Wesseling et al., 2014).

## **Static Optimization**

acrshortso is an inverse approach, that calculates muscle forces and/or excitations for a given movement by distributing the load across synergistic actuators (OpenSim, 2021c). It does this by solving an optimization problem for minimal activation levels over the entire musculoskeletal model (OpenSim, 2021d; Thelen & Anderson, 2006). This is done for each time step independently, following the predefined motion trajectory. OpenSim describes it as:

“Static Optimization is a method for estimating muscle activations and muscle forces that satisfy the positions, velocities, accelerations, and external forces [...] of a motion. The technique is called "static" since calculations are performed at each time frame, without integrating the equations of motion between time steps. Because there is no integration, Static Optimization can be very fast and efficient, but it does ignore activation dynamics and tendon compliance.” (OpenSim, 2021o).

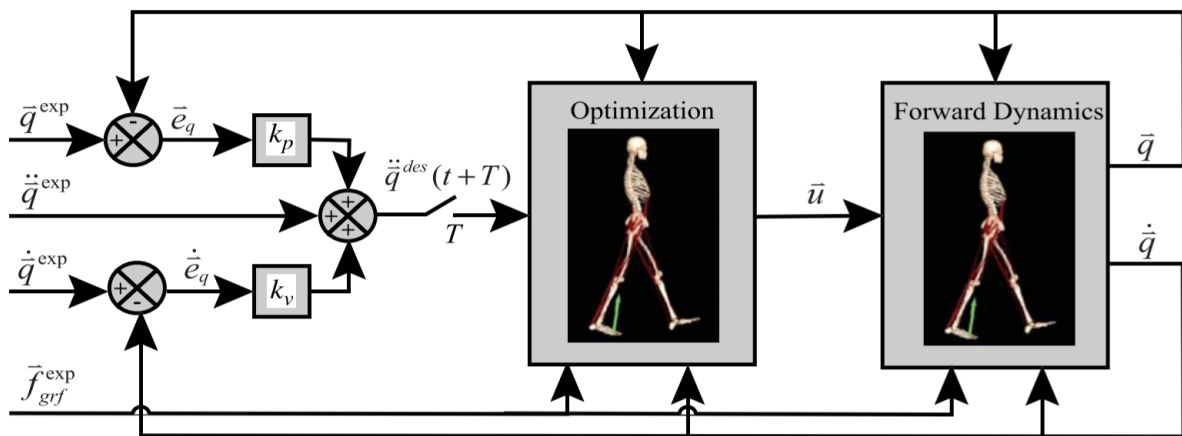
## **Computed Muscle Control**

Just like SO, CMC calculates muscle forces and/or activations for a given motion. It combines SO with forward dynamics, where calculated forces drive a given model (OpenSim, 2021c).

The process is depicted in figure 3. Initial states for joint angles, speeds and muscle states are computed in the first 0.03 s of the movement and should not be used as valid results (OpenSim, 2021c).

First the desired accelerations for generalized coordinates (e.g. joint angles) for the next time step are calculated using a PD (proportional-differential) controller. They drive the coordinates towards the predefined trajectory (Thelen & Anderson, 2006). As a next step the actuator controls are calculated that will achieve

the desired accelerations. The previously described SO algorithm is used for this. As an optimization criterion to be minimized a “weighted sum of squared actuator controls” (OpenSim, 2021c) (slow target) is used in one of two formulations. Either the “sum of desired acceleration errors” or “a set of equality constraints that require the desired accelerations to be achieved within the tolerance” (OpenSim, 2021c) is applied in the criterion. Which of the two is used depends on the desired speed and robustness of the simulation. When it is likely that the algorithm will have trouble following the trajectory - that is for e.g. noisy motion data - reserve actuators can be added for each generalized coordinate. (OpenSim, 2021c) The last step of one CMC algorithm iteration is a forward dynamic simulation for the duration of one time interval. The computed actuator controls are used to compute actuator forces, which drive the dynamic model along the kinematic trajectory. (OpenSim, 2021c) These steps are repeated until the end of the motion data has been reached.



**Figure 3:** Workflow of the CMC algorithm (Thelen & Anderson, 2006).  
 $\mathbf{q}$  are the model's generalized coordinates  
 $\mathbf{u}$  are the model's generalized coordinate speeds  
 $f^{exp}$  are (optional) experimental forces, e.g. ground reaction forces in a walking trial

### Joint Reaction Analysis

As by definition of JRFs in chapter 2.2.1, a simulation tool for these will “calculate the joint forces and moments transferred between consecutive bodies as a result of all loads acting on the model” (OpenSim, 2021e). This is done via basic mechanics in an analysis of each time step. Just as in a free-body-diagram, the forces are calculated as if the joint was replaced by the reaction forces. Or as OpenSim puts it:

“[Joint Reaction Analysis] reports the joint reaction loads from a model. For a given joint, the reaction load is calculated as the forces and moments required to constrain the body motions to satisfy the joint as if the joint did not exist. The reaction load acts at the joint center (mobilizer frame) of both the parent and child bodies [...]” (OpenSim, 2021a).

An algorithm for this is implemented in OpenSim as the Joint Reaction Analysis (JRA) tool. It uses muscle, actuator and external forces and moments acting on the model, as well as the motion file to calculate JRFs for each time step of the simulation. Each musculoskeletal model has a ground body. From this

body outwards, each subsequently attached body is the child of the one attached more closely to this ground. For example, with the thorax as a ground body, the forearm bones are children of the upper arm's bone. The reaction forces can be reported as acting on the child or parent body in either reference or the ground frame (OpenSim, 2021e).

#### **2.2.4. Dependency on Musculoskeletal Model Setup**

JRFs are closely linked to muscle forces (Amis et al., 1980; DeMers, 2011). These strongly depend on the models used in the musculoskeletal simulations (Baltzopoulos & McErlain-Naylor, 2020). Models may vary in their amount, setup and complexity of joints, bones and muscles, as well as the representation of passive forces like ligaments.

Early research of 2D models already showed this dependency. A study on joint reaction forces in the jaw joint more specifically found for this joint that “[...] the magnitude of the joint reaction force is less sensitive to error than the calculation of joint reaction force direction” (Throckmorton, 1985). JRFs were especially sensitive to varying muscle moment arms (Throckmorton & Throckmorton, 1985). One study compared simulation results of different upper body musculoskeletal models with real world muscle force data. They found that more complex models containing more muscles produced more accurate results (Quental, Folgado, Ambrósio, & Monteiro, 2013). Another study found that the muscle forces and activations of a gait simulation were sensitive to the choice of the underlying model as well (Roelker et al., 2020).

To the best knowledge of the author, studies investigating the dependency of muscle forces or joint reaction forces on the representation of an exoskeleton within such a musculoskeletal model are not present in current literature.

### **2.3. Musculoskeletal Modeling of Exoskeletons**

As explained in chapter 2.1.1 there is a large variety of exoskeletons and applications in the academic and industrial world. Thus the purposes, tools and modeling approaches in simulations can also vary greatly. This chapter aims at giving insights into why musculoskeletal simulations are performed and some implementations thereof.

#### **2.3.1. Goal and Purpose**

The way an exoskeleton is modeled and simulated depends on the goals and purposes to be achieved. The impact of exoskeletons on the human body is not entirely understood in terms of safety and ergonomics and is difficult to analyze (Gull et al., 2020; Bae, 2013). A virtual model and analysis assists in studying the effect of the exoskeleton on the body, thus allowing for evaluation and possibly optimization of design and control (Gull et al., 2020; Agarwal et al., 2013). The physical human-machine-interaction and its implications for the user can be analyzed via musculoskeletal modeling.

One goal of musculoskeletal simulation can be to determine the amount of support an exoskeleton provides (Gull et al., 2020). This can be done via analyzing e.g. the metabolic cost, individual and overall muscle activations or joint reaction forces (Gull et al., 2020; Zhou, Li, & Bai, 2017; Bae, 2013; Tröster et

al., 2018). Other goals can be to investigate the forces acting directly onto the user and to estimate the level of comfort (Gull et al., 2020; Shourijeh et al., 2017; Shao, Tang, & Yi, 2014).

### 2.3.2. Modeling Examples

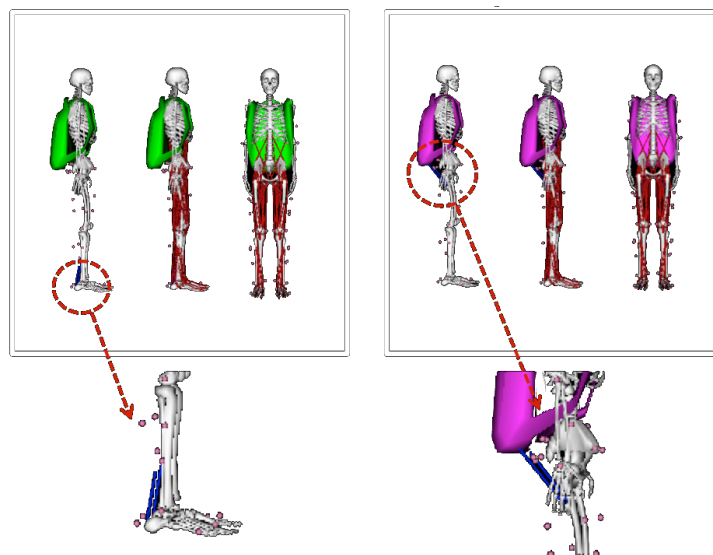
#### Gait Support Exosuit by (Bae, 2013)

(Bae, 2013) investigated a gait supporting exosuit similar to the one by Harvard Biodesign Lab using OpenSim. Support for hip extension and ankle plantarflexion is provided by a path actuator on each side of the user.

Path actuators are a subclass of actuators within OpenSim, that applies a “controllable tension along a geometry path” (OpenSim, 2021). The geometry path is defined by two (in this case) or more points attached to different bones or body segments.

In these simulations, only the exposed part of the cables are implemented, ignoring the parts within the textile (see figure 4). In one configuration, plantarflexion is supported via a connection from heel to tibia. In a second, hip extension is implemented by connecting the femur to the backpack that is part of the exoskeleton. In a third configuration, both joint supports are combined as a biarticular actuator reaching from the backpack over the knee to the heel.

With this model, the effect of the exoskeleton on the metabolic cost was investigated, where the configuration of with pure ankle support was found to be most effective out of the three. It was mentioned, that more realistic representations of the exosuit would be required (Bae, 2013). This could be via more precise paths, more accurate attachments to the body or a combination of active and passive support forces.



**Figure 4:** Exosuit implemented as simple path actuators by (Bae, 2013). The path actuators are marked in blue and support the ankle (left) and the hip (right).

### **Upper Extremity “Stuttgart Exo-Jacket” Modeled by (Tröster et al., 2018)**

(Tröster et al., 2018) performed musculoskeletal simulations for the “Stuttgart Exo-Jacket”, an assistive rigid upper extremity exoskeleton with active and passive support elements. A 3D CAD model was imported into AMS and connected to the human model via a back plate fixed to the hip. Two spring elements additionally connected the back plate to the thorax, which allowed for relative movement. The springs were passive gas spring elements with a constant unidirectional force. Additionally, active elements were implemented as motors supporting elbow flexion and shoulder elevation. (Tröster et al., 2018)

This model was used to investigate muscle activation and JRFs via inverse dynamics simulation as a function of the “gravity compensation factor” (Tröster et al., 2018), a form of control strategy. A dependency of the activations and JRFs on this control is found with different optimal support levels depending on which of the two metrics is being minimized (Tröster et al., 2018). This means that a control strategy optimized for minimal muscle activations does not equal minimal JRFs. The exoskeleton model itself is quite detailed and is not mentioned as a limitation to the quality of the study’s results.

### **Finger Exoskeleton Modeled by (Agarwal et al., 2013)**

(Agarwal et al., 2013) presented a virtual prototyping framework for iterative improvement of an exoskeleton. As an evaluation step they used musculoskeletal simulations to investigate a rigid, cable-driven index finger exoskeleton for rehabilitation including passive spring elements.

The virtual model includes a 3D CAD model of all exoskeleton parts, which is connected segment-wise to a human finger model in OpenSim. Four mechanical springs are implemented as spring elements. Three cables are modeled as muscle-tendon-units connecting the exoskeleton parts. CMC was then used to calculate muscle forces and JRFs in order to optimize the exoskeleton design. (Agarwal et al., 2013)

The framework proved to be a powerful tool in experimenting with and analyzing an exoskeleton prototype. For the finger exoskeleton investigated, higher stiffness in the passive spring elements was found to increase the simulated JRFs (Agarwal et al., 2013). For the presented exoskeleton model, no limitations on the results stemming from a lack of detail or quality in modeling is mentioned.

### **Elbow Exoskeleton Modeled by (Gonzalez-Mendoza et al., 2019)**

(Gonzalez-Mendoza et al., 2019) evaluated a proportional-derivative controller for an exoskeleton using the CMC algorithm of OpenSim in combination with three different human musculoskeletal upper limb models. The elbow exoskeleton was modeled in OpenSim as a pure torque actuator. As the description of the exoskeleton prototype itself lacked clarity, it can not be derived, how accurate this representation is. However, as force was applied via a cable or rope from an external machine, this modeling approach seems to be a simplification.

The setup is deemed appropriate for controller design of movements up to 300 °/s (Gonzalez-Mendoza et al., 2019) and finally it is mentioned that more simplified musculoskeletal models will be tested in the future to speed up the simulations for real-time applications.

### **Elbow Exoskeleton Modeled by (de Kruif, Schmidhauser, Stadler, & O'Sullivan, 2017)**

(de Kruif et al., 2017) investigated the control of a rigid elbow exoskeleton including a human response model via simulated muscle activations. The calculations were then used to predict the response to external perturbations.

The exoskeleton was modeled in OpenSim to a three-muscle upper extremity model via two external bodies each connected to the humerus and the forearm and weighting 0.75 kg. A torque was applied between these two. Together with the model from (Gonzalez-Mendoza et al., 2019) this presents one of the simplest exoskeleton models encountered in the literature.

The model and simulation framework allowed for a prediction of the response to perturbations including the use of the exoskeleton. It is indicated, that a model containing more individual muscles might yield more realistic results.

### **Modeling of Elbow Exosuit LfE Prototype from (Harbauer et al., 2020)**

In previous work at the LfE a cable-driven elbow supporting exosuit was developed, manufactured, simulated and improved by (Harbauer et al., 2020; Nguyen, 2018; Bandmann, 2020) and (Sugiarto, 2020). The prototype comprises a textile jacket of varying elasticity with a cable guided through low-friction tubing. The cable is actuated via electric motors sitting at the back of the user. Both ends insert to one actuator each with the cable running in a closed loop to distribute loads between both sides (Harbauer et al., 2020). The path runs from one motor over the shoulder, where it exits the tubing and heads towards the medial side of a brace attached at the jacket's forearm segment. There, it follows a tubing path on the brace towards the underside of the forearm up to the wrist. It makes a sharp turn and runs back on the under- and then the lateral side of the forearm. Here it is lead out of the tubing and away from the brace back towards the shoulder, where it reenters the last tubing segment, which guides it over the shoulder to the user's back until the other motor. To get a better picture of the setup, a picture is provided in figure 5.



**Figure 5:** Elbow exosuit prototype developed at the LfE with two actuated and linked cables running in parallel.

Musculoskeletal simulations were conducted for this exoskeleton by (Bandmann, 2020) with the goal of finding an optimal attachment position in the forearm's coordinate system for the cables. In OpenSim, the two cable sections outside the tubing were modeled as one single path actuator attached to the humerus and forearm. This model was used to calculate JRFs. The musculoskeletal model as well as the resulting JRFs levels and patterns are described in chapter 2.5.

The study found attachment points the furthest apart from the elbow joint center vertically and horizontally to be a theoretical optimum. For practical reasons, this optimum was then shifted towards the joint center again. Limitations to the setup and modeling strategy were documented, including the reduced amount of unlocked degrees of freedom in the model, specifically the radio-ulnar articulation. Additionally, a lack of representation of the gripping force was mentioned as a possible source of inaccuracy in the computed JRF.

In another JRF simulation for the same exosuit by (Sugiarto, 2020), parts of the jacket were modeled as CAD parts including their approximate weight and imported into OpenSim. The active support was represented by a torque actuator, whose control was designed to counter the torque placed on the elbow flexion coordinate by the weight. As with (Bandmann, 2020), the simulated JRF values and patterns during the lifting movement are described in chapter 2.5. It is shown that the model manages to relieve the elbow in terms of JRFs, better so for lighter weights compared to heavier ones.

(Sugiarto, 2020) points out that the virtual exosuit model could still use some improvement in the form of passive damper and spring elements. The idea to represent the prototype's cables as paths is not mentioned.

## Overview

Table 1 provides an overview of the different exoskeletons and modeling strategies applied for creating musculoskeletal models. Overall the models had a broad field of representations of different exoskeletons ranging from simple torque actuators - like (Gonzalez-Mendoza et al., 2019) or (de Kruif et al., 2017) - over path actuators applying a force between two bodies - like (Bae, 2013) or (Bandmann, 2020) - to elaborate models including passive and active elements, as well as additional bodies - like (Agarwal et al., 2013). Some studies included possible improvements to the virtual models. They concerned either the human musculoskeletal or the exoskeleton representation. For the muscle models, more detail in the muscles, motion and degrees of freedom were suggested. In the exoskeleton representation more realistic models were sometimes suggested.

Generally all cited publications performed musculoskeletal simulations exclusively using one single exoskeleton model. Studies comparing different implementations of the same prototype or product were not found.



| Researcher(s)           | support of        | exoskeleton type                       | modeling strategy                                     |
|-------------------------|-------------------|--|---|
| Bae (2013)              | leg / gait        | exosuit                                | path actuators (consisting of 2 path points each)     |
| Troester et al. (2018)  | upper extremities | rigid                                  | passive (gas) springs, torque actuators               |
| Agarwal et al. (2013)   | index finger      | rigid, cable-driven, including springs | spring elements, muscle-tendon units (several points) |
| Gonzalez-Mendoza (2019) | upper extremity   | unclear                                | torque actuator                                       |
| de Kruif et al. (2017)  | elbow             | virtual                                | torque actuator                                       |
| Bandmann (2020)         | elbow             | exosuit                                | path actuator (consisting of 2 path points)           |
| Sugiarto (2020)         | elbow             | exosuit                                | torque actuator                                       |

**Table 1:** Overview of exoskeleton modeling strategies in musculoskeletal simulations by different researchers.

## 2.4. Human Elbow Physiology

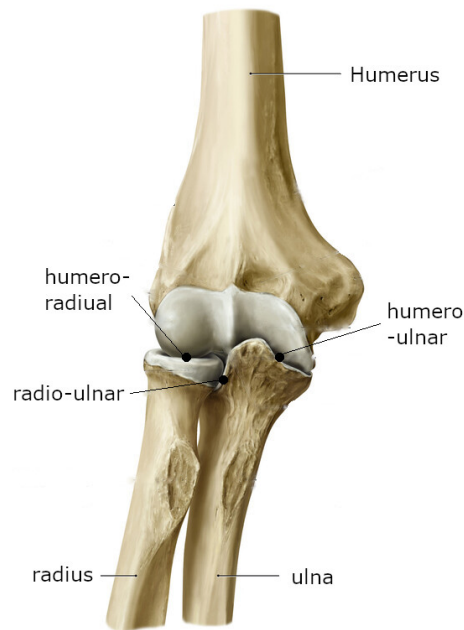
The very complex human elbow joint (Rüther & Simmen, 2013, p. 7) is only explained in the amount of detail necessary for this thesis.

In the elbow, three bones are at play: the humerus in the upper arm, radius and ulna in the forearm (see figure 6). It is capable of two types of movement. These are operated by three articulations, one between each of the three bones (humero-radial, humero-ulnar and radio-ulnar). (Celli, Celli, & Morrey, 2010, p. 1) (Rüther & Simmen, 2013, p. 7)

Firstly, the elbow enables pronation-supination (Philipp Zimmer, 2020, p. 74), which allows for turning the hand without moving the upper arm. This happens by turning the radius, which articulates in contact with the humerus (humero-radial) and ulna (radio-ulnar) around the forearm axis (Rüther & Simmen, 2013, p. 16).

Elbow flexion-extension is the second mode of movement (Celli et al., 2010, p. 8), where the entire forearm can be rotated from a fully extended arm to fully flexed. The range of motion from approximately 0 to 140° (Celli et al., 2010, p. 8) or 150° (Rüther & Simmen, 2013, p. 16) is realized by the humero-radial and humero-ulnar articulations. The main joint surface involved in the forearm and providing most of the stability is the one on the ulna (Rüther & Simmen, 2013, p. 9).

The joint is stabilized actively by different muscles and passively by ligaments (Celli et al., 2010, p. 1). Many of the muscles act on both modes of motion, e.g. the brachioradialis as a extensor-supinator muscle (Celli et al., 2010, p. 6).



**Figure 6:** Bones and articulations of the human elbow (Schuenke, Schulte, & Schumacher, 2011)

## 2.5. Simulated Elbow JRFs

There were no measured JRFs found for the elbow joint in the literature. However, using the tools presented in chapter 2.2.3, different researchers investigated the JRFs within the elbow joint. This was done for a variety of movements, which are not always easy to compare to one another. This chapter gives an overview of different elbow JRF simulation results.

### 2.5.1. Magnitude

A study of elbow JRFs over flexion angles whilst pulling against a static force acting at the hand used an upper extremity musculoskeletal model containing 13 muscles (Amis et al., 1980). To calculate the muscle forces in each step, a hypothesis of equal fiber stress was applied for co-operative muscles (Amis et al., 1980). The maximum JRFs for an external force of 50 N were estimated to be about 800 N for humero-coronoid forces (a part of the ulna), 725 N for radio-humeral forces and 560 N for resultant humero-ulnar forces (Amis et al., 1980). Adding up the humero-ulnar and humero-ulnar forces this gives an overall elbow JRF of 1 285 N.

In a second trial applying maximum antagonistic and isometric (= static) co-contraction of the elbow joint, humero-ulnar JRFs of around 3 000 to 3 200 N were computed (Amis et al., 1980; Chadwick & Nicol, 2000; Raikova, 2009). All JRFs were acting as compressive forces for every flexion angle (Amis et al., 1980). It was also mentioned that forces acting perpendicular to the plane of elbow flexion-extension were small compared to the ones in this plane.

One early study investigated elbow JRFs during everyday activities like eating, reaching, pulling on an object and dip-like motion on parallel bars. The JRFs they calculated reached up to 2 450 N for the

humero-ulnar joint and 1 500 N in the humero-radial joint (Nicol, 1977; Chadwick & Nicol, 2000), taking into account the two articulations participating in elbow flexion. Summed up, this gives a total force of up to 3 950 N on the humerus.

For “occupational pick and place activities” (Chadwick & Nicol, 2000) that involved gripping an object, a study investigated the elbow JRFs using an upper extremity musculoskeletal model with 15 muscles and an optimization algorithm. The optimization minimized the overall minimum muscle stress and the sum of muscle and ligament forces (unlike SO, which minimizes total muscle activations). The computed forces reached maximums of 1 600 N for the humero-ulnar and 800 N for the radio-ulnar articulation.

Baseball pitchers encounter net compressive joint forces of up to 800 N (Werner et al., 1993). Another study simulated the subsequent internal JRFs of 4 to 7 times the body weight using a 12-segment whole body musculoskeletal model (Buffi, Werner, Kepple, & Murray, 2014; Vigotsky et al., 2019). For a 50<sup>th</sup> percentile male of around 74 kg ( AirSlate Legal Forms, Inc., 2021; FRA, 2008) this would translate to roughly 3 000 to 5 000 N. The JRFs were determined using the CMC and JRA tools of OpenSim.

The peak elbow JRFs found for a 50<sup>th</sup> percentile male without the use of an exoskeleton ranged from 500 N (Bandmann, 2020) up to about 4 (Nicol, 1977) and 5 kN (Buffi et al., 2014) in motions involving higher muscle forces. Table 2 gives an overview of the peak elbow JRFs encountered in the different studies with and without exoskeleton support.

### **Studies Including an Exoskeleton Model**

A master’s thesis from TUM simulated elbow JRFs using a right side upper extremity model for a slow and pure flexion-extension movement carrying a 5 kg weight in its hand in OpenSim (Bandmann, 2020). The model also included the path actuator model of the LfE exosuit presented in chapter 2.3.2. The musculoskeletal model consisted of six muscles and two degrees of freedom, one of them being the elbow flexion. SO was used to calculate the muscle forces during the motion and JRA was used to compute the JRFs, both in OpenSim. The control for the exoskeleton actuator was entirely managed by the SO algorithm by providing the actuator with a very high theoretical maximum force of 500 000 N. The optimization algorithm thus preferred activating this actuator instead of the model’s muscles. Within the study, the exoskeleton attachment points were modified to find an optimal design. The models lead to great variations in the simulated JRF values. In a trial with neither the weight, nor the exoskeleton included in the model, the JRFs peaked at 500 N. Depending on the exoskeleton setup, the highest JRF values ranged from 65 to 120 000 N. The excessive forces present in some of the models were then excluded as not representative. For the implementations with optimized design according to the simulation results, the peak values ranged from 65 to 85 N.

One explanation for these relatively low JRFs in comparison to studies presented before could be the heavily simplified musculoskeletal “Arm26” model (see chapter 4.1.1).

A different study of the same LfE elbow exosuit by (Sugiarto, 2020) investigated the JRFs using a more complex musculoskeletal model called “Upper Extremity Dynamic Model” and a torque actuator representing the exosuit. A similar motion lifting different weights was simulated with the CMC to determine the muscle forces and JRA to then derive the internal joint reactions. The actuator’s support was controlled by prescribing predefined torque values. These were calculated to match the torque on the elbow arising from lifting the weight.

Due to unreliable values at the start of the motion, the results were restricted to the flexion values between  $10^\circ$  and  $100^\circ$  elbow flexion. Within this range, the highest JRFs observed when lifting the 5 kg weight reached about 2 530 N without and around 2 240 N including the exosuit (Sugiarto, 2020). The mean total JRF with this weight was 722 N with and 991 N without exoskeleton support. The control strategy used by (Sugiarto, 2020) was a compensation of the torque imposed on the elbow by the weight and the estimated weight of the exosuit itself of about 60 g.

The peak elbow JRFs found for a 50<sup>th</sup> percentile male using an exoskeleton in a pure flexion motion ranged from around 65 N for one configuration in (Bandmann, 2020) up to 2 240 N in (Sugiarto, 2020) and 120 000 N for one configuration in (Bandmann, 2020). The wide range in computed JRF values shows that reliable calculation of the JRFs is not trivial and may strongly depend on factors like the exoskeleton control strategy. Table 2 gives an overview of the peak elbow JRFs encountered in the different studies with and without exoskeleton support.

### **2.5.2. Pattern for Elbow Flexion**

The kinematics of articulations and muscles in the elbow change over different flexion angles. Some studies investigating simulated JRFs for this joint gave insights about the resulting correlation between flexion angle and internal joint forces.

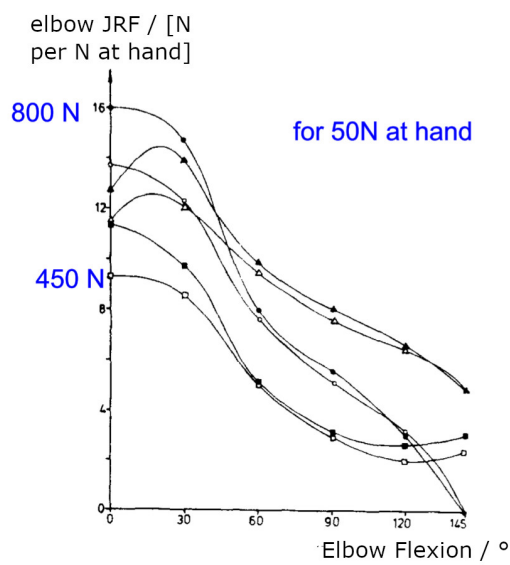
Generally, the elbow JRFs tended to decrease over a higher flexion angle (Morrey, An, & Stormont, 1988) for a constant required joint torque (see figure 7). This was also shown in the plots included in (Amis et al., 1980), where the lowest JRFs coincided with higher flexion angles during the motions for a constant force at the hand. Furthermore, the greatest force transmission was observed between  $0$  and  $30^\circ$  of flexion and were higher in a pronated compared to a supinated position (Morrey et al., 1988).

(Bandmann, 2020) investigated a flexion-extension movement with a weight in the hand, resulting in a changing torque requirement over the elbow flexion angle. Maximum JRF was not observed at the point of greatest required torque however, but at about  $40^\circ$  flexion (Bandmann, 2020), indicating again that the JRFs don’t simply scale with the required external force or torque. For this study, the lowest JRFs were observed at the lowest flexion angle, where the required torque was still low.

The JRFs simulated in the study by (Sugiarto, 2020) showed a relatively steady decrease in force over the elbow flexion angle during the lifting motion (Sugiarto, 2020, p. 86). It is characterized by a monotonous decline of the Y-direction vector component and a lower, concave shaped curve for the X-direction with its maximum around  $60^\circ$  flexion. The Z-component is way smaller than the other two, as it is perpendicular to plane the elbow flexion occurs in.

| Researcher(s)           | Motion   | Method / Model                            | peak elbow JRFs [N] | articulation(s)                 |
|-------------------------|--|---|---------------------|---------------------------------|
| Nicol et al. (1977)     | everyday activities  | -   | 3950                | humero-ulnar +<br>humero-radial |
| Amis et al. (1980)      | pulling against a static force                             | upper extremity model, 13 muscles         | 1285                | humero-ulnar +<br>humero-radial |
| Chadwick & Nicol (2000) | occupational pick and place activities                     | upper extremity model, 15 muscles         | 1600                | humero-ulnar                    |
| Buffi et al. (2014)     | baseball pitch   | 12-segment whole-body model               | 5000                | entire elbow                    |
| Bandmann (2020)         | pure flexion-extension (without exoskeleton)               | upper extremity model, 6 muscles          | 500                 | entire elbow                    |
| Bandmann (2020)         | pure flexion-extension (with different exoskeleton models) | upper extremity model, 6 muscles          | 65 - 120 000        | entire elbow                    |
| Sugiarto (2020)         | pure flexion-extension (without exoskeleton)               | upper extremity dynamic model, 50 muscles | 2530                | entire elbow                    |
| Sugiarto (2020)         | pure flexion-extension (with exoskeleton)                  | upper extremity dynamic model, 50 muscles | 2240                | entire elbow                    |

**Table 2:** Overview of studies simulating elbow JRFs for different models and motions, some including exoskeleton support.



**Figure 7:** Different elbow JRFs over flexion angles according to (Amis et al., 1980) with values for 50 N at the hand: Variation of joint forces during flexion, with and without triceps antagonism, per unit of force at the hand. Key: ●, ○: Humero-coronoid antagonism ; ▲, △ : Humero-radial force, with and without forces: ■, □ : Resultant humero-ulnar forces (Amis et al., 1980)

## 3. Methodology

### 3.1. Different Implementations of the Same Exoskeleton and Setup

As stated in chapter 1, the this thesis investigates the effect of different modeling approaches of the same exoskeleton on the JRFs calculated in musculoskeletal simulations.

In order to be able to make a proper comparison, the variability of other factors and setup configurations should be kept as low as possible.

Most importantly, all simulation setups will model the same physical object. The elbow supporting exosuit LfE prototype presented in chapter 2.3.2 is selected as the exoskeleton, that all models will represent.

For the setup to change as little as possible, all other factors will remain the same for the simulation. These include, but are not limited to:

- simulation program and set of tools/algorithms
- musculoskeletal model (including degrees of freedom and weight)
- motion
- equal support from the exoskeletons  
(as this is a major factor for JRFs → see chapter 2.5)

As a simulation program, OpenSim is chosen over competitors like AMS mentioned in 2.2.3, in order to be able to compare it to previous work at LfE. In addition, its open-source availability provides easy access, freely available models from contributors and support through an online forum. The tools comprise of an algorithm to calculate the muscle forces required to perform the given motion with each model, as well as a tool to derive joint reaction forces from these and other forces acting on the body. To calculate the muscle forces, the CMC algorithm is chosen over SO, as the utilized MobL model is evaluated with it. The settings used for these tools will be portrayed in detail in chapter 4.1.

The motion is provided via a predefined file consisting of pure elbow flexion. Every model with its controlled force is subjected to the exact same, artificially created movement, eliminating any inconsistencies between simulations.

Finally, to ensure comparability between the models, they are to provide the same amount of support to the user. The exoskeleton's main function is the support of the elbow flexion moment. Equal support is herein defined as providing the ever same torque to the elbow joint as a result of the respective exoskeleton model for each step of the motion. This is guaranteed by predefining a torque curve that is present during the motion. It is translated into a torque or force that the actuator representing the exoskeleton provides in the simulation.

## 3.2. Comparison of Models via JRFs

The strain put on the user can be expressed in various metrics like metabolic cost, muscle activation or JRFs.

Investigation of metabolic cost allows for a generalized metric displaying the amount of support the user gets from an exoskeleton. However, muscles have different forces and moment arms acting on the various joints. This type of analysis does not give insights on how specific muscles and joint forces may be subjected to strain resulting from a motion. Analysis of muscle activation patterns allows for a closer look at how each muscle is used during a motion.

As described in chapter 2.2.1, JRA displays the forces between adjacent bones as a result of muscle-tendon-units pulling on bones via insertion points and other forces like those of an exoskeleton. This type of analysis does not look at the contributions of individual muscles and is not a scalar measure of overall relief / strain. However, JRFs are an indicator for strain on a user's joints and thus are a meaningful way of showing the support / relief that the use of an exoskeleton provides. Previous work at the LfE also used JRFs to investigate the effect of the LfE exosuit prototype presented in chapter 2.3.2 and so a comparison to other results is possible. Thus, this thesis uses JRFs as a metric for strain from a defined motion with differently modeled implementations of the same exoskeleton.

With this selection made it should be kept in mind that the support evaluation differs when using muscle activations vs. JRFs as evaluation and optimization criterion of an exoskeleton design (Tröster et al., 2018).

## 3.3. Simulative Workflow

With the goal of this thesis being to compare differently modeled exoskeletons in terms of the resulting simulated JRFs, a fixed simulation setup and workflow is to be established to run every model through.

### **Setup for All Models**

Firstly, some files are set up, which are used for the simulation of all models in OpenSim. They are:

- a motion file,
- torque curve(s),
- CMC setup files,
- JRA setup file
- and the Muscle Analysis (MA) setup file.

They ensure that each simulation is using the same movement with the same amount of elbow joint support and with the same algorithm parameters to calculate muscle forces and JRFs during the motion. The CMC, JRA and MA setup files also define the placement of input and output files for the corresponding OpenSim tool.



## Process for Each Individual Model

To make sure every model provides the same amount of support to the user in the form of torque to the elbow joint, the actuator representing the exoskeleton needs to be controlled accordingly. This happens via a prescribed torque or force specified in a file. However, this force needs to be computed first.

For a torque actuator this is relatively simple, as it just uses the given torque provided by the torque curve. For actuators that act with a moment arm a force needs to be calculated via the following formula:

$$F_{actuator} = \frac{\tau_{given}}{r_{actuator}}$$

The required actuator force  $F$  results from the division of the given torque  $\tau$  by the actuator's moment arm  $r$ . This calculation is not trivial, as the moment arms of the actuator may vary greatly throughout the motion. To compensate for this, the moment arm is calculated dynamically over the course of the movement. OpenSim has a tool to do this for muscles, called Muscle Analysis (MA). However, this tool only calculates the moment arms of muscles, not actuators. Thus, a copy of the model needs to be created in an intermediate step, representing the actuator's geometry as a muscle. This copy is only used for MA and not in any of the following steps. Using the moment arms from the MA, the prescribed actuator force is calculated for every simulation time step and saved as a file. It is later used in the muscle force calculations.

As a next step, the muscle forces present during the motion are computed with the CMC OpenSim tool as explained in chapter 2.2.3. The tool takes the model file, two additional setup files, the previously created prescribed actuator force file as well as the motion file as inputs. The results files created by the tool include muscle forces throughout the movement, which are the main resource used in the next step.

CMC calculates the muscle forces that are required to move track the trajectory defined by the motion. It takes into account the prescribed force/torque already provided by the actuator representing the exoskeleton and optimizes the muscle activations for a cost index as described in chapter 2.2.3.

The muscle forces are used to calculate the JRFs. This is done in OpenSim's JRA tool. It takes the forces, as well as the motion file as inputs and computes the desired JRFs as vector components in X-, Y- and Z-direction of the desired reference frame. These are stored in a table-like format.

In order to make the results interpretable and more visually appealing, they are then read and displayed via a MATLAB script.

## Batch Processing

In order to be able to run the simulations for several models with the same setup, the processes are automated using MATLAB. There are two main stages for a batch of models to be run.

The first is the generation of the prescribed actuator torques or forces as files. The files are then placed in the right folder. The second one is the automated process of running CMC and JRA for the same batch. It includes the creation of plots representing the JRFs over the elbow flexion angle. For the last part, the motion file is used again to determine the flexion angle for the different time steps. JRFs are not simply

plotted over time to render the results easier to interpret without knowledge of the precise motion. It also facilitates the comparison to other work that does not use the same motion.

### 3.4. Approach and Generation of Different Models

In chapter 2.3.2 different modeling strategies for exoskeletons in musculoskeletal simulations are presented. Many of these used torque actuators to represent different types of structures and designs. Some more elaborate models include passive elements like springs or gas springs or used path actuators for cable-actuated exosuits and -skeletons. The path actuators represent the cables in varying detail.

These approaches are utilized to create representations of the exosuit from LfE presented in chapter 2.3.2 as a torque and a single-stringed path actuator with a start and an end point. From this point on, the models are expanded towards a full representation of the cable path within the entire exoskeleton by different models of the two cables running in parallel and along the forearm.

In the creation of each path actuator, the path point positions are defined within the frame of a bone body of the underlying musculoskeletal model, to which they apply the force. This attachment is one of the factors to be investigated in terms of its effect on the simulated JRFs. Models with the equivalent path actuator geometry within the model, but points seated in different bone reference frames are thus created. A complete list and description of the models used in this thesis is detailed in chapter 4.3.2.

## 4. Implementation

### 4.1. OpenSim Tools and Setup

#### 4.1.1. Musculoskeletal Model

##### OpenSim Upper Extremity Models and MoBL-ARMS

As a first step in setting up an OpenSim environment, a musculoskeletal model is required. The section of the body that is represented should be chosen according to the purpose of the simulation. As this thesis investigates the effect on the elbow JRFs, a model containing this joint, as well as the adjacent ones is to be chosen. A whole body simulation would most likely not give any better insights, but only increase computational effort. Also, the full body models are often derived from other body part models or contain them.

OpenSim provides an overview of publically available models on its website. These include its very own “core [...] example models” (OpenSim, 2021h), as well as user-contributed ones. Table 3 gives an overview of models from this list that represent the upper body, meaning mostly from the thorax or shoulder up to the hand. They vary greatly in their complexity and applicability.

| Model                          | DOFs per Arm | # Muscles per Arm | Description   |
|--------------------------------|--------------|-------------------|---|
| Arm26                          | 2            | 6                 | very simple, for educational & demonstrational purposes   |
| Delft Shoulder and Elbow Model | 5            | 31                | used for kinematic & dynamic analysis, deprecated model for OpenSim 4.0 and newer   |
| Stanford VA Upper Limb Model   | 15           | 50                | derived from experimental data, used for kinematic analysis, includes thumb & index finger                                    |
| MoBL-ARMS                      | 7            | 50                | 50 <sup>th</sup> male percentile, research-grade kinematics and dynamic simulation, derived from Stanford VA Upper Limb Model |

**Table 3:** Upper extremity musculoskeletal models in OpenSim

As described in chapter 2.2.4, the joint reaction forces depend on the model they are simulated with. The more detailed and accurate the musculoskeletal model, the more accurate are the computed JRFs. The relatively simple Arm26 model comprises 6 muscles for only 2 degrees of freedom (DOF) in one arm. It is mainly meant for educational or demonstrational purposes (OpenSim, 2021h). It was used in previous work at the LfE to optimize the design of an exosuit prototype (Bandmann, 2020).

The Delft Shoulder and Elbow Model is made up of 31 muscles, as stated in the ‘readme’ in the download

files. However, it is fairly old and not compatible with the current OpenSim versions 4.0 and later anymore. The Stanford VA Upper Limb Model was created in 2005 (Holzbaur, Murray, & Delp, 2005) as a kinematic model. As inertias are not present for the body segments, it is not meant for dynamic analysis (OpenSim, 2021h), which CMC is. Furthermore, it does not include joint limiting forces representing ligaments and other joint tissue (Sugiarto, 2020).

With the Stanford VA Upper Limb Model as a predecessor, the Upper Extremity Dynamic Model or MoBL-ARMS Dynamic Musculoskeletal Model (MobL) was developed incrementally by several researchers (OpenSim, 2021h; Saul et al., 2014; McFarland, McCain, Poppo, & Saul, 2019). The readme file contained in the download urges the user to describe the most recently updated model as follows: “The computer model was modified from Saul et al. (2015) as described by McFarland et al. (2019) to include an updated range of motion at the shoulder, ligaments models representing the glenohumeral and coracohumeral ligaments, and updated muscle model (Millard et al., 2013) with force-length and tendon curves matching the original model’s respective curves.”<sup>1</sup> It features inertial properties for all included body segments, which makes it suitable for dynamic analysis. It is deemed suitable for “research-grade kinematics and dynamic simulation of shoulder and arm movement” (OpenSim, 2021h). The size and mass properties correspond to those of a 50<sup>th</sup> percentile adult male (Saul et al., 2014). MobL was compared to in-vivo gleno-humeral (shoulder) JRFs using an instrumented shoulder endoprosthesis: “results indicated a reasonable compatibility between model and measured data” (Nikooyan et al., 2010). It is mentioned though, that the model will have to be adjusted for individual patients (Nikooyan et al., 2010).

The MobL as a dynamic and detailed model that is compatible with OpenSim 4.2 and that has in part been validated with in-vivo data, it is selected as musculoskeletal model for the simulations in this thesis.

## Degrees of Freedom and Coordinates

The coordinates that make up the elbow within the MobL model are called 'elbow\_flexion' and 'pro\_sup'. The first of the two represents the humero-ulnar joint and was renamed to 'r\_elbow\_flex' due to some earlier usage of the code with other models. The latter takes over the function of the radioulnar joint which enables pronation and supination. JRFs will be reported for 'r\_elbow\_flex', the only connection to the humerus and thus towards the thorax and ground of the model. It thus gives an insight into which forces are present in the elbow overall.

In some initial simulations, all other joints (= coordinates) were locked, which lead to noisy JRF data. Also, this would mean that the locked joint would not need to be stabilized by the muscles themselves. As this wasn't deemed realistic for the present case, all other coordinates representing the wrist and shoulder joints are left in the unlocked default for the model.

Regarding the musculoskeletal model's degrees of freedom, one should always keep in mind that a lot of muscles are biarticulate, meaning that they act on more than one joint or articulation. This counts for the musculoskeletal model's muscles as well. By investigating the MobL model's muscles individually, it

---

<sup>1</sup> This note can be found in the readme file within the MobL model download files available from (OpenSim, 2021g).

was found that 64 % muscles (32 out of 50) were biarticulate <sup>2</sup> (see Appendix E). About 3/4 (17 out of 22) of the muscles involved in the elbow flexion are biarticulate. Involvement in elbow flexion is defined as having an attachment to the humerus and at least one of the ulna or radius.

## Weight

In order to represent the lifting of a 10kg weight with both hands, it has to be added to the musculoskeletal model. A point mass of 5kg is thus attached to the hand centered around the palm. For this the body “box” is added to the model, represented by a sphere. It is attached to the hand body via a weldjoint “weight\_grip”, which does not allow for relative movement of hand and weight. Its precise location was adjusted visually in the OpenSim GUI. The detailed code snippets can be found in Appendix F.

### 4.1.2. Motion File

The motion used for all simulations is the lifting a weight of 10kg with both hands in front of the body, as was the case for previous work at LfE (Bandmann, 2020).

To keep this parameter of the simulation as simple as possible, an artificial sinusoidal movement is created. Another reason for this simplification is that even slight inaccuracies in the kinematics can lead to miscalculations in the moment arms of muscles and actuators and thus muscle forces (Blache & Begon, 2018). Recorded motions are also often prone to error due to imperfections from e.g. skin markers, leading to soft-tissue-artifacts (Blache & Begon, 2018).

The movement ranges from one limit of the model to the other in a duration of precisely one second, meaning a movement from 0° to 130° in the coordinate ‘r\_elbow\_flex’. The peak angular velocity is 204°/s in the middle of the motion and the peak angular acceleration 641°/s<sup>2</sup> - positive at the start and negative at the very end of the motion. The sampling rate is set to 100 Hz, similar to the ones used in previous work at LfE (Bandmann, 2020). Preliminary tests showed it to be a good tradeoff between accuracy and computational effort. OpenSim shows a sampling rate of 60 Hz in its tutorial on motion capture tracking files (OpenSim, 2021f), showing that it is sufficient for following calculations. The elbow flexion angle over time for this motion can be seen in figure 8.

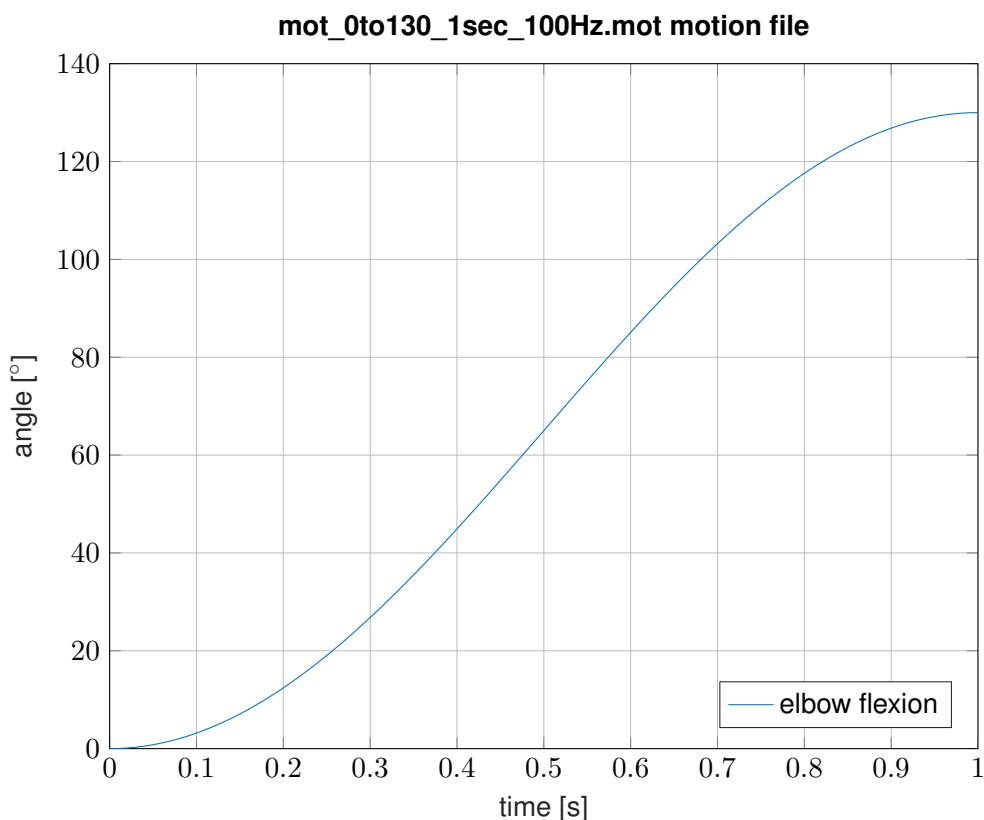
OpenSim motion files can be stored in the ‘.mot’ file format which includes a table with columns for the time steps, as well as the coordinate angles in degrees. For this a MATLAB script is written to generate these files automatically. The motion is described by the following equation with  $t$  as the time and 130° as the maximum flexion angle:

$$elbow\_flexion(t) = -\left(\frac{130^\circ}{2}\right) \cdot \cos\left(\left(\frac{\pi}{1 \text{ second}}\right) \cdot t\right) + \left(\frac{130^\circ}{2}\right)$$

All other coordinates are not part of the motion file. Instead, the default angles are set to zero in the model file, which meant that they would stay in this position and be held in place there by the muscles. Figure 9 shows the model at the start and end of the motion.

