DTU Autonomous Systems
Department of Elektro

A Simulation Study: Developing an EMG-Based Quasi-Static Controller for Stroke Rehabilitation

Nuria Fe Garcia del Valle

A Simulation Study: Developing an EMG-Based Quasi-Static Controller for Stroke Rehabilitation

Master Thesis
August, 2023

By
Nuria Fe Garcia del Valle

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Elektro, Brovej, Building 118, 2800 Kgs. Lyngby Denmark
www.byg.dtu.dk
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Approval

This thesis has been prepared over six months at the Section for Indoor Climate, Department of Civil Engineering, at the Technical University of Denmark, DTU, in partial fulfilment for the degree Master of Science in Engineering, MSc Eng.

It is assumed that the reader has a basic knowledge in the areas of statistics.

Nuria Fe Garcia del Valle - s202421

.....

Signature

.....

Date

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Acknowledgements

Nuria Fe Garcia del Valle, MSc Civil Engineering, DTU
Creator of this thesis template.

[Name], [Title], [affiliation]
[text]

[Name], [Title], [affiliation]
[text]

Contents

1 Introduction

1.1 Research Question and Problem Statement

Stroke is the leading cause of disability, and neurorehabilitation aims to enhance brain plasticity to promote rapid recovery and restore lost motor functions. Functional Electrical Stimulation (FES) has been shown to improve rehabilitation outcomes. FES effectiveness increases when combined with a patient's voluntary effort, which current open-loop-based or triggered controllers fail to encourage in hospital or rehabilitation center settings. Newer technologies are emerging, which use EMG sensors to capture muscle signals and encourage active patient contribution. Unfortunately, most of these devices are threshold-based or focus only on detecting the intention of movement, and lack the design of a controller with the ability to learn, generalize, and adapt not only to different stroke survivors but also to individual stroke survivors over time. Furthermore, while many studies have examined the importance of wrist flexion, not enough attention has been paid to arm flexion and the recovery of reaching movements.

- Is it possible to develop a technology that can provide FES stimulation tailored to each patient's current muscular activity?
- Is it feasible to create a system that can autonomously respond and adapt to a patient's input without any human interaction?
- Is it possible to create a framework that resembles the arm dynamics that will allow different control methods to be tested?

1.2 Research Goals

This project aims to achieve the following reaching goals:

1. **Identify Gaps in Current FES Technology**:
    - Conduct an exhaustive literature review to recognize limitations and shortcomings in the present Functional Electrical Stimulation (FES) technology.
    - Provide recommendations or potential avenues for advancement based on identified gaps.
2. **Analyze Current FES Control Methods**:
    - Review and evaluate prevalent FES control methods and strategies from recent literature.
    - Determine and select the most suitable control method for integration with this project based on performance, reliability, and compatibility metrics.
3. **Comprehend and Replicate an Established Model**:
    - Understand the intricacies and functionalities of the chosen controller.
    - Successfully replicate the dynamic arm simulator's movement using the aforementioned controller.
4. **Develop and Validate Arm Dynamics and Neural Modelling**:
    - Design a model that will represent the arm dynamics.

- Evaluate the relationship between neural excitations and arm dynamic.

5. **Simulate Arm Movements Using Inverted Models**:
    - Utilize the inverse of the models to simulate the arm's movement in the dynamic arm simulator.
    - Test with different paths and movements.
6. **Integration of Neural Excitation Data**:
    - Record comprehensive neural excitation data for varied arm movements.
    - Incorporate this data into the simulator to enhance the authenticity and responsiveness of the arm's movements.
    - Record arm movements simulated impaired neural excitation, simulating stroke survivors movement.
7. **Develop an Additional Layer for an EMG-inspired Control**:
    - Design and implement an additional layer in the control system to provide supplementary neural excitation, resembling the FES devices, ensuring smooth and accurate arm movement that will resemble an impaired arm.

1.3 Project Overview

2 Motivation

2.1 Background

Stroke is one of the **most significant cause of disability** worldwide, with a wide range of physical, cognitive, and psychological consequences. The World Health Organization (WHO) states that stroke is the primary cause of disability and death among the senior population after heart disease [1].

Because of cell death brought on by hypoxia and inflammation that affects both the gray and white matter pathways, stroke can cause localized lesions in the brain. A wide range of disorders, including hemiparesis, abnormal posture, spatial neglect, aphasia, spasticity, affective and cognitive problems such as chronic pain and depression, can develop after a stroke. Depending on the severity of it and the patient's general health, the onset and duration of these deficits may differ [2].