---

<sup>2</sup> These are muscles that act on several articulations and not just one.



**Figure 8:** Elbow flexion angle over time in the artificially created motion file

#### 4.1.3. Muscle Forces via Computed Muscle Control (CMC)

The selected musculoskeletal MobL model has been developed for usage with the CMC algorithm. As explained in chapter 2.2.3, CMC also includes passive forces, which are ignored in SO.

For these reasons, the CMC algorithm is chosen over SO to calculate the muscle forces throughout the motion for the given motion, model and actuator.

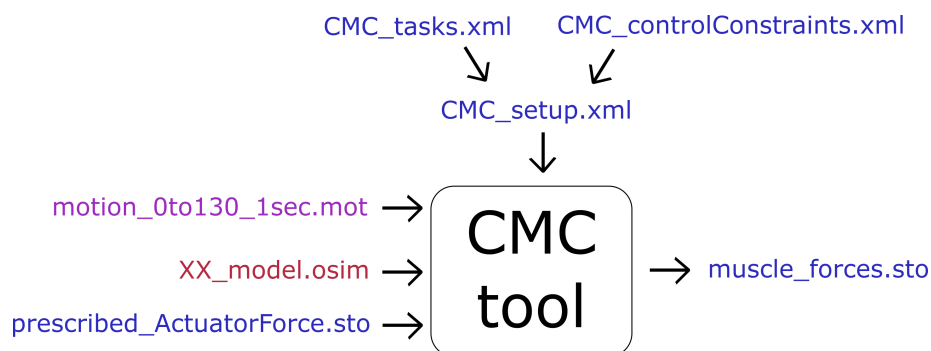
For the CMC simulations conducted, a baseline setup file is created. It serves as a stock setup, for which the inputs, outputs and settings are later adjusted by a MATLAB script (see `CMCrunner.m` in chapter 4.2.3). This script also runs the actual simulation using the modified setup.

The setup for CMC consists of three files. The main setup file references two others called “Control Constraints” and “Tasks” (see figure 10). The main setup file defines the simulation inputs and outputs to OpenSim’s CMC tool. They are presented in figure 10. OpenSim model, motion file and the prescribed exoskeleton actuator force/torque are inputs to the algorithm. They have to be provided as paths to the corresponding files. As output, the muscle forces of each muscle in every time step are given as a table within a storage file. The path to create this file needs to be defined in the setup. CMC takes the prescribed actuator force into account during the static optimization step as given and optimizes the remaining muscle forces accordingly. Other outputs that are not necessary for further steps in this workflow are not further documented. The setup file can be found in Appendix F. The tasks file tells the algorithm which coordinates to track with a specifiable tracking weight (OpenSim, 2021b). In this thesis, all seven coordinates in shoulder, elbow and wrist (see chapter 4.1.1) were tracked with equal



**Figure 9:** Visualization of the lifting motion from 0 to 130° elbow flexion. Screenshots at different time steps during the movement are overlaid in an image editing program.

tracking weights, as no tracking errors were encountered in previous tests. The control constraints file specifies limitations to certain muscles or reserve actuators (OpenSim, 2021b). The default controls were set to a minimum activation of 0.02 and a maximum activation of 1, limiting all muscles in the model. The minimum activation level was chosen due to a recommendation on the OpenSim website due to the behaviour of muscles under CMC (OpenSim, 2021c). Higher activations than 1 should not be physically possible. Depending on the analyzed motion, CMC simulations require reserve actuators which are added via another setup file. Their purpose is to compensate for a model not diverging from the desired trajectory at certain joints e.g. during noisy motion data. However, as the model's thorax is fixed to the ground and the artificial motion is completely smooth, there is no need for a reserve actuator file. This was tested and confirmed in early simulation trials.



**Figure 10:** Flow chart depicting inputs and outputs of OpenSim's CMC tool in the configuration of this thesis.

#### 4.1.4. Joint Reaction Forces via Joint Reaction Analysis (JRA)

OpenSim incorporates a tool to compute the JRFs resulting from a set of muscle forces throughout a movement: the JRA tool.

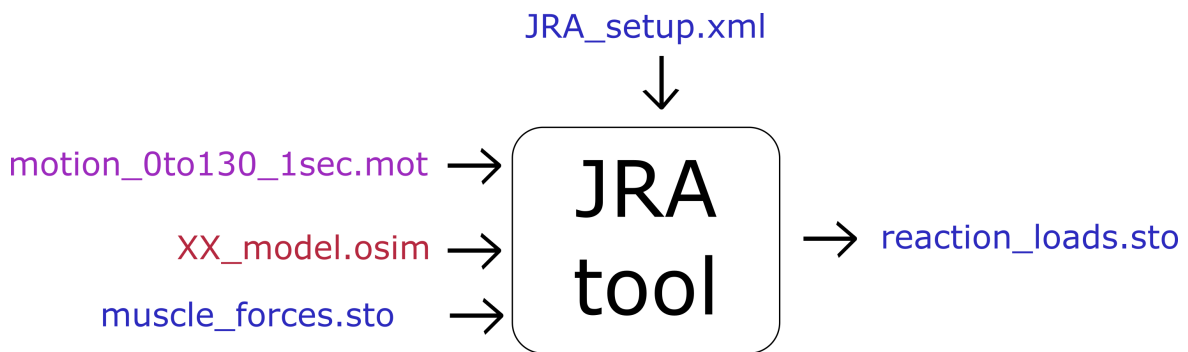
Just like with the CMC tool, a baseline setup file is established, which can be modified through MATLAB scripts, as will be explained in chapter 4.2. The only parameters being altered are the input and output

file paths. The setup file can be created by creating a general analysis tool in OpenSim and then adding “Joint Reaction” to the analysis set.

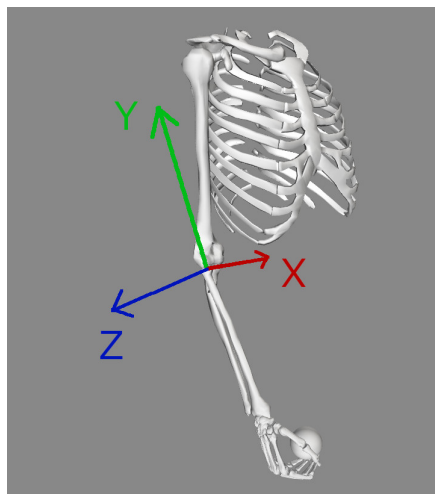
As inputs, this tool needs the paths to the OpenSim model file, the motion file, as well as the file containing the muscle forces during the movement. The latter is the one that was previously generated as an output from the CMC tool. An overview of the inputs and outputs for the JRA tool is given in figure 11.

The setup file also contains the settings on the nature of the JRFs to be computed. They are calculated for all of the model’s joints and all bodies / bones, each of them in the “child” reference frame. This means that the JRFs are provided in the reference frame of the distal body with the more proximal body closer to the thorax being the parent. In the elbow flexion coordinate “r\_elbow\_flex” this means that the JRFs are reported as the force acting from the humerus (parent) on the ulna (child) in the coordinate system / reference frame of the ulna (child). The coordinates of the ulna can be seen in figures 12 and 13. This convention for positive JRF vector components means that negative values portray a compression of the joint.

Lastly, the output path needs to be set. It will contain the computed JRFs as an OpenSim storage file with table-like data included for every time step as a line. Each column shows the vector components of the JRFs and moments.

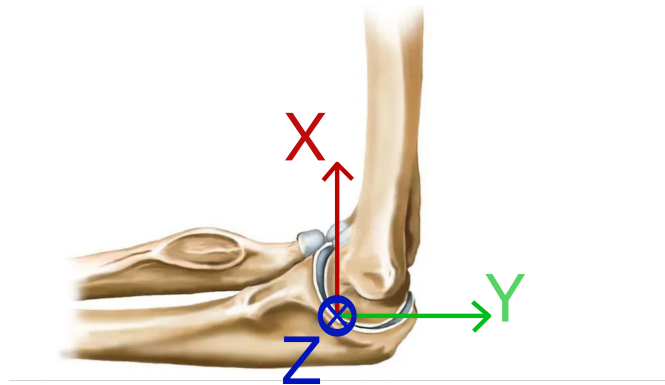


**Figure 11:** Flow chart depicting inputs and outputs of OpenSim’s JRA tool in the configuration of this thesis.



**Figure 12:** Coordinate system of the ulna body in the MobL Model





**Figure 13:** Convention for positive vector components of the reported elbow JRFs. Negative values in the XY-plane imply a compressive force in the joint. The coordinate system is fixed to the ulna and rotates around the humerus' head during flexion.

#### 4.1.5. Exoskeleton Moment Arms via Muscle Analysis

As explained in chapter 3.1, the support of the exoskeleton is to be held equal for all exoskeleton models in terms of the elbow flexion torque they provide. For models containing a path actuator this torque needs to be translated into a force acting along the path. This happens via the previously shown equation 3.3, which requires the path actuator's moment arm about the elbow flexion coordinate. The difficulty in calculating this moment arm is that it changes significantly during the motion and thus needs to be known for each time step.

To obtain the moment arm of muscles, OpenSim provides a tool called MA (MA). However, it does not compute the moment arms of actuators. Thus a copy of the desired model is created first, including a muscle with the same path points as the actuator. Forces are not relevant for this tool, as it only checks the kinematics.

With paths to this model copy and the motion file as inputs, the MA tool is able to compute the moment arm of the exoskeleton actuator as a muscle. The result is stored as an OpenSim storage file with the moment arms over time in a table-like format. They are then used in a MATLAB script to calculate the prescribed exoskeleton actuator force, as will be explained in chapter 4.2.

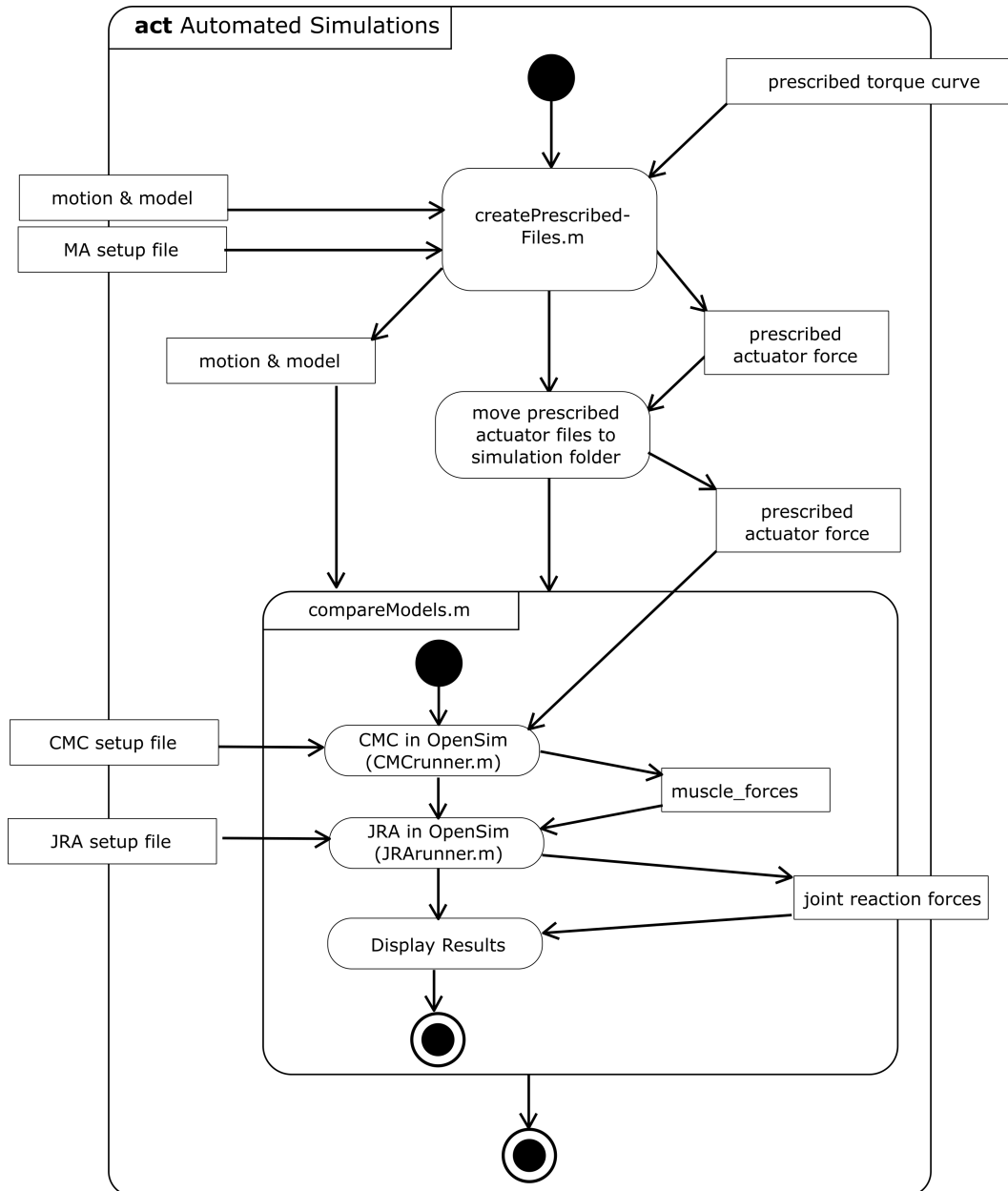
## 4.2. Automation and Workflow in MATLAB

The following chapter describes the implemented process that was developed and used in this thesis to automatically set up and run the simulation tools with the proper inputs and outputs. It is also employed to calculate intermediate results like the prescribed actuator files. Finally, MATLAB scripts are used to structure, evaluate and visualize the simulation results. In order to work with OpenSim, MATLAB needs to be set up using a script that is automatically installed along with OpenSim (OpenSim, 2021m).

### 4.2.1. Code and File Structure

The underlying folder structure for MATLAB, setup and results files is as follows. The folder containing all files and subfolders is called "Simulation". OpenSim model files, prescribed exoActuator, motion file, as well as the main two MATLAB scripts lie directly in this folder. It also serves as the current directory for all

MATLAB scripts and functions. Subfolders contain further MATLAB functions (“MATLAB\_functions”), the batch processed simulation results of CMC, JRA and the deduced plots (all three in “Results”), as well as an archive of previously used prescribed exoActuator files (“PrescribedController\_archive”).



**Figure 14:** Flow chart depicting the automated simulation workflow in MATLAB with its inputs and outputs.

Simulations are run as a two-step process. Firstly, the prescribed exoActuator force files are generated for all models via the “createPrescribedFiles.m” script in a subfolder of the prescribed files archive. These files then have to be placed in the “Simulation” folder, where they are found by the “compareModels.m” script. This second step runs through CMC and JRA and creates and saves the resulting data and plots (see figure 14).

Each function that is used by either of these scripts or functions has a description of inputs, outputs and

purpose in the first lines. Most functions lie in the “MATLAB\_functions” folder, only few that are only used once are saved as local functions within the parent function.

#### **4.2.2. Generation of Prescribed Force Files**

As described in chapter 3.3, this script creates the files to control the different exoskeleton actuators during the CMC tool’s simulation.

First a folder with the current date and time is created within the prescribed files archive. All temporarily used files and resulting OpenSim storage files are saved here. Next, the parameters for the script like models and torque curve are saved for documentation purposes. The moment arms file required for the calculation is generated in a local function running OpenSim’s MA tool.

The “computeCableForce.m” function then computes the actual cable force from this moment arm file and the selected torque curve. Using the previously stated equation 3.3 (see chapter 3.1), forces are calculated for path actuator models. Equal actuator moment arms of different models will lead to the exact same prescribed torque file. For models using a coordinate actuator, the torque curve is directly used as `exoActuator` torque.

For path actuators running forth and back along a trajectory like a loose pulley, the moment arms will be double that of what they would be with just one direction. The MA tool takes effects of complex actuators like these into account (Sherman, Seth, & Delp, 2010). As stated in chapter 4.3.1, some torque files result from other OpenSim tools, which do not always have synchronised time steps corresponding to the ones from the moment arm file of the MA tool. Thus, torque values are interpolated via a spline interpolation to enable calculation in the discrete time steps defined by the motion file’s sampling rate. Interpolation is justifiable for the smooth torque curves (see figure 15).

The torque curves sometimes involved negative values, which would mean a pushing force on the exosuit cables in a real setup. As this is not possible, negative values are set to zero in the MATLAB program calculating the actuator forces. To render an OpenSim storage file that is readable for the CMC tool, another function is used. It is called “motionGenerator”, as it was initially only used to generate the motion file. However, it is capable of creating storage files for all sorts of input. As it takes MATLAB function handles as input, the calculated prescribed `exoActuator` curve has to be handed over using a spline interpolated function handle again. Thus one should be careful in interpreting JRF results that do not differ significantly.

Finally, the files names of the prescribed files are displayed as a code snippet, that can be copied into the “compareModels.m” script of chapter 4.2.3. However, this is only necessary, when OpenSim model names or the motion file are altered from their default.

The code of the `createPrescribedFiles.m` function can be found in Appendix G.

#### **4.2.3. Batch Simulation of Models**

The MATLAB script “compareModels.m” allows for batch simulation of one up to all 14 models for one exoskeleton control strategy (see chapter 4.3.1 and 4.3.2). It runs the CMC and JRA tools and saves and plots the results. The functions `CMCrunner.m`, `JRARunner.m` and `evaluateJRFs.m` are used in the script as shown in figure 14.

## Preparation and Parameters

This script should be run only after having put the prescribed `exoActuator` files generated with the script from chapter 4.2.2. When using other OpenSim models or a motion file other than “`mot_0to130_1sec_100Hz`”, the code snippet generated in “`createPrescribedFiles`” has to be copied into the script section “CORRESPONDING PRESCRIBED EXOACTUATOR FILE”.

The user of the script can define which models are to be processed as an array containing the model numbers to be run (e.g. `[4 5 6]`). Also, the motion file can be selected. Keep in mind, that it should correspond to the one the torque file was meant for and the prescribed `exoActuator` files were created for. The model names and corresponding prescribed files are then automatically selected for the desired models. All other parameters can remain at their default. Also, an output folder for all subsequent result files is created with the current date and time within the “Results” folder.

## CMCrunner.m and JRArunner.m

As a next step the function `CMCrunner.m` is called for each model with the input arguments of model name and path, prescribed `exoActuator` file path, the previously defined output path and the motion file to execute the CMC tool with the correct setup.

This function sets up an OpenSim environment with the CMC baseline setup file as described in chapter 4.1.3. The setup is modified according to the arguments stated above. Prescribed files for the exoskeleton actuator are only applied when a file has been specified for the model within `compareModels.m`. Another setting that is being adjusted is the ending time for the tool using the function `createFinalTime.m` (see chapter 4.2.4). Finally, the CMC tool is started from within the `CMCrunner.m` function.

Similarly to `CMCrunner.m`, this executes OpenSim’s JRA tool in a function called by `compareModels.m`. It also sets up the OpenSim environment with the baseline JRA setup file and modifies it according to the arguments of model and motion file, as well as the path to the results folder. The latter is used to hand over the force file created by the CMC tool in `CMCrunner.m` to the JRA algorithm. Also, the ending time, as well as the output path are changed in the tool setup. Finally, the JRA tool is run by the function.

Both functions save a copy of the final tool setup just before running it for documentation purposes within the corresponding results folder.

## evaluateJRFs.m

This function, as the last major one within `compareModels` collects, plots and saves the data generated in the CMC and JRA functions.

The arguments it is run with are the results folder path, the model names and the motion file. Some deprecated parameters like the muscle model or the muscle force algorithm from preceding tests can be ignored and left at their default. The function then uses the `readSto.m` function to read data from the JRF results files, as well as the forces file from the CMC tool’s results. Subsequently, the `exoActuator` force

and elbow JRF data are extracted and saved as a MATLAB file (.mat). The total resulting JRF (absolute value) is computed as a root of squared sums (RSS) of the three vector components according to the following equation:

$$JRF_{total} = \sqrt{JRF_X^2 + JRF_Y^2 + JRF_Z^2}$$

The motion file is read as well to extract the motions's elbow flexion angle over time. Finally, the JRFs and exoskeleton actuator force / torque are plotted over the flexion angle.

Using the `readSto.m` auxiliary function (see 4.2.4), other data is extracted and plotted to obtain the graphs present in this thesis. The conversion into LaTeX format is conducted using a `matlab2tikz.m` script found on github<sup>3</sup>.

The code of the `compareModels` function, as well as `CMCrunner.m`, `JRARunner.m` and `evaluateJRFs.m` can be found in ??.

#### 4.2.4. Auxiliary Functions

Some auxiliary functions are used in the preceding functions. They are used as tools in several of them and are handy when handling the files in general. Thus they are presented in this additional section.

The code of all auxiliary MATLAB functions can be found in the digital appendix.

##### **readSto.m**

All results files from CMC and JRA are stored in an OpenSim specific storage file format. This format consists of a few header lines. They contain information like the number of data rows and columns. Sometimes additional information, such as if angles are displayed in degrees or radians, is included as well (OpenSim, 2021n). After the keyword `endheader` the actual data is presented in columns with column titles, the first of which is always the time steps (`time`).

In order to read these files and handle the data in MATLAB, the function `readSto.m` is used. It takes a file path as an input and extracts the column names and the corresponding data into a MATLAB table. If only specific columns are to be read, their name can be included as an input argument. The table is the returned output.

##### **motionGenerator.m**

The `motionGenerator.m` function is the counterpiece to the `readSto.m` function, as it can generate the OpenSim storage files for a given data input. As the OpenSim storage and motion files are closely tied (OpenSim, 2021n), this also holds for the latter. As this was its initial purpose, the name persisted. The

---

<sup>3</sup> <https://github.com/matlab2tikz/matlab2tikz>

inputs it uses to create such a file are file name (including path) and type, data column names, total and sampling time and function handle(s) for the data to be saved. The time steps are deduced from total and sampling time. The values are the function handles' output for the time steps. After creating the header lines including number of rows and columns, the data column titles are written as set by the column name input argument. After that, the columns are filled for `time` and the values calculated from the function handles.

The function does not return anything, but rather creates the file at the defined file name and path.

### **createFinalTime.m**

This function is used by `CMCrunner.m` and `JRARunner.m` to set the final time in the respective tool setup. This is required for OpenSim to know how far to run the simulation and is adjusted for each motion file depending on its duration.

`createFinalTime.m` uses `readSto.m` to search for the last entry in the `time` column and returns it as a scalar.

## **4.3. Exoskeleton Modeling**

### **4.3.1. Exoskeleton Actuators and Control**

To represent the exoskeleton in the musculoskeletal model, there are several options available in OpenSim. The most common way is to add them to an OpenSim model file as an actuator. The actuator(s) must then be controlled to define the force they apply to the body. However, the exoskeleton could theoretically also be represented via e.g. external forces acting on the model, which are later added in a simulation.

#### **Actuator Types**

Within OpenSim 4.2 there are 5 different actuators available (OpenSim, 2021i):

- coordinate actuator
- path actuator
- point actuator
- pointToPoint actuator
- torque actuator

In this thesis, two of the most common ones found in other publications are tested, namely the Coordinate and Path Actuators, the latter in different implementations (see chapter 3.4). Another reason for the selection of these two is that they are able to represent the cables driving the soft exosuit.

Coordinate Actuators directly apply a scalar force/torque along a generalized coordinate, e.g. a torque to the coordinate `"r_elbwo_flex"` in case of the elbow flexion. Path Actuators attach to fixed path points located on different bones and can exert a force along the path connecting the path points. They can wrap

around bones and joints when told to. The points can spread across one, two or more bone segments. Each path point is locked within the bone's reference frame it has been placed in.

## **Controlling the Actuator**

Another important step for the implementation of the exoskeleton within an OpenSim model is the way its support is controlled. These are some of the ways imaginable for the present use-case:

- as part of the CMC's optimization problem
- passive actuator
- active actuator with prescribed force/torque

When giving the actuator some control and force limits, the CMC algorithm is capable of using the actuator as if it were a muscle and part of the body. However, this implementation would make it hard to compare different models, as they might end up providing the body with different amounts of support.

A second possibility is implementing the exoskeleton as a passive actuator that provides force/torque according to e.g. its displacement or elongation. However, as the exosuit to be modeled is actively controlled, it does not represent it very well.

Another way of controlling the exoskeleton model's support is to implement a prescribed controller within OpenSim. This means that the actuator force/torque will follow a predefined curve throughout the motion, which has to be provided in an OpenSim storage file. As explained in chapter 3.3, this method was used for controlling the actuators in this thesis' simulations.

In order to use the prescribed force / torque file during the simulation it has to be inserted as a command to the CMC's main setup file. This is done via a MATLAB script as shown in chapter 4.2. First attempts at adding the prescribed controller directly to each model failed, as OpenSim did not take the forces into account during the simulation.

## **Torque Curves**

As one last step toward the control of the exosuit model actuators, the torque curve for the given motion had to be defined. As presented in chapters 3.1 and 4.2, this torque is translated into an equivalently supporting force for models with path actuators.

Three control strategies with resulting torque curves are chosen and run for all models:

- cost-optimized torque from CMC
- constant torque
- 100 % support of additional weight

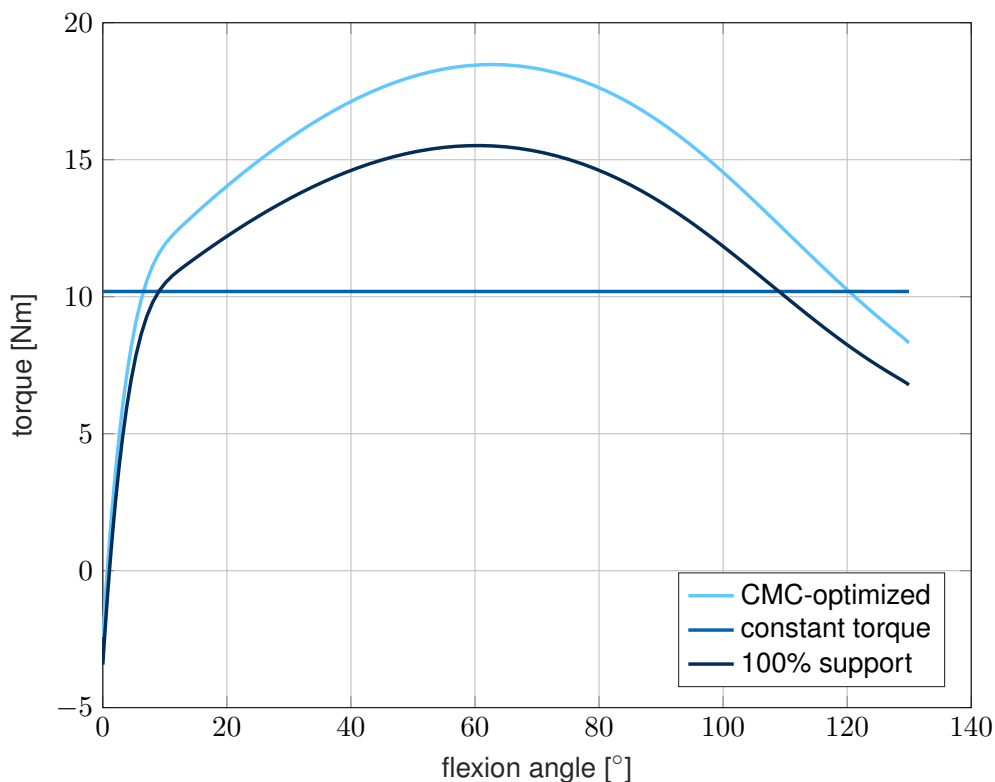
The first strategy follows an optimized control strategy in a way. The torque curve is created by running CMC with a model containing a coordinate actuator without a limitation on its maximum force so that the muscle force calculation would use as much force as is optimal for the algorithm's cost index. This strategy is called "CMC-optimized" support in the subsequent chapters. This strategy is pursued by (Bae, 2013) and (Bandmann, 2020) to represent exoskeletons in musculoskeletal simulations. However, this control

strategy may be hard to implement on a real exoskeleton when the user's exact muscle forces are hardly known and the motion has not been predefined.

A more easily implementable control strategy in the real world is to apply a constant elbow flexion torque via the exosuit. With the CMC-optimized torque curve as a baseline for a reasonable amount of support, the torque level for this control is set as its average. The torque values of the CMC-optimized controller of each time step are summed up and then divided by the number of time steps, resulting in a time-averaged torque. This torque level sits at 10.19 N m.

Another control strategy that implemented here is the full support of the 5 kg weight (see chapter 4.1.1). 100 % support is defined as the additional torque necessary to lift the weight. This is calculated using yet another OpenSim tool called "Inverse Dynamics". This tool calculates the generalized forces and moments along the model coordinates required to perform a given motion. The torque on the "r\_elbow\_flex" coordinate is such a generalized force. A new model was created, in which the mass from all body segments was set to zero, except the 5 kg weight. The tool then ran with the lifting motion file as input, providing the required torque for lifting the weight only as an output. This torque is then used as another prescribed torque for the exoskeleton models.

The resulting torque curves are shown in figure 15.



**Figure 15:** Three torque curves from different control strategies for the exoskeleton support.

#### 4.3.2. Models 0 - 13

All in all, 14 different OpenSim models were created in this thesis to investigate the effect of different exoskeleton modeling approaches on the resulting simulated JRFs. All except the two first models (num-

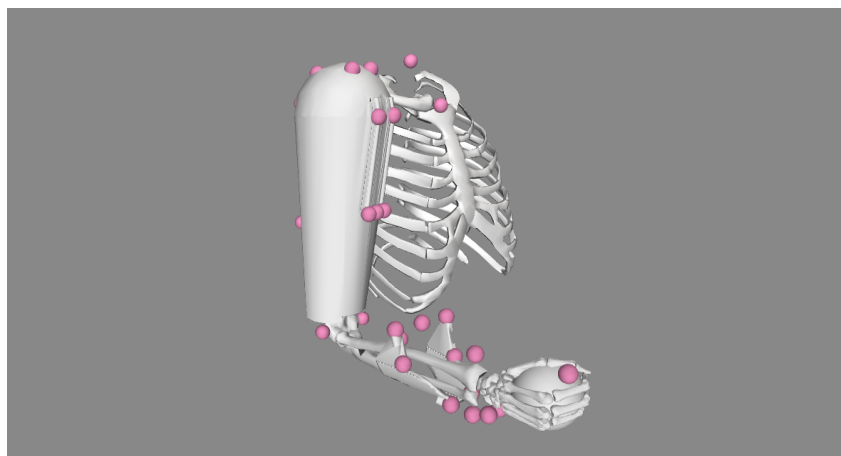


bers 0 and 1) were created by adding the exoskeleton in the form of an actuator to the musculoskeletal OpenSim model mentioned in chapter 4.1.1, including the 5 kg weight. The models differ in actuator type, as well as in the number and reference frames of path points for the path actuators. All of this amounts to different levels of detail in the modeling.

Table 4 gives an overview of all models, including the numbering and naming used in this thesis.

### Path Points Placement

In order to represent the exoskeleton as a path actuator in OpenSim, the path points need to be defined within the OpenSim model. The path points are start, intermediate and end points of a path, which can be fixed within a bone's frame of reference. In order to properly represent the exoskeleton's cables, these points are adjusted to a variety of references. This includes the actual prototype, the forearm brace of which had been designed to match the point "5G" described in (Bandmann, 2020). This point is set to be 15 cm along the forearm and 4 cm in palmar direction in a supinated position (Bandmann, 2020). A 3D model of the brace had been provided by Mr. Martin Fleischer from the LfE can be seen in figure 16. All path points along the forearm are set to match the cable path within this brace model file. Additionally, a model created by (Sugiarto, 2020) is used to match path points to the shoulder and upper arm segments. The exact position of each point is adjusted visually in the OpenSim GUI to match these visual references via markers. Every point along the forearm is defined in the reference frames of both ulna and radius to be able to select either one for each model. As the motion does not include pronation-supination movement, two equivalent points in ulna or radius coordinates stay exactly the same throughout the movement from chapter 4.1.2. The desired marker positions are then copied to those of path points of the corresponding models. A model including the forearm brace, as well as the shoulder part with matched markers can be seen in figure 16.



**Figure 16:** Marker positions used for path actuators with the prototype's reference geometries. The upper arm exosuit part was copied from (Sugiarto, 2020). The forearm brace was provided by Martin Fleischer from LfE.

| Number | Name                             | Actuator Type | # of Path Points | Forearm Attachment                       |
|--------|----------------------------------|---------------|------------------|--|
| 0      | 00_noExo_noWeight                | none          | -                | -  |
| 1      | 01_noExo                         | none          | -                | -  |
| 2      | 02_coordinateActuator            | coordinate    | -                | none (actuates elbow flexion coordinate) |
| 3      | 03_singlePath_ulna               | path          | 2                | ulna                                     |
| 4      | 04_singlePath_radius             | path          | 2                | radius                                   |
| 5      | 05_doublePath_ulna_radius        | path          | 4                | ulna & radius (closest in frontal plane) |
| 6      | 06_doublePath_radius             | path          | 4                | radius                                   |
| 7      | 07_doublePath_ulna               | path          | 4                | ulna                                     |
| 8      | 08_doublePath_connectedBelow     | path          | 6                | ulna & radius (closest)                  |
| 9      | 09_doublePath_AtWrist            | path          | 10               | ulna & radius (closest)                  |
| 10     | 10_doublePath_AtWrist_onlyRadius | path          | 10               | radius                                   |
| 11     | 11_doublePath_AtWrist_onlyUlna   | path          | 10               | ulna                                     |
| 12     | 12_doublePath_withShoulder       | path          | 20               | ulna & radius (closest)                  |
| 13     | 13_ulna_withShoulder             | path          | 14               | ulna                                     |

**Table 4:** Overview of all OpenSim models (0-13). Models 00\_noExo\_noWeight and 01\_noExo don't have any exoskeleton support implemented.

## Models for Moment Arm Calculations

As mentioned in chapter 4.1.5, a copy of each model including a path actuator is required to calculate the moment arms during a motion via MA. This model comprises a muscle with the same path points as the exoskeleton actuator. The model naming for these in the “simulation” folder is the same as the regular models 3 to 13, only with the numbering increased by 10, e.g. 14\_singlePath\_radius\_asMuscle for the original model 04\_singlePath\_radius. The muscle of the type Millard2012EquilibriumMuscle is copied and modified for this purpose from the “SUP” supinator, as it does not comprise any special wrapping features. An example of this kind of muscle with the same path as the exoskeleton path actuator can be found in Appendix H.

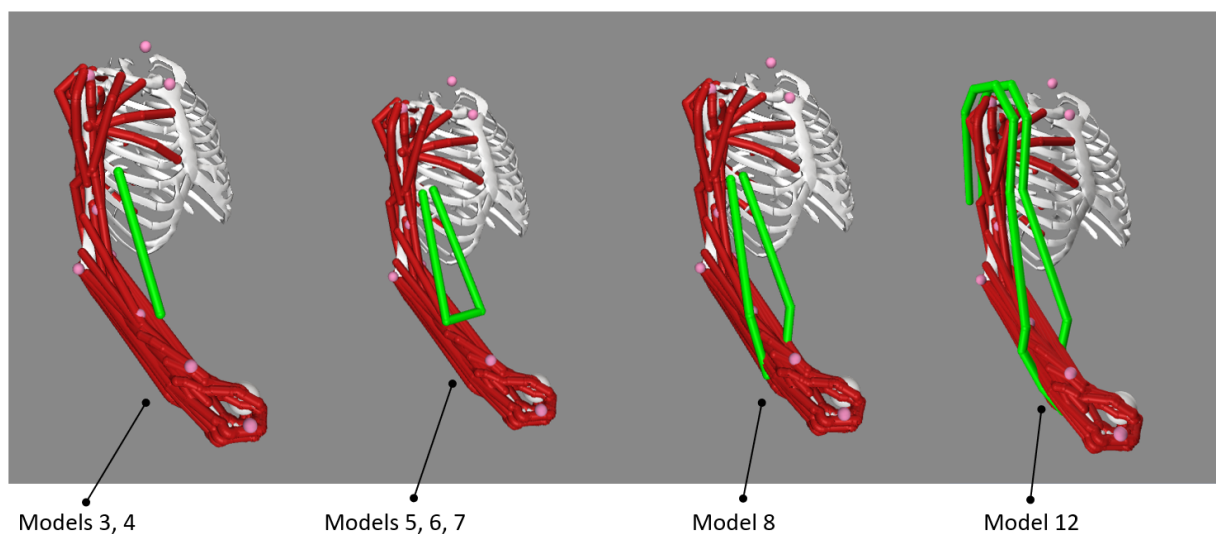
## Reference Models without Exoskeleton Actuator (models 0, 1)

The first two models 00\_noExo\_noWeight and 01\_noExo are created mainly as a reference and baseline for a model without an actuator representing an exoskeleton. They both include a coordinate actuator in order for the MATLAB scripts to run error-free. Model 01\_noExo includes the 5 kg weight, which is excluded for model 00\_noExo\_noWeight. Model 01\_noExo thus shows the simulated JRFs for the motion without the exoskeleton without the weight and model 00\_noExo\_noWeight even without the weight. The purpose of this is to see how well the exoskeleton actuator compensates for the added load.

## Coordinate / Torque Actuator (model 2)

Model 02\_coordinateActuator represents the exoskeleton as a simple torque to the elbow flexion coordinate, as was done for the representation of exoskeletons in numerous publications (see chapter 2.3.2). The OpenSim actuator type used is called `coordinate actuator`, as it operates on one specific coordinate only (here: “r\_elbow\_flex”).

Earlier trials using a `torque actuator` type acting on the model’s adjacent and force-transferring bodies humerus and ulna (see chapter 4.1.1) show the exact same results. Thus only the coordinate actuator version is used in the end. The two terms are used as synonyms in chapter 5. The source code for the actuator can be found in the digital appendix. An exemplary code snippet is shown in Appendix H.



**Figure 17:** OpenSim models including different exosuit actuators developed in this thesis.

### Single-Stringed Path Actuator (models 3, 4)

Researchers (Bae, 2013) and (Bandmann, 2020) had used simple two-point path actuators to represent exosuits in musculoskeletal modeling. Similarly, models `03_singlePath_ulna` and `04_singlePath_radius` implement the exoskeleton as a simple path actuator with one start and end point each. The difference between the two lies in the placement of the point on the forearm. For both it is located at the “5G” spot mentioned before, right between the two cable entries of the forearm brace. Thus they appear the same in the OpenSim GUI’s visualization. However, for `03_singlePath_ulna`, the point is fixed to the ulna’s reference frame, and for model `04_singlePath_radius` to to radius’s. An image of the OpenSim model including the path actuator can be seen in figure 17. The source code for the actuators can be found in the digital appendix. An exemplary code snippet is shown in Appendix H.

### Simple Double-Stringed Path Actuator (models 5, 6, 7)

(Agarwal et al., 2013) used more elaborate muscle-tendon-unit like actuators to represent a cable-actuated finger exoskeleton, similar in topology to a more complex path actuator comprising more than two path points.

The next three models `05_doublePath_ulna_radius`, `06_doublePath_radius` and `07_doublePath_ulna` represent the two exoskeleton cables running in parallel in a simplified way whilst ignoring the cable part within the brace and over the shoulder. The path starts at the shoulder outlets, runs through two points on the forearm and then back to the shoulder cable outlet. The path actuator is thus made up of four path points.

As with the single-stringed models `03_singlePath_ulna` and `04_singlePath_radius`, these three models differ in the attachment of the forearm path points. Model `05_doublePath_ulna_radius` uses the bones lying closest to the respective path point in the frontal plane. Thus the medial point is fixed in the ulna’s coordinate system, while the lateral point sits in the radius’s. This is done to force the points to be

in different reference frames in order to investigate it's effect. Models 6 and 7 attach both forearm points to the radius and ulna respectively. A visual representation of the three models, which - again - look the same in the OpenSim GUI's visualizer, can be found in figure 17. The source code for the actuator can be found in Appendix H.

### **More Complex Forearm Paths Actuator (models 8, 9, 10, 11)**

Models 8 to 11, named 08\_doublePath\_connectedBelow, 09\_doublePath\_connectedAtWrist, 10\_doublePath\_connectedAtWrist\_onlyRadius and 11\_doublePath\_connectedAtWrist\_onlyUlna expand the double-stringed models by adding more points in between the two brace cable outlets. These points follow the real prototype's cable path in two levels of detail.

Model 8 adds two more points, making the exoskeleton actuator pass on the underside of the forearm. This model can be seen in figure 17. Additionally, models 9 to 11 implement the entire cable path reaching up to the wrist along the underside of the forearm (see chapter 2.3.2). Just as with the simple double-stringed models, they differ in the forearm path point attachment frames. They look just like model 12\_doublePath\_withShoulder without the shoulder path points (see figure 17). 10\_doublePath\_connectedAtWrist\_onlyRadius and 11\_doublePath\_connectedAtWrist\_onlyUlna have all forearm points attached to one reference frame only, radius and ulna respectively. The forearm path points of 09\_doublePath\_AtWrist are mostly placed in the ulna's reference frame, as it is closer than the radius. The first one or two most proximal points on each side of the forearm are in the radius' frame for the same reason (see Appendix H). An image of models 09\_doublePath\_AtWrist, 10\_doublePath\_AtWrist\_onlyRadius and 11\_doublePath\_AtWrist\_onlyUlna with identical visual appearance can be found in figure 17. The source code for the actuators can be found in Appendix H.

### **Models with Path over Shoulder (models 12, 13)**

Model 12\_doublePath\_withShoulder is the same as 09\_doublePath\_connectedAtWrist, except for 10 more path points which are added to implement the cables running over the shoulder and inserting into the electric motor and gearbox (see figure 17). Model 13\_ulna\_withShoulder also has the path over the shoulder implemented, but comprises the same forearm attachment as model 07\_doublePath\_ulna (see chapter 4.3.2). All points along and over the shoulder are fixed in the humerus's coordinate system. The first and last path point connected to the gearboxes and motors sit in the thorax reference frame, as they would be attached to the user's back. The source code for the actuators can be found in Appendix H.

## 5. Results

### 5.1. Overview of JRFs (Models 0 – 13)

For each of the models presented in chapter 4.3.2, the JRFs were simulated according to the workflow and tools presented in chapters 3.3 and 4. The results are gathered and presented in this chapter. Different models are compared to one-another and to the reference models without an actuator (see chapter 4.3.2).

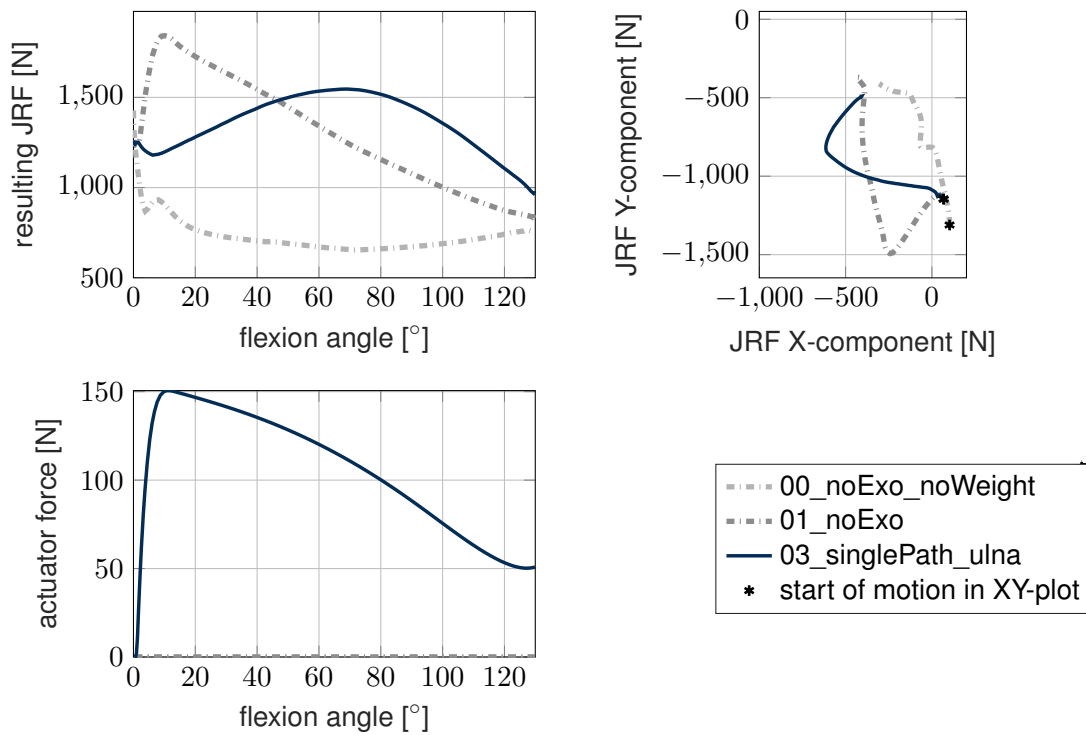
It is important to note that the sections comparing specific models do so mainly for the 100%-support control strategy. This is done as the trends observed when comparing different models are often redundant in the other two. At the end of each section, however, special observations in the other two strategies are mentioned in a separate paragraph.

Two types of plots are used to visualize the simulated JRFs. Firstly, one type of figure is provided for each section comparing two to four models for the 100%-support control strategy (see e.g. figure 18). Each of these contains three subplots showing the exoskeleton actuator's force or torque and the resulting JRF of each investigated model over the motion's elbow flexion angle. The third subplot depicts the JRF's vector components in the ulna's reference frame. As explained in chapter 2.5 and according to the simulation results, the Z-component plays only a minor role compared to the other two coordinates. The directional plot thus only shows the XY-plane. In each of the subplots the JRFs of models `00_noExo_noWeight` and `01_noExo` are plotted in thick dotted lines to serve as a reference for the other models.

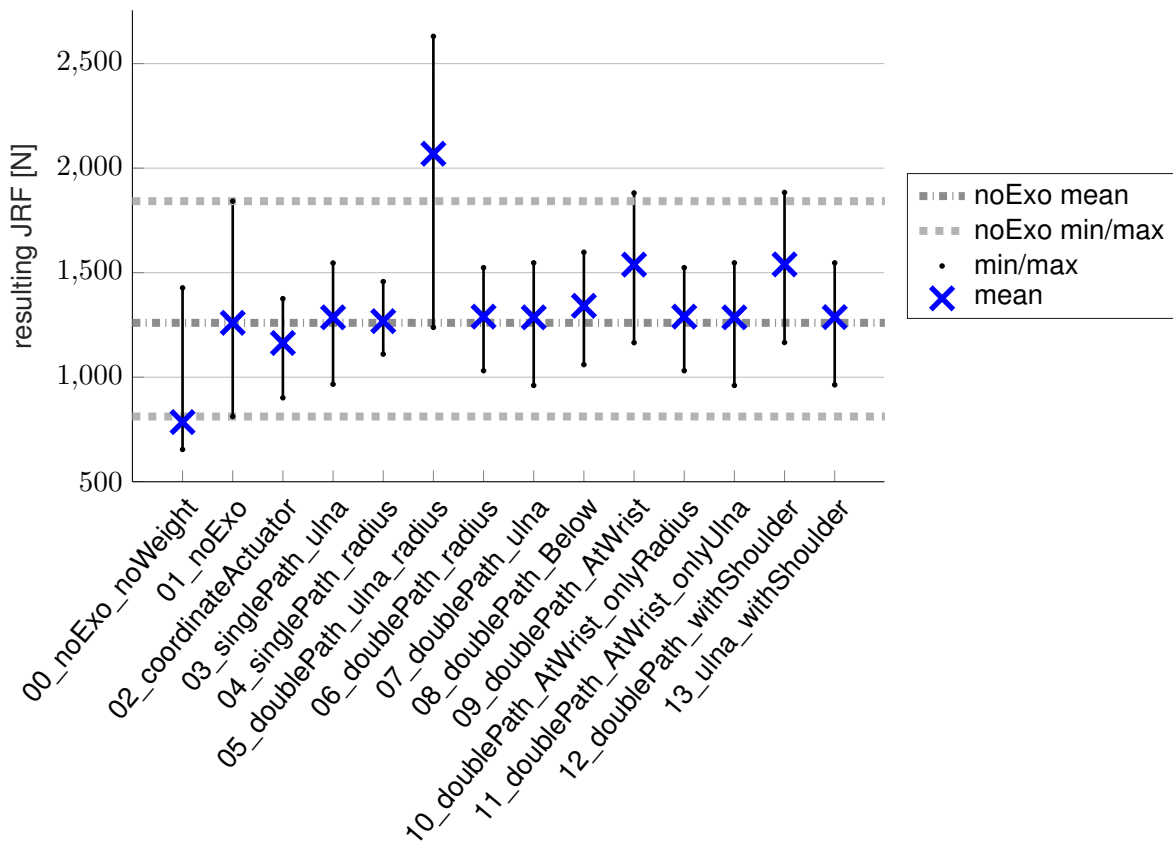
Secondly, a boxplot showing the minimum, maximum and mean value of all models gives an overview on the level of JRF present in the different models (see figure 19). The mean has been calculated as an average of the JRF values of each time step. The values of Models 0 and 1 are shown as thick, horizontal lines to give an insight on the change imposed by using each exoskeleton model compared to these references. Figure 19 shows the JRF levels for all models using the 100%-support controller. Similar boxplots for the CMC-optimized and constant torque control strategies can be found in Appendix A.

In figure 19 it is apparent that for most models, the mean, minimum and maximum values are held within the range of those of model `01_noExo`. Maximum values usually decrease compared to model `01_noExo` and minimum values increase. The mean value increases slightly for most models, except for model `02_coordinateActuator` containing a coordinate actuator. Models `05_doublePath_ulna_radius`, `09_doublePath_AtWrist` and `12_doublePath_withShoulder` have mean values significantly higher than model `01_noExo`. The maxima of these three also lie above model `01_noExo`'s. A comparison for the mean, minimum and maximum values with the other two control strategies is given in the following subsection.

The minimum, maximum and mean JRF values plotted in the boxplot are also documented in a table in Appendix C for all three control strategies. And so are tables containing the increase or decrease in these three metrics for each model and controller compared to those of model `01_noExo` with no exoskeleton (see Appendix D).



**Figure 18:** Example plot of resulting JRF, JRF vector components and actuator force of models 01\_noExo\_noWeight, 01\_noExo and 03\_singlePath\_ulna. This type of plot is used to compare different models' JRFs in subsequent sections.



**Figure 19:** Minimum, maximum and mean elbow JRF values of all 14 models for the 100%-support controller. This type of plot can be found for the other two control strategies in Appendix A.

## 5.2. Comparison of Control Strategies

In order to compare the JRFs resulting from different control strategies presented in chapter 4.3.1, they are being compared using one of the simplest models `03_singlePath_ulna` with a single-stringed path actuator attached to the ulna. For the prescribed actuator force and the resulting JRF as well as its vector components, most models follow similarly shaped curves, with only a few exceptions. The general shapes are explained here, whereas the following sections in chapter 5.3 compare specific models and go into detail on their characteristics aside from the general shape.

As stated before, the comparison in the following chapter 5.3 focuses mostly on the 100%-support strategy. For the other two control strategies, only peculiarities are noted.

The exoskeleton path actuator force is calculated from the prescribed torque (see chapters 3.3 and 4.2), which is defined as the control strategy here. Thus, for each strategy, different prescribed forces can be observed in the actuator force plot (see bottom left in figure 20). The actuator force is set to zero at the very beginning for the CMC-optimized and the constant torque controllers, as the underlying torque is negative (see chapter 4.2.2). After that both of them rise steeply to a maximum at around  $9^\circ$  flexion and then decrease steadily. The CMC-optimized controller's force rises again at the end of the motion. The exoskeleton actuator force from the constant torque control strategy, however starts at its maximum and decreases until roughly  $110^\circ$  flexion, after which it increases again up to the end of the motion. Due to a loose pulley effect, the prescribed actuator forces of models with two interconnected, parallel strings in one actuator only have half the force throughout the entire motion. An example for this is found in chapter 5.3.3.

Looking at the boxplots in figure 19, one can see that the 100%-support controller generally lead to a decrease in mean and maximum JRF compared to `01_noExo`. For the constant torque and the CMC-optimized controller, most mean JRF are similar to model `01_noExo`'s (see Appendix A and Appendix D). The maxima are usually decreased for the CMC-optimized and increased for the constant control strategy. These general assumptions stand for most models. Outliers and detailed comparisons between models follow in chapter 5.3.

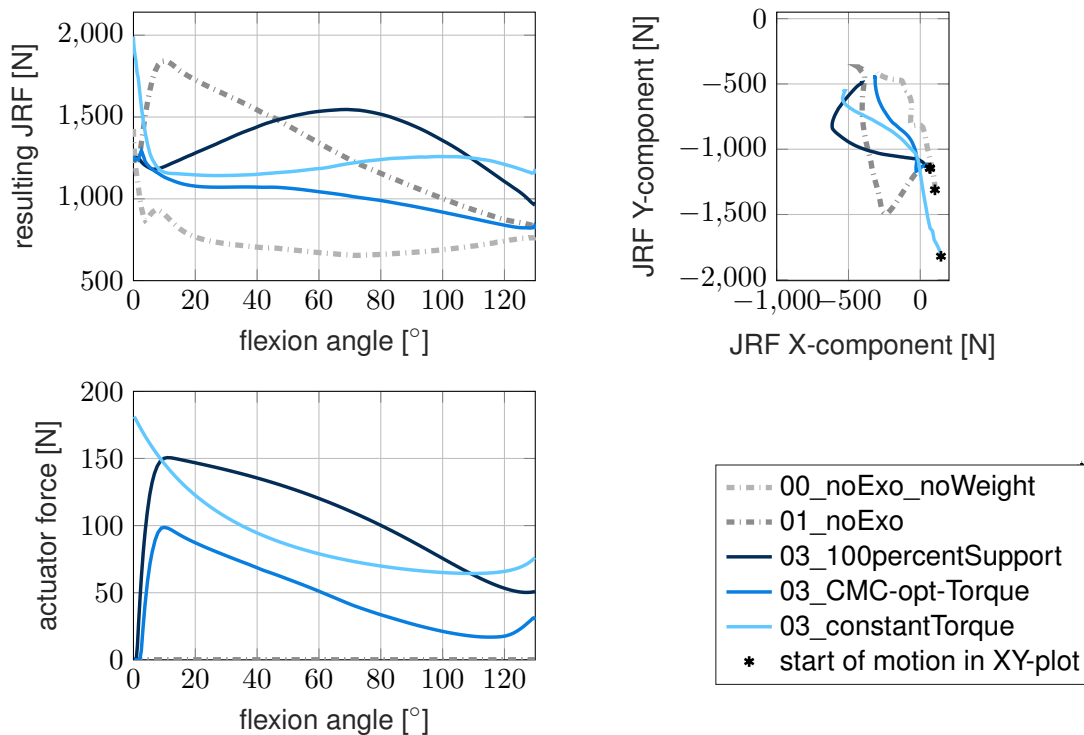
The JRF curve with the 100%-support control strategy mostly describes a concave shape, similar to a slightly distorted parabola open towards the bottom (see figure 20). The start of the motion forms an exception, as the JRF decreases from its starting point until the first few degrees of flexion. It reaches its maximum around the halfway point of the motion and its minimum at the very end. Vector component and direction-wise the 100%-support strategy starts in its maximum Y-compression and a slight X-decompression. From then on, Y-compression slowly decreases and X-compression increases until maximum total JRF. Towards the end of the motion, both X- and Y-compression decrease.

The CMC-optimized controller's JRF start with its maximum and experience a gentle decline over the course of the motion (see figure 20). In comparison to the other two control strategies, the minimum



and maximum levels lie relatively close together. The decrease happens mainly in the Y-compression direction, whereas the X-compression increases throughout the motion.

The JRF curve for the constant torque control strategy generally starts at its highest point as well (see figure 20). However, the peak is more pronounced than the other two strategies' maxima. After a significant drop in JRF within the first 10° flexion, the force stays relatively constant until the end of the motion compared to the other two strategies. A local maximum is reached towards the end of the motion, after which the JRF decreases until the very end. The main contributor to the initially high JRF is the compression in the Y-direction, while the X-direction is in slight decompression. After the first decline, X-compression increases, while Y-compression decreases. Only at the very end do both compression components decline.



**Figure 20:** JRFs and actuator forces of model 03\_singlePath\_ulna with the three different control strategies

### 5.3. JRFs for different Models

After having explained the general patterns observed with different control strategies, this chapter focuses on the comparison of specific models' JRFs with the 100%-support strategy as a default.

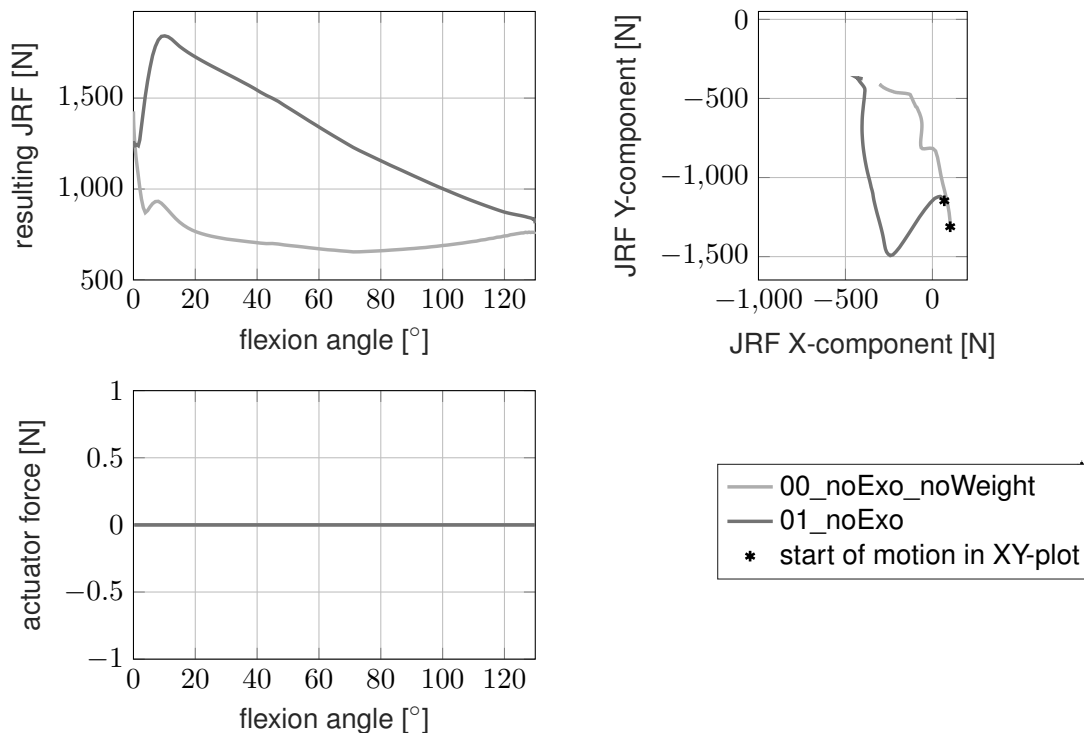
#### 5.3.1. Models without Exoskeleton Support (Models 0 & 1)

The simulation results of the first two models 00\_noExo\_noWeight and 01\_noExo are used as references in the subsequent sections. They show the simulated JRFs during the motion described in chapter 4.1.2 for the models without any exoskeleton support (see figures 19 and 21). Model 01\_noExo includes the weight, in contrary to model 00\_noExo\_noWeight.

The curves of models 00\_noExo\_noWeight and 01\_noExo seen in figure 21 are included in the other JRF curve plots as thick, grey, dotted lines (see e.g. figure 20) to give an easily accessible insight on the JRFs in comparison to a model with exoskeleton support.

The JRFs of 01\_noExo are higher than the ones of 00\_noExo\_noWeight throughout the entire motion. Their mean values are 785 N (model 00\_noExo\_noWeight) and 1 260 N (model 01\_noExo) respectively. The curve of 00\_noExo\_noWeight stays relatively constant after its maximum at the very beginning, whereas 01\_noExo has a sharp increase at the start and then monotonously decreases until its minimum value. Towards the end of the movement, both models' JRFs almost converge to 765 N (model 00\_noExo\_noWeight) and 812 N (model 01\_noExo).

Regarding the direction of the JRFs, the X-, as well as the Y-component are larger for 01\_noExo than 00\_noExo\_noWeight at all times. The decrease in resulting JRF for model 01\_noExo can be largely attributed to a relief of the Y-component, which is the one in the direction of the forearm. Except for a brief decompression in the X-direction at the start of the motion, all force components are compressive.



**Figure 21:** JRFs of models without exoskeleton support.

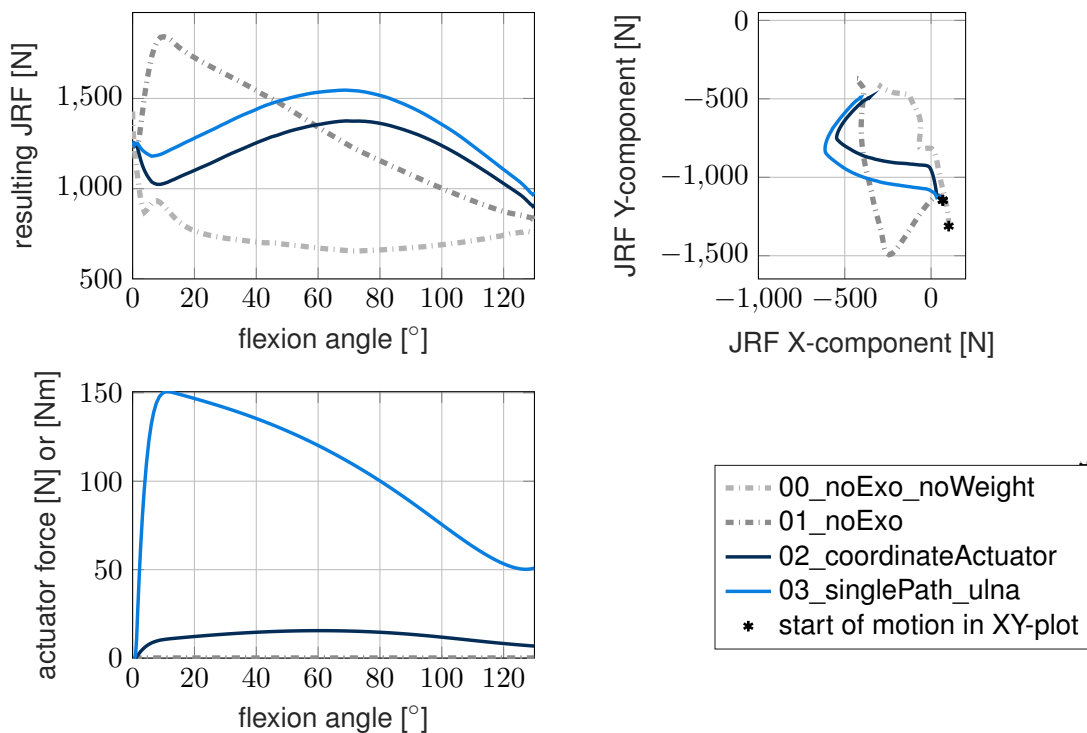
### 5.3.2. Coordinate and Path Actuator (Models 2 & 3)

Model 02\_coordinateActuator and 03\_singlePath\_ulna have the exosuit implemented as a torque actuator and one of the most simple path actuators attached to the ulna. The simulation results for the JRFs can be seen in the boxplot of figure 19 and figure 22.

The exoskeleton actuator torque of model 02\_coordinateActuator reaches its maximum at about 60 ° flexion (see figure 22). Due to a changing moment arm of the path actuator, model 03\_singlePath\_ulna's prescribed force rises steeply to about 150 N at 12° flexion and then monotonously declines until the end of the motion.

The pattern of the JRFs simulated with these two models both follow the general shape described in

chapter 5.2, except for a brief decrease at the beginning (see figure 22). Both curves run almost in parallel from  $10^\circ$  to  $60^\circ$  flexion and approach each other towards the end of the motion. Up to the point around the maximum JRF the compression in X-direction grows for both exoskeleton models. After that the decrease is mainly a result from decreasing compression in Y-direction. The JRFs of 03\_singlePath\_ulna are higher than those of 02\_coordinateActuator throughout the full range of motion. This is true for all three control strategies (see Appendix B). For the 100% support control the difference is 123 N on average with a mean of 1163 N for the torque actuator and 1286 N for the path actuator model. It should also be mentioned, that the JRFs of the path actuator model are found to be lower than all other path actuator models for every control strategy. In comparison to 01\_noExo, the JRFs of models 02\_coordinateActuator and 03\_singlePath\_ulna are lower in terms of resulting JRF until  $45^\circ$  (model 03\_singlePath\_ulna) and  $55^\circ$  (model 02\_coordinateActuator) flexion and stay higher for the rest of the movement (see figure 22). The mean value of model 02\_coordinateActuator stays below that of the reference model 01\_noExo by 7.6%, whereas the path actuator model is slightly above it by 1.8%. The maximum values of both these exoskeleton models stay below that of 01\_noExo by 25.3% (model 02\_coordinateActuator) and 16.1% (model 03\_singlePath\_ulna).



**Figure 22:** JRFs and actuator forces of models with coordinate and path actuators. The dotted lines are the JRFs in the models without exosuit support and included in every JRF plot.

For the other two control strategies with CMC-optimized and constant torque (see Appendix B), the patterns are different in that the JRF curves are more stable during the motion, except for a large spike in total JRF at the start of the constant torque's curve. This leads to the maximum values being about the same as 01\_noExo for model 02\_coordinateActuator and 150 N higher for model 03\_singlePath\_ulna. For both, the torque actuator always stays below the path actuator. In the optimized control's JRFs, the mean

values are far below that of 01\_noExo and the maximum values lie around the mean of it. In terms of JRF direction, the lower values observed for most of the motion can be attributed to lower compression forces in the X-direction.

### 5.3.3. Single- and Double-Stringed Path Actuators on the Same Body (Models 3 & 7 and 4 & 6)

Models 03\_singlePath\_ulna and 07\_doublePath\_ulna both attach to the ulna only, but represent the exoskeleton as a single- or interconnected double-stringed path actuator as described in chapter 4.3.2. The same goes for the radius body with models 04\_singlePath\_radius and 06\_doublePath\_radius. The models are compared to each other in these pairs using figures 23 and 19. The results for paths with different attachment points follow in chapter 5.3.4.

Looking at the actuators' forces one can see that the models with double-stringed path actuators have half the force compared to their single-stringed counterparts. This is due to the loose pulley effect. Prescribed forces of both single-stringed models are basically identical, as are the double-stringed ones.

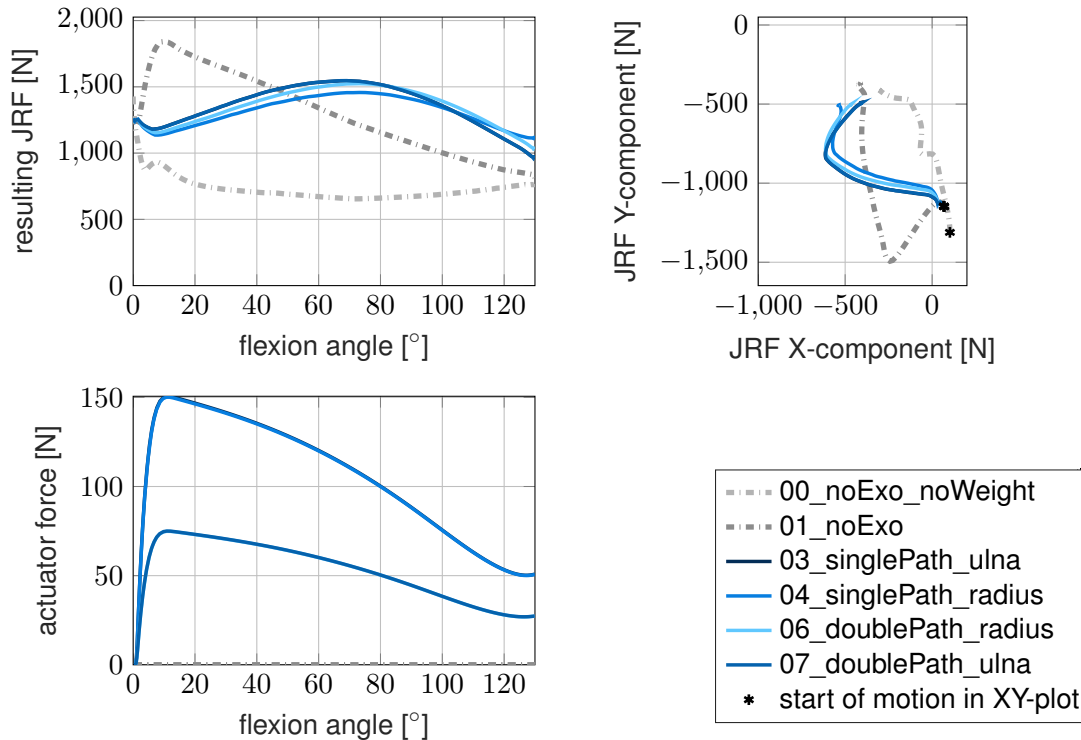
The curves of the resulting JRFs for models 03\_singlePath\_ulna and 04\_singlePath\_radius, as well as 06\_doublePath\_radius and 07\_doublePath\_ulna all share the general JRF pattern described for the 100%-support control strategy in chapter 5.2. The lines of both ulna-attached models share the same JRF values and thus appear as one in the plot (see figure 23). There is thus no significant difference between these two for the whole motion. The ulna-attached model's JRF directions are generally speaking very similar to the ones observed for most of the 100%-support strategy (see chapter 5.2). For the radius-attached models, the double-stringed variant shows higher compression in the Y-direction for most of the movement and lower compression in X-direction towards the end. The increase for the single-stringed model 04\_singlePath\_radius in X-compression differs from the default pattern described in chapter 5.2 and is not observed in most other models.

The latter manifests in higher mean and maximum JRF for model 06\_doublePath\_radius compared to the single-stringed model 04\_singlePath\_radius. The difference is 22 N for the mean and 67 N for the maximum. For both ulna-attached models 03\_singlePath\_ulna and 07\_doublePath\_ulna the mean is 1286 N and the maximum 1465 N. Compared to 01\_noExo, this means a difference of 2.1% for the mean and 16% for the maximum value. For the radius-attached models the mean JRF experienced a minor increase by 0.6% (model 04\_singlePath\_radius) and 2.4% (model 06\_doublePath\_radius). However, the maxima are reduced by 20.9% (model 04\_singlePath\_radius) and 17.3% (model 06\_doublePath\_radius) in comparison to 01\_noExo. Up until around 46 (models 03\_singlePath\_ulna and 07\_doublePath\_ulna) to 52° (model 04\_singlePath\_radius) flexion the four JRF curves are below model 01\_noExo's and higher for the remainder.

For the constant control strategy a large spike in JRFs is present in all models at the start of the motion as a result of the very short actuator moment arm in this position. One difference compared to the 100% support control is a rise in JRF in model 04\_singlePath\_radius towards the end, which does not occur in the others.

All JRF curves are very close together in the CMC-optimized controlled simulations. All 4 are only separated by a maximum of about 10 N until roughly 115° flexion. Only then do they diverge, as the actuator forces rise. Model 04\_singlePath\_radius has an elbow JRF about 55 N higher than its double-stringed

counterpart. The JRFs of the ulna-attached models are the same for both other control strategies as well.



**Figure 23:** JRFs and actuator forces of models with single- and double-stringed path actuators fixed in the ulna's or radius' coordinate system  
(Actuator forces of models 03\_singlePath\_ulna and 04\_singlePath\_radius as well as 06\_doublePath\_radius and 07\_doublePath\_ulna are the same.)

### 5.3.4. Different Attachment Point Placements

As mentioned in chapter 4.3.2, the path points of the path actuators in models 3 to 13 are each fixed in one body's coordinate system. This chapter shows the simulated JRFs for models with the same topology, but with points attached in different frames, namely the two forearm bones radius and ulna. This is done for single- and simple double-stringed path actuators, as well as for more complex ones with more points along the forearm to show more detail (see chapter 4.3.2).

#### Single-Path Actuators (Models 3 & 4)

Models 03\_singlePath\_ulna and 04\_singlePath\_radius represent the exoskeleton as single-stringed path actuators with just one attachment point on each the forearm and the upper arm. The forearm attachment point sits in the ulna (Model 03\_singlePath\_ulna) or radius (model 04\_singlePath\_radius). The plots depicting the elbow JRFs and exoskeleton actuator force can be seen in figure 23. Therein the plot line of model 03\_singlePath\_ulna follows the same curve as model 07\_doublePath\_ulna and is thus covered by it.

The first thing to mention is the force of the exoskeleton actuator. As the models have the same topology

and resulting moment arms around the elbow joint, the prescribed force is equal throughout the motion. However, the resulting JRFs differ. The general pattern follows the default concave one described for the 100%-support strategy in chapter 5.2. As mentioned in chapter 5.3.3, the radius-attached model's JRF has an inflection point towards the end of the flexion movement as a peculiarity, which model 03\_singlePath\_ulna does not have. In terms of the directionality, model 03\_singlePath\_ulna has a higher Y-compression component until about the maximum JRF is reached. After that, the X-compression component of model 03\_singlePath\_ulna decreases faster than that of the ulna-attached model 04\_singlePath\_radius.

The mean value for model 03\_singlePath\_ulna (ulna) is 1286 N - 18 N more than that of model 04\_singlePath\_radius. In terms of the maximum, model 03\_singlePath\_ulna surpasses its counterpart by 89 N with the peak at 1564 N. The spread between minimum and maximum JRF is smaller for the radius-attached model (see figure 19). In comparison to model 01\_noExo, the models have mean JRFs higher by 2.1% (model 03\_singlePath\_ulna) and 0.6% (model 04\_singlePath\_radius). The maximum values, however are reduced by 16.1% for model 03\_singlePath\_ulna and 20.9% for model 04\_singlePath\_radius. The JRFs are lower than the ones without exoskeleton support until 45° and 53.53° flexion.

When looking at the simulated JRFs of the CMC-optimized control strategy, the curves of models 03\_singlePath\_ulna and 04\_singlePath\_radius are very close to one-another for most of the movement. They only diverge by more than 20 N at the very end, where the compression in X-direction is higher in radius-attached model 04\_singlePath\_radius. The mean values are way lower than the ones with 100% support control and lower than those of model 01\_noExo, as is the case for most models (see chapter 5.2). In fact, the maximum values observed are both in the vicinity of the mean value of model 01\_noExo with 1013 N (model 03\_singlePath\_ulna) and 1028 N (model 04\_singlePath\_radius).

For the constant torque control, the JRF of ulna-attached model 03\_singlePath\_ulna are higher than those of model 04\_singlePath\_radius for all except the last part of the movement, similar to the results with the 100%-support strategy. The maximum values observed for both models exceeded those of model 01\_noExo due to the large peak of JRFs at the start of the flexion - motion, as is observed in previous models. The mean JRF values for the two single-stringed path actuator models with this control strategy are similar to those of model 01\_noExo.

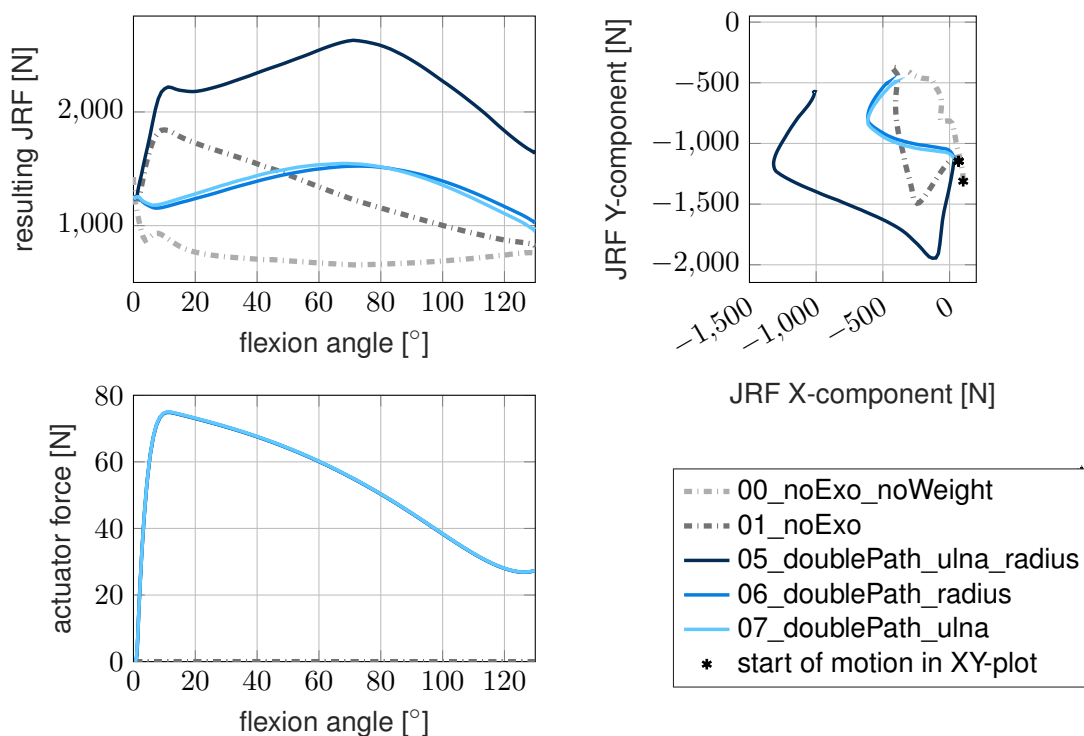
### **Simple Double-Stringed Path Actuators (Models 5, 6 & 7)**

The models investigated in this section are all path actuators with four path points arranged in the same geometric positions. The points on the forearm are set in the radius' reference frame for model 06\_doublePath\_radius, in the ulna's for 07\_doublePath\_ulna and in the body frame closest in the frontal plane in 05\_doublePath\_ulna\_radius (see chapter 4.3.2). The same geometric path point setup results in equal moment arms around the elbow flexion coordinate and thus the same prescribed exoskeleton actuator forces (see figure 24).

Models 06\_doublePath\_radius and 07\_doublePath\_ulna describe the concave function similar to the one described in chapter 5.2 except for the first 10° of flexion with its maximum around 70° (model

07\_doublePath\_ulna) and 75° flexion (model 06\_doublePath\_radius), as can be seen in the JRF plot in figure 24. The JRF direction in the XY-plane does not reveal any particularities either. Model 05\_doublePath\_ulna\_radius however, has a sharp increase of JRF at the beginning, mainly in Y-direction compression. After that, its X-compression increases during a decline in Y-compression until the maximum total JRF is reached at about 70° flexion. Finally, both compressions decrease again until the end of the motion.

In terms of magnitude, the mean values of both models attached to just one forearm bone show a slight increase in comparison to model 01\_noExo's 1 260 N of about 2% each (see Appendix C and Appendix D). For the maximum values of 1 524 N (model 07\_doublePath\_ulna) and 1 547 N (model 06\_doublePath\_radius) a decrease of 16 % (model 07\_doublePath\_ulna) and 17.3 % (model 06\_doublePath\_radius) to model 01\_noExo is observed. Model 05\_doublePath\_ulna\_radius with different attachment bodies on the forearm shows the highest minimum, mean and maximum JRFs observed in all models with the 100%-support control strategy. This manifests in an increase of 64.2 % in mean JRF compared to 01\_noExo with 2 069 N and an increase in its maximum by 42.8 % at 2 630 N (see Appendix C and Appendix D).



**Figure 24:** JRFs and actuator forces of models with double-stringed path actuators (4 path points)  
(Actuator force curves of models 05\_doublePath\_ulna\_radius, 06\_doublePath\_radius and 07\_doublePath\_ulna are equivalent.)

In simulated JRFs using the other two control strategies, 05\_doublePath\_ulna\_radius also has higher values than all other exoskeleton models (see Appendix C). This is true for each of the vector components as well. It is the only model, where for the CMC-optimized control the mean JRF increases in comparison to 01\_noExo (by 4.7 %). However, the maximum value is reduced by 15.7 %. For model 05\_doublePath\_ulna\_radius an increase in total JRF at the end of the movement from a rising X-compression coincides

with an increase in actuator force (model 05\_doublePath\_ulna\_radius, see Appendix B). The difference between models 06\_doublePath\_radius and 07\_doublePath\_ulna is even smaller than those with 100% support for the CMC-optimized torque.

For the constant torque, model 05\_doublePath\_ulna\_radius has a relatively high plateau of JRFs until about 25° flexion (see Appendix B). After that it drops to a relatively constant level between 2000 and 2100 N.

### **Complex Double-Path Actuators (Models 9, 10 & 11)**

The difference between the models in this section in comparison to the previous double-stringed ones is that the paths are now made up of ten path points instead of just four (see chapter 4.3.2). The points along the forearm are fixed to either the radius (10\_doublePath\_AtWrist\_onlyRadius), the ulna (11\_doublePath\_AtWrist\_onlyUlna) or the bone body sitting closest to each point (09\_doublePath\_AtWrist). With the same topology and thus actuator moment arms, the prescribed actuator force is equal for all three models (see figure 25).

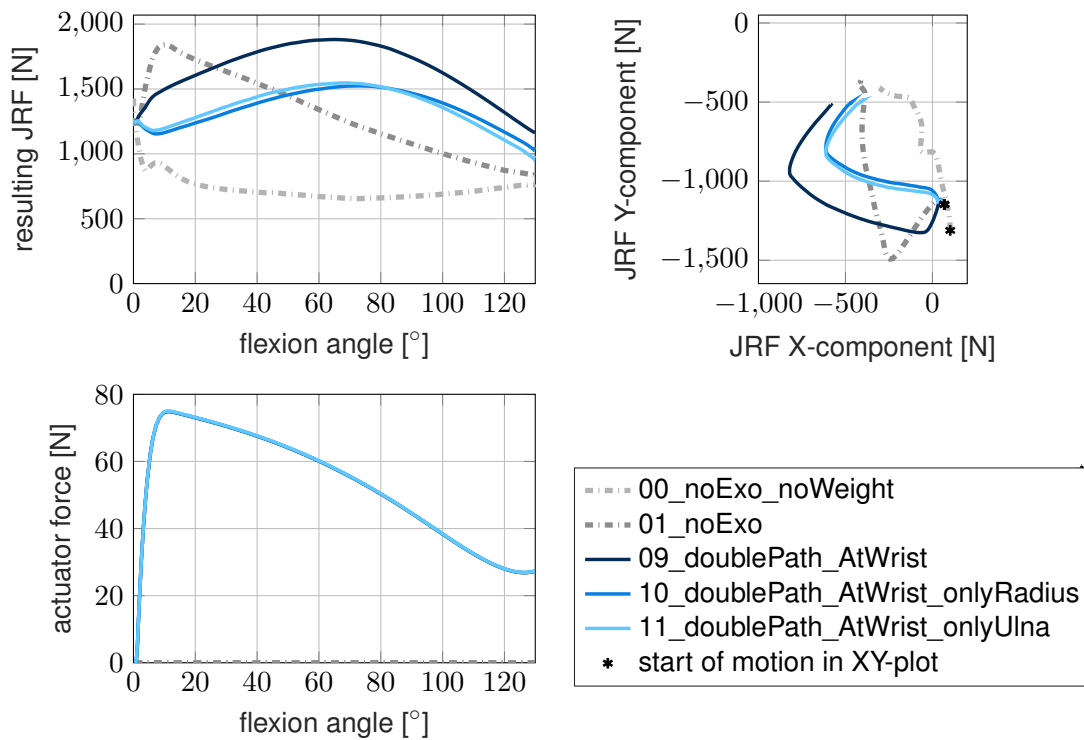
Except for the first approximately 10° of flexion, all models have the concave shape mentioned in chapter 5.2. Their directions also follow the same pattern of X- and Y-compression and appear to be scaled around the origin - except for the first part of the motion, where they are still the same due to zero prescribed force.

The mean values of all 3 models are higher than those reported for 01\_noExo by 22.1% (model 09\_doublePath\_AtWrist), 2.4% (model 10\_doublePath\_AtWrist\_onlyRadius) and 2.1% (for model 11\_doublePath\_AtWrist\_onlyUlna) with the highest mean JRF in model 09\_doublePath\_AtWrist at 1539 N (see Appendix C and Appendix D). The maximum values of models 10\_doublePath\_AtWrist\_onlyRadius (1290 N) and 11\_doublePath\_AtWrist\_onlyUlna (1286 N) are almost identical and comparable to those of model 01\_noExo. Model 09\_doublePath\_AtWrist reaches a maximum JRF of 1881 N, an increase of 2.1 % compared to model 01\_noExo. It is one of only three models that increase the maximum value for the 100%-support strategy. The JRFs of the ulna-attached model are higher until about 70° flexion and are then surpassed by the ones of ulna-attached model 10\_doublePath\_AtWrist\_onlyRadius.

In both other control strategies, model 09\_doublePath\_AtWrist's JRFs is also higher than the two with only one attachment body (models 10\_doublePath\_AtWrist\_onlyRadius and 11\_doublePath\_AtWrist\_onlyUlna) at all times. With the CMC-optimized control, models 10\_doublePath\_AtWrist\_onlyRadius and 11\_doublePath\_AtWrist\_onlyUlna follow almost the same curve. Only towards the end of the movement do they diverge by about 25 N, as the prescribed actuator force increases once more (see Appendix B). In general it can be said that the differences between all three models are smaller when using the CMC-optimized controller compared to the other two.

For the constant torque controller, the JRFs of model 11\_doublePath\_AtWrist\_onlyUlna were above those of model 10\_doublePath\_AtWrist\_onlyRadius until 70° flexion and lower after that, just like for the 100% support variant. In terms of the JRF direction there are no notable particularities.





**Figure 25:** JRFs and actuator forces of models with double-stringed path actuators (10 path points)  
 (The actuator forces of models 09\_doublePath\_AtWrist, 10\_doublePath\_AtWrist\_onlyRadius and 11\_doublePath\_AtWrist\_onlyUlna are equivalent.)

### 5.3.5. Path Actuators of Varying Complexity

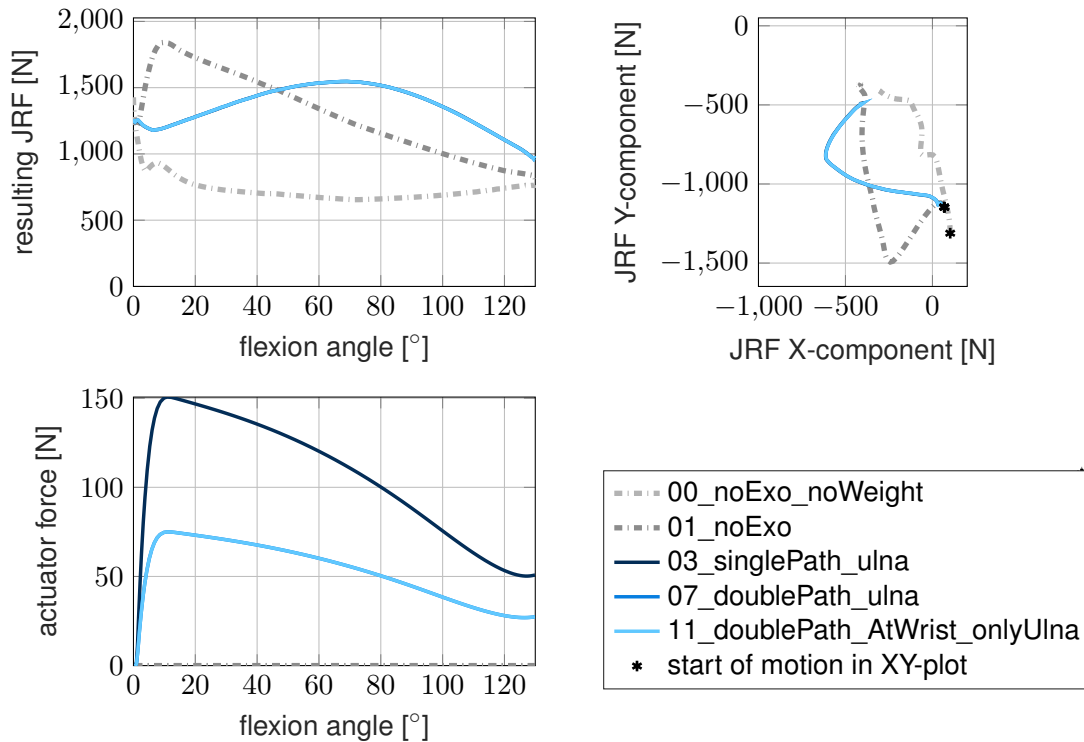
In this section, models with the same path point attachment strategy, but with varying topologies and complexity are compared to one-another.

#### Ulna-Attached (Models 3, 7 & 11)

Figure 26 shows the JRF plots for models with path actuators whose forearm-attached point(s) are fixed in the ulna's reference frame only. This includes the single-stringed path actuator model 03\_singlePath\_ulna and the two double-stringed models 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna, the latter with the path along the forearm represented more in detail than the first. The prescribed actuator forces for models 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna are double that of single-stringed model 03\_singlePath\_ulna due to the previously mentioned loose pulley effect.

The concave shape of the JRF curve observed in preceding sections is almost exactly the same for all three models (see figure 26). The same goes for the X- and Y-components. All share the same mean JRF of 1286 N, an increase of 2.1% in comparison to 01\_noExo. The maximum values are 1547 N for models 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna and 1546 N for model 03\_singlePath\_ulna, which means a decrease of about 16% to model 01\_noExo for all three (see Appendix C and Appendix D). In fact, the curves of models 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna lie directly on top of each other, while the JRFs of model 03\_singlePath\_ulna are off by a

margin of 0.8 N on average and up to 5.8 N compared to the double-stringed variants (see Appendix C). The same general situation is observed in the other control strategies (see Appendix B) following the typical patterns presented in chapter 5.2.



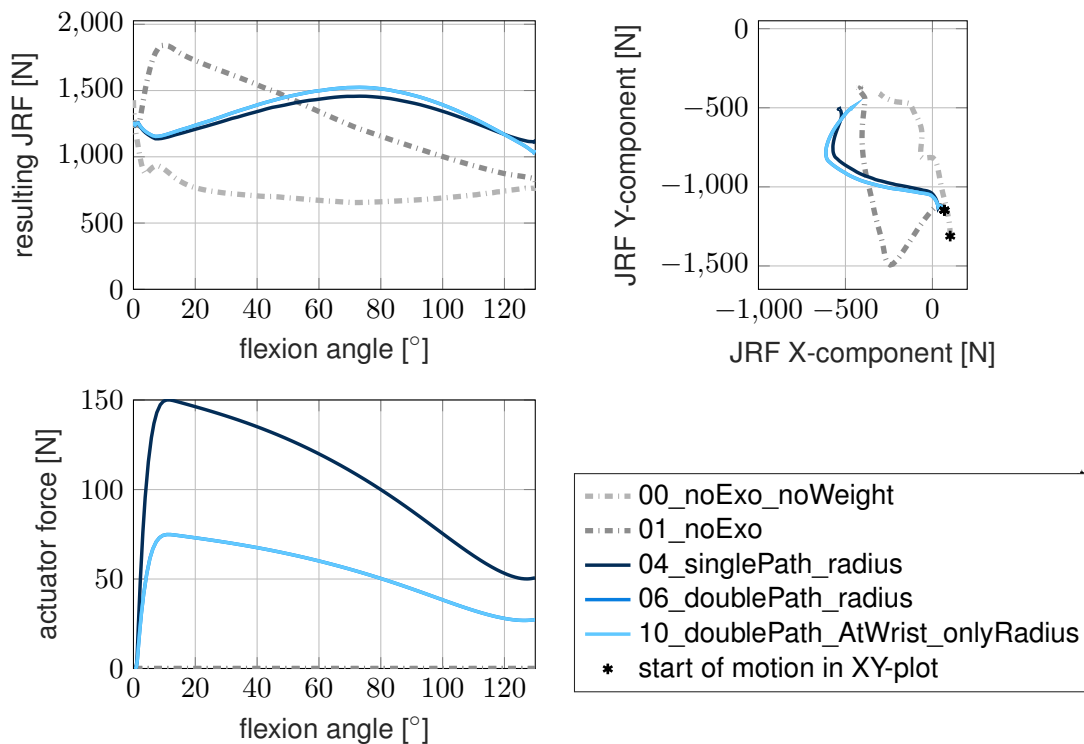
**Figure 26:** JRFs and actuator forces of models with path actuators attached to the ulna (The JRF curves of models 03\_singlePath\_ulna, 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna are (almost) equivalent. The actuator forces of models 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna are equivalent.)