According to current statistics, the number of individuals aged 60 years or older is estimated to be around 650 million. However, projections indicate that **this number will rise significantly** and reach up to 2 billion by the year 2050 [3]. It is important to note that the elderly population is at a heightened risk of experiencing a stroke, emphasizing the need for preventative measures and effective healthcare policies to mitigate the impact of stroke on this demographic.

Despite advances in prevention and treatment, stroke remains a **significant public health issue,** with an expected increase in the stroke population by 2025 [4]. This could result in a notable economic strain on society due to the high survival rate of stroke compared to other illnesses.

The cost of post-stroke care varies widely, ranging from a minimum of 752 USD/month to a maximum of 4850 USD/month. In Europe, there are 1 million new cases of stroke reported annually, with a total of 6 million stroke survivors. The estimated annual cost of post-stroke care and treatment in 27 EU countries is 27 billion euros, with 8.5 billion euros in indirect costs and 18.5 billion euros in direct medical expenses. Denmark has the second-highest per-patient cost per month, with 3022 USD/month. [5]. The main contributors to the **high cost of care** are nursing care and rehabilitative therapies. Therefore it is essential to understand the significant cost drivers and fill the information aperture to help develop effective public health and rehabilitation policies.

The aftermath of stroke can be devastating and can severely **impact the affected individual's quality of life.** Various developed therapies have been reviewed for their effectiveness in treating stroke including Functional Electrical Stimulation (FES), Non-invasive Brain Stimulation (NIBS), transcranial Direct Current Stimulation (t-DCS), transcranial Magnetic Stimulation (t-MS), Virtual Reality (VR), task-oriented therapy and robot-assisted rehabilitation. These therapies are essential for individuals who have experienced a stroke, as they help them regain their independence and improve their overall standard of living.

However, there is a **lack of** research and studies on stroke survivors' performance during rehabilitation and **tailored-made rehabilitation** that adapt not only to the survivors but to its own personal progress [6]. Investigating these aspects of stroke rehabilitation could provide valuable insights into the effectiveness of different therapies and help to tailor

rehabilitation programs to the individual needs of stroke survivors to achieve an overall improved rehabilitation.

In conclusion, stroke is a major cause of disability and death, particularly among the elderly population, and its prevalence is expected to rise in the coming years. The economic burden of stroke is significant, and nursing care and rehabilitative therapies are the main contributors to the high cost of care. Despite advances in prevention and treatment, there is a need for effective healthcare policies and preventative measures to mitigate the impact of stroke on society.Various therapies have been reviewed for their effectiveness in treating stroke, but further investigation should be developed with special focus on adaptable, personalized treatments and effectiveness of treatment during rehabilitation. Investigating these aspects could provide valuable impacts on stroke recover and reduce the demand of caretakers and healthcare personnel.

2.2 Post-Stroke Upper Limb Rehabilitation

Most stroke survivors experience weakness in their upper extremities, which limits their ability to perform daily activities. Arm deficits are particularly significant, as 50% of the reduction in quality of life after a stroke is due to the inability to use the arm effectively (Upper extremity function in stroke subjects: Relationships between the international classification of functioning, disability, and health domains).

Unifying the results of 11 rehabilitation pilot studies, it was observed that there is a functional improvement during the whole post-stroke rehabilitation. However, motor recovery usually progresses non linearly, reaching asymptotic levels a few months after the damage. In other words, there is a crucial window for recovery that lasts for longer than a year after a stroke, during which stroke survivors appear more receptive to therapy (A critical time window for recovery extends beyond one-year post-stroke). In conclusion, early therapy is essential for maximizing recovery potential. Nevertheless therapy should also be extended to chronic stroke survivors.

Exercise or physical therapy has been observed to restore sensory-motor function in stroke survivors with some residual muscle activity thanks to brain plasticity (Effects on upper extremity function and cortical plasticity in individuals with incomplete cervical spinal cord injury). It has been found that repetitive tasks cause an impact on cortical plasticity, improving motor and muscle function. Training therapies can influence the reorganizational mechanism in the cerebral cortex promoting the functional recovery. Even though the mechanism of neurological recovery after stroke is not yet well understood there are significant evidence that intervention of one or more therapeutic technique are helpful for rehabilitation of the neural pathways. (Masiero S., Carraro E. Upper limb movements and cerebral plasticity in post-stroke rehabilitation)

In this project, we explore Functional Electrical Stimulation (FES) that is commonly used in two main assistive therapies for rehabilitation: robot-aided rehabilitation, task-oriented rehabilitation. Below is a brief explanation presented for each of the three topics.