### Radius-Attached (Models 4, 6 & 10)

The models compared in this section all attach to the radius only on the forearm. This includes the single-stringed 04\_singlePath\_radius, as well as the double-stringed models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius. The difference between the latter two is again the number of path points along the forearm - two points for model 06\_doublePath\_radius and eight points for model 10\_doublePath\_AtWrist\_onlyRadius (see chapter 4.3.2). The simulated JRFs can be seen in figures 27 and 19. As with the ulna-attached models in the previous section, the single-stringed path actuator model's prescribed actuator force is double that of the double-stringed ones.

All three models show the previously described concave shape except for the short decrease at the start. model 04\_singlePath\_radius deteriorates from this pattern at the very end of the motion (see figure 27). The double-stringed models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius's JRFs are the exact same for the entire motion. Up until 120° of flexion, the double-stringed models' JRFs are higher than those of model 04\_singlePath\_radius. After that an inflection point in model 04\_singlePath\_radius's curve leads to higher resulting JRF. This coincides with a larger decrease in X-compression in models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius. The

maximum of models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius of 1524 N is lower than that of model 04\_singlePath\_radius with 1457 N. In comparison to 01\_noExo that means a reduction of 17.3% (models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius) and 20.9% (model 04\_singlePath\_radius). The mean values are closer together with 1268 N (model 04\_singlePath\_radius) and 1290 N (both models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius), which means an increase in mean JRF of 0.6% for the single-stringed model and 2.4% for the double-stringed ones in comparison to model 01\_noExo. JRFs are lower than those of 01\_noExo until 48 (models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius) to 52° elbow flexion.



**Figure 27:** JRFs and actuator forces of models with path actuators attached to the radius  
(The JRF and actuator force curves of models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius are equivalent.)

When looking at the results of other control strategies (see Appendix B and Appendix A), one can see that the JRFs of all three radius-attached models are very similar for the CMC-optimized control. Contrary to the observation made for the 100% support control, the JRF of single-stringed model 04\_singlePath\_radius is higher than those of the double-stringed ones until about 85° flexion. After that, the JRF of model 04\_singlePath\_radius rises up 56 N more than its double-stringed counterparts, mainly due to an increase in its X-component. The maximum and mean values of all three models are decreased in comparison to 01\_noExo.

For the constant torque control strategy, model 04\_singlePath\_radius's JRFs are again lower than those of models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius. The mean JRF is about the same as with the 100% support control and model 01\_noExo for all three models, while their

maximum values are higher than those of 01\_noExo resulting from the peak at the start of the motion described in chapter 5.2.

### **Closest-Bone Attachment (Models 5, 8 & 9)**

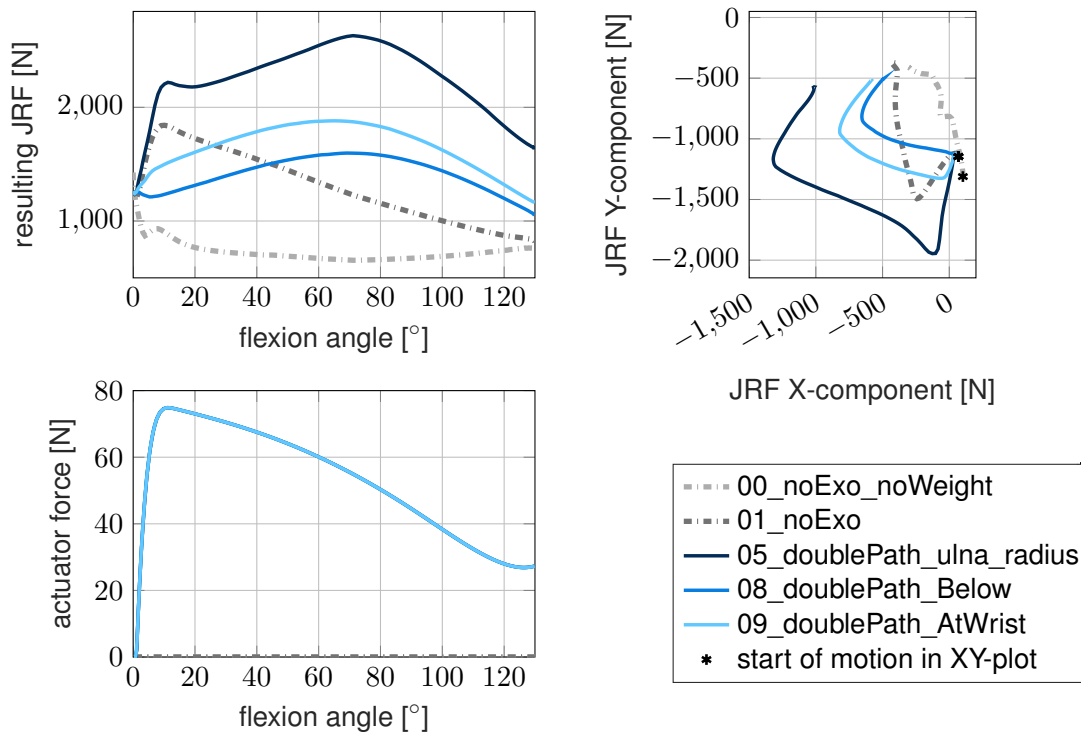
This section compares the JRFs of models attaching to both the radius and ulna in paths of different levels of detail. Model 05\_doublePath\_ulna\_radius has a total of four path points, 08\_doublePath\_Below has two more along the forearm and 09\_doublePath\_AtWrist has a total of ten by adding four more to model 08\_doublePath\_connectedBelow along the forearm (see chapter 4.3.2). A single-stringed model is not included here, as their forearm attachment only comprised one bone. The attachment points along the forearm are fixed to the bone closest to them. In model 05\_doublePath\_ulna\_radius, the bone closest in the frontal plane is chosen. Due to the same moment arms on the elbow flexion coordinate, the prescribed actuator forces are the same for all three models (see figure 28).

Model 09\_doublePath\_AtWrist's JRFs describe the typical concave arc with its maximum around 65° flexion (see figure 28). Model 08\_doublePath\_connectedBelow has a similar shape, but rises at the start of the motion instead of falling. The pattern of model 05\_doublePath\_ulna\_radius's JRFs have a sharp increase at the start. After a small dip, they rise again until 70° flexion and then decline steeper than the two other models. Except for the sharp rise at the beginning of model 05\_doublePath\_ulna\_radius's JRFs, which stems mostly from a fast increase of the Y-component, all three models follow similar patterns for the vector components. The differences between the models' JRFs are largest at their peak values and smallest at the very beginning and towards the end of the motion.

As in chapter 5.3.4, the least detailed of the three (model 05\_doublePath\_ulna\_radius) shows by far the highest JRFs with a mean of 2069 N, an increase of 64.2% compared to 01\_noExo and a maximum value of 2630 N, an increase of 42.8% to reference model 01\_noExo (see Appendix C and Appendix D). Model 08\_doublePath\_connectedBelow has lower JRFs for the entire motion compared to the more detailed model 09\_doublePath\_AtWrist. Their mean values of 1342 N (model 08\_doublePath\_connectedBelow) and 1539 N mean an increase to model 01\_noExo by 6.5% (model 08\_doublePath\_connectedBelow) and 22.1% (model 09\_doublePath\_AtWrist). In the maximum values, model 08\_doublePath\_connectedBelow experiences a decrease of 13.3% and model 09\_doublePath\_AtWrist an increase of 2.1% compared to 01\_noExo.

Using the CMC-optimized control strategy, all three models 05\_doublePath\_ulna\_radius, 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist reduce the maximum JRFs and both 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist the mean value as well compared to 01\_noExo (see Appendix A). Model 05\_doublePath\_ulna\_radius sees a plateau-like JRF until 50° flexion after the first sharp increase (see Appendix B). After that it follows the general pattern observed for this type of controller. A rise at the end stems mostly from the X-component. For the constant torque controller, all three models have a higher mean and maximum value than model 01\_noExo (see Appendix A). Just as with the 100% controller, the JRFs of model 05\_doublePath\_ulna\_radius greatly exceed the other two for the entire motion (see Appendix A). JRFs of models 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist are almost parallel throughout the motion with the difference becoming smaller

towards the end. Model 05\_doublePath\_ulna\_radius however shows two levels, the first one above 2800 N until about 25° flexion and a second, fairly steady one at 2000 to 2100 N. Even here, at its minimum, model 05\_doublePath\_ulna\_radius's JRF is still higher than the maximum of model 01\_noExo.



**Figure 28:** JRFs and actuator forces of models with path actuators attached to the radius and the ulna (The actuator forces of models 05\_doublePath\_ulna\_radius, 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist are equivalent.)

### 5.3.6. Complex Models with(out) Path over Shoulder (Models 9 & 12 and 7 & 13)

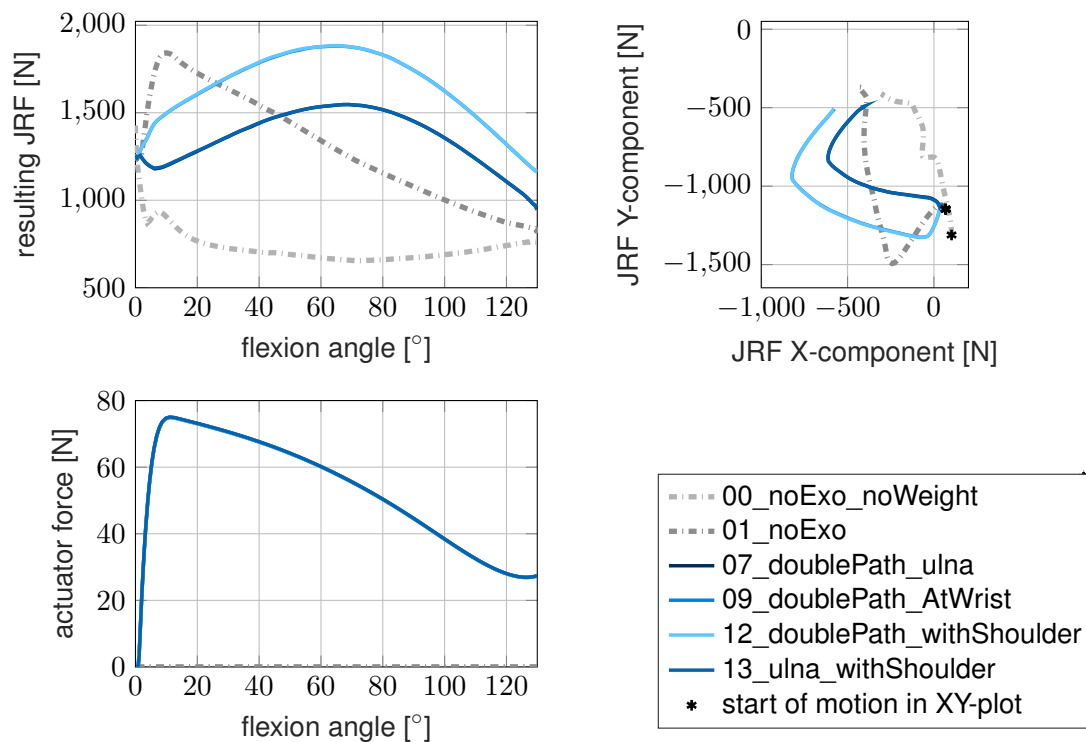
Model 12\_doublePath\_withShoulder is an extended version of model 09\_doublePath\_AtWrist by adding eight path points in the humerus' frame over the shoulder and two more points connecting to the thorax body at the back (see chapter 4.3.2). Model 12\_doublePath\_withShoulder thus marks the most complex path actuator model investigated in this thesis. The shoulder path is also implemented for model 13\_ulna\_withShoulder, which uses the forearm attachment of model 07\_doublePath\_ulna via two path points to the ulna only (see chapter 4.3.2).

The JRFs of both models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder follow the same trajectory (see figure 29) with the largest difference between them being 4.5 N and model 12\_doublePath\_withShoulder having on average about 0.9 N higher total JRF than model 09\_doublePath\_AtWrist. The similarity goes for the vector components as well. Between models 07\_doublePath\_ulna and 13\_ulna\_withShoulder, the largest difference also is 4.5 N and the mean 1.5 N. They follow similar curves as well. Both models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder have a mean JRF value of 1539 N, which means an increase of 22.1% in respect to 01\_noExo (see Appendix C and Appendix D). The maximum value of about 1882 N means an increase by 2.2% compared to model 01\_noExo. For both

models 07\_doublePath\_ulna and 13\_ulna\_withShoulder, the mean JRF is 1287N, an increase of 2.2% compared to model 01\_noExo. The maximum JRFs are reduced by 16% with 1547N (see Appendix C and Appendix D).

In both other control strategies, models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder, as well as 7 and 13 generally also have similar JRF curves and vector components (see Appendix B). For the constant torque controller, the difference between model 09\_doublePath\_AtWrist and 12's JRFs is larger at the beginning of the motion with up to 49N (see Appendix B). After the JRF curve stabilizes at about 35° of elbow flexion, the difference is never larger than 5N. Compared to 01\_noExo, both these models increase the mean and maximum JRF. Models 07\_doublePath\_ulna and 13\_ulna\_withShoulder show no significant differences in JRF for this controller.

The JRFs simulated using the CMC-optimized controller in models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder have less than 1N between them on average. Both models greatly reduce the mean and maximum JRF compared to model 01\_noExo with this controller (see Appendix A). Models 07\_doublePath\_ulna and 13\_ulna\_withShoulder follow a similar curve for most of the motion with this controller, but differ by up to 100N at the very end with higher X-compression in model 07\_doublePath\_ulna (see Appendix B).



**Figure 29:** JRFs and actuator forces of models including path actuators. Models 12 and 13 are extended versions of models 7 and 9 by adding 10 more path points over the shoulder. (The JRFs of models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder are (almost) equivalent. The JRFs of models 07\_doublePath\_ulna and 13\_ulna\_withShoulder are (almost) equivalent. The actuator forces of models 07\_doublePath\_ulna, 09\_doublePath\_AtWrist, 12\_doublePath\_withShoulder and 13 are equivalent.)

## 6. Discussion

The simulation results presented and described in chapter 5 are now discussed. This includes an analysis of the JRFs for different models, as well as the implications of it for creating an elbow exosuit model in musculoskeletal simulations. The simulated forces are compared to values found in the literature and later challenged by explaining limitations to their interpretation.

### 6.1. Comparison/Evaluation with Literature

#### Models without Exoskeleton Support

As most of the literature examples for simulated elbow JRFs do not use an exoskeleton, the model and simulation setup and workflow are evaluated using the simulation results mainly from the model without exoskeleton support.

Compared to the maximum JRF from (Bandmann, 2020) of 500 N, the values from this thesis with model 01\_noExo reach a higher maximum of more than three times as much. The motion and load is similar to the ones from this thesis. However, the underlying musculoskeletal model only has one degree of freedom to stabilize and is mainly intended for educational purposes rather than research-grade simulations. Also, the tool used to calculate the muscle forces is SO, which tends to produce lower JRFs (Roelker et al., 2020).

The elbow JRFs of up to 5 000 N computed by (Buffi et al., 2014) on the other hand are way higher in magnitude and have higher loads during the more vigorous baseball pitch motion. They use similar simulation pipeline and tools as described in chapter 4.1 which shows that the JRFs from this thesis are not improbably high.

The values simulated by (Chadwick & Nicol, 2000) for occupational pick and place activities are in a similar range as the ones from model 01\_noExo presented in chapter 5.3.1 with 1 600 N. The ones by (Sugiarto, 2020) of up to 2 530 N are a bit higher than the JRF values of model 01\_noExo, despite using a previous version of the same musculoskeletal model and OpenSim with the CMC and JRA tools in his simulation of a similar motion. One possible explanation for the diverging results between this thesis and (Sugiarto, 2020)'s - each excluding the exoskeleton support - is the different amount of unlocked degrees of freedom in the MobL musculoskeletal model. With other articulations to be tracked by the CMC algorithm, especially biarticulate muscles acting on the elbow and other joints may be activated differently. About 70 % of muscles acting on the elbow were found to be biarticulate in the MobL model (see Appendix E). The slightly different motion files are another possible contributor to the difference.

Regarding the pattern in the JRFs, (Amis et al., 1980), (Morrey et al., 1988) and (Sugiarto, 2020) all observed a decline over the elbow flexion angle. This is also observed in chapter 5.3.1 and gives confidence in the correctness of the simulated JRFs. For the simulation of (Bandmann, 2020) this decline has only been observed for the first 40° of elbow flexion, which might again be caused by the underlying, relatively simple musculoskeletal model.

## Models with Exoskeleton Support

Only two of the publications listed in chapter 2.5 include an exoskeleton in their elbow JRF simulations: (Bandmann, 2020) and (Sugiarto, 2020).

The values computed by (Bandmann, 2020) depended strongly on the geometry of the parameterized model used. For one model, the peak JRF is as low as 65 N. The model that uses similar attachment points - number "5G" in (Bandmann, 2020) - as model `03_singlePath_ulna` with a single-stringed path actuator produces a mean of 104 N and a maximum of 129 N in resultant force (Bandmann, 2020). Forces this low are not achieved by any of the models in this thesis - not even model `00_noExo_noWeight` without an external weight to lift. As previously stated, the different levels of detail in degrees of freedom that are actuated, as well as the amount of muscles in the models and the simulation workflow itself could be a reason for this. It is unlikely that the control strategies cause this large gap in JRF data, as even the CMC-optimized strategy similar to the one used by (Bandmann, 2020) produces way higher JRFs.

(Sugiarto, 2020) uses a torque actuator to implement the exosuit's support. Compared to the torque actuator's JRFs from this thesis with 100%-support control (similar strategies), the maximum value of (Sugiarto, 2020)'s simulation is higher by 864 N (62.8 %) and the mean value lower by 441 N (37.9 %). The control strategies are not completely equivalent and may be part of the explanation of the difference between (Sugiarto, 2020)'s values and the ones in model `02_coordinateActuator`. However, as the baseline simulations without support already differed, the main reason is likely to be in the different motion and model files used. While a very similar model - the predecessor of `MobL` was used, (Sugiarto, 2020) had only unlocked the elbow flexion coordinate. This can change the forces of biarticulate muscles, which lead to a change in forces when stabilizing other coordinates like the pronation-supination or the shoulder and wrist articulations. In terms of the shape of the JRF curve, the patterns observed with the three different control strategies in this thesis differ from the ones seen in the results by (Sugiarto, 2020). Therein, the JRF closely follow the monotonously decreasing shape seen in the simulations without exoskeleton support, instead of the patterns observed with the three control strategies in this thesis (see chapter 5.2). For the path actuators, this a possible explanation for this are the different exoskeleton modeling approaches. With coordinate actuator model `02_coordinateActuator`, the main difference to (Sugiarto, 2020)'s values might stem from the control strategy applied.

## 6.2. Differences in JRFs

### 6.2.1. Actuator Types

When looking at the JRFs produced by different types of actuators, namely the coordinate and path actuators in chapter 5.3.2, it is apparent that the internal joint forces are lower in the coordinate actuator model. This is actually the case for all path models and for each of the control strategies. One possible explanation could be that the path actuators do not only act with a rotational moment on the elbow joint, but also with translational forces on humerus and forearm that are then transferred through the joint. These can occur in each of the vector components. A second explanation lies in the possibility of path actuators to act on more than one degree of freedom, even when only attached on two points. This would be the case, when two bones are connected that aren't directly connected themselves. For example, when attaching to the humerus and radius, moments will be exerted on the elbow flexion, as well on the



pronation-supination coordinate. For the MobL model this only counts for radius-attached path actuators. In terms of directionality of the internal forces, there are no notable differences when excluding the part, where both actuators are not yet active at the very beginning of the motion (see figure 22). This shows that the additional translational forces from the path actuator model produce JRFs in the same direction as the underlying human muscles do when using the torque actuator. The JRF direction including the elbow exosuit can thus be simulated using a simple torque actuator model, as long as the magnitude is not of interest.

For exosuits like the one analyzed in this thesis, the difference in total JRF values implies that using a torque or coordinate actuator to represent forces acting along cables leads to an underestimation of the JRF magnitude. It lacks the translational forces imposed by a path actuator and in the case of radius-attached actuators does not consider the effect on more than one degree of freedom.

Regarding the difference between torque and coordinate actuator the following can be noted: As the elbow flexion coordinate is implemented as an articulation between only two bones in the MobL model, there is no observable difference between using a coordinate actuator or a torque actuator acting around the ulna's Z-axis (see chapter 4.3.2). For models having humero-ulnar and humero-radial articulations implemented, this assumption will have to be revisited.

### **6.2.2. Single- and Double-Stringed Path Actuators**

When comparing models with single- or double-stringed actuators, two different observations are made for the path actuator models attaching to the ulna and the radius.

The ulna-attached models show the same JRFs throughout the motion regardless of whether the model uses a single- or double-stringed actuator (see chapter 5.3.3). One explanation for this is that the placement of either path point has the same effect not only on the elbow flexion coordinate, but also on other coordinates like the pronation-supination articulating the two attached bones. With the MobL model having only the humero-ulnar and not the humero-radial articulation implemented, the forces applied to the ulna can in fact not affect the pronation-supination, as they are all transferred to the humerus towards the ground body. The equivalent JRFs may however not be observed when the single-stringed actuator's forearm attachment point does not sit between the points of the double-stringed one, as this can lead to the actuators applying forces in other directions, imposing a change in JRF between the two connected bodies.

The radius-attached models, however, show different JRFs over the course of the movement (see chapter 5.3.3). Here, the double-stringed model leads to higher mean and maximum JRF. In all likelihood, the moment on the pronation-supination coordinate plays the main role here. The effect on this coordinate differs with path points placed in positions with different moment arms on it. Here, the path points of the double-stringed path seem to have larger moment arms, leading to a stronger torque on the pronation-supination and resulting in higher muscle forces to counter it. This in turn increases the simulated JRFs compared to the single-stringed model.

Model 04\_singlePath\_radius has the exception in both the 100%-support and the constant torque control to have an increase in X-compression at the end of the motion. This again implies that the implementation of the presented exosuit in a single- or double-stringed path actuator attached to the radius makes a difference not only in the magnitude of produced JRFs, but also in the direction thereof.

Overall, this means that using a single- or double-stringed path actuator can cause a change in simulated JRF's magnitude and direction, depending on the attachment bone. For ulna-attached models, the JRFs stay the same when using the MobL model, due to a lack of the humero-radial articulation. For the radius-attached models, the double-stringed variant produces higher values.

### 6.2.3. Effect of Attachment Point Placement

The JRFs of models with three types of topology are analyzed and compared in chapter 5.3.4. Each of these topologies is implemented with the path actuator points fixed in different bone's coordinate systems, either the ulna's, the radius' or both (see chapter 4.3.2).

In the single-stringed path actuator models, the reductions of mean JRF to the model without exoskeleton (01\_noExo) is a mere 1.5 % better in the radius-attached model compared to the ulna variant. However, the difference in the maximum reduction is more pronounced with 4.8 % lower JRF in the radius-attached model (see chapter 5.3.4). Lower JRFs in the radius-attached models suggest that the additional torque placed on the pronation-supination coordinate helps to counter other torques acting on it.

For the simple and more detailed double-stringed path actuator models, the attachment in the ulna or radius frame has smaller differences in the mean and maximum JRF reductions. The radius-attached model has lower JRFs than the ulna-attached one for the first half of the motion, while the opposite is the case for the second half (see chapters 5.3.4 & 5.3.4). Here, the torque on the pronation-supination coordinate by the radius-attached model seems to be beneficial only for the the first half. For both variants of the double-stringed models attaching to both the ulna and radius with the same actuator (models 05\_doublePath\_ulna\_radius and 09\_doublePath\_AtWrist), the JRFs are increased compared to the models only using one reference frame and the same topology. Model 05\_doublePath\_ulna\_radius has the highest JRFs observed overall, but model 09\_doublePath\_AtWrist also presents a big change. These changes are supposed to arise from strong forces being exerted on the pronation-supination coordinate, as path points apply forces between ulna and radius directly and with a much greater moment arm than the radius-attached models. This force thus has a much stronger effect. This moment on the pronation-supination coordinate is strongest in model 05\_doublePath\_ulna\_radius as it is the one with the largest moment arm on this degree of freedom (see chapter 4.3.2).

Model 09\_doublePath\_AtWrist retains the general curve shape introduced in chapter 5.2, which model 05\_doublePath\_ulna\_radius does not - possibly due to too large forces on the pronation-supination coordinate. For all but these two models, the general shape is followed. The difference within each group lays in the JRF of the ulna-attached models being higher than the radius-attached ones for the first half to two-thirds of the motion. After that, the radius-attached models show higher values, mainly ascribed to a greater X-compression. This is again likely to be linked to the effects on the pronation-supination coordinate. This pattern is observed in both the 100%-support control strategy, as well as for the constant torque and in all three topologies. In the CMC-optimized control, the model's JRFs lie too close together to be able to make these kinds of distinctions with certainty.

All in all it can be said that the attachment of the path actuator points on the forearm plays an important role for simulated JRFs. This is independent of the complexity of the setup and counts for single-, as well as double-stringed actuators. The difference is the biggest when setting the points in both the ulna and

the radius for the same actuator due to the effect on the pronation-supination coordinate. But differences in magnitude and pattern of the JRFs are also observed for models using the ulna's or radius' frame only for the same reason. One issue with path actuators attached to the radius or both forearm bones that did not matter for the motion analyzed in this thesis is that these points will turn along with the radio-ulnar articulation if it is involved in a movement. This will lead to inaccurate representations of the exoskeleton.

#### **6.2.4. Effect of Complexity of Path Along Forearm**

One major difference between the different path actuator models is the amount of path points and thus complexity in the representation of the exosuit's cable. In chapter 5.3.5, the models are grouped by their attachment strategy to only describe the differences in simulated JRFs resulting from the level of detail.

For the ulna-attached models it is apparent that the variants with a single- and two types of double-stringed path actuators produce almost the same JRFs (see chapter 5.3.5). The double-stringed models actually produce the exact same JRFs. For these only very slight differences, it can not be guaranteed, that the differences do not arise from numerical effects in e.g. the generation of the prescribed actuator force or the CMC or JRA simulations themselves. As previously mentioned in chapter 6.2.3, the evanescent differences in the ulna-attached models presumably result from no other coordinates being actuated by them due to the articulations of the MobL model. The equivalent values from the double-stringed models show that the forces between two path points don't have any effect on the forces between bones, as long as they are fixed to the same body. This would only lead to internal compressive or decompressive forces within that body, which is not regarded in OpenSim's multi-body simulation environment.

In the radius-attached models both double-stringed path actuators also bear the exact same JRF values (see chapter 5.3.5). This indicates again that when adding several path points in the same reference frame one after the other, only the ones coming from or going to other bones do in fact matter for the simulated internal joint forces. The single-stringed version, however has higher JRFs for the 100%-support and the constant torque control strategies than the double-stringed versions due to different moment arms on the pronation-supination coordinate, as was explained in chapter 6.2.2.

When attaching the forearm path points of a path actuator to the closest bone, the JRFs of models with a varying amount of path points all display different values (see chapter 5.3.5). This is true for all three control strategies and means that for this type of attachment, the level of path actuator detail plays a role in the resulting JRFs. There is no clear trend linking the magnitude to the complexity with the models presented in chapter 5.3.5. All of these models display mean and maximum JRF values higher than those of path actuators attached to one bone only. The presumed reason for this are the direct forces acting on the radio-ulnar articulation resulting from the attachment on both bodies. This effect was already mentioned in chapter 6.2.3. As this force is not really present in the physical exosuit prototype it is questionable, if this attachment method is even a valid modeling approach.

By far the highest JRFs were encountered in model 05\_doublePath\_ulna\_radius, which also showed a deterioration from the general JRF pattern for each controller. It should be considered though that the two points sitting in the ulna's and radius' frame in this model are located in the radius' frame only for the two following models 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist. They thus have a much larger moment arm on the pronation-supination coordinate, which in turn requires more muscle activation to counter this effect and results in higher JRFs.

For model 06\_doublePath\_radius, however, the point placement is equivalent to the one in models 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist. Comparing the JRFs of models 06\_doublePath\_radius, 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist, one can observe a trend for the more complex models to have higher JRFs (see figure 30). With the prescribed actuator force of all models and thus the moment arm on the elbow flexion coordinate being the same, this is not a plausible explanation for the trend. A possible causal connection is that the two transition points of each model crossing the path from one body to the other are placed in different spots in each of the models. This leads to different moment arms on the pronation-supination coordinate for the path actuator and can thus lead to changes in muscle activations resulting in different JRFs.

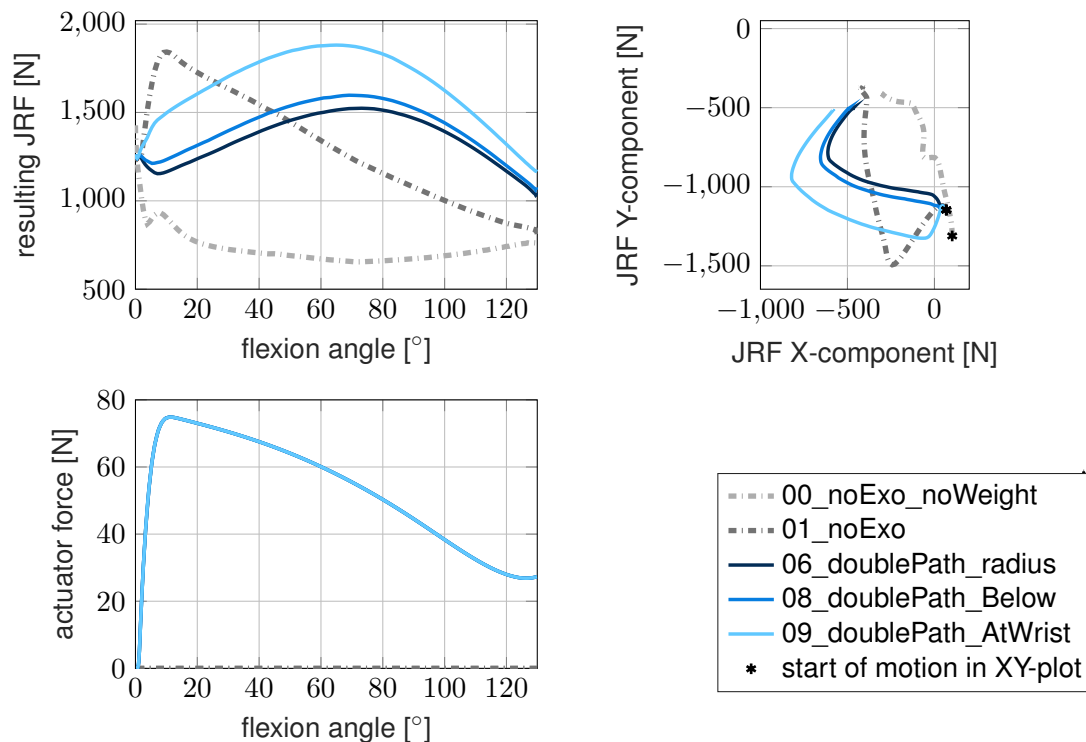
Judging from the three types of attachment, one can infer that the level of detail in a path actuator (measured via the number of path points) can - but does not necessarily - influence the simulated JRFs for the elbow exosuit. Besides the effect of single- and double-stringed path actuator models discussed in chapter 6.2.2, the amount of path points on the forearm only affects the JRFs when points leading to or coming from other bodies are altered. Thus when using a double-stringed path actuator attached to just one forearm body, only the two points coming after and before the humerus attachment points are relevant.

For models including both ulna and radius as fixations, the level of detail in the cable path representation along the forearm influences the simulated JRFs, as long as the path points coming from or leading to the other bone play a role. For this type of attachment, the elbow JRF is especially sensitive to changes, as it directly articulates the radio-ulnar joint, which may need compensation from the muscles.

### **Effect of Path over Shoulder**

To investigate the effect of the cable path reaching over the shoulder within the PTFE tubing, models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder are compared, as well as 7 and 13. Both models of each pair share the same path points except for the extension of the two path ends towards the user's back, where they attach to the torso body (see chapter 4.3.2).

Looking at the simulation results of chapter 5.3.6, it becomes clear that this change does not substantially influence the simulated JRFs for the 100%-support control strategies with maximum deterioration between each of the two models of 5 N. The same thing can not be said about the constant torque control, where the differences between models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder amount up to 49 N towards the start of the motion. At this point, the moment arms change relatively fast and the prescribed actuator force is high, leading to computational errors having a large impact. This might lead to the differences observed. Similarly, some difference is observed between models 07\_doublePath\_ulna and 13\_ulna\_withShoulder for the CMC-optimized control strategy at the very end of the motion. For the rest and the constant torque controller the curves are almost identical. One possible explanation why the implementation of the cable path over the shoulder does not affect the elbow JRFs is that this leads to a support of mainly the shoulder joint elevation coordinate. Within MobL there are only the two biceps muscles involved in the elbow flexion and the shoulder joint as well. However, the biceps is only marginally involved in the shoulder articulation (Antwerpes & Rezaie, 2021). This lack of biarticulate muscles involved in both the elbow and shoulder means that changes in shoulder muscle activations will not result in big changes in elbow JRF.



**Figure 30:** JRFs and actuator forces of models with path actuators attached to the radius and the ulna. A trend towards higher JRFs with increasing number of path points can be observed. (The actuator forces of models 06\_doublePath\_radius, 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist are equivalent.)

Overall it can be said that the additional path points over the shoulder do not influence the elbow JRFs to a relevant degree for most of the motion with the three controllers. There seem to be some minor differences in any case, especially towards the start and end of the motion, likely due to the biceps involvement. The control strategy appears to determine in part how strong they are. When one is interested in the shoulder JRFs as well, it is imperative though to include this part of the actuator.

### 6.2.5. Effect of Actuator Control

For each of the models presented in chapter 4.3.2, the musculoskeletal simulations are performed with three different control strategies and a derived prescribed actuator force. This is done to verify that JRF differences between models were not merely present in one of them, but independent of it.

The JRF curves follow similarly shaped trajectories for almost all models with each of the controls. It can thus be said that the control strategy plays a major role for their pattern over the movement and also their general magnitude (see chapter 5.2). The magnitude similarities manifest in similar minimum, maximum and mean JRFs across the 14 models compared to the simulation results of other control strategies (see chapter 5.1 & boxplots in Appendix A). From this general type of JRF curve, each model then has some specific changes that are discussed in the previous sections, but resemble the default shape. Model 05\_doublePath\_ulna\_radius proves to be an outlier in this respect due to the effect on the pronation-supination coordinate discussed in chapters 6.2.3 and 6.2.4.

The control strategy mainly analyzed in this thesis, providing 100%-support, decreases the maximum JRF values compared to not using an exosuit actuator and doesn't alter the mean values of most models by much. This means that the relief provided by using this strategy has mainly an effect on the maximum JRF experienced by the elbow.

The CMC-optimized controller, however, manages to strongly reduce the internal joint loads in terms of the maximum and mean values and their vector components in most models, in the course providing real relief for the elbow joint's tissues throughout all models compared to model 01\_noExo (except model 05\_doublePath\_ulna\_radius).

The same can not be said about the constant torque controller. Due to the large spike in JRF at the start of the motion for all models (see chapter 5.2, the maximum value is higher in comparison to the model without exoskeleton (see Appendix A). The mean values stay about the same as in model 01\_noExo. It is thus questionable, if an exoskeleton with this type of controller would even provide any relief to the elbow joint for the user in the analyzed motion. For the torque actuator model 02\_coordinateActuator, however maximum and mean JRF are reduced and can thus give the impression of improvement.

Looking at the overall picture of different models' simulated JRFs with the three control strategies, one can observe that the difference between the models are smallest with the CMC-optimized controller. The differences between the models' values are bigger when using the 100%-support and constant torque strategies. One explanation could be that the prescribed actuator forces are bigger with these two (see figure 20 in chapter 5.2). The bigger forces could in turn increase unwanted forces arising from the exoskeleton actuator, which would have to be compensated for by the muscles and thus increase the JRFs.

Having the lowest JRFs coincide with the control strategy with the lowest actuator force could lead to the assumption that a reduced exoskeleton force generally provides more JRF relief. However, a model with no actuator force would most likely produce the same JRF as model 01\_noExo. Thus, one could assume that there is an optimum control strategy and actuator force for the presented elbow exosuit in terms of minimum joint reaction forces somewhere between that of the 100%-support controller and no exosuit at all.

The comparison of models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder shows the importance of having simulated the JRFs for three different control strategies, as there are peculiarities that only appear in one of them. However, it is not ultimately clear, why the simulation results only differ by a relevant amount for the constant torque controller.

### 6.3. Implications for Musculoskeletal Modeling of the Elbow Exosuit

As stated before, a model should always match its purpose. The musculoskeletal modeling of exoskeletons especially with regard to the simulation of JRFs is usually used to improve the physical design of an exosuit or to develop a control strategy and algorithm to operate it. The JRFs therein serve as an indicator of the strain that is put on the user, which is to be minimized.

Different representations of the soft, cable-driven exosuit from LfE presented in chapter 2.3.2 in such a

simulation are implemented, categorized and tested in this thesis. Almost all of the identified representation parameters like the actuator type, the attachment to the forearm bones, single- or double-stringed path actuators, the overall complexity and level of detail, as well as the control strategy influence the results in some way.

Some examples from the literature used simple torque actuators as a representation of exoskeletons to design and optimize their control. In the simulation results from this thesis, the control strategy with the lowest JRFs could be chosen according to the results obtained using such a coordinate actuator alone. However, the magnitude of the JRFs would likely be underestimated, as all other path actuator exosuit models produce higher values due to the reasons presented in chapter 6.2.1. This would be especially important when comparing the values to a model without exoskeleton, like `01_noExo`, as the torque actuator model with the 100%-support strategy would suggest a JRF relief, where other models say otherwise.

If the exact values are not of interest and only a relative comparison of control strategies is sought for, this a simplified exosuit representation via a torque actuator can be utilized. It also does not seem to underestimate the JRFs in extreme positions any more than for the rest of the motion.

Especially when the magnitude of simulated JRFs is of interest, path actuator models should be chosen to represent the cable-driven exosuit. They all show higher internal forces than the torque actuator model. However, the present data does not allow for judgment, which of the simulated models' JRF values match the reality best. For this, a thorough validation via experiments is required, e.g. similar to the ones described in chapter 2.2.2.

What can be said is that one should always be aware that parameters like the ones stated above can influence the simulated JRFs. Almost identical JRFs in single- and double-stringed, ulna-attached models `03_singlePath_ulna`, `07_doublePath_ulna` and `11_doublePath_AtWrist_onlyUlna` are one example, where changing a parameter have no influence due to the same resulting force on the forearm through the ulna. This is supposed to stem from the articulations implemented in the underlying MobL musculoskeletal model (see chapter 6.2.3). However, for the radius-attached models, different detail in the cable representation does make a difference in elbow JRFs, as additional forces are imposed by the change in moment arms on the pronation-supination coordinate (see chapter 6.2.3).

When representing the LfE elbow exosuit prototype by (Harbauer et al., 2020) in a musculoskeletal simulation of JRFs, the following recommendations can be made according to the insights gained in this thesis: A single-stringed path actuator that attaches to the ulna only can be used for pure flexion motions, as it produces the same results as more complex double-stringed ulna-attached actuators (see chapter 6.2.2). The ulna seems suitable here mainly as the path points attached to the radius will rotate with pronation-supination movement. Ulna-attachment also doesn't lead to forces being passed through the radio-ulnar articulation due to the elbow joint implementation in the MobL musculoskeletal model, which would be the case with a radius-attachment. For more complex movements articulating more of the model's degrees of freedom than the elbow flexion, it is possible that the double-stringed, ulna-attached path-actuator produces different results, as the directions of the cables can change. Only the two path points for which the path is coming from and going back to the another bone like the humerus are then necessary, requiring

only a simple 4-point actuator. Attaching the path points to both ulna and radius in the forearm seems to lead to an overestimation of elbow JRFs for the investigated elbow exosuit. They directly exert a moment on the pronation-supination coordinate, that has to be countered by the model's muscles. This moment, however, is not present in the physical prototype.

The cable path over the shoulder is less relevant when assessing the elbow JRFs with the MobL model. When evaluating the shoulder JRFs, however, the path should be included.

All these considerations lead to the conclusion that the double-stringed, ulna-attached model `07_doublePath_ulna` is an appropriate model to compute JRFs with the simulation setup presented in this thesis.

## 6.4. Alternative Modeling Approaches

The 12 models created for this thesis are not the only ways to implement the LfE elbow exosuit prototype in musculoskeletal simulations and are created according to what is deemed suitable for the current prototype.

One aspect that could be implemented differently when modeling the exosuit is the actuator attachment to the MobL musculoskeletal model. Path actuators like the ones used in 11 of the exosuit models could attach not directly to the bones, but rather to an additional body similar to the brace that is part of the exosuit or even parts of the textile structure. This body can then be linked to the underlying bones via different connections. Preliminary tests showed that attaching a mass-less external body completely fixed to a bone would not make a difference, as long as the path points have the same positioning as when fixed to the bone directly.

Connections representing relative motion to the bones via a movement over or together with the user's skin could make a difference in the simulated JRFs. OpenSim offers the option of joints behaving like a slider or articulations with completely customizable translational and rotational degrees of freedom (coordinates) (OpenSim, 2021k). The forces limiting these coordinates could then be represented via e.g. passive springs or other OpenSim actuators like coordinate limiting forces (OpenSim, 2021j). The studies required to find the parameters to properly represent the behavior of the prototype with this setup would, however, have exceeded the scope of this thesis and are opportunities for further research.

The modeling approach in this thesis is focused solely on the implementation and implications of the active contribution of the exosuit's elbow support. But when wearing such a system, it is likely that different passive forces will affect the musculoskeletal system as well. These passive forces could arise from e.g. the pressure exerted on bones, joints and tissue in extreme positions. For example, the textile sleeve can apply pressure on the arm when in full elbow flexion like a tight fitting shirt would. In a similar fashion, the friction between skin and textile, as well as between cable and tubing could be modeled as passive forces that affect the muscle forces and thus the JRFs computed in the simulations.

Another part that is not modeled due to uncertain parameters is the support provided to the user from the exosuit's wrist brace, which provides passive forces stabilizing the joint when flexed or extended.



## 6.5. Limitations to the Results and their Interpretation

There are some limitations to the results and interpretations presented and discussed in this thesis, which should be considered when working with them.

In the workflow of simulating the JRFs, the OpenSim tools CMC and JRA are used. The results stemming from JRA follow simple mechanical laws and should thus match the expected values for the given muscle forces. CMC uses an optimization algorithm to calculate these forces in the first place and it is not guaranteed that these forces are precisely the ones that a user's body would generate. However, this type of analysis provides reasonable estimations.

The motion file used here is created artificially and simplified in the sense that it only consists of movement in one degree of freedom. Practical application scenarios for the presented exosuit most likely include other articulations as well. This would very likely lead to larger differences in the models, as the radius-attached path points would be rotating along with the bone in pronation-supination movements. These models might then not even be reasonable representations of the exoskeleton for these types of movement anymore (see chapter 6.2.3). Also, articulation of the shoulder joint may lead to larger differences between models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder due to the biarticularity property of the biceps muscles (see chapter 6.2.4).

Regarding the underlying musculoskeletal model, the sophisticated, state-of-the-art and research-grade MobL upper extremity model is used in this thesis. One disadvantage of this model when investigating the elbow JRFs is that only two of the three elbow articulations are included. The humero-radial one is missing and thus all forces transmitted through the humerus pass through the humero-ulnar contact. They can thus only be interpreted as forces passing from the upper to the forearm and not as plain humero-ulnar. Missing the radio-humeral articulation and with the hand only attaching to the radius, this also means that all forces coming from the hand and wrist will have to pass the radio-ulnar joint, which is most likely not a reasonable assumption to make. Altogether this provides the opportunity of evaluating not the forces in the three individual articulations, but rather all forces traversing the elbow joint as a whole in one JRF with its three vector components. Additionally, the model represents a 50<sup>th</sup> percentile male and the force magnitudes thus only fit a user of these proportions.

The weight introduced to the model would realistically have to be held by the user's hand. In this thesis it is merely fixed to the hand's reference frame. With many biarticulate muscles, the normally required gripping force would most likely affect the subsequent wrist and elbow JRF as well. However, this effect is not considered in this thesis. A possible weakness of the exosuit models lies in the attachment of to the musculoskeletal MobL model. The real prototype's cable path is not precisely fixed to the bones' reference frames, but rather has some degrees of freedom by moving with and over the soft tissue in between. However, by assuming little movement in these directions, the fixed attachments can still be considered a reasonable approach. Other possible modeling strategies are discussed in chapter 6.4.

Regarding the control strategy, it should be kept in mind that this simulation setup was entirely artificial and the motion to be supported was known beforehand. The control strategies implemented in real exosuits

most likely differ from the ones presented in chapter 4.3.1 as they need to dynamically adjust to the user's intent (see chapter 2.1.2). With the control strategy as a main influencing factor for change in elbow JRF, this inevitably changes the values and shape of the observed forces.

Taking all these considerations into account, it is important to keep in mind that the results presented in chapter 5 are simulation results, not actual measurements. In order to validate them, they need to be compared to actual measurements. The investigation is aimed at the effect of different modeling strategies for an exosuit in musculoskeletal simulations on the JRFs, not at producing the most accurate forces. For this purpose, the established methodology described in chapter 3, which makes use of the state-of-the-art musculoskeletal simulation tools CMC and JRA combined with a research-grade human model, is deemed suitable. The comparison to the values found in the literature in chapter 6.1 shows that the values calculated are in the proper range that can be expected for the presented setup.

## 7. Conclusion

### 7.1. Summary

This thesis' goal is to assess the effect the modeling strategy of a soft, cable driven exosuit in musculoskeletal simulation has on the computed joint reaction forces.

The effect is investigated by simulating JRFs in a predefined procedure for different implementations of the same exoskeleton, a prototype exosuit by LfE. Parameters influencing the results are identified and a simulation setup established to guarantee the same conditions for all models. This setup consists of a research-grade musculoskeletal model, that is analyzed using the Computed Muscle Control and Joint Reaction Analysis tools of OpenSim. An artificial motion file is created that mimics lifting a 5 kg weight from full elbow extension to 130° flexion.

The different representations of the exosuit comprise of torque and path actuators, the latter in ten variants with different fixation methods and topologies. Each of these is intended to match the real design of the LfE prototype to be modeled. Additionally, two models without any exoskeleton support are simulated to serve as a reference for the other models, as well as to evaluate the simulation pipeline with elbow JRF values from the literature. During the movement each exoskeleton model is supposed to provide the same support, which is defined as the torque on the elbow flexion coordinate. This is implemented by calculating the force or torque for the exoskeleton actuator in advance and providing it as a given input to the simulation. The actuators' predefined support is controlled by three different control strategies: one providing full support of the elbow torque from the 5 kg weight, one CMC-optimized control for low muscle activation and one providing constant torque.

Finally, the processes of creating the prescribed actuator files, as well as running the OpenSim tools as a batch process is implemented in MATLAB including a range of subfunctions.

The simulations are performed for all of the 12 different exosuit models with each of the three control strategies. The JRFs of different models are then compared in different groups to show the effect of varying exosuit modeling parameters like the actuator type or single- and double-stringed path actuators. Other categories are different attachment point strategies on the forearm for the same topology, the same attachment strategy with varying topology and models with or without inclusion of a path over the user's shoulder.

The key findings emerging from these comparisons are presented in the subsequent section 7.2. A comparison to the findings of other research and limitations to the results from this thesis are presented in chapters 6.1 and 6.5. Also, the implications for modeling a two-stringed elbow exosuit in musculoskeletal simulations are presented in chapter 6.3.

## 7.2. Key Findings

The simulation results show that the actuator type has an influence on the simulated JRFs. Using a coordinate / torque actuator, they are lower than any of the path actuator models and one will thus most likely underestimate the forces present in the elbow. The general pattern stays the same.

Using single- or double-stringed path actuators may - but won't necessarily - affect the JRFs. Whether or not they differ is connected to the attachment strategy and positioning. The ulna-attached models have the same values, while the radius-attached ones differ from each other. The elbow articulation setup in the MobL model is in part responsible for this property.

With two bones in the forearm, path points can be fixed to either the ulna, the radius or a combination of both. Either strategy produces different JRFs. Different attachments to the ulna lead to equivalent elbow JRFs. Attachment to the radius produces values of similar magnitudes, but with slightly different patterns. Combining the two shows the highest JRFs with large discrepancies compared to other attachments. However, it is questionable whether this attachment strategy is justifiable or not.

When comparing models with the same attachment strategy, but different levels of detail in the representation of the cable path, changes in JRF are possible, but not forcibly present, depending on the attachment strategy. For attachments to one bone only, the path points leading to and from other bones (e.g. the humerus) are relevant. For the closest-bone attachment large changes can result from different levels of detail in the path along the forearm.

The cable path over the shoulder does not seem to have a large influence on the simulated JRFs for the presented setup. However, it is likely that the shoulder JRFs are affected.

Overall it can be said that with the torque actuator chosen in many musculoskeletal simulations, the general JRF patterns and directions in the XY-plane are most likely correct. However, the magnitudes are likely to be underestimated when modeling cable-actuated exosuits.

The shape of the JRF curve in general is mostly directed by the implemented control strategy and the musculoskeletal model used. Except for model 05\_doublePath\_ulna\_radius as the main outlier due to unrealistic forces on the radio-ulnar articulation, the 12 different implemented exosuit models then result in minor changes only. For future JRF simulations with the MobL model, usage of either the single- or double-stringed path actuator attached to the ulna only is recommended.

## 7.3. Further Research Opportunities

With the results gathered in this thesis it is not yet possible to determine which exosuit model produces the most accurate JRFs that can be experienced for a real user. To do this, in vitro or in vivo measurements similar to the ones presented in chapter 2.2.2 are required. The comparison to the internal elbow forces from the literature lead to the conclusion that the established simulation setup and process is able to compute JRFs with a reasonable range of magnitude. The precise pattern and size of the forces can not be verified though due to a lack of research with experimental elbow JRF data.

For now, this thesis gives insights into how different exosuit modeling approaches and parameters in musculoskeletal simulations can influence the simulated JRFs. The findings from this thesis can already help improving the evaluation of exoskeleton designs and control strategies. The exoskeleton or exosuit models should be created with the implications mentioned in chapter 6.3 in mind. Once it is established which model produces the most accurate JRF values via measurements, this information can then be used to have more confidence in the applicability of design and control optimizations using the presented workflow and modeling strategies.

Also, the simulation process established in this thesis can be used to design and evaluate control strategies that will minimize the elbow JRFs and thus relieve the strain on the user. An optimization of the physical design and control algorithm of the LfE elbow exosuit prototype for a next generation using the recommended modeling strategy and established simulation workflow poses itself as an ideal use of this thesis' findings.

Another research opportunity lies in further analysis and/or development of the models produced in this thesis. For example, more realistic motion data can be recorded to identify differences not observed for pure flexion. One detail in the models that should be examined further is the interface between the musculoskeletal model and the exoskeleton. In this thesis, the attachments are fixed to one bone's coordinate system, whereas this does not necessarily match the real conditions sufficiently. Other ways of attachment such as interfaces allowing for rotational and/or translational relative motion are discussed in chapter 6.4. They could lead to different JRF results, especially for more complex movements.

One more possible area of research lies in the investigation of shoulder JRFs with the LfE exosuit prototype in musculoskeletal simulations. The same or a similar workflow as in this thesis can be used for the investigation. The effect of the cable path reaching over the shoulder included in models 12\_-doublePath\_withShoulder and 13\_ulna\_withShoulder can then be researched in detail.

## References

- AirSlate Legal Forms, Inc. (2021). *50th-percentile adult male law and legal definition*. Website, Retrieved on 28 October 2021. Retrieved from <https://definitions.uslegal.com/5/50th-percentile-adult-male/>
- Agarwal, P., Fox, J., Yun, Y., O'Malley, M. K., & Deshpande, A. D. (2015). An index finger exoskeleton with series elastic actuation for rehabilitation: Design, control and performance characterization. *The International Journal of Robotics Research*, *34*(14), 1747–1772. doi: 10.1177/0278364915598388
- Agarwal, P., Kuo, P.-H., Neptune, R. R., & Deshpande, A. D. (2013). A novel framework for virtual prototyping of rehabilitation exoskeletons. In *2013 IEEE 13th international conference on rehabilitation robotics (ICORR)*. IEEE. doi: 10.1109/icorr.2013.6650382
- Al-Fahaam, H., Davis, S., & Nefti-Meziani, S. (2018, jan). The design and mathematical modelling of novel extensor bending pneumatic artificial muscles (EBPAMs) for soft exoskeletons. *Robotics and Autonomous Systems*, *99*, 63–74. doi: 10.1016/j.robot.2017.10.010
- Amis, A., Dowson, D., & Wright, V. (1980). Elbow joint force predictions for some strenuous isometric actions. *Journal of Biomechanics*, *13*(9), 765–775. doi: 10.1016/0021-9290(80)90238-9
- Antwerpes, D. F., & Rezaie, H. (2021). *Musculus biceps brachii*. DocCheck Flexikon Website, Retrieved on 04 December 2021. Retrieved from [https://flexikon.doccheck.com/de/Musculus\\_biceps\\_brachii#Wirkung\\_auf\\_das\\_Schultergelenk](https://flexikon.doccheck.com/de/Musculus_biceps_brachii#Wirkung_auf_das_Schultergelenk)
- AnyBody Technology A/S. (2021). *anybodytech.com: Software*. Website, Retrieved on 27 May 2021. Retrieved from <https://www.anybodytech.com/software/>
- Apreleva, M., Parsons, I., Warner, J. J., Fu, F. H., & Woo, S. L.-Y. (2000). Experimental investigation of reaction forces at the glenohumeral joint during active abduction. *Journal of Shoulder and Elbow Surgery*, *9*(5), 409–417. doi: 10.1067/mse.2000.106321
- Asbeck, A., Rossi, S. D., Galiana, I., Ye, D., & Walsh, C. (2014). Stronger, smarter, softer: Nextgeneration wearable robots. *Robotics & Automation Magazine*, *21*(4), 22-33.
- Bae, J. (2013). *Modeling, evaluation and control optimization of exosuit with opensim*. Website, Retrieved on 03 May 2021. Retrieved from [https://simtk-confluence.stanford.edu/display/OpenSim/Modeling ,Evaluation,andControlOptimizationofExosuitwithOpenSim](https://simtk-confluence.stanford.edu/display/OpenSim/Modeling,+Evaluation,andControlOptimizationofExosuitwithOpenSim)
- Baltzopoulos, B., & McErlain-Naylor, S. (2020). *Inverse dynamics, joint reaction forces, and loading - bill baltzopoulos*. Lecture (YouTube Video), Retrieved on 30 April 2020. Retrieved from <https://www.youtube.com/watch?v=b0dX-hS1mUY>
- Bandmann, C. E. M. (2020). *Analysis and design optimisation approach for a wearable cable-driven elbow exoskeleton using biomechanical simulation* (Masters Thesis). Technische Universität München.
- Bergmann, G., Graichen, F., Bender, A., Rohlmann, A., Halder, A., Beier, A., & Westerhoff, P. (2011, may). In vivo gleno-humeral joint loads during forward flexion and abduction. *Journal of Biomechanics*, *44*(8), 1543–1552. doi: 10.1016/j.jbiomech.2011.02.142
- Blache, Y., & Begon, M. (2018, apr). Influence of shoulder kinematic estimate on joint and muscle mechanics predicted by musculoskeletal model. *IEEE Transactions on Biomedical Engineering*, *65*(4), 715–722. doi: 10.1109/tbme.2017.2716186
- BoB Biomechanics Ltd. (2021). *The biomechanics of bodies (bob) software*. Website, Retrieved on 03

May 2021. Retrieved from <https://www.bob-biomechanics.com/>

- Buffi, J. H., Werner, K., Kepple, T., & Murray, W. M. (2014, oct). Computing muscle, ligament, and osseous contributions to the elbow varus moment during baseball pitching. *Annals of Biomedical Engineering*, 43(2), 404–415. doi: 10.1007/s10439-014-1144-z
- Cardona, M., Solanki, V. K., & Cena, C. E. G. (2020). *Exoskeleton robots for rehabilitation and healthcare devices*. Springer Singapore. doi: 10.1007/978-981-15-4732-4
- Celli, A., Celli, L., & Morrey, B. F. (Eds.). (2010). *Treatment of elbow lesions*. Springer Milan. Retrieved from [https://www.ebook.de/de/product/14528786/treatment\\_of\\_elbow\\_lesions.html](https://www.ebook.de/de/product/14528786/treatment_of_elbow_lesions.html)
- Chadwick, E., & Nicol, A. (2000). Elbow and wrist joint contact forces during occupational pick and place activities. *Journal of Biomechanics*, 33(5), 591–600. doi: 10.1016/s0021-9290(99)00184-0
- Copaci, D., Cano, E., Moreno, L., & Blanco, D. (2017). New design of a soft robotics wearable elbow exoskeleton based on shape memory alloy wire actuators. *Applied Bionics and Biomechanics*, 2017, 1–11. doi: 10.1155/2017/1605101
- DARPA. (2021). *Defense advanced research projects agency - warrior web (archived)*. Website, Retrieved on 27 April 2021. Retrieved from <https://www.darpa.mil/program/warrior-web>
- de Kruif, B. J., Schmidhauser, E., Stadler, K. S., & O'Sullivan, L. W. (2017). Simulation architecture for modelling interaction between user and elbow-articulated exoskeleton. *Journal of Bionic Engineering*, 14(4), 706–715. doi: 10.1016/s1672-6529(16)60437-7
- Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., . . . Thelen, D. G. (2007, nov). OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11), 1940–1950. doi: 10.1109/tbme.2007.901024
- DeMers, M. (2011). *Estimating joint loads in opensim*. Webinar, Retrieved on 11 May 2021. Retrieved from [https://opensim.stanford.edu/support/event\\_details.php?id=13&title=Webinar-Estimating-Joint-Loads-in-OpenSim](https://opensim.stanford.edu/support/event_details.php?id=13&title=Webinar-Estimating-Joint-Loads-in-OpenSim) & <https://www.youtube.com/watch?v=ly4rhOOifOO>
- DGUV, D. G. U. e. (2019). *Fbhl-006 einatz von exoskeletonen an gewerblichen arbeitsplätze*. Industry Guideline. Retrieved from <https://publikationen.dguv.de/widgets/pdf/download/article/3579>
- DiGiovine, N. M., Jobe, F. W., Pink, M., & Perry, J. (1992, jan). An electromyographic analysis of the upper extremity in pitching. *Journal of Shoulder and Elbow Surgery*, 1(1), 15–25. doi: 10.1016/s1058-2746(09)80011-6
- EduExo. (2017). *Exoskeleton history: A brief history of robotic exoskeletons*. Website, Retrieved on 05 March 2021. Retrieved from <https://www.eduexo.com/resources/articles/exoskeleton-history/>
- Edwards, W. B. (2018). Modeling overuse injuries in sport as a mechanical fatigue phenomenon. *Exercise and Sport Sciences Reviews*, 46(4), 224-231. doi: 10.1249/jes.0000000000000163
- FRA. (2008). *Passenger equipment safety standards subpart a - general section 238.5*. US Regulatory Information - Code of Federal Regulations. Retrieved from <https://www.govinfo.gov/app/details/CFR-2011-title49-vol4/CFR-2011-title49-vol4-sec238-5>
- Fregly, B. J., Besier, T. F., Lloyd, D. G., Delp, S. L., Banks, S. A., Pandy, M. G., & D'Lima, D. D. (2011, dec). Grand challenge competition to predict in vivo knee loads. *Journal of Orthopaedic Research*, 30(4), 503–513. doi: 10.1002/jor.22023
- Gallagher, S., & Schall Jr., M. C. (2016). Musculoskeletal disorders as a fatigue failure process: evidence, implications and research needs. *Ergonomics*, 60(2), 255–269. doi: 10.1080/00140139.2016.1208848

- General Electric. (2016). *Do you even lift, bro? hardiman was ge's muscular take on the human-machine interface*. Website, Retrieved on 14 September 2021. Retrieved from <https://www.ge.com/news/reports/do-you-even-lift-bro-hardiman-and-the-human-machine-interface>
- Gonzalez-Mendoza, A., Lopez-Gutierrez, R., Perez-SanPablo, A. I., Salazar-Cruz, S., Quinones-Uriostegui, I., Tho, M.-C. H. B., & Dao, T.-T. (2019, sep). Upper limb musculoskeletal modeling for human-exoskeleton interaction. In *2019 16th international conference on electrical engineering, computing science and automatic control (CCE)*. IEEE. doi: 10.1109/iceee.2019.8884537
- Grand View Research, I. (2021). *Exoskeleton market size, share & trends report - report overview*. Website, Retrieved on 25 November 2021. Retrieved from <https://www.grandviewresearch.com/industry-analysis/exoskeleton-market>
- Gull, M. A., Bai, S., & Bak, T. (2020). A review on design of upper limb exoskeletons. *Robotics*, 9(1), 16. doi: 10.3390/robotics9010016
- Harbauer, C. M., Fleischer, M., Nguyen, T., Bos, F., & Bengler, K. (2020). Too close to comfort? a new approach of designing a soft cable-driven exoskeleton for lifting tasks under ergonomic aspects. In *2020 3rd international conference on intelligent robotic and control engineering (IRCE)*. IEEE. doi: 10.1109/irce50905.2020.9199238
- Harvard Biodesign Lab. (2021). *Soft exosuits*. Website, Retrieved on 28 April 2021. Retrieved from <https://biodesign.seas.harvard.edu/soft-exosuits>
- Holzbour, K. R. S., Murray, W. M., & Delp, S. L. (2005, jun). A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control. *Annals of Biomedical Engineering*, 33(6), 829–840. doi: 10.1007/s10439-005-3320-7
- Islam, M. R., Spiewak, C., Rahman, M. H., & Fareh, R. (2017). A brief review on robotic exoskeletons for upper extremity rehabilitation to find the gap between research porotype and commercial type. *Advances in Robotics & Automation*, 06(03). doi: 10.4172/2168-9695.1000177
- Lessard, S., Pansodtee, P., Robbins, A., Baltaxe-Admony, L. B., Trombadore, J. M., Teodorescu, M., ... Kurniawan, S. (2017, jul). CRUX: A compliant robotic upper-extremity exosuit for lightweight, portable, multi-joint muscular augmentation. IEEE. doi: 10.1109/icorr.2017.8009482
- Mao, Y., & Agrawal, S. K. (2010). Wearable cable-driven upper arm exoskeleton - motion with transmitted joint force and moment minimization. *2010 IEEE International Conference on Robotics and Automation*. doi: <https://doi.org/10.1109/ROBOT.2010.5509823>
- Masarati, P., Morandini, M., & Mantegazza, P. (2014, jul). An efficient formulation for general-purpose multibody/multiphysics analysis. *Journal of Computational and Nonlinear Dynamics*, 9(4). doi: 10.1115/1.4025628
- McFarland, D. C., McCain, E. M., Poppo, M. N., & Saul, K. R. (2019, mar). Spatial dependency of glenohumeral joint stability during dynamic unimanual and bimanual pushing and pulling. *Journal of Biomechanical Engineering*, 141(5). doi: 10.1115/1.4043035
- Moon, D.-H., Kim, D., & Hong, Y.-D. (2019, oct). Intention detection using physical sensors and electromyogram for a single leg knee exoskeleton. *Sensors*, 19(20), 4447. doi: 10.3390/s19204447
- Morrey, B. F., An, K. N., & Stormont, T. J. (1988). Force transmission through the radial head. *The Journal of Bone & Joint Surgery*, 70(2), 250-256. Retrieved from [https://journals.lww.com/jbjsjournal/Abstract/1988/70020/Force\\_transmission\\_through\\_the\\_radial\\_head\\_.14.aspx](https://journals.lww.com/jbjsjournal/Abstract/1988/70020/Force_transmission_through_the_radial_head_.14.aspx)
- Nguyen, T. (2018). *Konzept zur entwicklung einer aktorgetriebenen kinematik eines exoskeletts für den*



*arm* (mathesis). Chair of Ergonomics, Technical University of Munich.

- Nicol, A. (1977). *Elbow joint prosthesis design: biomechanical aspects* (Unpublished doctoral dissertation). University of Strathclyde.
- Nikooyan, A., Veeger, H., Westerhoff, P., Graichen, F., Bergmann, G., & van der Helm, F. (2010, nov). Validation of the delft shoulder and elbow model using in-vivo glenohumeral joint contact forces. *Journal of Biomechanics*, 43(15), 3007–3014. doi: 10.1016/j.jbiomech.2010.06.015
- OpenSim. (2021a). *Getting started with analyses*. Website, Retrieved on 26 October 2021. Retrieved from <https://simtk-confluence.stanford.edu:8443/display/OpenSim/GettingStartedwithAnalyses>
- OpenSim. (2021b). *Getting started with CMC*. Website, Retrieved on 07 May 2021. Retrieved from <https://simtk-confluence.stanford.edu/display/OpenSim/GettingStartedwithCMC>
- OpenSim. (2021c). *How cmc works*. Website, Retrieved on 12 October 2021. Retrieved from <https://simtk-confluence.stanford.edu:8443/display/OpenSim/HowCMCWorks>
- OpenSim. (2021d). *How static optimisation works*. Website, Retrieved on 15 May 2021. Retrieved from <https://simtk-confluence.stanford.edu/display/OpenSim/HowStaticOptimizationWorks>
- OpenSim. (2021e). *Joint reactions analysis*. Website, Retrieved on 05 May 2021. Retrieved from <https://simtk-confluence.stanford.edu:8443/display/OpenSim/JointReactionsAnalysis>
- OpenSim. (2021f). *Marker (.trc) files*. Website, Retrieved on 13 October 2021. Retrieved from [https://simtk-confluence.stanford.edu:8443/display/OpenSim/Marker\(.trc\)Files](https://simtk-confluence.stanford.edu:8443/display/OpenSim/Marker(.trc)Files)
- OpenSim. (2021g). *Mobl-arms dynamic upper limb*. Website, Retrieved on 22 November 2021. Retrieved from [https://simtk.org/frs/?group\\_id=657](https://simtk.org/frs/?group_id=657)
- OpenSim. (2021h). *Musculoskeletal models*. Website, Retrieved on 06 June 2021. Retrieved from <https://simtk-confluence.stanford.edu/display/OpenSim/MusculoskeletalModels>
- OpenSim. (2021i). *Opensim::actuator class reference*. Website, Retrieved on 14 October 2021. Retrieved from [https://simtk.org/api\\_docs/opensim/api\\_docs/classOpenSim\\_1\\_1Actuator.html](https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1Actuator.html)
- OpenSim. (2021j). *OpenSim::CoordinateLimitForce Class Reference*. Website, Retrieved on 01 July 2021. Retrieved from [https://simtk.org/api\\_docs/opensim/api\\_docs/classOpenSim\\_1\\_1CoordinateLimitForce.html](https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1CoordinateLimitForce.html)
- OpenSim. (2021k). *Opensim models*. Website, Retrieved on 30 November 2021. Retrieved from <https://simtk-confluence.stanford.edu:8443/display/OpenSim/OpenSimModels#OpenSimModels-Joints>
- OpenSim. (2021l). *Pathactuator class reference*. Website, Retrieved on 22 October 2021. Retrieved from [https://simtk.org/api\\_docs/opensim/api\\_docs/classOpenSim\\_1\\_1PathActuator.html](https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1PathActuator.html)
- OpenSim. (2021m). *Scripting with matlab*. Website, Retrieved on 15 October 2021. Retrieved from <https://simtk-confluence.stanford.edu:8443/display/OpenSim/ScriptingwithMatlab>
- OpenSim. (2021n). *Storage (.sto) files*. Website, Retrieved on 15 October 2021. Retrieved from [https://simtk-confluence.stanford.edu:8443/display/OpenSim/Storage\(.sto\)Files](https://simtk-confluence.stanford.edu:8443/display/OpenSim/Storage(.sto)Files)
- OpenSim. (2021o). *Working with static optimisation*. Website, Retrieved on 08 July 2021. Retrieved from <https://simtk-confluence.stanford.edu:8443/display/OpenSim/WorkingwithStaticOptimization>
- Pandy, M. G. (2001, aug). Computer modeling and simulation of human movement. *Annual Review of Biomedical Engineering*, 3(1), 245–273. doi: 10.1146/annurev.bioeng.3.1.245
- Philipp Zimmer, H.-J. A. (2020). *Funktionelle anatomie*. Springer-Verlag GmbH. Retrieved from [https://www.ebook.de/de/product/39703146/philipp\\_zimmer\\_hans\\_joachim\\_appell\\_funktionelle\\_anatomie.html](https://www.ebook.de/de/product/39703146/philipp_zimmer_hans_joachim_appell_funktionelle_anatomie.html)

- Pons, J. L. (2008). *Wearable robots*. John Wiley & Sons. Retrieved from [https://www.ebook.de/de/product/15114146/jose\\_l\\_pons\\_wearable\\_robots.html](https://www.ebook.de/de/product/15114146/jose_l_pons_wearable_robots.html)
- Quental, C., Folgado, J., Ambrósio, J., & Monteiro, J. (2013, oct). Critical analysis of musculoskeletal modelling complexity in multibody biomechanical models of the upper limb. *Computer Methods in Biomechanics and Biomedical Engineering*, 18(7), 749–759. doi: 10.1080/10255842.2013.845879
- Raikova, R. T. (2009). INVESTIGATION OF THE INFLUENCE OF THE ELBOW JOINT REACTION ON THE PREDICTED MUSCLE FORCES USING DIFFERENT OPTIMIZATION FUNCTIONS. *Journal of Musculoskeletal Research*, 12(01), 31–43. doi: 10.1142/s021895770900216x
- Roelker, S. A., Caruthers, E. J., Hall, R. K., Pelz, N. C., Chaudhari, A. M., & Siston, R. A. (2020, aug). Effects of optimization technique on simulated muscle activations and forces. *Journal of Applied Biomechanics*, 36(4), 259–278. doi: 10.1123/jab.2018-0332
- Rüther, W., & Simmen, B. R. (Eds.). (2013). *Ae-manual der endoprothetik*. Springer-Verlag GmbH. Retrieved from [https://www.ebook.de/de/product/19990163/ae\\_manual\\_der\\_endoprothetik.html](https://www.ebook.de/de/product/19990163/ae_manual_der_endoprothetik.html)
- Saul, K. R., Hu, X., Goehler, C. M., Vidt, M. E., Daly, M., Velisar, A., & Murray, W. M. (2014, jul). Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model. *Computer Methods in Biomechanics and Biomedical Engineering*, 18(13), 1445–1458. doi: 10.1080/10255842.2014.916698
- Schuenke, M., Schulte, E., & Schumacher, U. (2011). *Prometheus: Lernatlas der anatomie*. Georg Thieme Verlag. doi: 10.1055/b-004-134445
- Sergeyev, A., Alaraje, N., Seidel, C., Carlson, Z., & Breda, B. (2013, mar). Design of a pneumatically powered wearable exoskeleton with biomimetic support and actuation. In *2013 IEEE aerospace conference*. IEEE. doi: 10.1109/aero.2013.6496857
- Shao, Z.-F., Tang, X., & Yi, W. (2014). Optimal design of a 3-DOF cable-driven upper arm exoskeleton. *Advances in Mechanical Engineering*, 6, 157096. doi: 10.1155/2014/157096
- Sherman, M., Seth, A., & Delp, S. (2010). *How to compute muscle moment arm using generalized coordinates*. Website, Retrieved on 27 October 2021. Retrieved from <https://simtk-confluence.stanford.edu:8443/download/attachments/2624184/HowToComputeMuscleMomentArm.pdf?version=1&modificationDate=1341887981184&api=v2>
- Shourijeh, M. S., Jung, M., Ko, S.-T., McGrath, M., Stech, N., & Damsgaard, M. (2017). Simulating physiological discomfort of exoskeletons using musculoskeletal modelling. *Gait & Posture*, 57, 83–84. Retrieved from [https://www.researchgate.net/profile/Moon\\_Jung2/publication/318449202\\_Simulating\\_Physiological\\_Discomfort\\_of\\_Exoskeletons\\_Using\\_Musculoskeletal\\_Modelling/links/5a87ff52a6fdcc6b1a3b6023/Simulating-Physiological-Discomfort-of-Exoskeletons-Using-Musculoskeletal-Modelling.pdf](https://www.researchgate.net/profile/Moon_Jung2/publication/318449202_Simulating_Physiological_Discomfort_of_Exoskeletons_Using_Musculoskeletal_Modelling/links/5a87ff52a6fdcc6b1a3b6023/Simulating-Physiological-Discomfort-of-Exoskeletons-Using-Musculoskeletal-Modelling.pdf) doi: 10.1016/j.gaitpost.2017.06.301
- Siciliano, B., & Khatib, O. (Eds.). (2008). *Springer handbook of robotics*. Springer. Retrieved from <https://link.springer.com/content/pdf/10.1007/978-3-540-30301-5.pdf>
- Stienen, A., Hekman, E., van der Helm, F., & van der Kooij, H. (2009). Self-aligning exoskeleton axes through decoupling of joint rotations and translations. *IEEE Transactions on Robotics*, 25(3), 628–633. doi: 10.1109/tro.2009.2019147
- Sugiarto, W. (2020). *Simulation eines weichen exoskeletts für den ellbogen* (Unpublished master's thesis). Technische Universität München. (Semester Thesis / Project)
- Thelen, D. G., & Anderson, F. C. (2006, jan). Using computed muscle control to generate forward dynamic

- simulations of human walking from experimental data. *Journal of Biomechanics*, 39(6), 1107–1115. doi: 10.1016/j.jbiomech.2005.02.010
- Throckmorton, G. S. (1985, jan). Quantitative calculations of temporomandibular joint reaction forces—II. the importance of the direction of the jaw muscle forces. *Journal of Biomechanics*, 18(6), 453–461. doi: 10.1016/0021-9290(85)90280-5
- Throckmorton, G. S., & Throckmorton, L. S. (1985, jan). Quantitative calculations of temporomandibular joint reaction forces—i. the importance of the magnitude of the jaw muscle forces. *Journal of Biomechanics*, 18(6), 445–452. doi: 10.1016/0021-9290(85)90279-9
- Troy, K. (2013). *What is the difference between hip contact and hip joint forces?* Website, Retrieved on 03 September 2021. Retrieved from [https://www.researchgate.net/post/What\\_is\\_the\\_difference\\_between\\_hip\\_contact\\_and\\_hip\\_joint\\_forces](https://www.researchgate.net/post/What_is_the_difference_between_hip_contact_and_hip_joint_forces)
- Tröster, M., Schneider, U., Bauerhansl, T., Rasmussen, J., & Andersen, M. (2018). Simulation framework for active upper limb exoskeleton design optimization based on musculoskeletal modeling. *Technische Unterstützungssysteme, die die Menschen wirklich wollen*.
- Viceconti, M. (2012). *Multiscale modeling of the skeletal system*. Cambridge: Cambridge University Press. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=409054>
- Vigotsky, A. D., K. E. Zelik, J. L., & Hinrichs, R. N. (2019). Mechanical misconceptions: Have we lost the “mechanics” in “sports biomechanics”? *Journal of Biomechanics*. doi: 10.1016/j.jbiomech.2019.07.005
- Werner, S. L., Fleisig, G. S., Dillman, C. J., & Andrews, J. R. (1993). Biomechanics of the elbow during baseball pitching. *Journal of Orthopaedic & Sports Physical Therapy*, 17(6), 274–278. doi: 10.2519/jospt.1993.17.6.274
- Wesseling, M., Derikx, L. C., de Groote, F., Bartels, W., Meyer, C., Verdonschot, N., & Jonkers, I. (2014). Muscle optimization techniques impact the magnitude of calculated hip joint contact forces. *Journal of Orthopaedic Research*, 33(3), 430–438. doi: 10.1002/jor.22769
- Westerhoff, P., Graichen, F., Bender, A., Rohlmann, A., & Bergmann, G. (2009). An instrumented implant for in vivo measurement of contact forces and contact moments in the shoulder joint. *Medical Engineering & Physics*, 31(2), 207–213. doi: 10.1016/j.medengphy.2008.07.011
- XO, G. (2021). *Sarcos technology and robotics corporation*. Website, Retrieved on 28 November 2021. Retrieved from <https://www.sarcos.com/products/guardian-xo-powered-exoskeleton/>
- Yang, Z., Gu, W., Zhang, J., & Gui, L. (2017). *Force control theory and method of human load carrying exoskeleton suit*. Springer Berlin Heidelberg. doi: 10.1007/978-3-662-54144-9
- Young, A. J., Gannon, H., & Ferris, D. P. (2017, jun). A biomechanical comparison of proportional electromyography control to biological torque control using a powered hip exoskeleton. *Frontiers in Bioengineering and Biotechnology*, 5. doi: 10.3389/fbioe.2017.00037
- Zhou, L., Li, Y., & Bai, S. (2017). A human-centered design optimization approach for robotic exoskeletons through biomechanical simulation. *Robotics and Autonomous Systems*, 91, 337–347. doi: 10.1016/j.robot.2016.12.012

## List of Figures

|    |   |    |
|----|---|----|
| 1  | The “Hardiman” exoskeleton by General Electric from 1965 (General Electric, 2016) . . . . .   | 10 |
| 2  | Exosuit as used by the Harvard Biodesign Lab (Harvard Biodesign Lab, 2021) . . . . .  | 12 |
| 3  | Workflow of the CMC algorithm (Thelen & Anderson, 2006). $\mathbf{q}$ are the model’s generalized coordinates $\mathbf{u}$ are the model’s generalized coordinate speeds $f^{exp}$ are (optional) experimental forces, e.g. ground reaction forces in a walking trial . . . . .   | 15 |
| 4  | Exosuit implemented as simple path actuators by (Bae, 2013). The path actuators are marked in blue and support the ankle (left) and the hip (right). . . . .  | 17 |
| 5  | Elbow exosuit prototype developed at the LfE with two actuated and linked cables running in parallel. . . . .   | 19 |
| 6  | Bones and articulations of the human elbow (Schuenke et al., 2011) . . . . .  | 22 |
| 7  | Different elbow JRFs over flexion angles according to (Amis et al., 1980) with values for 50 N at the hand: Variation of joint forces during flexion, with and without triceps antagonism, per unit of force at the hand. Key: $\bullet$ , $\circ$ : Humero-coronoid antagonism ; $\blacktriangle$ , $\triangle$ : Humero-radial force, with and without forces: $\blacksquare$ , $\square$ : Resultant humero-ulnar forces (Amis et al., 1980) . . . . . | 26 |
| 8  | Elbow flexion angle over time in the artificially created motion file . . . . .   | 34 |
| 9  | Visualization of the lifting motion from 0 to 130° elbow flexion. Screenshots at different time steps during the movement are overlaid in an image editing program. . . . .   | 35 |
| 10 | Flow chart depicting inputs and outputs of OpenSim’s CMC tool in the configuration of this thesis. . . . .  | 35 |
| 11 | Flow chart depicting inputs and outputs of OpenSim’s JRA tool in the configuration of this thesis. . . . .  | 36 |
| 12 | Coordinate system of the ulna body in the MobL Model . . . . .  | 36 |
| 13 | Convention for positive vector components of the reported elbow JRFs. Negative values in the XY-plane imply a compressive force in the joint. The coordinate system is fixed to the ulna and rotates around the humerus’ head during flexion. . . . .   | 37 |
| 14 | Flow chart depicting the automated simulation workflow in MATLAB with its inputs and outputs. . . . .   | 38 |
| 15 | Three torque curves from different control strategies for the exoskeleton support. . . . .  | 44 |
| 16 | Marker positions used for path actuators with the prototype’s reference geometries. The upper arm exosuit part was copied from (Sugiarto, 2020). The forearm brace was provided by Martin Fleischer from LfE. . . . .   | 45 |
| 17 | OpenSim models including different exosuit actuators developed in this thesis. . . . .  | 48 |

|      |  |    |
|------|--|----|
| 18   | Example plot of resulting JRF, JRF vector components and actuator force of models 01_noExo_noWeight, 01_noExo and 03_singlePath_ulna. This type of plot is used to compare different models' JRFs in subsequent sections. . . . .  | 51 |
| 19   | Minimum, maximum and mean elbow JRF values of all 14 models for the 100%-support controller. This type of plot can be found for the other two control strategies in Appendix A.  | 51 |
| 20   | JRFs and actuator forces of model 03_singlePath_ulna with the three different control strategies . . . . .   | 53 |
| 21   | JRFs of models without exoskeleton support. . . . .  | 54 |
| 22   | JRFs and actuator forces of models with coordinate and path actuators. The dotted lines are the JRFs in the models without exosuit support and included in every JRF plot. . . . .   | 55 |
| 23   | JRFs and actuator forces of models with single- and double-stringed path actuators fixed in the ulna's or radius' coordinate system (Actuator forces of models 03_singlePath_ulna and 04_singlePath_radius as well as 06_doublePath_radius and 07_doublePath_ulna are the same.) . . . . .   | 57 |
| 24   | JRFs and actuator forces of models with double-stringed path actuators (4 path points) (Actuator force curves of models 05_doublePath_ulna_radius, 06_doublePath_radius and 07_doublePath_ulna are equivalent.) . . . . .  | 59 |
| 25   | JRFs and actuator forces of models with double-stringed path actuators (10 path points) (The actuator forces of models 09_doublePath_AtWrist, 10_doublePath_AtWrist_onlyRadius and 11_doublePath_AtWrist_onlyUlna are equivalent.) . . . . .   | 61 |
| 26   | JRFs and actuator forces of models with path actuators attached to the ulna (The JRF curves of models 03_singlePath_ulna, 07_doublePath_ulna and 11_doublePath_AtWrist_onlyUlna are (almost) equivalent. The actuator forces of models 07_doublePath_ulna and 11_doublePath_AtWrist_onlyUlna are equivalent.) . . . . .  | 62 |
| 27   | JRFs and actuator forces of models with path actuators attached to the radius (The JRF and actuator force curves of models 06_doublePath_radius and 10_doublePath_AtWrist_onlyRadius are equivalent.) . . . . .  | 63 |
| 28   | JRFs and actuator forces of models with path actuators attached to the radius and the ulna (The actuator forces of models 05_doublePath_ulna_radius, 08_doublePath_connectedBelow and 09_doublePath_AtWrist are equivalent.) . . . . .   | 65 |
| 29   | JRFs and actuator forces of models including path actuators. Models 12 and 13 are extended versions of models 7 and 9 by adding 10 more path points over the shoulder. (The JRFs of models 09_doublePath_AtWrist and 12_doublePath_withShoulder are (almost) equivalent. The JRFs of models 07_doublePath_ulna and 13_ulna_withShoulder are (almost) equivalent. The actuator forces of models 07_doublePath_ulna, 09_doublePath_AtWrist, 12_doublePath_withShoulder and 13 are equivalent.) . . . . . | 66 |
| 30   | JRFs and actuator forces of models with path actuators attached to the radius and the ulna. A trend towards higher JRFs with increasing number of path points can be observed. (The actuator forces of models 06_doublePath_radius, 08_doublePath_connectedBelow and 09_doublePath_AtWrist are equivalent.) . . . . .  | 73 |
| A.31 | Minimum, maximum and mean elbow JRF values of all 14 models for the 100%-support controller. . . . .   | 95 |

|      |  |     |
|------|--|-----|
| A.32 | Minimum, maximum and mean elbow JRF values of all 14 models for the constant torque controller. . . . .  | 96  |
| A.33 | Minimum, maximum and mean elbow JRF values of all 14 models for the CMC-optimized controller. . . . .  | 97  |
| A.34 | JRFs and actuator forces of models with coordinate and path actuators with the constant torque control. . . . .  | 99  |
| A.35 | JRFs and actuator forces of models with single- and double-stringed path actuators fixed in the ulna's or radius' coordinate system and with the constant torque control. (Actuator forces of models 03_singlePath_ulna and 04_singlePath_radius as well as 06_doublePath_radius and 07_doublePath_ulna are the same.) . . . . .   | 100 |
| A.36 | JRFs and actuator forces of models with double-stringed path actuators (4 path points) using the constant torque control. (Actuator force curves of models 05_doublePath_ulna_radius, 06_doublePath_radius and 07_doublePath_ulna are equivalent.) . . . .   | 100 |
| A.37 | JRFs and actuator forces of models with double-stringed path actuators (10 path points) using the constant torque control. (The actuator forces of models 09_doublePath_AtWrist, 10_doublePath_AtWrist_onlyRadius and 11_doublePath_AtWrist_onlyUlna are equivalent.) . . . . .  | 101 |
| A.38 | JRFs and actuator forces of models with path actuators attached to the ulna using the constant torque control. (The JRF curves of models 03_singlePath_ulna, 07_doublePath_ulna and 11_doublePath_AtWrist_onlyUlna are (almost) equivalent. The actuator forces of models 07_doublePath_ulna and 11_doublePath_AtWrist_onlyUlna are equivalent.)   | 101 |
| A.39 | JRFs and actuator forces of models with path actuators attached to the radius using the constant torque control. (The JRF and actuator force curves of models 06_doublePath_radius and 10_doublePath_AtWrist_onlyRadius are equivalent.) . . . . .   | 102 |
| A.40 | JRFs and actuator forces of models with path actuators attached to the radius and the ulna using the constant torque control. (The actuator forces of models 05_doublePath_ulna_radius, 08_doublePath_connectedBelow and 09_doublePath_AtWrist are equivalent.)  | 102 |
| A.41 | JRFs and actuator forces of models including path actuators using the constant torque control. Models 12 and 13 are extended versions of models 7 and 9 by adding 10 more path points over the shoulder. (The JRFs of models 09_doublePath_AtWrist and 12_doublePath_withShoulder are (almost) equivalent. The JRFs of models 07_doublePath_ulna and 13_ulna_withShoulder are (almost) equivalent. The actuator forces of models 07_doublePath_ulna, 09_doublePath_AtWrist, 12_doublePath_withShoulder and 13 are equivalent.) . . . . . | 103 |
| A.42 | JRFs and actuator forces of models with coordinate and path actuators with the CMC-optimized control. . . . .  | 104 |
| A.43 | JRFs and actuator forces of models with single- and double-stringed path actuators fixed in the ulna's or radius' coordinate system and with the CMC-optimized control. (Actuator forces of models 03_singlePath_ulna and 04_singlePath_radius as well as 06_doublePath_radius and 07_doublePath_ulna are the same.) . . . . .   | 105 |

|      |  |     |
|------|--|-----|
| A.44 | JRFs and actuator forces of models with double-stringed path actuators (4 path points) using the CMC-optimized control. (Actuator force curves of models 05_doublePath_ulna_radius, 06_doublePath_radius and 07_doublePath_ulna are equivalent.) . . . .   | 105 |
| A.45 | JRFs and actuator forces of models with double-stringed path actuators (10 path points) using the CMC-optimized control. (The actuator forces of models 09_doublePath_AtWrist, 10_doublePath_AtWrist_onlyRadius and 11_doublePath_AtWrist_onlyUlna are equivalent.) . . . . .  | 106 |
| A.46 | JRFs and actuator forces of models with path actuators attached to the ulna using the CMC-optimized control. (The JRF curves of models 03_singlePath_ulna, 07_doublePath_ulna and 11_doublePath_AtWrist_onlyUlna are (almost) equivalent. The actuator forces of models 07_doublePath_ulna and 11_doublePath_AtWrist_onlyUlna are equivalent.)   | 106 |
| A.47 | JRFs and actuator forces of models with path actuators attached to the radius using the CMC-optimized control. (The JRF and actuator force curves of models 06_doublePath_radius and 10_doublePath_AtWrist_onlyRadius are equivalent.) . . . . .   | 107 |
| A.48 | JRFs and actuator forces of models with path actuators attached to the radius and the ulna using the CMC-optimized control. (The actuator forces of models 05_doublePath_ulna_radius, 08_doublePath_connectedBelow and 09_doublePath_AtWrist are equivalent.)  | 107 |
| A.49 | JRFs and actuator forces of models including path actuators using the CMC-optimized control. Models 12 and 13 are extended versions of models 7 and 9 by adding 10 more path points over the shoulder. (The JRFs of models 09_doublePath_AtWrist and 12_doublePath_withShoulder are (almost) equivalent. The JRFs of models 07_doublePath_ulna and 13_ulna_withShoulder are (almost) equivalent. The actuator forces of models 07_doublePath_ulna, 09_doublePath_AtWrist, 12_doublePath_withShoulder and 13 are equivalent.) . . . . . | 108 |

## List of Tables

|     |  |     |
|-----|--|-----|
| 1   | Overview of exoskeleton modeling strategies in musculoskeletal simulations by different researchers. . . . .                         | 21  |
| 2   | Overview of studies simulating elbow JRFs for different models and motions, some including exoskeleton support. . . . .              | 25  |
| 3   | Upper extremity musculoskeletal models in OpenSim . . . . .  | 31  |
| 4   | Overview of all OpenSim models (0-13). Models 00_noExo_noWeight and 01_noExo don't have any exoskeleton support implemented. . . . . | 46  |
| A.5 | Table with minimum, mean and maximum elbow JRF values for each model with the 100%-support controller. . . . .                       | 109 |
| A.6 | Table with minimum, mean and maximum elbow JRF values for each model with the constant torque controller. . . . .                    | 110 |

|      |  |     |
|------|--|-----|
| A.7  | Table with minimum, mean and maximum elbow JRF values for each model with the CMC-optimized controller. . . . .                                    | 111 |
| A.8  | Elbow JRF relief with the 100%-support control strategy in comparison to model 01_noExo as a percentage thereof for all OpenSim models. . . . .    | 112 |
| A.9  | Elbow JRF relief with the constant torque control strategy in comparison to model 01_noExo as a percentage thereof for all OpenSim models. . . . . | 112 |
| A.10 | Elbow JRF relief with the CMC-optimized control strategy in comparison to model 01_noExo as a percentage thereof for all OpenSim models. . . . .   | 113 |
| A.11 | Analysis of the biarticularity property of the muscles in the MobL musculoskeletal model. "1" means true; "0" means false. . . . .                 | 115 |



# Abbreviations

## **A**

AMS - AnyBody Modeling System.....

## **C**

CMC - Computed Muscle Control.....

## **D**

DOF - degree of freedom.....

DOFs - degrees of freedom.....

## **J**

JRA - Joint Reaction Analysis.....

JRF - joint reaction force.....

JRFs - joint reaction forces.....

## **L**

LfE - Chair of Ergonomics at the Technical University of Munich.....

## **M**

MA - Muscle Analysis.....

MobL - MobL-ARMS Dynamic Upper Limb.....

## **S**

SO - Static Optimisation.....

## **T**

TUM - Technical University of Munich.....

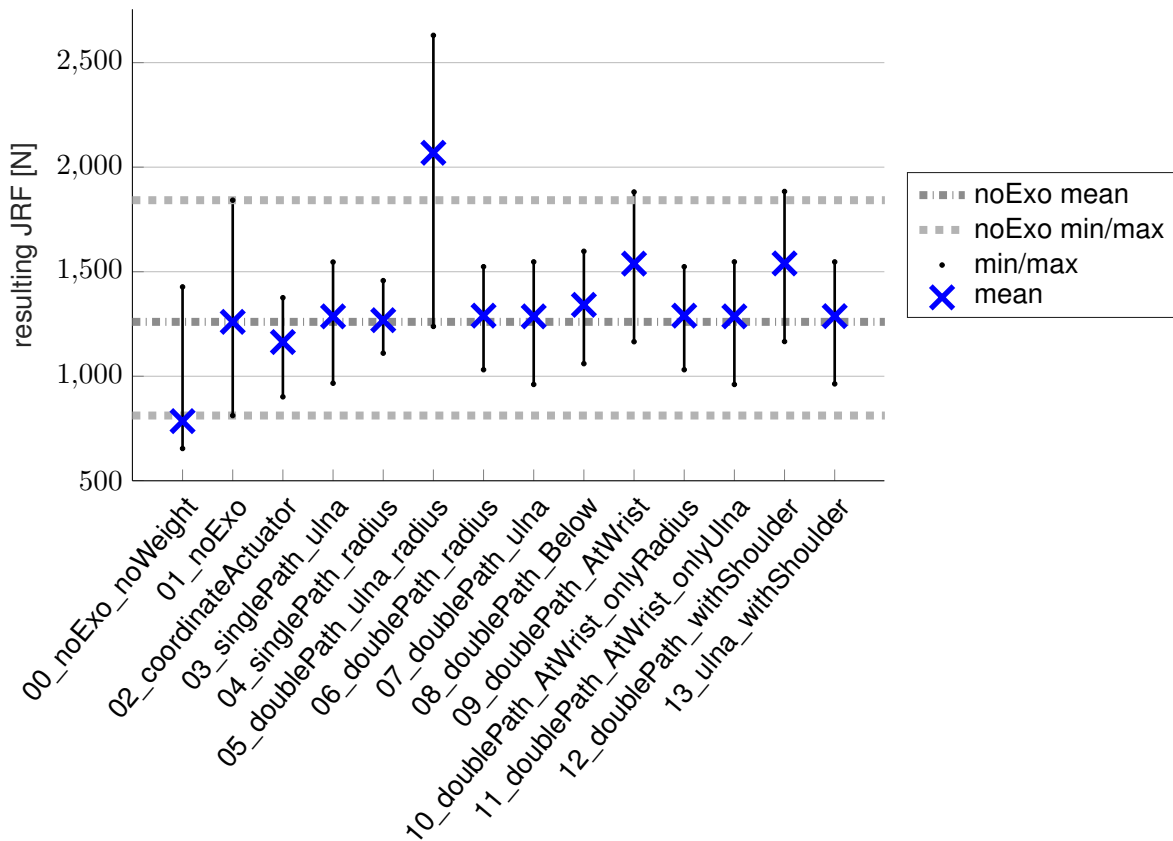
# Appendix

The appendix consists of the following sections:

- **Appendix A** - Boxplots with minimum, mean and maximum JRF values for all three control strategies
- **Appendix B** - JRF plots for the constant torque and CMC-optimized control strategies.
- **Appendix C** - Tables with minimum, maximum and mean elbow JRF for all three control strategies in each model
- **Appendix D** - Tables with elbow JRF relief compared to model 01\_noExo
- **Appendix E** - Analysis of biarticularity of the MobL model's muscles
- **Appendix F** - Code for model modifications and setup files in OpenSim created in this thesis
- **Appendix G** - Code for MATLAB scripts and functions created in this thesis
- **Appendix H** - Exemplary code for exosuit actuator models created in this thesis

## Appendix A

Boxplot-like plot displaying the minimum, maximum and mean elbow JRF values for all 14 models - one plot for each control strategy.



**Figure A.31:** Minimum, maximum and mean elbow JRF values of all 14 models for the 100%-support controller.

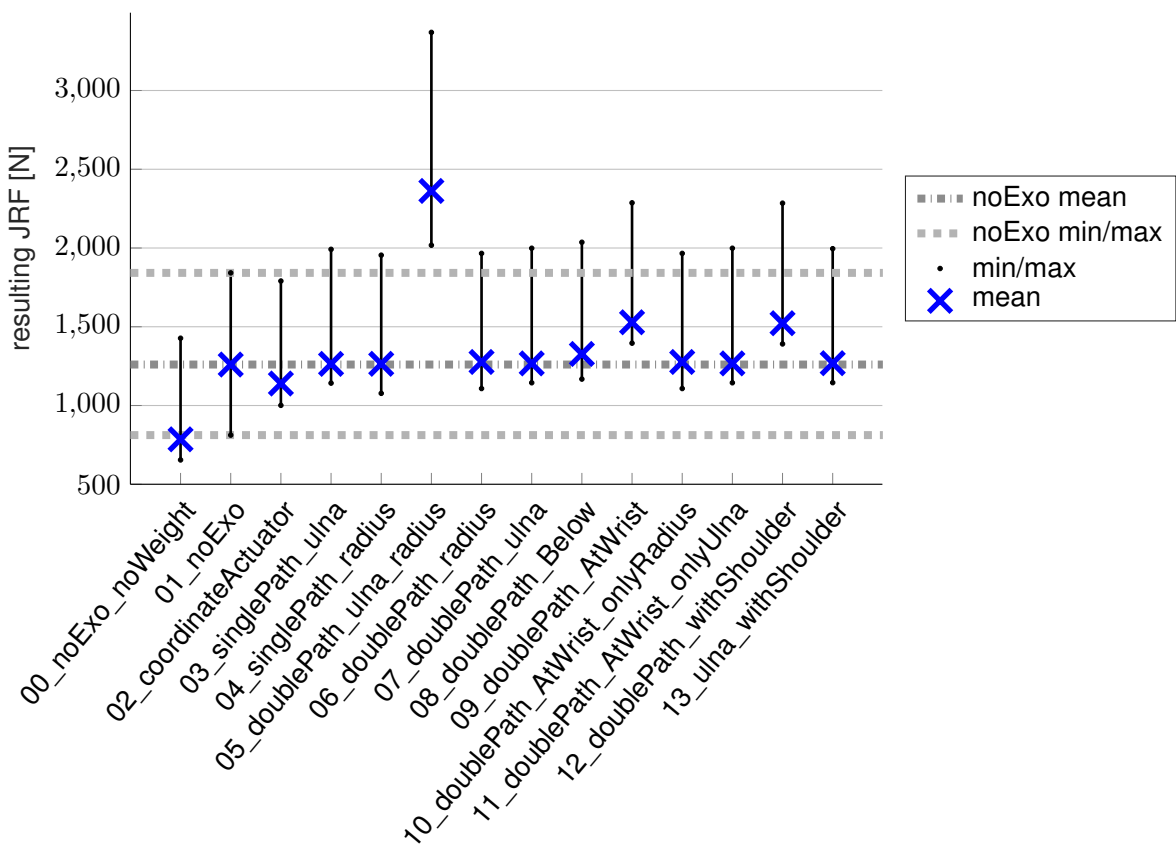


Figure A.32: Minimum, maximum and mean elbow JRF values of all 14 models for the constant torque controller.

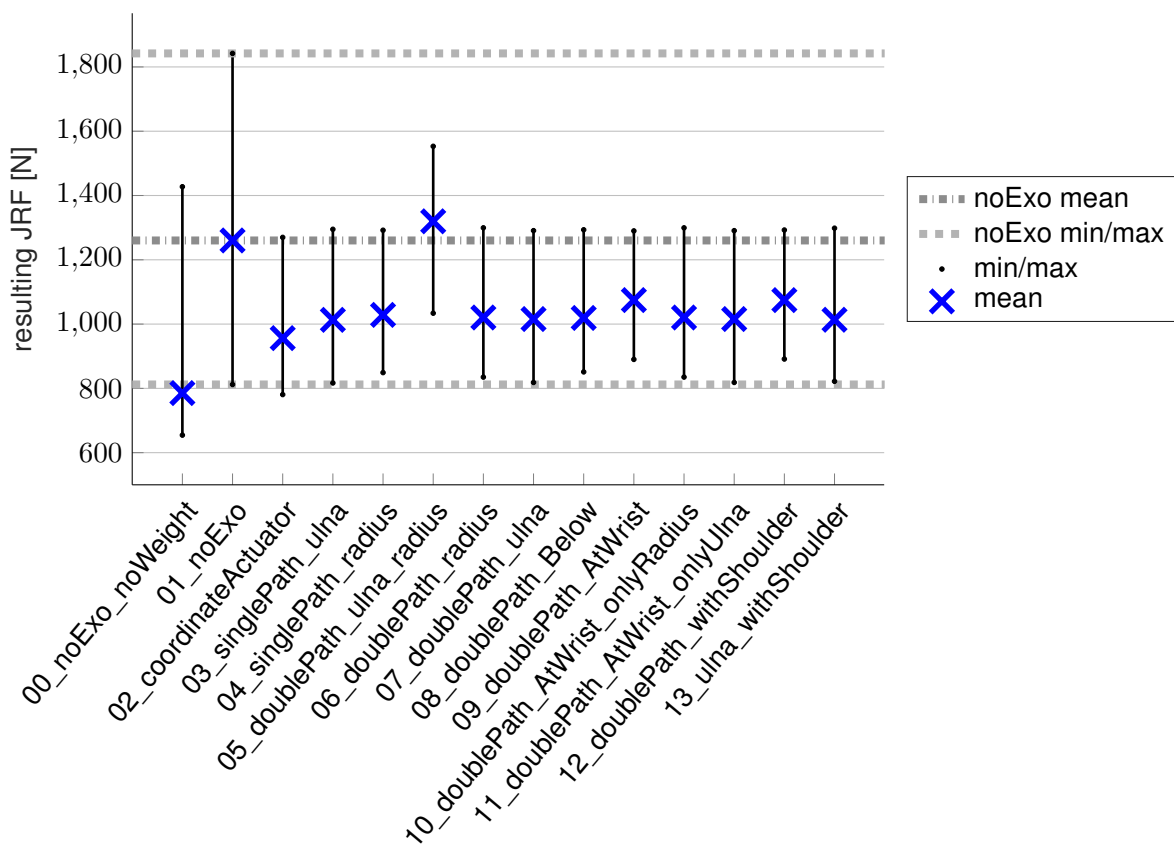
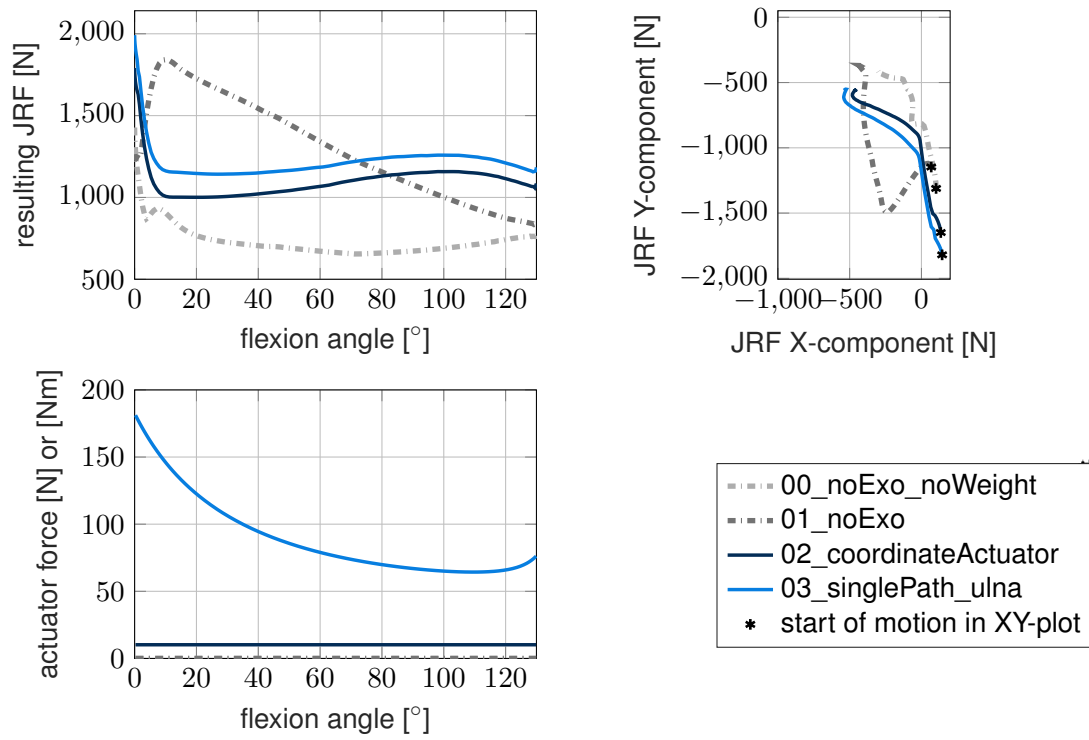


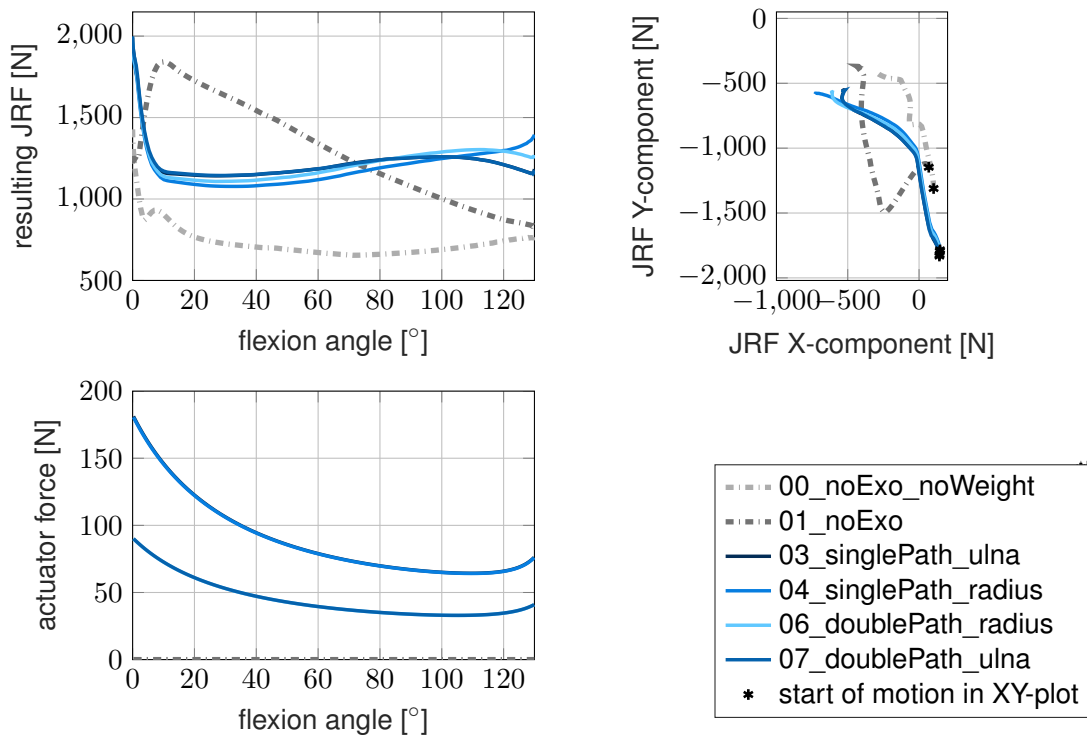
Figure A.33: Minimum, maximum and mean elbow JRF values of all 14 models for the CMC-optimized controller.

## Appendix B

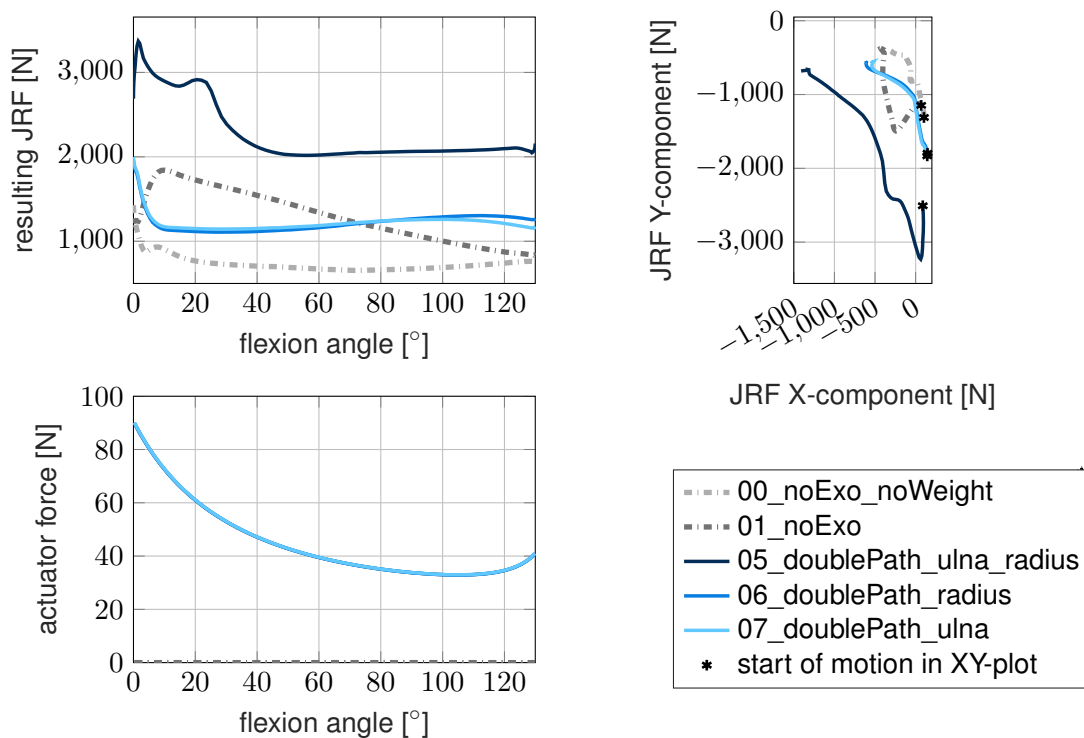
This appendix section shows the JRF plots for all exosuit models with the constant torque and CMC-optimized controllers. The models are grouped in the same way as in chapter 5.3.



**Figure A.34:** JRFs and actuator forces of models with coordinate and path actuators with the constant torque control.

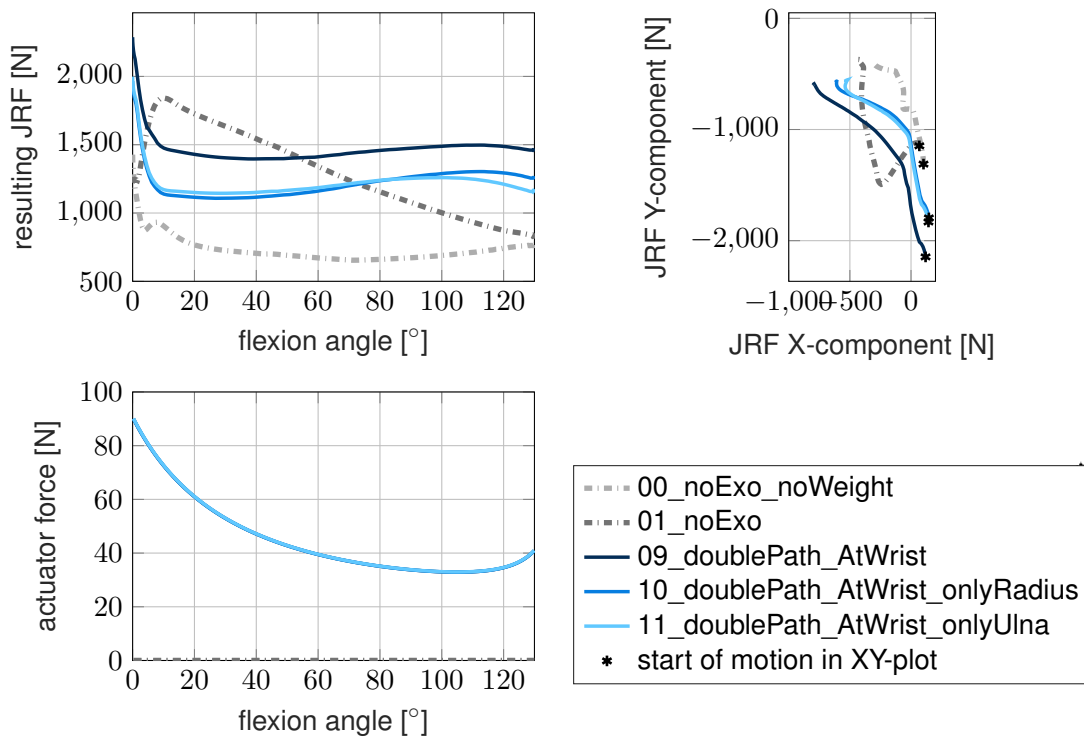


**Figure A.35:** JRFs and actuator forces of models with single- and double-stringed path actuators fixed in the ulna's or radius' coordinate system and with the constant torque control. (Actuator forces of models 03\_singlePath\_ulna and 04\_singlePath\_radius as well as 06\_doublePath\_radius and 07\_doublePath\_ulna are the same.)

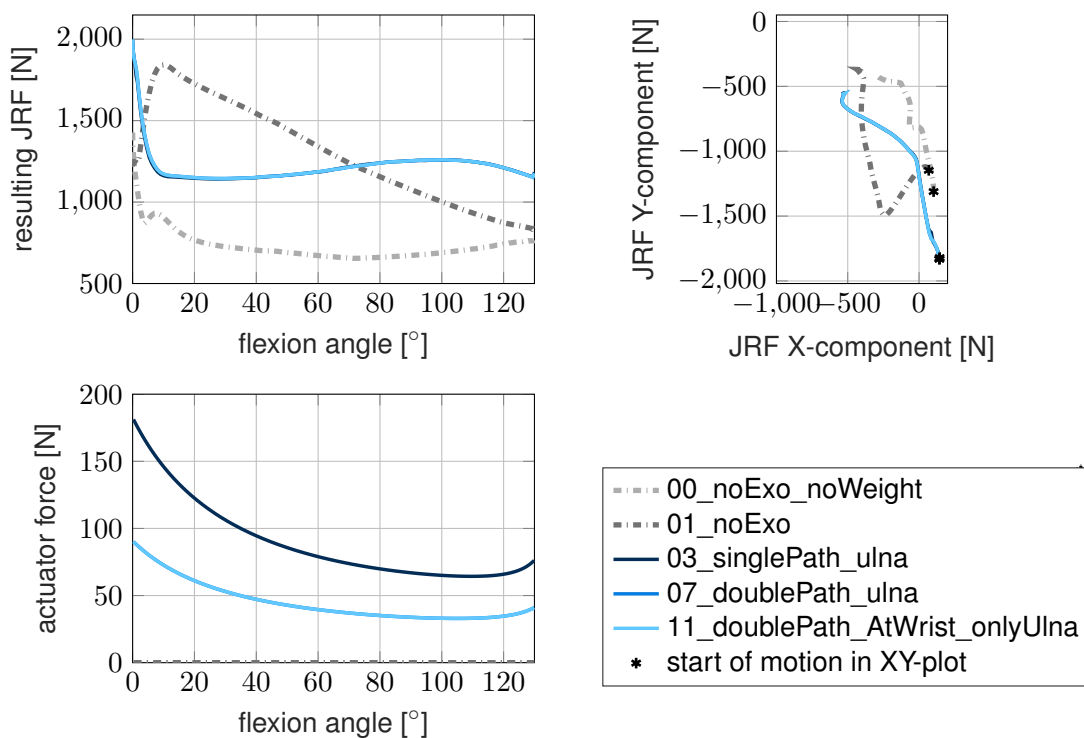


**Figure A.36:** JRFs and actuator forces of models with double-stringed path actuators (4 path points) using the constant torque control. (Actuator force curves of models 05\_doublePath\_ulna\_radius, 06\_doublePath\_radius and 07\_doublePath\_ulna are equivalent.)

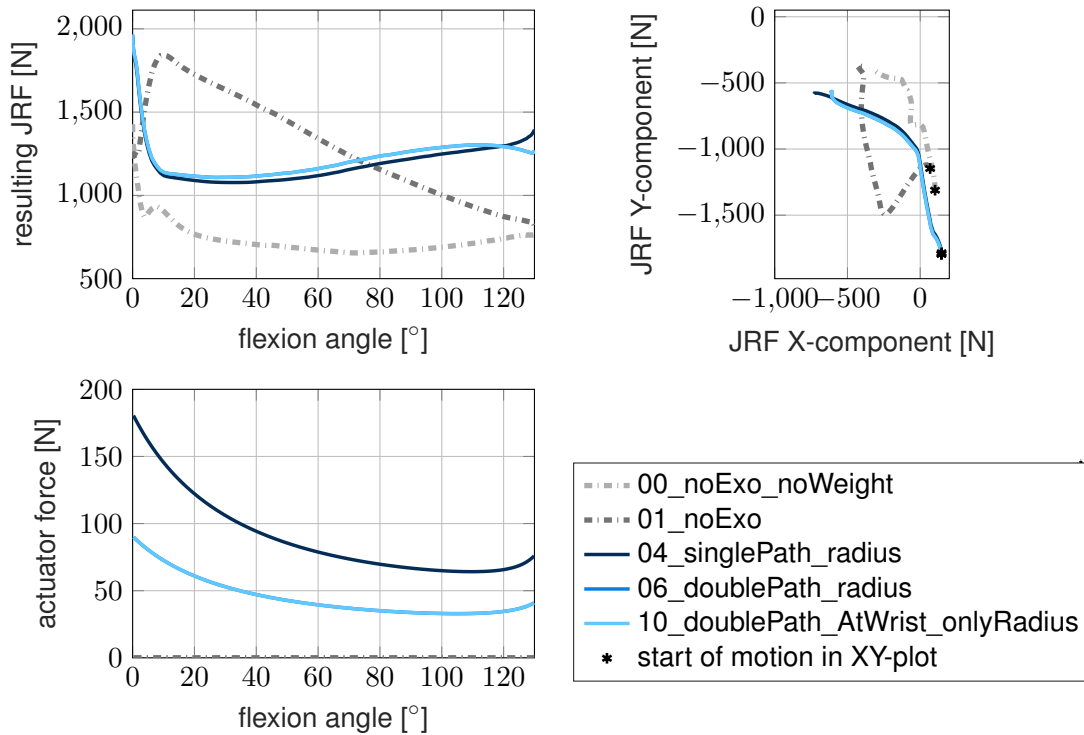




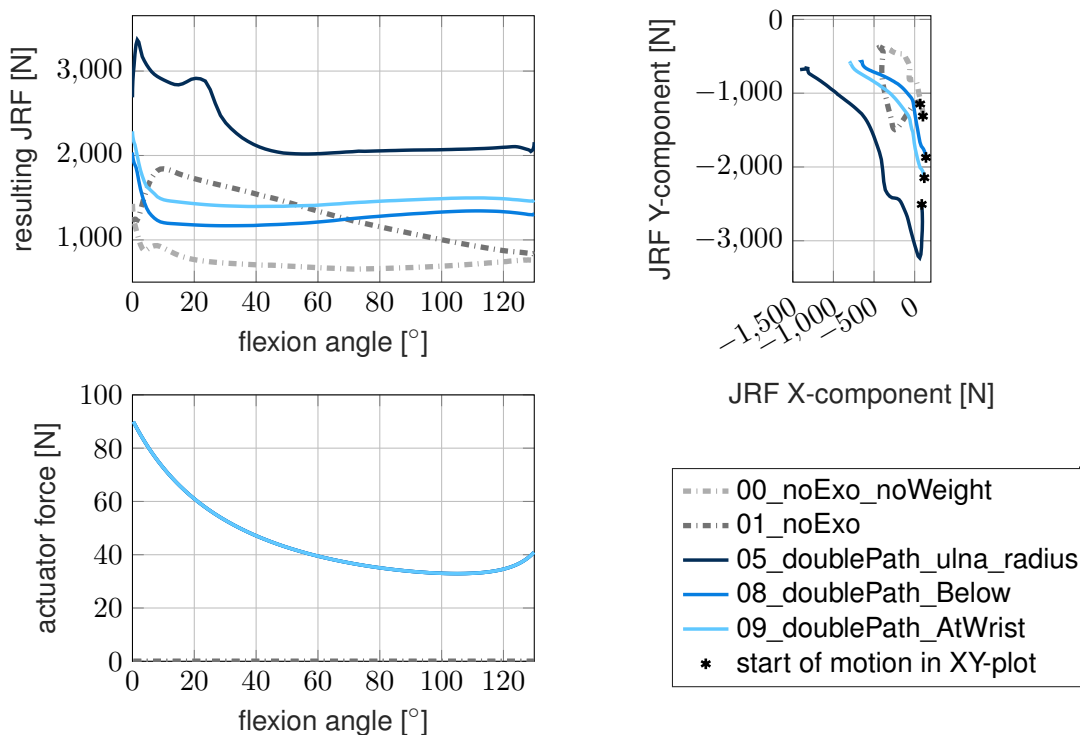
**Figure A.37:** JRFs and actuator forces of models with double-stringed path actuators (10 path points) using the constant torque control. (The actuator forces of models 09\_doublePath\_AtWrist, 10\_doublePath\_AtWrist\_onlyRadius and 11\_doublePath\_AtWrist\_onlyUlna are equivalent.)



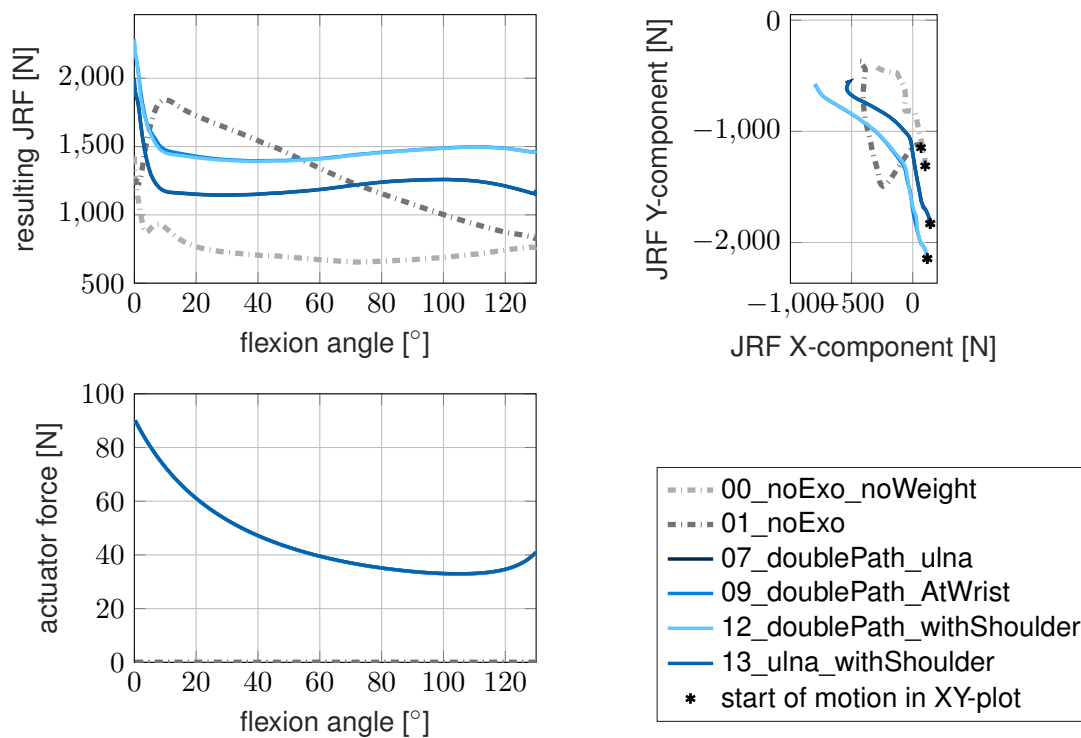
**Figure A.38:** JRFs and actuator forces of models with path actuators attached to the ulna using the constant torque control. (The JRF curves of models 03\_singlePath\_ulna, 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna are (almost) equivalent. The actuator forces of models 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna are equivalent.)



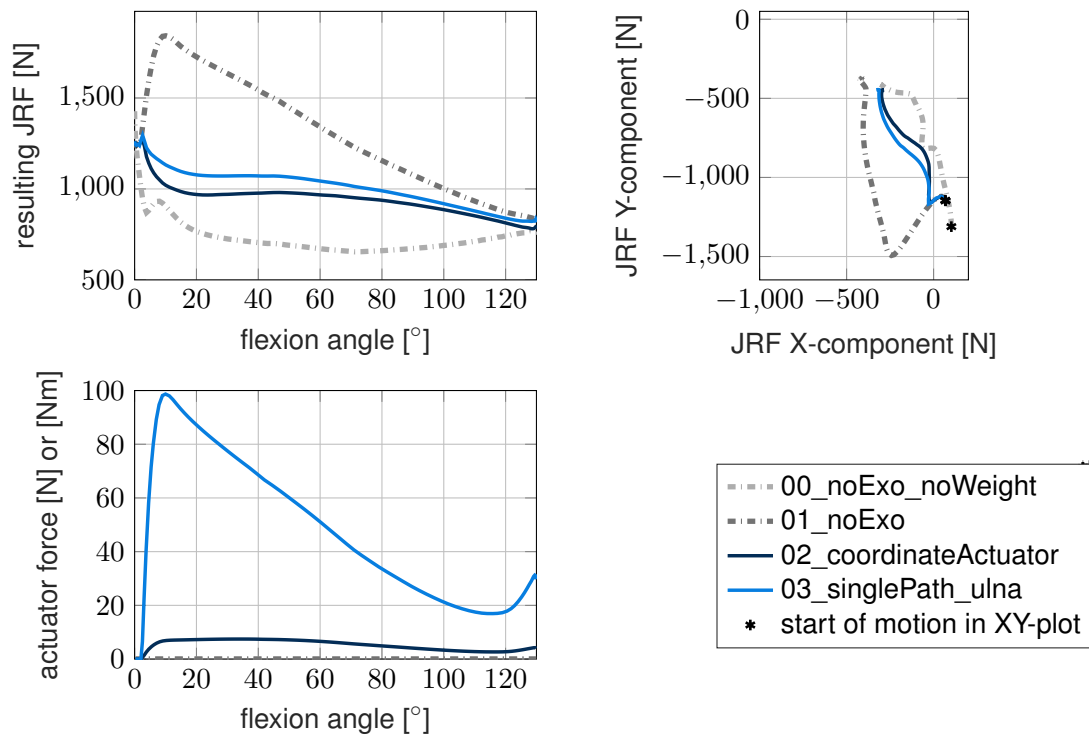
**Figure A.39:** JRFs and actuator forces of models with path actuators attached to the radius using the constant torque control. (The JRF and actuator force curves of models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius are equivalent.)



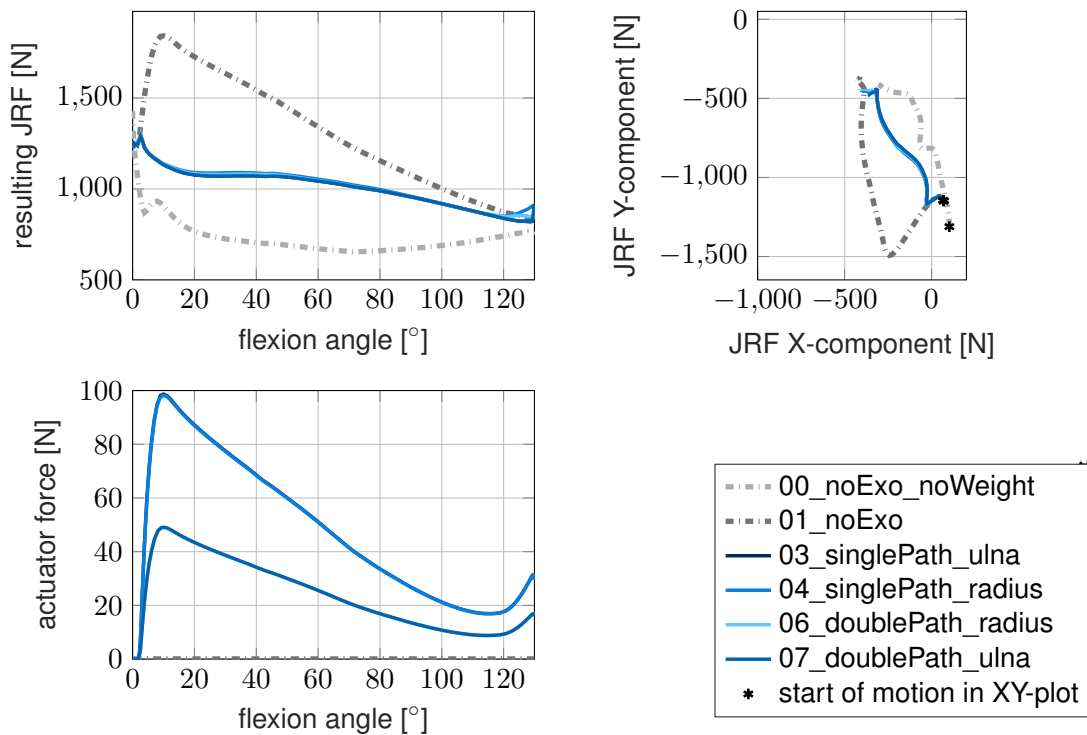
**Figure A.40:** JRFs and actuator forces of models with path actuators attached to the radius and the ulna using the constant torque control. (The actuator forces of models 05\_doublePath\_ulna\_radius, 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist are equivalent.)



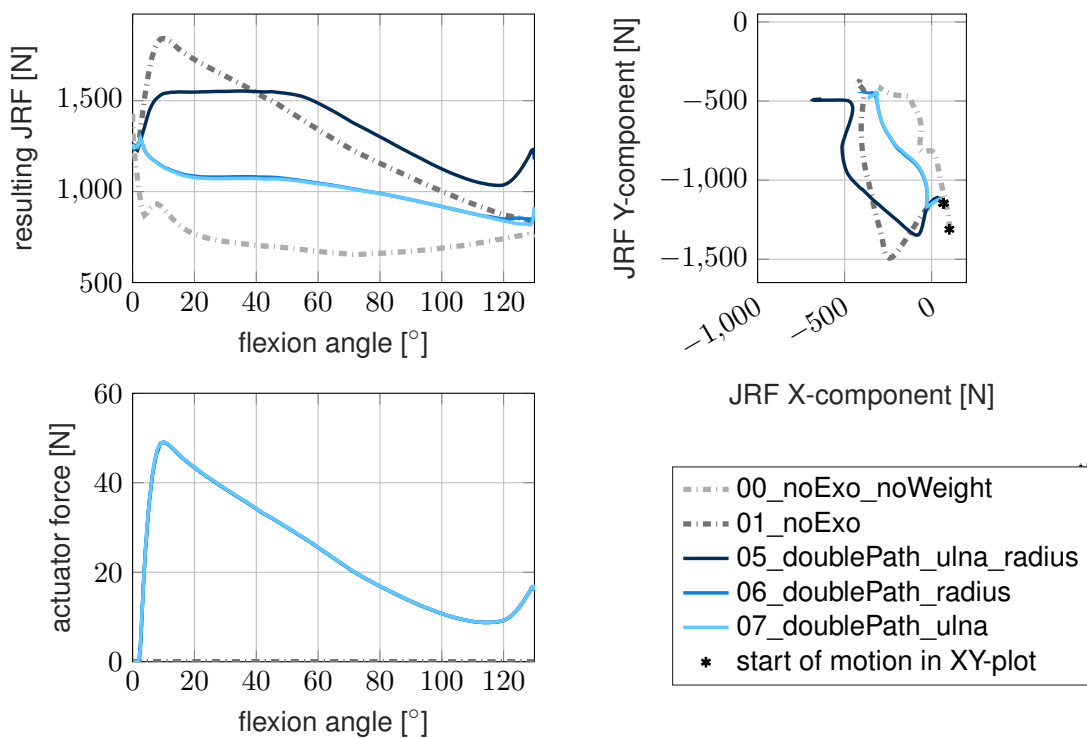
**Figure A.41:** JRFs and actuator forces of models including path actuators using the constant torque control. Models 12 and 13 are extended versions of models 7 and 9 by adding 10 more path points over the shoulder. (The JRFs of models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder are (almost) equivalent. The JRFs of models 07\_doublePath\_ulna and 13\_ulna\_withShoulder are (almost) equivalent. The actuator forces of models 07\_doublePath\_ulna, 09\_doublePath\_AtWrist, 12\_doublePath\_withShoulder and 13 are equivalent.)



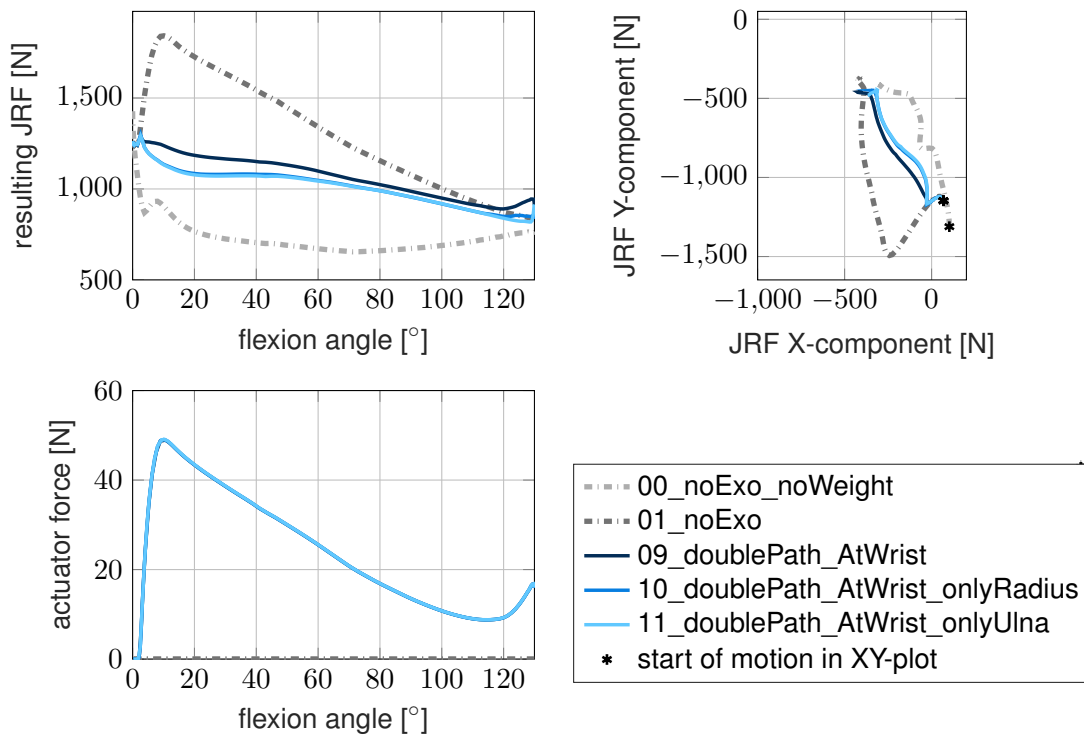
**Figure A.42:** JRFs and actuator forces of models with coordinate and path actuators with the CMC-optimized control.



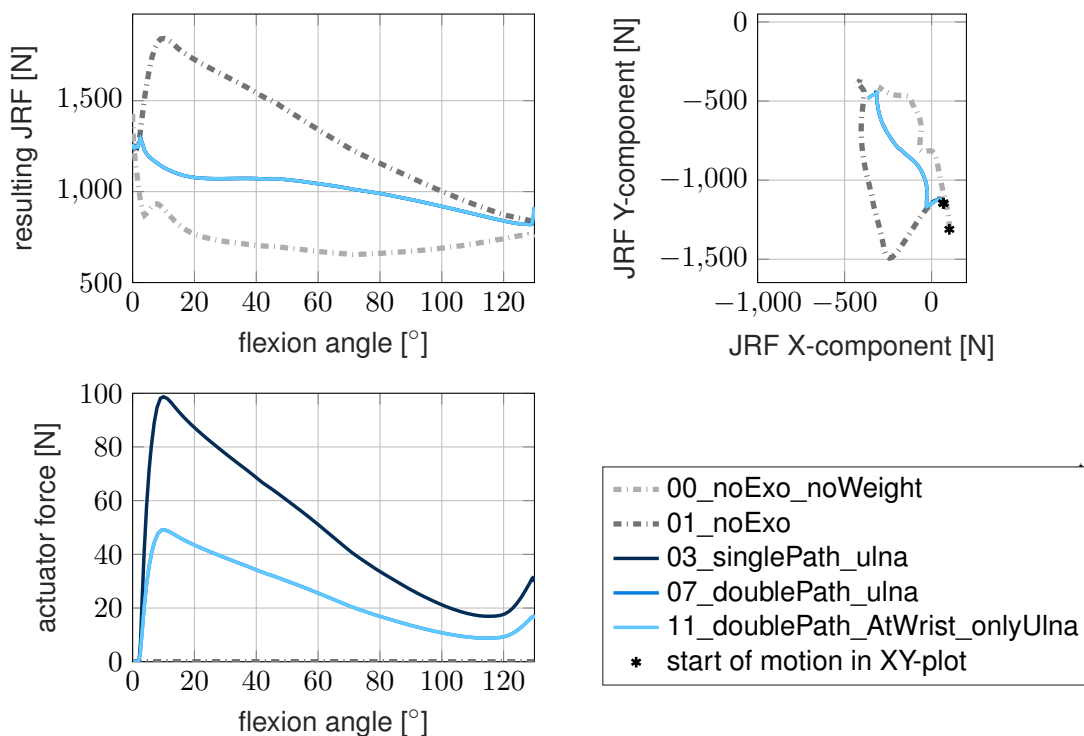
**Figure A.43:** JRFs and actuator forces of models with single- and double-stringed path actuators fixed in the ulna's or radius' coordinate system and with the CMC-optimized control. (Actuator forces of models 03\_singlePath\_ulna and 04\_singlePath\_radius as well as 06\_doublePath\_radius and 07\_doublePath\_ulna are the same.)



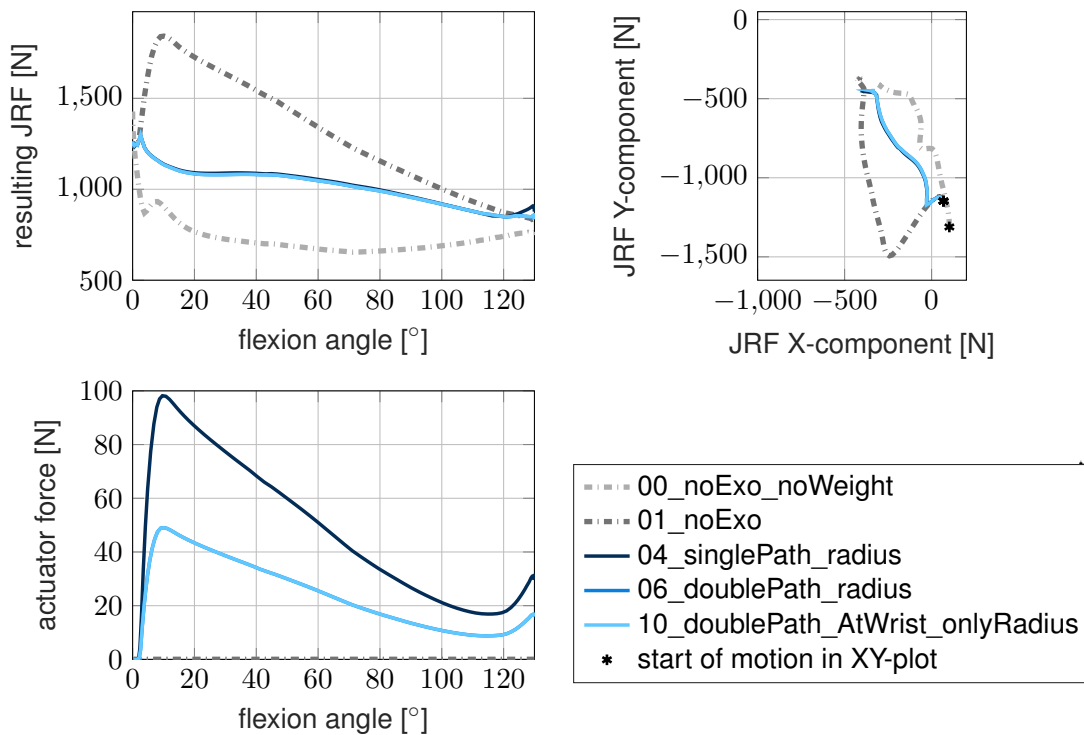
**Figure A.44:** JRFs and actuator forces of models with double-stringed path actuators (4 path points) using the CMC-optimized control. (Actuator force curves of models 05\_doublePath\_ulna\_radius, 06\_doublePath\_radius and 07\_doublePath\_ulna are equivalent.)



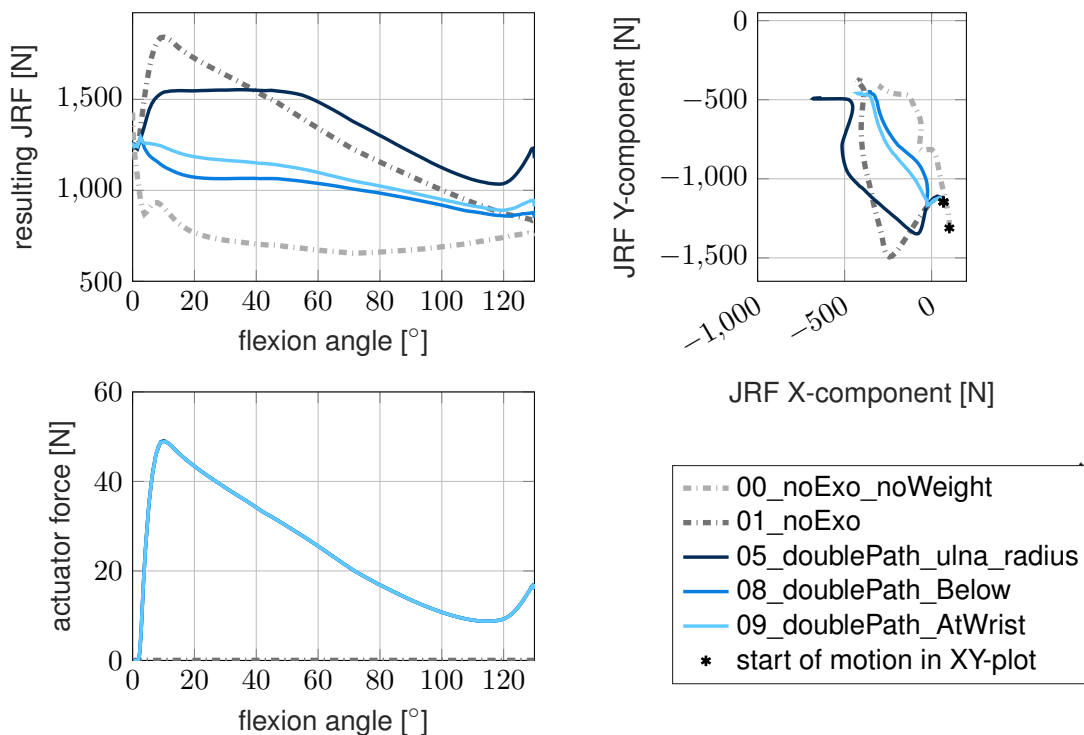
**Figure A.45:** JRFs and actuator forces of models with double-stringed path actuators (10 path points) using the CMC-optimized control. (The actuator forces of models 09\_doublePath\_AtWrist, 10\_doublePath\_AtWrist\_onlyRadius and 11\_doublePath\_AtWrist\_onlyUlna are equivalent.)



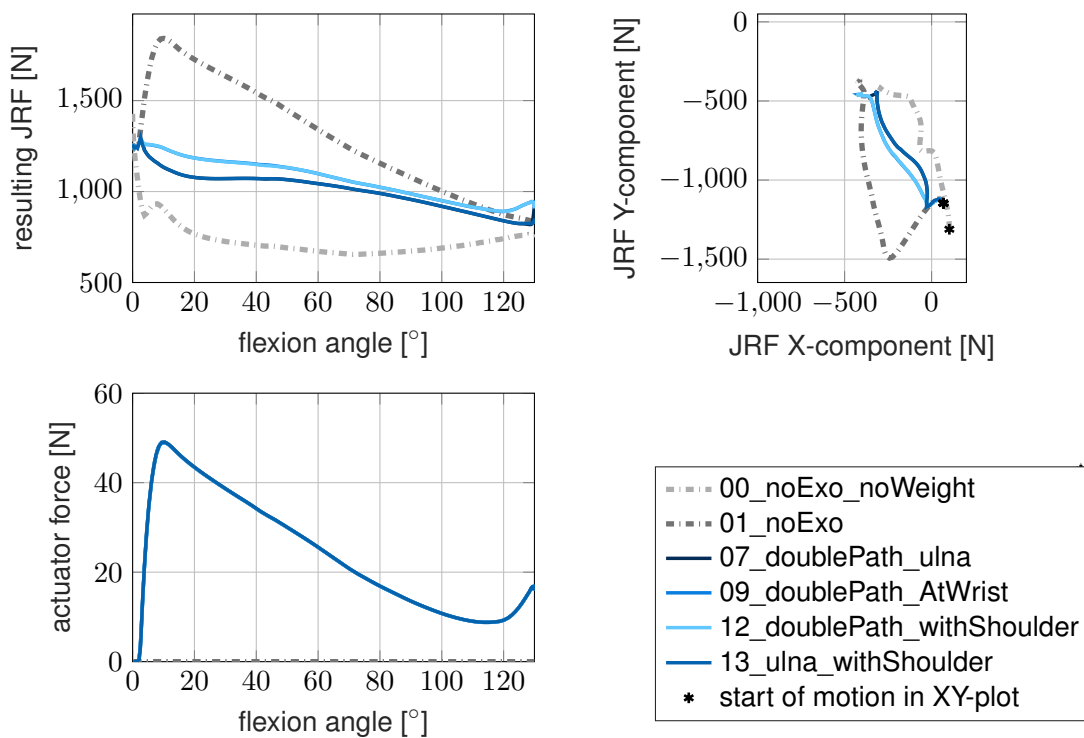
**Figure A.46:** JRFs and actuator forces of models with path actuators attached to the ulna using the CMC-optimized control. (The JRF curves of models 03\_singlePath\_ulna, 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna are (almost) equivalent. The actuator forces of models 07\_doublePath\_ulna and 11\_doublePath\_AtWrist\_onlyUlna are equivalent.)



**Figure A.47:** JRFs and actuator forces of models with path actuators attached to the radius using the CMC-optimized control. (The JRF and actuator force curves of models 06\_doublePath\_radius and 10\_doublePath\_AtWrist\_onlyRadius are equivalent.)



**Figure A.48:** JRFs and actuator forces of models with path actuators attached to the radius and the ulna using the CMC-optimized control. (The actuator forces of models 05\_doublePath\_ulna\_radius, 08\_doublePath\_connectedBelow and 09\_doublePath\_AtWrist are equivalent.)



**Figure A.49:** JRFs and actuator forces of models including path actuators using the CMC-optimized control. Models 12 and 13 are extended versions of models 7 and 9 by adding 10 more path points over the shoulder. (The JRFs of models 09\_doublePath\_AtWrist and 12\_doublePath\_withShoulder are (almost) equivalent. The JRFs of models 07\_doublePath\_ulna and 13\_ulna\_withShoulder are (almost) equivalent. The actuator forces of models 07\_doublePath\_ulna, 09\_doublePath\_AtWrist, 12\_doublePath\_withShoulder and 13 are equivalent.)



## Appendix C

Tables with minimum, mean and maximum elbow JRF values for all three control strategies - 100%-support, constant torque and CMC-optimized.

| Model                            | Min JRF [N] | Mean JRF [N] | Max JRF [N] |
|----------------------------------|-------------|--------------|-------------|
| 00_noExo_noWeight                | 655         | 785          | 1,427       |
| 01_noExo                         | 812         | 1,260        | 1,842       |
| 02_coordinateActuator            | 902         | 1,164        | 1,376       |
| 03_singlePath_ulna               | 967         | 1,286        | 1,546       |
| 04_singlePath_radius             | 1,110       | 1,268        | 1,457       |
| 05_doublePath_ulna_radius        | 1,238       | 2,069        | 2,630       |
| 06_doublePath_radius             | 1,031       | 1,290        | 1,524       |
| 07_doublePath_ulna               | 961         | 1,286        | 1,547       |
| 08_doublePath_connected-Below    | 1,060       | 1,342        | 1,598       |
| 09_doublePath_AtWrist            | 1,165       | 1,539        | 1,881       |
| 10_doublePath_AtWrist_onlyRadius | 1,031       | 1,290        | 1,524       |
| 11_doublePath_AtWrist_onlyUlna   | 961         | 1,286        | 1,547       |
| 12_doublePath_withShoulder       | 1,166       | 1,540        | 1,884       |
| 13_ulna_withShoulder             | 964         | 1,287        | 1,547       |

**Table A.5:** Table with minimum, mean and maximum elbow JRF values for each model with the 100%-support controller.

| Model                            | Min JRF [N] | Mean JRF [N] | Max JRF [N] |
|----------------------------------|-------------|--------------|-------------|
| 00_noExo_noWeight                | 655         | 785          | 1,427       |
| 01_noExo                         | 812         | 1,260        | 1,842       |
| 02_coordinateActuator            | 1,001       | 1,140        | 1,790       |
| 03_singlePath_ulna               | 1,142       | 1,264        | 1,992       |
| 04_singlePath_radius             | 1,077       | 1,265        | 1,955       |
| 05_doublePath_ulna_radius        | 2,018       | 2,362        | 3,369       |
| 06_doublePath_radius             | 1,108       | 1,277        | 1,966       |
| 07_doublePath_ulna               | 1,144       | 1,267        | 1,999       |
| 08_doublePath_connected-Below    | 1,168       | 1,327        | 2,037       |
| 09_doublePath_AtWrist            | 1,396       | 1,529        | 2,287       |
| 10_doublePath_AtWrist_onlyRadius | 1,108       | 1,277        | 1,966       |
| 11_doublePath_AtWrist_onlyUlna   | 1,144       | 1,267        | 1,999       |
| 12_doublePath_withShoulder       | 1,391       | 1,522        | 2,285       |
| 13_ulna_withShoulder             | 1,145       | 1,269        | 1,996       |

**Table A.6:** Table with minimum, mean and maximum elbow JRF values for each model with the constant torque controller.

| Model                            | Min JRF [N] | Mean JRF [N] | Max JRF [N] |
|----------------------------------|-------------|--------------|-------------|
| 00_noExo_noWeight                | 655         | 785          | 1,427       |
| 01_noExo                         | 812         | 1,260        | 1,842       |
| 02_coordinateActuator            | 781         | 956          | 1,270       |
| 03_singlePath_ulna               | 817         | 1,013        | 1,295       |
| 04_singlePath_radius             | 849         | 1,029        | 1,292       |
| 05_doublePath_ulna_radius        | 1,034       | 1,320        | 1,553       |
| 06_doublePath_radius             | 835         | 1,020        | 1,299       |
| 07_doublePath_ulna               | 818         | 1,016        | 1,291       |
| 08_doublePath_connected-Below    | 851         | 1,019        | 1,293       |
| 09_doublePath_AtWrist            | 890         | 1,075        | 1,290       |
| 10_doublePath_AtWrist_onlyRadius | 835         | 1,020        | 1,299       |
| 11_doublePath_AtWrist_onlyUlna   | 818         | 1,016        | 1,291       |
| 12_doublePath_withShoulder       | 891         | 1,075        | 1,293       |
| 13_ulna_withShoulder             | 822         | 1,014        | 1,298       |

**Table A.7:** Table with minimum, mean and maximum elbow JRF values for each model with the CMC-optimized controller.

## Appendix D

Tables with elbow JRF relief compared to model 01\_noExo.

| Model                            | Change in Min JRF [%] | Change in Mean JRF [%] | Change in Max JRF [%] |
|----------------------------------|-----------------------|------------------------|-----------------------|
| 00_noExo_noWeight                | -19.4                 | -37.7                  | -22.5                 |
| 01_noExo                         | 0.0                   | 0.0                    | 0.0                   |
| 02_coordinateActuator            | 11.0                  | -7.7                   | -25.3                 |
| 03_singlePath_ulna               | 19.0                  | 2.1                    | -16.1                 |
| 04_singlePath_radius             | 36.7                  | 0.6                    | -20.9                 |
| 05_doublePath_ulna_radius        | 52.5                  | 64.2                   | 42.8                  |
| 06_doublePath_radius             | 27.0                  | 2.4                    | -17.3                 |
| 07_doublePath_ulna               | 18.3                  | 2.1                    | -16.0                 |
| 08_doublePath_connectedBelow     | 30.6                  | 6.5                    | -13.3                 |
| 09_doublePath_AtWrist            | 43.5                  | 22.1                   | 2.1                   |
| 10_doublePath_AtWrist_onlyRadius | 27.0                  | 2.4                    | -17.3                 |
| 11_doublePath_AtWrist_onlyUlna   | 18.3                  | 2.1                    | -16.0                 |
| 12_doublePath_withShoulder       | 43.6                  | 22.2                   | 2.3                   |
| 13_ulna_withShoulder             | 18.6                  | 2.2                    | -16.0                 |

**Table A.8:** Elbow JRF relief with the 100%-support control strategy in comparison to model 01\_noExo as a percentage thereof for all OpenSim models.

| Model                            | Change in Min JRF [%] | Change in Mean JRF [%] | Change in Max JRF [%] |
|----------------------------------|-----------------------|------------------------|-----------------------|
| 00_noExo_noWeight                | -19.4                 | -37.7                  | -22.5                 |
| 01_noExo                         | 0.0                   | 0.0                    | 0.0                   |
| 02_coordinateActuator            | 23.2                  | -9.6                   | -2.8                  |
| 03_singlePath_ulna               | 40.6                  | 0.3                    | 8.1                   |
| 04_singlePath_radius             | 32.6                  | 0.4                    | 6.1                   |
| 05_doublePath_ulna_radius        | 148.4                 | 87.5                   | 82.9                  |
| 06_doublePath_radius             | 36.4                  | 1.3                    | 6.7                   |
| 07_doublePath_ulna               | 40.9                  | 0.6                    | 8.5                   |
| 08_doublePath_connectedBelow     | 43.8                  | 5.3                    | 10.6                  |
| 09_doublePath_AtWrist            | 71.9                  | 21.3                   | 24.2                  |
| 10_doublePath_AtWrist_onlyRadius | 36.4                  | 1.3                    | 6.7                   |
| 11_doublePath_AtWrist_onlyUlna   | 40.9                  | 0.6                    | 8.5                   |
| 12_doublePath_withShoulder       | 71.3                  | 20.8                   | 24.0                  |
| 13_ulna_withShoulder             | 41.0                  | 0.7                    | 8.3                   |

**Table A.9:** Elbow JRF relief with the constant torque control strategy in comparison to model 01\_noExo as a percentage thereof for all OpenSim models.

| Model                            | Change in Min JRF [%] | Change in Mean JRF [%] | Change in Max JRF [%] |
|----------------------------------|-----------------------|------------------------|-----------------------|
| 00_noExo_noWeight                | -19.4                 | -37.7                  | -22.5                 |
| 01_noExo                         | 0.0                   | 0.0                    | 0.0                   |
| 02_coordinateActuator            | -3.9                  | -24.1                  | -31.1                 |
| 03_singlePath_ulna               | 0.6                   | -19.6                  | -29.7                 |
| 04_singlePath_radius             | 4.5                   | -18.4                  | -29.9                 |
| 05_doublePath_ulna_radius        | 27.3                  | 4.7                    | -15.7                 |
| 06_doublePath_radius             | 2.8                   | -19.0                  | -29.5                 |
| 07_doublePath_ulna               | 0.8                   | -19.4                  | -29.9                 |
| 08_doublePath_connectedBelow     | 4.8                   | -19.1                  | -29.8                 |
| 09_doublePath_AtWrist            | 9.6                   | -14.7                  | -30.0                 |
| 10_doublePath_AtWrist_onlyRadius | 2.8                   | -19.0                  | -29.5                 |
| 11_doublePath_AtWrist_onlyUlna   | 0.8                   | -19.4                  | -29.9                 |
| 12_doublePath_withShoulder       | 9.7                   | -14.7                  | -29.8                 |
| 13_ulna_withShoulder             | 1.2                   | -19.6                  | -29.5                 |

**Table A.10:** Elbow JRF relief with the CMC-optimized control strategy in comparison to model 01\_noExo as a percentage thereof for all OpenSim models.

## Appendix E

Analysis of the biarticularity property of the muscles in the MobL musculoskeletal model.

| Muscle              | other | Shoulder | Humerus | Ulna | Radius | Hand | attached bodies | biarticulate (>2 bodies) | elbow flexion?        | elbow & biarticulate |
|---------------------|-------|----------|---------|------|--------|------|-----------------|--------------------------|-----------------------|----------------------|
| DEL1                | 1     | 1        | 1       | 0    | 0      | 0    | 3               | 1                        | 0                     | 0                    |
| DEL2                | 0     | 1        | 1       | 0    | 0      | 0    | 2               | 0                        | 0                     | 0                    |
| DEL3                | 0     | 1        | 1       | 0    | 0      | 0    | 2               | 0                        | 0                     | 0                    |
| SUPSP               | 0     | 1        | 1       | 0    | 0      | 0    | 2               | 0                        | 0                     | 0                    |
| INFSP               | 0     | 1        | 1       | 0    | 0      | 0    | 2               | 0                        | 0                     | 0                    |
| SUBSC               | 0     | 1        | 1       | 0    | 0      | 0    | 2               | 0                        | 0                     | 0                    |
| TMIN                | 0     | 1        | 1       | 0    | 0      | 0    | 2               | 0                        | 0                     | 0                    |
| TMAJ                | 0     | 1        | 1       | 0    | 0      | 0    | 2               | 0                        | 0                     | 0                    |
| PECM1               | 1     | 1        | 1       | 0    | 0      | 0    | 3               | 1                        | 0                     | 0                    |
| PECM2               | 1     | 1        | 1       | 0    | 0      | 0    | 3               | 1                        | 0                     | 0                    |
| PECM3               | 1     | 1        | 1       | 0    | 0      | 0    | 3               | 1                        | 0                     | 0                    |
| LAT1                | 1     | 1        | 1       | 0    | 0      | 0    | 3               | 1                        | 0                     | 0                    |
| LAT2                | 1     | 1        | 1       | 0    | 0      | 0    | 3               | 1                        | 0                     | 0                    |
| LAT3                | 1     | 1        | 1       | 0    | 0      | 0    | 3               | 1                        | 0                     | 0                    |
| CORB                | 0     | 1        | 1       | 0    | 0      | 0    | 2               | 0                        | 0                     | 0                    |
| TRIlong             | 0     | 1        | 1       | 1    | 0      | 0    | 3               | 1                        | 1                     | 1                    |
| TRIlat              | 0     | 0        | 1       | 1    | 0      | 0    | 2               | 0                        | 1                     | 0                    |
| TRImed              | 0     | 0        | 1       | 1    | 0      | 0    | 2               | 0                        | 1                     | 0                    |
| ANC                 | 0     | 0        | 1       | 1    | 0      | 0    | 2               | 0                        | 1                     | 0                    |
| SUP                 | 0     | 0        | 0       | 1    | 1      | 0    | 2               | 0                        | 0                     | 0                    |
| BIClong             | 0     | 1        | 1       | 0    | 1      | 0    | 3               | 1                        | 1                     | 1                    |
| BICshort            | 0     | 1        | 1       | 0    | 1      | 0    | 3               | 1                        | 1                     | 1                    |
| BRA                 | 0     | 0        | 1       | 1    | 0      | 0    | 2               | 0                        | 1                     | 0                    |
| BRD                 | 0     | 0        | 1       | 0    | 1      | 0    | 2               | 0                        | 1                     | 0                    |
| ECRL                | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| ECRB                | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| ECU                 | 0     | 0        | 1       | 1    | 1      | 1    | 4               | 1                        | 1                     | 1                    |
| FCR                 | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| FCU                 | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| PL                  | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| PT                  | 0     | 0        | 1       | 1    | 1      | 0    | 3               | 1                        | 1                     | 1                    |
| PQ                  | 0     | 0        | 0       | 1    | 1      | 0    | 2               | 0                        | 0                     | 0                    |
| FDSL                | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| FDSR                | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| FDSM                | 0     | 0        | 0       | 1    | 1      | 1    | 3               | 1                        | 0                     | 0                    |
| FDSI                | 0     | 0        | 0       | 1    | 1      | 1    | 3               | 1                        | 0                     | 0                    |
| FDPL                | 0     | 0        | 0       | 1    | 1      | 1    | 3               | 1                        | 0                     | 0                    |
| FDPR                | 0     | 0        | 0       | 1    | 1      | 1    | 3               | 1                        | 0                     | 0                    |
| FDPM                | 0     | 0        | 0       | 1    | 1      | 1    | 3               | 1                        | 0                     | 0                    |
| FDPI                | 0     | 0        | 0       | 1    | 1      | 1    | 3               | 1                        | 0                     | 0                    |
| EDCL                | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| EDCR                | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| EDCM                | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| EDCI                | 0     | 0        | 1       | 0    | 1      | 1    | 3               | 1                        | 1                     | 1                    |
| EDM                 | 0     | 0        | 1       | 1    | 1      | 1    | 4               | 1                        | 1                     | 1                    |
| EIP                 | 0     | 0        | 0       | 1    | 1      | 1    | 3               | 1                        | 0                     | 0                    |
| EPL                 | 0     | 0        | 0       | 1    | 1      | 1    | 3               | 1                        | 0                     | 0                    |
| EPB                 | 0     | 0        | 0       | 0    | 1      | 1    | 2               | 0                        | 0                     | 0                    |
| FPL                 | 0     | 0        | 0       | 0    | 1      | 1    | 2               | 0                        | 0                     | 0                    |
| APL                 | 0     | 0        | 0       | 0    | 1      | 1    | 2               | 0                        | 0                     | 0                    |
| 50 muscles in total |       |          |         |      |        |      |                 | 32                       | 22                    | 17                   |
|                     |       |          |         |      |        |      | That's          | 64.0%                    | of all and            | 77.3%                |
|                     |       |          |         |      |        |      |                 |                          | of all elbow muscles. |                      |

**Table A.11:** Analysis of the biarticularity property of the muscles in the MobL musculoskeletal model. "1" means true; "0" means false.

## Appendix F

**This appendix section shows the source code for the following in OpenSim:**

- Adding the body for the 5 kg weight in the OpenSim model file.
- Attaching the 5 kg weight to the hand via a WeldJoint in the OpenSim model file.
- CMC xml setup file
- JRA xml setup file
- MA xml setup file

**Adding the body for the 5 kg weight in the OpenSim model file:**

```
[...]  
<Body name="box">  
  <!--The geometry used to display the axes of this Frame.-->  
  <FrameGeometry name="frame_geometry">  
    <!--Path to a Component that satisfies the Socket 'frame' of type Frame.-->  
    <socket_frame>..</socket_frame>  
    <!--Scale factors in X, Y, Z directions respectively.-->  
    <scale_factors>0.20000000000000001 0.20000000000000001 0.20000000000000001</  
      scale_factors>  
  </FrameGeometry>  
  <!--List of geometry attached to this Frame. Note, the geometry are treated  
    as fixed to the frame and they share the transform of the frame when  
    visualized-->  
  <attached_geometry>  
    <Mesh name="box_geom_1">  
      <!--Path to a Component that satisfies the Socket 'frame' of type Frame.-->  
      <socket_frame>..</socket_frame>  
      <!--Scale factors in X, Y, Z directions respectively.-->  
      <scale_factors>0.050000000000000003 0.050000000000000003 0.050000000000000003  
        </scale_factors>  
      <!--Default appearance attributes for this Geometry-->  
      <Appearance>  
        <!--The opacity used to display the geometry between 0:transparent, 1:opaque.  
          -->  
        <opacity>1</opacity>  
        <!--The color, (red, green, blue), [0, 1], used to display the geometry. -->  
        <color>1 1 1</color>  
      </Appearance>
```



```

<!--Name of geometry file.-->
<mesh_file>sphere.vtp</mesh_file>
</Mesh>
</attached_geometry>
<!--Set of wrap objects fixed to this body that GeometryPaths can wrap over.
This property used to be a member of Body but was moved up with the
introduction of Frames.-->
<WrapObjectSet name="wrapobjectset">
<!--All properties of this object have their default values.-->
</WrapObjectSet>
<!--The mass of the body (kg)-->
<mass>5</mass>
<!--The location (Vec3) of the mass center in the body frame.-->
<mass_center>0 0 0</mass_center>
<!--The elements of the inertia tensor (Vec6) as [Ixx Iyy Izz Ixy Ixz Iyz]
measured about the mass_center and not the body origin.-->
<inertia>0.010985 0.010985 0.010985 0 0 0</inertia>
</Body>
[...]
```

#### Attaching the 5 kg weight to the hand via a weldJoint in the OpenSim model file:

```

[...]
```

```

<WeldJoint name="weight\_grip">
<!--Path to a Component that satisfies the Socket 'parent_frame' of type
PhysicalFrame (description: The parent frame for the joint).-->
<socket_parent_frame>box_hand_offset</socket_parent_frame>
<!--Path to a Component that satisfies the Socket 'child_frame' of type
PhysicalFrame (description: The child frame for the joint).-->
<socket_child_frame>box_offset</socket_child_frame>
<!--Physical offset frames owned by the Joint that are typically used to
satisfy the owning Joint's_parent_and_child_frame_connections_(sockets).
PhysicalOffsetFrames_are_often_used_to_describe_the_fixed_transformation_
from_a_Body's origin to another location of interest on the Body (e.g.,
the joint center). When the joint is deleted, so are the
PhysicalOffsetFrame components in this list.-->
<frames>
<PhysicalOffsetFrame name="box_hand_offset">
<!--The geometry used to display the axes of this Frame.-->
<FrameGeometry name="frame_geometry">
<!--Path to a Component that satisfies the Socket 'frame' of type Frame.-->
<socket_frame>..</socket_frame>
```

```

<!--Scale factors in X, Y, Z directions respectively.-->
<scale_factors>0.20000000000000001 0.20000000000000001 0.20000000000000001</
  scale_factors>
</FrameGeometry>
<!--Path to a Component that satisfies the Socket 'parent' of type C (
  description: The parent frame to this frame.)-->
<socket_parent>/bodyset/hand</socket_parent>
<!--Translational offset (in meters) of this frame's_origin_from_the_parent_
  frame's origin , expressed in the parent frame.-->
<translation>0.02 -0.05000000000000003 -0.05000000000000003</translation>
<!--Orientation offset (in radians) of this frame in its parent frame,
  expressed as a frame-fixed x-y-z rotation sequence.-->
<orientation>0 0 0</orientation>
</PhysicalOffsetFrame>
<PhysicalOffsetFrame name="box_offset">
<!--The geometry used to display the axes of this Frame.-->
<FrameGeometry name="frame_geometry">
<!--Path to a Component that satisfies the Socket 'frame' of type Frame.-->
<socket_frame>..</socket_frame>
<!--Scale factors in X, Y, Z directions respectively.-->
<scale_factors>0.20000000000000001 0.20000000000000001 0.20000000000000001</
  scale_factors>
</FrameGeometry>
<!--Path to a Component that satisfies the Socket 'parent' of type C (
  description: The parent frame to this frame.)-->
<socket_parent>/bodyset/box</socket_parent>
<!--Translational offset (in meters) of this frame's_origin_from_the_parent_
  frame's origin , expressed in the parent frame.-->
<translation>0 0 0</translation>
<!--Orientation offset (in radians) of this frame in its parent frame,
  expressed as a frame-fixed x-y-z rotation sequence.-->
<orientation>0 0 0</orientation>
</PhysicalOffsetFrame>
</frames>
</WeldJoint>
[ ... ]

```

**CMC xml setup file:**

```

<?xml version="1.0" encoding="UTF-8" ?>
<OpenSimDocument Version="30000">
<CMCTool name="CMC_results">

```

```

<!--Name of the .osim file used to construct a model.-->
<model_file />
<!--Replace the model's_force_set_with_sets_specified_in_<force_set_files>?_
    If_false,_the_force_set_is_appended_to.-->
<replace_force_set>>false</replace_force_set>
<!--List_of_xml_files_used_to_construct_a_force_set_for_the_model.-->
<force_set_files/>
<!--Directory_used_for_writing_results.-->
<results_directory>Results-CMC</results_directory>
<!--Output_precision._It_is_8_by_default.-->
<output_precision>20</output_precision>
<!--Initial_time_for_the_simulation.-->
<initial_time>0</initial_time>
<!--Final_time_for_the_simulation.-->
<final_time>0.18</final_time>
<!--Flag_indicating_whether_or_not_to_compute_equilibrium_values_for_states_
    other_than_the_coordinates_or_speeds._For_example,_equilibrium_muscle_
    fiber_lengths_or_muscle_forces.-->
<solve_for_equilibrium_for_auxiliary_states>>true</
    solve_for_equilibrium_for_auxiliary_states>
<!--Maximum_number_of_integrator_steps.-->
<maximum_number_of_integrator_steps>300000000</
    maximum_number_of_integrator_steps>
<!--Maximum_integration_step_size.-->
<maximum_integrator_step_size>1</maximum_integrator_step_size>
<!--Minimum_integration_step_size.-->
<minimum_integrator_step_size>0.0001</minimum_integrator_step_size>
<!--Integrator_error_tolerance._When_the_error_is_greater,_the_integrator_
    step_size_is_decreased.-->
<integrator_error_tolerance>0.0005</integrator_error_tolerance>
<!--Set_of_analyses_to_be_run_during_the_investigation.-->
<AnalysisSet_name="Analyses">
<objects>
<Kinematics_name="Kinematics">
<!--Names_of_generalized_coordinates_whose_kinematics_are_to_be_recorded.-->
<coordinates>all</coordinates>
<!--Flag_(true_or_false)_specifying_whether_on._True_by_default.-->
<on>>true</on>
<!--Start_time.-->
<start_time>0</start_time>
<!--End_time.-->
<end_time>0.18</end_time>

```

```

<!-- Specifies how often to store results during a simulation. More
specifically, the interval (a positive integer) specifies how many
successful integration steps should be taken before results are recorded
again.-->
<step_interval>10</step_interval>
<!-- Flag (true or false) indicating whether the results are in degrees or not
.-->
<in_degrees>true </in_degrees>
</Kinematics>
<Actuation_name="Actuation">
<!-- Flag (true or false) specifying whether on. True by default.-->
<on>true </on>
<!-- Start time.-->
<start_time>0</start_time>
<!-- End time.-->
<end_time>0.18</end_time>
<!-- Specifies how often to store results during a simulation. More
specifically, the interval (a positive integer) specifies how many
successful integration steps should be taken before results are recorded
again.-->
<step_interval>10</step_interval>
<!-- Flag (true or false) indicating whether the results are in degrees or not
.-->
<in_degrees>true </in_degrees>
</Actuation>
</objects>
<groups_/>
</AnalysisSet>
<!-- Controller objects in the model.-->
<ControllerSet_name="Controllers">
<objects_/>
<groups_/>
</ControllerSet>
<!--XML file (.xml) containing the forces applied to the model as
ExternalLoads.-->
<external_loads_file_/>
<!--Motion (.mot) or storage (.sto) file containing the desired point
trajectories.-->
<desired_points_file_/>
<!--Motion (.mot) or storage (.sto) file containing the desired kinematic
trajectories.-->
<desired_kinematics_file>mot_0to130_1sec_100Hz.mot</desired_kinematics_file>

```

```

<!-- File containing the tracking tasks. Which coordinates are tracked and
      with what weights are specified here.-->
<task_set_file>CMC_Tasks.xml</task_set_file>
<!-- File containing the constraints on the controls.-->
<constraints_file>CMC_ControlConstraints.xml</constraints_file>
<!-- File containing the controls output by RRA. These can be used to place
      constraints on the residuals during CMC.-->
<rra_controls_file />
<!-- Low-pass cut-off frequency for filtering the desired kinematics. A
      negative value results in no filtering. The default value is -1.0, so no
      filtering.-->
<lowpass_cutoff_frequency>-1</lowpass_cutoff_frequency>
<!-- Time window over which the desired actuator forces are achieved. Muscles
      forces cannot change instantaneously, so a finite time window must be
      allowed. The recommended time window for RRA is about 0.001 sec, and for
      CMC is about 0.010 sec.-->
<cmc_time_window>0.01</cmc_time_window>
<!-- Flag (true or false) indicating whether to use the fast CMC optimization
      target. The fast target requires the desired accelerations to be met. The
      optimizer fails if the accelerations constraints cannot be met, so the
      fast target can be less robust. The regular target does not require the
      acceleration constraints to be met; it meets them as well as it can, but
      it is slower and less accurate.-->
<use_fast_optimization_target>>false</use_fast_optimization_target>
<!-- Preferred optimizer algorithm (currently support "ipopt" or "cfsqp", the
      latter requiring the osimCFSQP library.-->
<optimizer_algorithm>ipopt</optimizer_algorithm>
<!-- Step size used by the optimizer to compute numerical derivatives. A value
      between 1.0e-4 and 1.0e-8 is usually appropriate.-->
<numerical_derivative_step_size>0.0001</numerical_derivative_step_size>
<!-- Convergence tolerance for the optimizer. The smaller this value, the
      deeper the convergence. Decreasing this number can improve a solution,
      but will also likely increase computation time.-->
<optimization_convergence_tolerance>0.0001</
      optimization_convergence_tolerance>
<!-- Maximum number of iterations for the optimizer.-->
<optimizer_max_iterations>2000</optimizer_max_iterations>
<!-- Print level for the optimizer, 0-3. 0=no printing, 3=detailed printing,
      2=in between-->
<optimizer_print_level>0</optimizer_print_level>
<!-- True-false flag indicating whether or not to turn on verbose printing for
      _cmc.-->

```

```

<use_verbose_printing>>false </use_verbose_printing>
</CMCTool>
</OpenSimDocument>

```

**JRA xml setup file:**

```

<?xml version="1.0" encoding="UTF-8" ?>
<OpenSimDocument Version="40000">
<AnalyzeTool name="JRAanalysis">
<!--Name of the .osim file used to construct a model.-->
<model_file />
<!--Replace the model's force_set_with_sets specified in <force_set_files>?_
If false , the force_set is appended to.-->
<replace_force_set>>false </replace_force_set>
<!--List of xml files used to construct a force_set for the model.-->
<force_set_files />
<!--Directory used for writing results.-->
<results_directory>Results-CMC</results_directory>
<!--Output precision . It is 8 by default.-->
<output_precision>20</output_precision>
<!--Initial time for the simulation.-->
<initial_time>0</initial_time>
<!--Final time for the simulation.-->
<final_time>0.9899999999999999</final_time>
<!--Flag indicating whether or not to compute equilibrium values for states_
other than the coordinates or speeds . For example , equilibrium_muscle_
fiber_lengths or muscle_forces.-->
<solve_for_equilibrium_for_auxiliary_states>>false </
solve_for_equilibrium_for_auxiliary_states>
<!--Maximum number of integrator steps.-->
<maximum_number_of_integrator_steps>20000</maximum_number_of_integrator_steps
>
<!--Maximum integration step size.-->
<maximum_integrator_step_size>1</maximum_integrator_step_size>
<!--Minimum integration step size.-->
<minimum_integrator_step_size>1e-08</minimum_integrator_step_size>
<!--Integrator error tolerance . When the error is greater , the integrator_
step_size is decreased.-->
<integrator_error_tolerance>1.0000000000000001e-05</
integrator_error_tolerance>
<!--Set of analyses to be run during the investigation.-->
<AnalysisSet_name="Analyses">

```

```

<objects >
<JointReaction_name="JointReaction">
<!--Flag_(true_or_false)_specifying_whether_on_._True_by_default.-->
<on>true </on>
<!-- Start_time.-->
<start_time >0</start_time >
<!--End_time.-->
<end_time>0.9899999999999999</end_time>
<!-- Specifies_how_often_to_store_results_during_a_simulation_._More_
specifically_ ,_the_interval_(a_positive_integer)_specifies_how_many_
successful_integration_steps_should_be_taken_before_results_are_recorded_
again.-->
<step_interval >1</step_interval >
<!--Flag_(true_or_false)_indicating_whether_the_results_are_in_degrees_or_not_
.-->
<in_degrees>true </in_degrees>
<!--The_name_of_a_file_containing_forces_storage_._If_a_file_name_is_provided_ ,
_the_forces_for_all_actuators_will_be_applied_according_to_values_
specified_in_the_forces_file_instead_of_being_computed_from_the_states_._
This_option_should_be_used_to_calculate_joint_reactions_from_static_
optimization_results.-->
<forces_file >C:\Users\MC_Schniggelzzz\Dropbox\Uni\Masterarbeit_MA\Simulation\
Results-CMC\CMC_Actuation_force.sto </forces_file >
<!--Names_of_the_joints_on_which_to_perform_the_analysis_._The_key_word_'All'_
indicates_that_the_analysis_should_be_performed_for_all_joints.-->
<joint_names>_ALL</joint_names>
<!--Choice_of_body_('parent'_or_'child')_for_which_the_reaction_loads_are_
calculated_._Child_body_is_default_._The_array_must_either_have_one_entry_
or_the_same_number_of_entries_as_joints_specified_above_._If_the_array_has_
one_entry_only ,_that_selection_is_applied_to_all_chosen_joints.-->
<apply_on_bodies>_child </apply_on_bodies>
<!--Names_of_frames_in_which_the_calculated_reactions_are_expressed ,_or_the_
keyword_'child'_or_'parent'_to_indicate_the_joint's 'child' or 'parent'
Frame. ground is default. If a Frame named 'child' or 'parent' exists and
the keyword 'child' or 'parent' is used, the analysis will use that
Frame. The array must either have one entry or the same number of entries
as joints specified above. If the array has one entry only, that
selection is applied to all chosen joints.-->
<express_in_frame> child</express_in_frame>
</JointReaction>
</objects>
<groups />

```

```

</AnalysisSet>
<!-- Controller objects in the model.-->
<ControllerSet name="Controllers">
<objects />
<groups />
</ControllerSet>
<!--XML file (.xml) containing the forces applied to the model as
      ExternalLoads.-->
<external_loads_file />
<!--Storage file (.sto) containing the time history of states for the model.
      This file often contains multiple rows of data, each row being a time-
      stamped array of states. The first column contains the time. The rest of
      the columns contain the states in the order appropriate for the model.
      In a storage file, unlike a motion file (.mot), non-uniform time spacing
      is allowed. If the user-specified initial time for a simulation does not
      correspond exactly to one of the time stamps in this file, interpolation
      is NOT used because it is sometimes necessary to use an exact set of
      states for analyses. Instead, the closest earlier set of states is used.
      -->
<states_file />
<!--Motion file (.mot) or storage file (.sto) containing the time history of
      the generalized coordinates for the model. These can be specified in
      place of the states file.-->
<coordinates_file>mot_0to130_1sec_100Hz.mot</coordinates_file>
<!--Storage file (.sto) containing the time history of the generalized speeds
      for the model. If coordinates_file is used in place of states_file,
      these can be optionally set as well to give the speeds. If not specified,
      speeds will be computed from coordinates by differentiation.-->
<speeds_file />
<!--Low-pass cut-off frequency for filtering the coordinates_file data (
      currently does not apply to states_file or speeds_file). A negative value
      results in no filtering. The default value is -1.0, so no filtering.-->
<lowpass_cutoff_frequency_for_coordinates>-1</
      lowpass_cutoff_frequency_for_coordinates>
</AnalyzeTool>
</OpenSimDocument>

```

#### MA xml setup file:

```

<?xml version="1.0" encoding="UTF-8" ?>
<OpenSimDocument Version="40000">
<AnalyzeTool name="MuscleAnalysis">

```



```

<!--Name of the .osim file used to construct a model.-->
<model_file>64_prescribed_singlePathActuator_radius_asMuscle_INCOMPLETE.osim<
  /model_file>
<!--Replace the model's force_set with sets specified in <force_set_files>?_
  If false, the force_set is appended to.-->
<replace_force_set>>false </replace_force_set>
<!--List of xml files used to construct a force_set for the model.-->
<force_set_files_ />
<!--Directory used for writing results.-->
<!--_EDIT_(NN) :_"Original"_Full_path:_C:\Users\MC_Schniggelzzz\Dropbox\Uni\
  Masterarbeit_MA\Simulation\Results\MuscleAnalysis_-->
<results_directory>Results\MuscleAnalysis</results_directory>
<!--Output_precision. It is 8 by default.-->
<output_precision>20</output_precision>
<!--Initial_time_for_the_simulation.-->
<initial_time>0</initial_time>
<!--Final_time_for_the_simulation.-->
<final_time>0.9899999999999999</final_time>
<!--Flag_indicating_whether_or_not_to_compute_equilibrium_values_for_states_
  other_than_the_coordinates_or_speeds. For example, equilibrium_muscle_
  fiber_lengths_or_muscle_forces.-->
<solve_for_equilibrium_for_auxiliary_states>>false </
  solve_for_equilibrium_for_auxiliary_states>
<!--Maximum_number_of_integrator_steps.-->
<maximum_number_of_integrator_steps>20000</maximum_number_of_integrator_steps
  >
<!--Maximum_integration_step_size.-->
<maximum_integrator_step_size>1</maximum_integrator_step_size>
<!--Minimum_integration_step_size.-->
<minimum_integrator_step_size>1e-08</minimum_integrator_step_size>
<!--Integrator_error_tolerance. When the error is greater, the integrator_
  step_size_is_decreased.-->
<integrator_error_tolerance>1.0000000000000001e-05</
  integrator_error_tolerance>
<!--Set_of_analyses_to_be_run_during_the_investigation.-->
<AnalysisSet_name="Analyses">
<objects>
<JointReaction_name="JointReaction">
<!--Flag_(true_or_false)_specifying_whether_on. True_by_default.-->
<on>>false </on>
<!--Start_time.-->
<start_time>0</start_time>

```

```

<!--End_time.-->
<end_time>0.9899999999999999</end_time>
<!-- Specifies how often to store results during a simulation. More
specifically, the interval (a positive integer) specifies how many
successful integration steps should be taken before results are recorded
again.-->
<step_interval>1</step_interval>
<!-- Flag (true or false) indicating whether the results are in degrees or not
.-->
<in_degrees>true</in_degrees>
<!--The name of a file containing forces storage. If a file name is provided,
the forces for all actuators will be applied according to values
specified in the forces file instead of being computed from the states.
This option should be used to calculate joint reactions from static
optimization results.-->
<forces_file>C:\Users\MC_Schniggelzzz\Dropbox\Uni\Masterarbeit_MA\Simulation\
Results-CMC\CMC_Actuation_force.sto</forces_file>
<!--Names of the joints on which to perform the analysis. The key word 'All'
indicates that the analysis should be performed for all joints.-->
<joint_names>_ALL</joint_names>
<!--Choice of body ('parent' or 'child') for which the reaction loads are
calculated. Child body is default. The array must either have one entry
or the same number of entries as joints specified above. If the array has
one entry only, that selection is applied to all chosen joints.-->
<apply_on_bodies>_child</apply_on_bodies>
<!--Names of frames in which the calculated reactions are expressed, or the
keyword 'child' or 'parent' to indicate the joint's 'child' or 'parent'
Frame. ground is default. If a Frame named 'child' or 'parent' exists and
the keyword 'child' or 'parent' is used, the analysis will use that
Frame. The array must either have one entry or the same number of entries
as joints specified above. If the array has one entry only, that
selection is applied to all chosen joints.-->
<express_in_frame> child</express_in_frame>
</JointReaction>
<MuscleAnalysis name="MuscleAnalysis">
<!--Flag (true or false) specifying whether on. True by default.-->
<on>true</on>
<!--Start time.-->
<start_time>0</start_time>
<!--End time.-->
<end_time>0.9899999999999999</end_time>

```

```

<!--Specifies how often to store results during a simulation. More
specifically, the interval (a positive integer) specifies how many
successful integration steps should be taken before results are recorded
again.-->
<step_interval>1</step_interval>
<!--Flag (true or false) indicating whether the results are in degrees or not
.-->
<in_degrees>>true</in_degrees>
<!--List of muscles for which to perform the analysis. Use 'all' to perform
the analysis for all muscles.-->
<muscle_list> all</muscle_list>
<!--List of generalized coordinates for which to compute moment arms. Use '
all' to compute for all coordinates.-->
<moment_arm_coordinate_list> r_elbow_flex</moment_arm_coordinate_list>
<!--Flag indicating whether moment-arms and/or moments should be computed.-->
<compute_moments>true</compute_moments>
</MuscleAnalysis>
</objects>
<groups />
</AnalysisSet>
<!--Controller objects in the model.-->
<ControllerSet name="Controllers">
<objects />
<groups />
</ControllerSet>
<!--XML file (.xml) containing the forces applied to the model as
ExternalLoads.-->
<external_loads_file />
<!--Storage file (.sto) containing the time history of states for the model.
This file often contains multiple rows of data, each row being a time-
stamped array of states. The first column contains the time. The rest of
the columns contain the states in the order appropriate for the model.
In a storage file, unlike a motion file (.mot), non-uniform time spacing
is allowed. If the user-specified initial time for a simulation does not
correspond exactly to one of the time stamps in this file, interpolation
is NOT used because it is sometimes necessary to use an exact set of
states for analyses. Instead, the closest earlier set of states is used.
-->
<states_file />
<!--Motion file (.mot) or storage file (.sto) containing the time history of
the generalized coordinates for the model. These can be specified in
place of the states file.-->

```

```
<coordinates_file>mot_0to130_1sec_100Hz_with70ProSup.mot</coordinates_file>
<!--Storage file (.sto) containing the time history of the generalized speeds
      for the model. If coordinates_file is used in place of states_file ,
      these can be optionally set as well to give the speeds. If not specified ,
      speeds will be computed from coordinates by differentiation.-->
<speeds_file />
<!--Low-pass cut-off frequency for filtering the coordinates_file data (
      currently does not apply to states_file or speeds_file). A negative value
      results in no filtering. The default value is -1.0, so no filtering.-->
<lowpass_cutoff_frequency_for_coordinates>-1</
      lowpass_cutoff_frequency_for_coordinates>
</AnalyzeTool>
</OpenSimDocument>
```

## Appendix G

This appendix section shows the source code for the following MATLAB scripts and functions:

- createPrescribedFiles.m
- compareModels.m
- CMCrunner.m
- JRArunner.m
- evaluateJRFs.m

The auxiliary functions used in them can be found in the digital appendix of this thesis.

### **createPrescribedFiles.m**

```
function [varargout] = createPrescribedFiles(varargin)
%% createPrescribedFiles
% DESCRIPTION:
%           This script uses a prescribed torque curve and calculates
prescribed forces for an
%           "exoActuator" according to the moment arms on the elbow flexion
coordinate. The latter
%           is computed via an OpenSim Muscle Analysis for each step of the
motion.
%           It is meant to be run in the directory 'Simulation'.
%
% INPUT:   varargout – distributed over the following:
%           motionFile – string of motion file to be used
%           modelNames – list of models (similar to compareModels)
%                   don't include the ".osim"!
%           torqueCurveFile – file with a given torque curve (as table
%                   variable "exoActuator")
%           outputPath – path where all files will be placed:
%                   * subfolder(s) with moment arm files
%                   * prescribedControl files
%                   * parameters file
%
% OUTPUT:  outputFiles – file names (with .sto) of prescribedControl files
```

%% ACTUAL CODE %%

```
addpath(genpath('MATLAB_functions'));

%% read inputs
switch nargin
case 0
motionFile = 'mot_0to130_1sec_100Hz.mot';
modelNameNames = { ...
    '02_coordinateActuator'; ...
    '03_singlePath_ulna'; ...
    '04_singlePath_radius'; ...
    '05_doublePath_ulna_radius'; ...
    '06_doublePath_radius'; ...
    '07_doublePath_ulna'; ...
    '08_doublePath_connectedBelow'; ...
    '09_doublePath_connectedAtWrist'; ...
    '10_doublePath_connectedAtWrist_onlyRadius'; ...
    '11_doublePath_connectedAtWrist_onlyUlna'; ...
    '12_doublePath_withShoulder'; ...
    '13_ulna_withShoulder' ...
};
    % don't forget to delete the last ";" when copying from
    compareModels.m
outputPath = ['PrescribedController_archive/prescribed_', datestr(now, '
ymmdd_HHMM')];
%torqueCurveFile = 'PrescribedController_archive/torqueCurves/
Torque_0to130_100of100percent.sto'; % XX can be 40/60/80/100
%torqueCurveFile = 'PrescribedController_archive/torqueCurves/
Torque_0to130_100of100percent.sto'; % meaning 100% support of the 5 kg
weight ( same as Torque_fromID_0to130_noArmMass.sto )
%torqueCurveFile = 'PrescribedController_archive/torqueCurves/
Torque_const_10Nm.sto'; % constant torque
torqueCurveFile = 'PrescribedController_archive/torqueCurves/
Torque_fromCMC_0to130_withHugeForce.sto'; % CMC-optimized torque

case 4 % for a new motion file or model (but torque curve exists – otherwise
create one!)
motionFile = varargin{1};
modelNameNames = varargin{2}; % don't include the ".osim"!
outputPath = varargin{3};
torqueCurveFile = varargin{4};
```

**end**

*%% create output directory & save parameters*

**if** ~**exist**(outputPath, 'dir') *% create folder, if it doesn't exist yet*  
mkdir(outputPath);

**end**

currentTimeString = [ '(' , datestr(now, 'HH') , ':' , datestr(now, 'MM') , ')' ' ];

**disp**( [ ' Script\_has\_started\_@\_ ' , currentTimeString , '\_... ' ])

*% save parameters*

parameters.motion = motionFile;

parameters.torque = torqueCurveFile;

parameters.models = modelNames;

**disp**( ' Prescribed\_Files\_are\_being\_created\_with\_these\_parameters: ' );

**disp**(parameters);

**save**( [ outputPath , '/parameters.mat' ] , 'parameters' );

*%% MuscleAnalysis: compute & save moment arms as file*

*% using only models and motion*

*% & dump in the right folder*

**for** m = 1:**length**(modelNames) *% for all models*

**if** **strcmp**(modelNames{m}(1:2), '02') *% only for coordAct (model 02)*

momentArmFiles{m} = '0';

**else**

resultsPaths{m} = [ outputPath , '/' , modelNames{m}(1:**end**) ];

mkdir(resultsPaths{m}); *% create the directory*

*% this needs to be done with the "1X...\_asMuscle" (or previously "8X...")  
models (thus the "+1")*

*% (HINT: The "asMuscle" versions always had the same numbering +10.)*

modelNamesAsMuscle{m} = [ **num2str**(**str2num**(modelNames{m}(1))+1) , ...

modelNames{m}(2:**end**) , '\_asMuscle.osim' ];

runMuscleAnalysis(modelNamesAsMuscle{m}, motionFile, resultsPaths{m});

momentArmFiles{m} = [ resultsPaths{m} , '/'

MuscleAnalysis\_MuscleAnalysis\_MomentArm\_r\_elbow\_flex.sto' ];

**end**

**end**

```

%% compute actuator / cable force & save as file
for m = 1:length(modelNames)
    if strcmp(modelNames{m}, '52_prescribed_coordinateActuator') % this one needs
        the torqueHandle (not the cable force)
        [~,~,exoActFunctionHandle] = computeCableForce(torqueCurveFile);
    elseif strcmp(modelNames{m}, '72_coordinateActuator') % this one needs the
        torqueHandle (not the cable force)
        [~,~,exoActFunctionHandle] = computeCableForce(torqueCurveFile);
    elseif strcmp(modelNames{m}, '02_coordinateActuator') % this one needs the
        torqueHandle (not the cable force)
        [~,~,exoActFunctionHandle] = computeCableForce(torqueCurveFile);
    else
        [~,exoActFunctionHandle] = computeCableForce(torqueCurveFile, momentArmFiles{m}
            });
    end

    prescribedFileNames{m} = ['prescribedControl_', motionFile(1:end-4), '_',
        modelNames{m}(1:2), '.sto'];
    motionGenerator(prescribedFileNames{m}, {exoActFunctionHandle}, {'exoActuator
        '}, outputPath);

    disp(['Prescribed_force_files_for_model_', modelNames{m}, '_was_created.'])
end

%% report names of prescribedControl files
% these can then be copied into/used in "compareModels.m"

showPrescribedNames(prescribedFileNames);
disp(['figureTitle_=_', ''', torqueCurveFile, ''', ';'']); % can be used in "
    compareModels.m" to name the figure
save([outputPath, '/prescribedFileNames.mat'], 'prescribedFileNames');

varargout{1} = prescribedFileNames;

currentTimeString = ['(', datestr(now, 'HH'), ':', datestr(now, 'MM'), ')'];
disp(['Script_finished_@_(', currentTimeString, ')'])

end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



```
%% AUXILIARY FUNCTIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [] = runMuscleAnalysis(modelPath, motionFile, outputPath)
% copy lines from JRRunner.m
```

```
import org.opensim.modeling.*
ModelVisualizer.addDirToGeometrySearchPaths(cd);
```

```
analyzeTool = AnalyzeTool('Setup_MuscleAnalysis.xml', false);
```

```
% edit analysis settings
analyzeTool.setCoordinatesFileName(motionFile);
analyzeTool.setFinalTime(createFinalTime(motionFile));
```

```
% set up model
currentModel = Model(modelPath);
currentModel.initSystem();
analyzeTool.setModel(currentModel);
analyzeTool.setLoadModelAndInput(true);
```

```
analyzeTool.setResultsDir(outputPath);
```

```
% Run analysis
analyzeTool.run();
```

```
end
```

```
function [] = showPrescribedNames(prescribedFileNames)
```

```
disp('
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

The following text can be copied into "compareModels.m" for the ', ...
'prescribedFiles_as-is:')

```

```
disp('_')
```

```
disp(['prescribedFiles (03: ', num2str(03+length(prescribedFileNames)), ', 1)_{_
... '])
```

```

% <-- this needs to be adjusted for models "XX_prescribed_..." (73 ->
    53 or -> 03)

for k = 1:length(prescribedFileNames)
    %disp(prescribedFileNames{k}) % just the name
    disp(['''',prescribedFileNames{k},''',';_... ']) % copyable version
end
disp(['''', 'placeholder',''','; ']);
disp('
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
')

end

```

## **compareModels.m**

```
function [] = compareModels()  
%% Description compareModels: run CMC, JRA and evaluate  
% This script runs computed muscle control (CMC) and joint reaction analysis  
  (JRA) for  
% selected models in OpenSim. Finally, it displays the resulting elbow JRFs  
% and exoskeleton actuator forces over time in 2D plots.  
%  
% INPUT / PARAMETERS  
% User Input happens via selection within the "Parameters" section.  
%  
% REQUIRED FILES  
% The script should be in the folder "Simulation". Other files & folders  
% should be present in it as well:  
% - folder "Results", where simulation and plot data will be stored  
% - folder "MATLAB_functions" with all (sub-)referenced functions  
% - motion ("mot_0to130_1sec_100Hz.mot") and osim model files (see list  
  below)  
% - "CMC_setup_NN_OpenSim3" & corresponding files (Reserve_Actuators,  
%   ControlConstraints, Tasks)  
% - JRA setup file (Setup_JRA_childRef_NN_CMC.xml)  
% - (optional) "prescribedControl" exoActuator files (.sto format), when  
%   they are to be used  
%  
%  
% online reference for linking MATLAB to OpenSim:  
% https://simtk-confluence.stanford.edu/display/OpenSim/Scripting+with+Matlab  
  
%% start (initialize script/function)  
for k = 1  
clear, close all;  
comparisonTime = tic;  
% add OpenSim MATLAB code files to path  
% HINT: When opening this file on a new PC, add your path here:  
  
try %@ my PC  
addpath(genpath('C:\Users\MC_Schniggelzzz\Documents\OpenSim\4.2\Code\Matlab'))  
)  
end  
try %@ Workstation / remote Desktop  
addpath(genpath('C:\Users\Niessen\Documents\OpenSim\4.2\Code\Matlab'))
```

**end**

*% add other paths*

`addpath('MATLAB_functions')`

`set(groot, 'defaultAxesColorOrder', 'remove')` *% reset to default MATLAB color scheme*

*% optional OpenSim log in MATLAB:*

`% Logger.addSink(JavaLogSink())`

**end**

*%% Parameters*

*% select models to simulate:*

`whichModels = [13];` *% e.g. [0 1 2 3 4 5 6 7 8 9 10 11 12], ...*

*% see list of names/numbering below or in folder "Simulation"*

`modelSelector = whichModels + 1;` *% matlab indexing starts at 1, the names start at "00\_..."*

*% if more than 7 models are used, one can adjust the default color order:*

*[https://de.mathworks.com/help/releases/R2019a/matlab/ref/matlab.graphics.axis.axes-properties.html#budumk7\\_sep\\_shared-ColorOrder](https://de.mathworks.com/help/releases/R2019a/matlab/ref/matlab.graphics.axis.axes-properties.html#budumk7_sep_shared-ColorOrder)*

**if** `length(whichModels) > 7`, `provideMoreColors()`, **end**

*% select MOTION file name*

`motion = 'mot_0to130_1sec_100Hz.mot';` *% "default"*

`%motion = 'mot_0to130_1sec_100Hz_start.mot';` *% "...\_start/middle/end/firstHalf"*

`%motion = 'mot_90const_short_100Hz.mot';` *% constant*

*% Choose a musculoskeletalModel (required for JRA setup adjustment)*

`muscleModel = 'MoBL';`

`%muscleModel = 'Arm26';` *% requires analysisType = 'separated'*

*% Choose an analysis type (separated or combined)*

`analysisType = 'separated';` *% preferred*

`%analysisType = 'combined';` *% so far only implemented for SO*

```

% Choose an algorithm for calculating the muscle forces (CMC or SO)
algorithmMuscles = 'CMC';
%algorithmMuscles = 'SO';

% if prescribed controllers are used for exoActuator, provide their file
names
% below in the variable "prescribedFiles"
% (one for each model / simulation in variable "whichModels")
% The names can be automatically generated in the "createPrescribedFiles.m"
script.

%% initialize paths & files according to selected Models
% all available MODEL NAMES:
modelName = { ...
    '00_noExo_noWeight'; ...
    '01_noExo'; ...
    '02_coordinateActuator'; ...
    '03_singlePath_ulna'; ...
    '04_singlePath_radius'; ...
    '05_doublePath_ulna_radius'; ...
    '06_doublePath_radius'; ...
    '07_doublePath_ulna'; ...
    '08_doublePath_connectedBelow'; ...
    '09_doublePath_connectedAtWrist'; ...
    '10_doublePath_connectedAtWrist_onlyRadius'; ...
    '11_doublePath_connectedAtWrist_onlyUlna'; ...
    '12_doublePath_withShoulder'; ...
    '13_ulna_withShoulder'; ...

    % models with exoActuator additionally as a muscle:
    '13_singlePath_ulna_asMuscle'; ...
    '14_singlePath_radius_asMuscle'; ...
    '15_doublePath_ulna_radius_asMuscle'; ...
    '16_doublePath_radius_asMuscle'; ...
    '17_doublePath_ulna_asMuscle'; ...
    '18_doublePath_connectedBelow_asMuscle'; ...
    '19_doublePath_connectedAtWrist_asMuscle'; ...
    '20_doublePath_connectedAtWrist_onlyRadius_asMuscle'; ...
    '21_doublePath_connectedAtWrist_onlyUlna_asMuscle'; ...
    '22_doublePath_withShoulder_asMuscle'; ...
    '23_ulna_withShoulder_asMuscle'; ...

```

```

    '99_tryoutModel'};
% ".osim" will be added for the model files by the program

modelNameNames = modelNameNames(modelSelector);

%% CORRESPONDING PRESCRIBED EXOACTUATOR FILE (to model)
% This code snippet can be copied from the console output of
% createPrescribedFiles.m when using a new model or motion file:

prescribedFiles(1:2,1) = {0}; % first 2 models (incl. model 00) don't
    use prescribed exoActuator forces
% (these have been generated via "createPrescribedFiles.m" and the text was
% copied from there)
% they need to be in the "Simulation" folder
% the "figureTitle" variable can later be used to name the jrfPlot
% depending on which models are used the first 2 (before: 52 or 72)
% entries need to be empty.
% The
prescribedFiles(03:15,1) = { ...
    'prescribedControl_mot_0to130_1sec_100Hz_02.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_03.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_04.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_05.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_06.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_07.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_08.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_09.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_10.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_11.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_12.sto'; ...
    'prescribedControl_mot_0to130_1sec_100Hz_13.sto'; ...
    'placeholder'};
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figureTitle = 'PrescribedController_archive/torqueCurves/Torque_XXX.sto';

prescribedFiles = prescribedFiles(modelSelector);

```

```

% INITIALIZE INPUT/OUTPUT PATHS
% this MATLAB script should be in the same folder as the setup, motion and
  model files
experimentModelsPath = num2str(cd);
experimentOutputPath = horzcat(num2str(cd), '\Results\simResults_', datestr(now
  , 'yymmdd_HHMM'));

% Import OpenSim Libraries
import org.opensim.modeling.* % not sure if even necessary in this script (or
  just in functions)

%% display parameter & model choices
disp(['Simulation_set_up_for_', num2str(size(modelSelector,2)), ...
  '_model(s)_with_the_following_parameters:'])
% save parameters in struct
parameters.whichModels = whichModels;
parameters.modelNames = modelNames;
parameters.prescribedExo = prescribedFiles;
parameters.motion = motion;
parameters.muscleModel = muscleModel;
parameters.analysisType = analysisType;
parameters.algorithmMuscles = algorithmMuscles;
disp(parameters)

%% run SO / CMC function for selected models
currentTimeString = ['(', datestr(now, 'HH'), ':', datestr(now, 'MM'), ')'];
disp(['Analysis_has_started_@_', currentTimeString, '_...'])
resultsPaths = cell(length(whichModels),1);

% this loop goes through all models
for k = 1:length(whichModels)
  modelName = cell2mat(modelNames(k));
  prescribedFile = cell2mat(prescribedFiles(k));
  switch algorithmMuscles
  case 'SO' % DEPRECATED / not used for MobL-ARMS model
    [musclesOutputPath] = SOrunner(modelName, experimentModelsPath,
      experimentOutputPath, motion, analysisType);
  case 'CMC'

```

```

disp([ 'CMC_for_model_' ,modelName, '_has_started. ' ])
[musclesOutputPath] = CMCrunner(modelName, experimentModelsPath ,
    experimentOutputPath , motion , prescribedFile );
end
% structExtension = horzcat('model_', num2str(whichModels(k))); % must begin
% with a character
% SOPaths.(structExtension) = SOoutputPath;
resultsPaths(k) = cellstr(musclesOutputPath);
end

%% run JRA function for selected models
% this loop goes through all models
if strcmp(analysisType, 'separated')
disp( 'JRA_analysis_has_started_...' )
for k = 1:length(whichModels)
modelName = cell2mat(modelNames(k));
inputPathSO = cell2mat(resultsPaths(k)); % or CMC
[JRAfilename] = JRARunner(modelName, experimentModelsPath , inputPathSO , motion ,
    algorithmMuscles);
% structExtension = horzcat('model_', modelName); % must begin with a
% character
% JRAfiles.(structExtension) = JRAfilename;
end
end

%% JRF comparison via plots

% evaluate/plot JRFs
[jrfData, jrfPlot, twoDimPlot] = evaluateJRFs(resultsPaths, modelNames, ...
    muscleModel, analysisType, algorithmMuscles, motion);

try
if exist('figureTitle', 'var') % rename in case a name is given before
figure(jrfPlot)
set(gcf, 'Name', figureTitle)
end
end

% save fig & mat files
savefig(jrfPlot, [experimentOutputPath, '\JRFplot.fig']);
if twoDimPlot ~= 0
savefig(twoDimPlot, [experimentOutputPath, '\twoDimPlot.fig']);

```



```

end
save([experimentOutputPath, '\JRFdata.mat'], 'jrfData');
save([experimentOutputPath, '\parameters.mat'], 'parameters');

%% program evaluation
disp(' ')
currentTimeString = ['(', datestr(now, 'HH'), ':', datestr(now, 'MM'), ')'];
disp(['The whole comparison took ', num2str(toc(comparisonTime)), ' seconds for
     ', ...
num2str(length(modelSelector)), ' model(s) .', currentTimeString])
disp(' ') % empty/new line

clear comparisonTime k modelName
set(groot, 'defaultAxesColorOrder', 'remove') % reset to default MATLAB color
scheme

end

```

### CMCranner.m

```

function [outputPath] = CMCrunner(modelName, inputPath, experimentOutputPath,
    motion, prescribedExo)
%% CMCrunner Description
% INPUT:    modelName – string with name (without ".osim" extension)
%           inputPath – path, where the models lie
%           experimentOutputPath – path, where the CMC output folder of the
%           ...
%           model should be stored
%           motion – motion file name string (should be in the same big
%           folder as
%           the models), e.g. 'IK_0to145.mot'
%           prescribedExo – path/file that contains the prescribed
%           force/torque for the exoActuator. If no prescribedExo is to
%           be used, it should just be a "0" (mat, not string)
% OUTPUT:   outputPath – path for model subfolder in experiment folder
%
% DESCRIPTION:
% This function executes the computed muscle control algorithm in OpenSim and
% saves the
% corresponding files.

```



```

cmc_tool = CMCTool(setupFile , false); % set up an OpenSim CMCTool from the
    setup:
% https://simtk.org/api\_docs/opensim/api\_docs/classOpenSim\_1\_1CMCTool.html

currentModel = Model(modelPath); % this gets the .osim file as input
currentModel.initSystem();

%% add prescribed controller

if prescribedExo ~= 0 % only done for models, where a prescribed controller
    is used
cmc_tool.setControlsFileName(prescribedExo); % needs to be done before "
    setModel"
end

%% further OpenSim Setup
cmc_tool.setModel(currentModel);
% cmc_tool.setLoadModelAndInput(true); % (todo): what did this do in SO?

% input motion file
motionFile = [inputPath, '\', motion]; % the motion file lies here too
%cmc_tool.setCoordinatesFileName(motionFile); % (todo): this is what it was
    called in SO
cmc_tool.setDesiredKinematicsFileName(motionFile);

% set final time for simulation
cmc_tool.setFinalTime(createFinalTime(motion));

% set results directory
cmc_tool.setResultsDir(outputPath);

%% save setup & run CMC
setupNameGenerated = 'Setup_CMC_generated.xml';
cmc_tool.print(fullfile([experimentOutputPath, '\', modelName],
    setupNameGenerated));

cmc_tool.run();

%% evaluate
currentTimeString = ['(', datestr(now, 'HH'), ':', datestr(now, 'MM'), ')'];
disp(['Elapsed_time_for_CMC_was_', num2str(toc), '_seconds_for_model_', ...

```

```
modelName, '□', currentTimeString])
```

```
end
```

## JRARunner.m

```
function [outputPath] = JRARunner(modelName,inputPathModel ,inputPathSO ,motion
    ,algorithmMuscles)
%% Description JRARunner
% INPUT:  modelName – string with name (without ".osim" extension)
%         inputPathModel – path, where the models lie
%         inputPathSO – path, where the SO force file lies, the output also
%         gets saved here
%         (requires the file 'Setup_JRA_altref_NN.xml' to be in the cd)
%         motion – motion file name string (should be in the same big folder
    as
%         the models), e.g. 'IK_0to145.mot'
%         muscleModel – string for definition of used musculoskeletal model
%         e.g. 'Arm26' or 'MoBL'
%
% OUTPUT: outputPath – path for model subfolder in experiment folder, JRA
%         setup and results file are stored here
%
% DESCRIPTION:
% This function executes the joint reaction analysis in OpenSim and saves the
% corresponding files. The necessary JRA setup file is generated before
%
%
%% %%%%%%%%%%% ACTUAL CODE %%%%%%%%%%%
tic % time measurement for JRA simulation

%% for tests without input (debugging etc.):
% this requires the inputs to be set to "varargin"
if nargin == 0
modelName = '00_noExo';
inputPathModel = num2str(cd); % this should be the path, where the model lies
inputPathSO = [num2str(cd), '\Results\simResults_210519_1250'];
motion = 'mot_0to145to0_2sec.mot';
muscleModel = 'Arm26';
end

%% set up input/output folders & paths
% create model file path for input
modelFile = horzcat(inputPathModel, '\',modelName, '.osim');
```

```

outputPath = inputPathSO;
% addpath(inputPathSO); % not needed for finding the proper setup file

%% set up OpenSim
import org.opensim.modeling.*

ModelVisualizer.addDirToGeometrySearchPaths(inputPathModel);

%% generate JRA setup file
% Setting up tool

% setupNameOriginal = 'Setup_JRA_altref_NN.xml'; % 'r_ulna_radius_hand' as
    frame
setupNameOriginal = 'Setup_JRA_childRef_NN.xml'; % 'child' as frame (--> here
    : ulna)
analyzeTool = AnalyzeTool(setupNameOriginal, false);
analysis = analyzeTool.getAnalysisSet().get(0); % creates general analysis
jra = JointReaction.safeDownCast(analysis); % creates JointReactionAnalysis
% <-- TODO: What does this do? Not 100% sure if I should use analyzeTool or
% jra afterwards.
name = 'JointReactionAnalysis';
jra.setName(name); % is used in the automatic file naming by OpenSim (I think
    )

% set results directory
analyzeTool.setResultsDir(inputPathSO); % the setup file should also go here

% set force input file
switch algorithmMuscles
case 'SO'
musclesResultsFile = [inputPathSO, '\SOanalysis_StaticOptimization_force.sto'
    ];
case 'CMC'
musclesResultsFile = [inputPathSO, '\CMC_results_Actuation_force.sto'];
end
jra.setForcesFileName(musclesResultsFile);

% input motion file
motionFile = [inputPathModel, '\', motion]; % the motion file lies here too
analyzeTool.setCoordinatesFileName(motionFile);

```

```

% set final time for simulation (using 'jra' didn't work)
analyzeTool.setFinalTime(createFinalTime(motion));

% Save setup file
setupNameGenerated = 'Setup_JRA_NN_generated.xml';
analyzeTool.print(fullfile(inputPathSO, setupNameGenerated));
setupFile = [inputPathSO, '\', setupNameGenerated]; % this is used in running
    the JRA

%% run JRA
% set up JRA tool
jra_tool = AnalyzeTool(setupFile, false);

% set up model
currentModel = Model(modelFile);
currentModel.initSystem();
jra_tool.setModel(currentModel);
jra_tool.setLoadModelAndInput(true);

% where to save results
analyzeTool.setResultsDir(inputPathSO);

% Run study
jra_tool.run();

%% evaluate
disp(['Elapsed_time_for_JRA_was_', num2str(toc), '_seconds_for_model"',
    modelName, ''])

end

```

## evaluateJRFs.m

```
function [varargout] = evaluateJRFs(varargin)
%% evaluateJRFs - description
% This function takes the SO and JRA result files (.sto files) and displays
% JRF (and exoActuator) plots.
%
% INPUT:   varargin - spread out over:
%           resultsPaths - folders, where SO and JRA results for each model
%           lie as a cell array.
%           modelNames - cell array of model names (without path and
%           extension)
%           muscleModel - string depicting used musculoskeletal Model (e.g.
%           'Arm26' or 'MoBL')
%           JRFfileName - string containing the name of the JRF results
%           file (including the .sto)
%           motionFile - the corresponding motion file (used to plot over
%           the angle as the x-axis)
% OUTPUT:  jrfPlot - figure data type (to be able to edit stuff afterwards)
%           jrfData - struct with JRF results (including time points)
%
%
%% %%%%%%%%%%% ACTUAL CODE %%%%%%%%%%%
tic % time measurement for JRF evaluation

% please select plot type (deprecated)
allInOne = 1;

%% VARARGIN
if nargin == 0
% for tests without input (debugging etc.):
modelName = { '00_noExo_withActuator'; '01_baseline_bandmann'; '02
    _simpleTorqueActuator' };
resultsPaths = { 'Results\simResults_210920_2250\70_noExo_noWeight'; ...
    'Results\simResults_210920_2250\71_noExo'; ...
    'Results\simResults_210920_2250\78_5G_doublePath_mostComplex' };
JRFfileName = 'JRAanalysis_JointReactionAnalysis_ReactionLoads.sto';
analysisType = 'separated';
algorithmMuscles = 'CMC';
muscleModel = 'MoBL';
motionFile = 'mot_0to130_1sec_100Hz.mot';
```



```

elseif nargin == 1
resultsPaths = varargin{1};
% create a generic list of modelNames:
for k = 1:length(resultsPaths)
modelNames{k} = ['model_',mat2str(k)];
end
muscleModel = 'MoBL';
analysisType = 'separated';
algorithmMuscles = 'CMC';
disp(['analysisType was assumed to be:_',analysisType,'_(in_evaluateJRFs)']
)
elseif nargin == 2
resultsPaths = varargin{1};
modelNames = varargin{2};
muscleModel = 'MoBL';
analysisType = 'separated';
algorithmMuscles = 'CMC';
elseif nargin == 3
resultsPaths = varargin{1};
modelNames = varargin{2};
muscleModel = varargin{3};
analysisType = 'combined';
elseif nargin == 4
resultsPaths = varargin{1};
modelNames = varargin{2};
muscleModel = varargin{3};
analysisType = varargin{4}; % --> influences JRFfileName and
    forceFileName
muscleModel = 'MoBL';
algorithmMuscles = 'CMC';
elseif nargin == 5
resultsPaths = varargin{1};
modelNames = varargin{2};
muscleModel = varargin{3};
analysisType = varargin{4}; % --> influences JRFfileName and forceFileName
algorithmMuscles = varargin{5}; % SO or CMC
elseif nargin == 6
resultsPaths = varargin{1};
modelNames = varargin{2};
muscleModel = varargin{3};
analysisType = varargin{4}; % --> influences JRFfileName and forceFileName
algorithmMuscles = varargin{5}; % SO or CMC
motionFile = varargin{6};

```

**end**

*%% select file names according to analysisType*

*% "combined" type is DEPRECATED!*

**if strcmp**(analysisType, 'combined') && **strcmp**(algorithmMuscles, 'SO')

JRFfileName = 'SOandJRA\_JointReaction\_ReactionLoads.sto';

forceFileName = 'SOandJRA\_StaticOptimization\_force.sto';

*% "separated" is USED!*

**elseif strcmp**(analysisType, 'separated') || **strcmp**(algorithmMuscles, 'CMC')

JRFfileName = 'JRAanalysis\_JointReactionAnalysis\_ReactionLoads.sto';

switch algorithmMuscles

case 'SO'

forceFileName = 'SOanalysis\_StaticOptimization\_force.sto';

case 'CMC'

forceFileName = 'CMC\_results\_Actuation\_force.sto';

**end**

**end**

*%% read motion file (to obtain elbow flexion values)*

motionData = readSto(motionFile);

flexionAngles = motionData.r\_elbow\_flex;

*%% read JRF .sto files*

**for** k = 1:length(resultsPaths) *% loop for each model*

stoFile = [cell2mat(resultsPaths(k)), '\', JRFfileName]; *% all JRA results files have this name*

stoModelName{k} = ['model\_', num2str(k)]; *% optional: make name from modelNames*

[jrfData.(stoModelName{k}).JRFs, jrfData.(stoModelName{k}).time] = ...  
readSto(stoFile, 'jrf', muscleModel);

*%% calculate resulting jrf (root of squared sums RSS)*

JRFs = jrfData.(stoModelName{k}).JRFs; *% vector of JRFs*

jrfRSS = **sum**(**sqrt**(JRFs.^2), 1); *% root of squared sums*

jrfData.(stoModelName{k}).resultingJRF = jrfRSS;

**clear** JRFs jrfRSS

**end**

*%% read exoActuator from SO .sto files*

**for** k = 1:length(resultsPaths)

stoFile = [cell2mat(resultsPaths(k)), '\', forceFileName]; *% all SO results files have this name*

[actuatorData.(stoModelName{k}).actuator, actuatorData.(stoModelName{k}).time]  
= ...

readSto(stoFile, 'exoActuator');

**end**

*%% plots*

*%% plotting parameters*

lineWidth = 1.25; *% used in plotting*

**if** allInOne == 1

*% get plots all together in one (contrary to deprecated versions)*

jrfPlot = **figure**('Name', 'Joint\_Reaction\_Analysis\_Results', 'Position', ...  
[150,120,600,400]);

howManyPlots = 3;

lightGrey00 = 0.68 .\* [1 1 1]; *% used for 01\_noExo plots*

lightGrey01 = 0.45 .\* [1 1 1]; *% used for 01\_noExo plots*

*% tiledlayout(2,2, 'Padding', 'none', 'TileSpacing', 'normal'); % use either this or subplots*

*% EXOACTUATOR FORCE*

**subplot**(2,2,3)

*% nexttile*

**for** m = 1:length(resultsPaths) *% for every model*

flexionAngles\_interpolated = **interp1**(motionData.time, motionData.r\_elbow\_flex, ...  
...

actuatorData.(stoModelName{m}).time); *% get flexion angles for each time step from motion file*

```

if m == 1 && length(resultsPaths) > 2 % mark noExo models 00 and 01 in thick
    grey
plot(flexionAngles_interpolated , actuatorData .(stoModelName{m}) . actuator , '-.' ,
    'LineWidth' ,2, 'Color' ,lightGrey00)
elseif m == 2 && length(resultsPaths) > 2 % mark noExo models 00 and 01 in
    thick grey
plot(flexionAngles_interpolated , actuatorData .(stoModelName{m}) . actuator , '-.' ,
    'LineWidth' ,2, 'Color' ,lightGrey01)
else % default
plot(flexionAngles_interpolated , actuatorData .(stoModelName{m}) . actuator , '
    LineWidth' ,lineWidth)
end

%plot (actuatorData .(stoModelName{m}) . time , actuatorData .(stoModelName{m}) .
    actuator , 'LineWidth' ,lineWidth) % old/obsolete: time as x-axis
hold on
end
xlim ([0 , max(motionData .r_elbow_flex) ])
grid on
% title ('exoActuator force ')
xlabel ('flexion_angle_[degrees] ')
ylabel ('actuator_force_[N] ') % TODO: use "Nm" for torque actuator

% RESULTING JRF
subplot(2,2,1)
%nexttile
for m = 1:length(resultsPaths) % for every model
if m == 1 && length(resultsPaths) > 2 % mark noExo models 00 and 01 in thick
    grey
plot(flexionAngles , jrfData .(stoModelName{m}) . resultingJRF , '-.' , 'LineWidth' ,2,
    'Color' ,lightGrey00)
elseif m == 2 && length(resultsPaths) > 2 % mark noExo models 00 and 01 in
    thick grey
plot(flexionAngles , jrfData .(stoModelName{m}) . resultingJRF , '-.' , 'LineWidth' ,2,
    'Color' ,lightGrey01)
else % default
plot(flexionAngles , jrfData .(stoModelName{m}) . resultingJRF , 'LineWidth' ,
    lineWidth)
end
hold on

```

```

end
ylim([500,inf]) % either 0 or 500
xlim([0, max(motionData.r_elbow_flex)])
grid on
% title('resulting JRF')
xlabel('flexion_angle_[degrees]')
ylabel('resulting_JRF_[N]')

% DIRECTION PLOT
subplot(2,2,2)
%nexttile
for m = 1:length(resultsPaths) % for each model
if m == 1 && length(resultsPaths) > 2 % mark noExo models 00 and 01 in thick
    grey
plot(jrfData.(stoModelName{m}).JRFs(1,:),jrfData.(stoModelName{m}).JRFs(2,:),
    '-.','Color',lightGrey00,'LineWidth',2)
elseif m == 2 && length(resultsPaths) > 2 % mark noExo models 00 and 01 in
    thick grey
plot(jrfData.(stoModelName{m}).JRFs(1,:),jrfData.(stoModelName{m}).JRFs(2,:),
    '-.','Color',lightGrey01,'LineWidth',2)
else % default
plot(jrfData.(stoModelName{m}).JRFs(1,:),jrfData.(stoModelName{m}).JRFs(2,:),
    '-','LineWidth',lineWidth)
end

hold on
end
for m = 1:length(resultsPaths) % start markers
% this is outside of the previous loop due to the plot color order
plot(jrfData.(stoModelName{m}).JRFs(1,1),jrfData.(stoModelName{m}).JRFs(2,1),
    '*','LineWidth',1,'Color','black')
end
% plot(0,0,'+','LineWidth',3,'Color','black') % optional: add origin for
    visualisation
xline(0) % origin
yline(0) % origin
xlim([-1000 200]) % to have sufficient width of the plot
ylim([-Inf 50]) % to have xline within plot
daspect([1 1 1]) % data aspect ratio X:Y = 1:1
grid on
% title('JRFs in the XY-plane')

```

```

xlabel( 'JRF_X-component_[N] ' )
ylabel( 'JRF_Y-component_[N] ' )

% LEGEND
%nexttile
legendTile = subplot(2,2,4);
legendTile.Position = [0.92 0.35 0.01 0.01]; % make plot insignificantly
    small & position legend properly

% pseudo plotting to get lines into legend

hold on
for m = 1:length(resultsPaths)
if m == 1 && length(resultsPaths) > 2 % mark noExo models 00 and 01 in thick
    grey
plot([0 1],[0 0], '-.', 'Color', lightGrey00, 'LineWidth', 2)
hold on
elseif m == 2 && length(resultsPaths) > 2 % mark noExo models 00 and 01 in
    thick grey
plot([0 1],[0 0], '-.', 'Color', lightGrey01, 'LineWidth', 2)
else % default
plot([0 1],[0 0], '-', 'LineWidth', lineWidth)
end
end
plot([0 0],[0 0], '*', 'LineWidth', 1, 'Color', 'black') % for start markers
set(gca, 'YTickLabel', []);
set(gca, 'XTickLabel', []);

% create legend (& entries)
for k = 1:length(resultsPaths)
legendEntries(k) = {strrep(cell2mat(modelNames(k)), '_ ', '\_')}; % replace "_"
legendEntries(k) = {strrep(cell2mat(legendEntries(k)), 'connected', '')}; %
    shorten "connectedAtWrist" model names for legend
end
legendEntries(end+1) = {'start_of_motion_in_XY-plot'}; % start markers in XY-
    plot
leg = legend(legendEntries, 'Location', 'northeast');
title(leg, 'models')
%leg.Position(1:2) = [.6 .2]; % deprecated, because this doesn't translate to
    the tikz version

```

**end**

*%% prepare varargout*

varargout{1} = jrfData;

varargout{2} = jrfPlot;

**if** allInOne == 0

varargout{3} = twoDimPlot;

**else**

varargout{3} = 0;

**end**

*%% evaluate function*

**disp**([ 'Elapsed\_time\_for\_JRF\_evaluation\_was: ', **num2str**(toc), ' seconds. ' ])

**end**

## Appendix H

This appendix section shows exemplary source code snippets for creating the models' exoskeleton actuators ("exoActuator") in OpenSim and the muscle version of it in the "...\_asMuscle". Each of the code snippets is located in the models' osim file as a ForceSet object.

- 01\_noExo
- 02\_coordinateActuator
- 03\_singlePath\_ulna
- Example code snippet showing the "...\_asMuscle" implementation for models used in the MA tool.

### "exoActuator" code snippet for the OpenSim model 01\_noExo

```
[...]  
<CoordinateActuator name="exoActuator">  
<min_control>0.01</min_control>  
<max_control>1</max_control>  
<optimal_force> 1e-03 </optimal_force>  
<coordinate> r_elbow_flex </coordinate>  
</CoordinateActuator>  
[...]
```

### "exoActuator" code snippet for the OpenSim model 02\_coordinateActuator

```
[...]  
<CoordinateActuator name="exoActuator">  
<min_control>0</min_control>  
<max_control>Inf</max_control>  
<optimal_force> 1 </optimal_force>  
<coordinate> r_elbow_flex </coordinate>  
</CoordinateActuator>  
[...]
```

### "exoActuator" code snippet for the OpenSim model 03\_singlePath\_ulna

```
[...]  
<PathActuator name="exoActuator">  
<!--Flag indicating whether the force is applied or not. If true the force is  
    applied to the MultibodySystem otherwise the force is not applied.NOTE:  
    Prior to OpenSim 4.0, this behavior was controlled by the 'isDisabled'  
</PathActuator>
```



```

    property, where 'true' meant that force was not being applied. Thus, if '
    isDisabled' is true, then 'appliesForce' is false.-->
<appliesForce>true </appliesForce>
<!--Minimum_allowed_value_for_control_signal._Used_primarily_when_solving_for
    _control_values.-->
<min_control>0</min_control>
<!--Maximum_allowed_value_for_control_signal._Used_primarily_when_solving_for
    _control_values.-->
<max_control>Inf </max_control>
<!--The_set_of_points_defining_the_path_of_the_actuator.-->
<GeometryPath_name="pathwrap">
<!--The_set_of_points_defining_the_path-->
<PathPointSet>
<objects>
points_R1-R2_in_radius' frame
points R3-R4 in ulnas frame
points L1-L4 in ulnas frame
points "shoulder" in humerus frame
points "thorax" in thorax frame

path runs from left (L / inside) over shoulder, upper arm and forearm to
    wrist an then back on the right (R / outside). -->

<PathPoint name="exo_shoulder">
<!--Path to a Component that satisfies the Socket 'parent_frame' of type
    PhysicalFrame (description: The frame to which this station is fixed.).--
    >
<socket_parent_frame>/bodyset/humerus</socket_parent_frame>
<!--The fixed location of the station expressed in its parent frame.-->
<location>0.0555 -0.112 -0.015</location>
</PathPoint>

<!-- <<< Left side (shoulder down to wrist) <<< -->

<!-- EDIT (NN): This snippet can be used to copy a marker position to a
    pathAct's_pathPoint:-->
<!--_<PathPoint_</PathPoint>_-->

<!--_>>>_Right_side_(wrist_up_to_shoulder)_>>>_-->

<PathPoint_name="exo_5G_brace_ulna">

```

```

<!--Path to a Component that satisfies the Socket 'parent_frame' of type
PhysicalFrame (description: The frame to which this station is fixed.)
.-->
<socket_parent_frame >/bodyset/ulna </socket_parent_frame >
<!--The fixed location of the station expressed in its parent frame.-->
<location >0.0627_-0.1478_0.0255</location >
</PathPoint >
</objects >
<groups_/ >
</PathPointSet >
<!--The wrap objects that are associated with this path-->
<PathWrapSet >
<objects >
</objects >
<groups_/ >
</PathWrapSet >
<!-- Default appearance attributes for this GeometryPath-->
<Appearance >
<!--The color (red, green, blue) [0, 1], used to display the geometry.-->
<color >0_1_0</color >
</Appearance >
</GeometryPath >
<!--The maximum force this actuator can produce.-->
<optimal_force >1</optimal_force >
</PathActuator >
[...]
```

**Example code snippet showing the “...\_asMuscle” implementation for models used in the MA tool for model 13\_singlePath\_ulna\_asMuscle:** 13\_singlePath\_ulna\_asMuscle is the “...\_asMuscle” version of model 03\_singlePath\_ulna.

```

[...]
```

```

<Millard2012EquilibriumMuscle name="exoActuator_asMuscle">
<!-- EDIT (NN): this muscle is only used for calculating the exoActuator's
moment_arm
this is because OpenSim only calculates moment_arms for muscles and not for
actuators in "MuscleAnalysis"
in order to not disturb the simulation, the property "appliesForce" is set to
"false"-->
<!--Flag indicating whether the force is applied or not. If true the force is
applied to the MultibodySystem otherwise the force is not applied. NOTE:
Prior to OpenSim 4.0, this behavior was controlled by the 'isDisabled' >
```

```

    property ,_where_ ' true ' _meant_ that_ force_ was_ not_ being_ applied . _Thus ,_ if_ '
    isDisabled' _is_ true ,_then_ ' appliesForce ' is false .-->
<appliesForce>>false</appliesForce>
<!--The set of points defining the path of the actuator.-->
<GeometryPath name="geometrypath">
<!--The set of points defining the path-->
<PathPointSet>
<objects>
<PathPoint name="exo_shoulder">
<!--Path to a Component that satisfies the Socket 'parent_frame' of type
PhysicalFrame (description: The frame to which this station is fixed.) .--
>
<socket_parent_frame>/bodyset/humerus</socket_parent_frame>
<!--The fixed location of the station expressed in its parent frame.-->
<location>0.0555 -0.112 -0.015</location>
</PathPoint>

<!-- <<< Left side (shoulder down to wrist) <<< -->

<!-- EDIT (NN): This snippet can be used to copy a marker position to a
pathAct's_pathPoint:_-->
<!--_<PathPoint_</PathPoint>_-->

<!--_>>>_Right_side_(wrist_up_to_shoulder)_>>>_-->

<PathPoint_name="exo_5G_brace_ulna">
<!--Path_to_a_Component_that_satisfies_the_Socket_'parent_frame'_of_type_
PhysicalFrame_(description:_The_frame_to_which_this_station_is_fixed_)
.-->
<socket_parent_frame >/bodyset/ulna </socket_parent_frame >
<!--The_fixed_location_of_the_station_expressed_in_its_parent_frame.-->
<location >0.0627_-0.1478_0.0255</location >
</PathPoint >
</objects >
<groups_ />
</PathPointSet >
<!--The_wrap_objects_that_are_associated_with_this_path-->
<PathWrapSet>
<objects >
</objects >
<groups_ />
</PathWrapSet>

```

```

<!--Default_appearance_attributes_for_this_GeometryPath-->
<Appearance>
<!--The_color_(red,_green,_blue),_[0,_1],_used_to_display_the_geometry.-->
<color>0.10000000000000004_0.10000000000000001_0.8000000000000001</color>
</Appearance>
</GeometryPath>
<!--Maximum_isometric_force_that_the_fibers_can_generate-->
<max_isometric_force>1e-02</max_isometric_force>
<!--Optimal_length_of_the_muscle_fibers-->
<optimal_fiber_length>0.33000000000000002</optimal_fiber_length>
<!--Resting_length_of_the_tendon-->
<tendon_slack_length>0.28000000000000001</tendon_slack_length>
<!--Angle_between_tendon_and_fibers_at_optimal_fiber_length_expressed_in_radians-->
<pennation_angle_at_optimal>0</pennation_angle_at_optimal>
<!--Maximum_contraction_velocity_of_the_fibers,_in_optimal_fiberlengths/second-->
<max_contraction_velocity>10</max_contraction_velocity>
<!--Compute_muscle_dynamics_ignoring_tendon_compliance._Tendon_is_assumed_to_be_rigid.-->
<ignore_tendon_compliance>true</ignore_tendon_compliance>
<!--Compute_muscle_dynamics_ignoring_activation_dynamics._Activation_is_equivalent_to_excitation.-->
<ignore_activation_dynamics>true</ignore_activation_dynamics>
<!--Assumed_initial_activation_level_if_none_is_assigned.-->
<default_activation>1</default_activation>
<!--Activation_lower_bound.-->
<minimum_activation>0.01</minimum_activation>
<!--Active-force-length_curve.-->
<ActiveForceLengthCurve_name="SUP_ActiveForceLengthCurve_exo">
<!--Normalized_fiber_length_where_the_steep_ascending_limb_starts-->
<min_norm_active_fiber_length>0.4440999999999999</min_norm_active_fiber_length>
<!--Normalized_fiber_length_where_the_steep_ascending_limb_transitions_to_the_shallow_ascending_limb-->
<transition_norm_fiber_length>0.72999999999999998</transition_norm_fiber_length>
<!--Normalized_fiber_length_where_the_descending_limb_ends-->
<max_norm_active_fiber_length>1.8123</max_norm_active_fiber_length>
<!--Slope_of_the_shallow_ascending_limb-->
<shallow_ascending_slope>0.86160000000000003</shallow_ascending_slope>
<!--Minimum_value_of_the_active-force-length_curve-->

```

```

<minimum_value>0</minimum_value>
</ActiveForceLengthCurve>
<!--Force-velocity_curve.-->
<ForceVelocityCurve_name="SUP_ForceVelocityCurve_exo">
<!--Curve_slope_at_the_maximum_normalized_concentric_(shortening)_velocity_(
    normalized_velocity_of_-1)-->
<concentric_slope_at_vmax>0</concentric_slope_at_vmax>
<!--Curve_slope_just_before_reaching_concentric_slope_at_vmax-->
<concentric_slope_near_vmax>0.25</concentric_slope_near_vmax>
<!--Curve_slope_at_isometric_(normalized_velocity_of_0)-->
<isometric_slope>5</isometric_slope>
<!--Curve_slope_at_the_maximum_normalized_eccentric_(lengthening)_velocity_(
    normalized_velocity_of_1)-->
<eccentric_slope_at_vmax>0</eccentric_slope_at_vmax>
<!--Curve_slope_just_before_reaching_eccentric_slope_at_vmax-->
<eccentric_slope_near_vmax>0.14999999999999999</eccentric_slope_near_vmax>
<!--Curve_value_at_the_maximum_normalized_eccentric_contraction_velocity-->
<max_eccentric_velocity_force_multiplier>1.3999999999999999</
    max_eccentric_velocity_force_multiplier>
</ForceVelocityCurve>
<!--Passive-force-length_curve.-->
<FiberForceLengthCurve_name="SUP_FiberForceLengthCurve_exo">
<!--Fiber_strain_at_zero_force-->
<strain_at_zero_force>0</strain_at_zero_force>
<!--Fiber_strain_at_a_tension_of_1_normalized_force-->
<strain_at_one_norm_force>0.69999999999999996</strain_at_one_norm_force>
<!--Fiber_stiffness_at_the_end_of_the_low-force_region-->
<stiffness_at_low_force>0.20000000000000001</stiffness_at_low_force>
<!--Fiber_stiffness_at_a_tension_of_1_normalized_force-->
<stiffness_at_one_norm_force>2.8571</stiffness_at_one_norm_force>
<!--Fiber_curve_bend_from_linear_(0)_to_maximum_bend_(1)-->
<curviness>0.75</curviness>
</FiberForceLengthCurve>
<!--Tendon-force-length_curve.-->
<TendonForceLengthCurve_name="SUP_TendonForceLengthCurve_exo">
<!--Tendon_strain_at_a_tension_of_1_normalized_force-->
<strain_at_one_norm_force>0.049000000000000002</strain_at_one_norm_force>
<!--Tendon_stiffness_at_a_tension_of_1_normalized_force-->
<stiffness_at_one_norm_force>28.061199999999999</stiffness_at_one_norm_force>
<!--Normalized_force_developed_at_the_end_of_the_toe_region-->
<norm_force_at_toe_end>0.66669999999999996</norm_force_at_toe_end>
<!--Tendon_curve_bend_from_linear_(0)_to_maximum_bend_(1)-->

```

```
<curviness>0.5</curviness>  
</TendonForceLengthCurve>  
</Millard2012EquilibriumMuscle>  
[...]
```