1. Functional Electrical Stimulation (FES): This therapy involves the use of electrical stimulation to activate muscles that are weak or paralyzed. The stimulation is delivered through electrodes placed on the skin or implanted in the muscle tissue. FES can help to improve muscle strength and control, reduce muscle spasms, and prevent muscle atrophy.

   For stroke survivors over 18 with a 2 month stroke duration, FES has been shown to be a successful treatment. However, in cases when FES treatment was started

more than a year after the stroke's onset, no appreciable improvement was seen. (Functional electrical stimulation enhancement of upper extremity functional recovery during stroke rehabilitation: A pilot study)

2. Robot-assisted rehabilitation: This therapy involves the use of robotic devices to assist stroke survivors with movement and exercises. The devices provide support, guidance, and resistance to help stroke survivors improve their range of motion, strength, and coordination.

   Functional activities of robot-assisted therapy have been extensively examined, giving positive but unsatisfactory results during clinical trials. There was no significant improvement in UL motor activity observed in the stroke survivors in comparison with routine clinical practice. (Masiero S., Poli P., Rosati G., Zanotto D., Iosa M., Paolucci S., Morone G. The value of robotic systems in stroke rehabilitation., Robot assisted training for the upper limb after stroke (RATULS): A multicentre randomised controlled trial. Lancet.)

3. Task-oriented (TO) rehabilitation: This therapy focuses on specific tasks and activities that are relevant to a patient's daily life. The goal is to help stroke survivors regain the skills and abilities they need to perform these tasks independently. Task-oriented rehabilitation may involve exercises that simulate real-life activities, such as reaching for objects, standing up from a chair, or walking on uneven surfaces.

   Task oriented muscle therapy in studies with randomized controlled trials included, concluded that TO training for stroke survivors is a reliable and safe way to enhance the functional outcomes and overall quality of life (Rensink M., Schuurmans M., Lindeman E., Hafsteinsdottir T. Task-oriented training in rehabilitation after stroke:)

2.3 Rehabilitation Task

The following section will discuss rehabilitation tasks and interventions aimed at improving upper limb function in stroke survivors.

Stroke survivors often exhibit decreased amplitudes at the shoulder and elbow joints during reaching movements, as well as a decrease in range of motion at the elbow joint with a tendency towards flexion. This hinders the ability to perform appropriate reaching movements and can lead to compensatory movements, such as excessive shoulder abduction. Moreover, hand spasticity limits the open and closing hand movements [7].

Grasping, holding, and manipulating objects are daily functions that remain deficient in 55% to 75% of stroke survivors 3 to 6 months post-stroke. Intervention studies have focused on increasing exercise duration and intensity, task-specific training, and enhancing training through surface neuromuscular electrical nerve stimulation. A pilot study with 15 stroke survivors during a 12-week training program showed that task-specific exercises significantly enhanced recovery when compared to task-specific exercise alone [8].

A bank of exercises shown below (see 2.1), including unilateral and bilateral training that can be performed by the patient or caregiver, has also been developed to improve hand functions of chronic stroke survivors in a home-based, self-administered program.

Table 2.1: Upper Extremity Functional Exercises

Exercise Category	Exercise Type	Description
Passive/Active ROM		
	Passive ROM	Wrist/elbow/shoulder, self or by family member
	Active/assistive ROM	Wrist/elbow/shoulder bilateral with dowel, cane
	Active ROM	Wrist/elbow/shoulder in sitting and standing
	Active ROM with resistance	Wrist/elbow/shoulder in sitting and standing
Weight bearing and supportive reaction		
	Seated weight bearing	Forearms on tabletop with affected upper extremity
	Extending arms	Seated or standing with bilateral upper extremity weight bearing on table
	Extended arms with transitional movements	Side lying to sit, sit to stand, dips
	Extended arms and wrist/hand on wall	From anterior and lateral, progress to wall push up
	Extended arms and wrist/hand on wall with change in base of support	Example: weight shifting, single lower extremity support, lateral wall walking
Reaching activities		
	Forward supported reach	Bilaterally with cane on tabletop (elbow extension)
	Forward supported reach with shoulder elevation	Elbow/wrist extension
	Reaching against gravity	In frontal and sagittal planes
	Reaching overhead	With active wrist/hand movements
	Dynamic reaching to a target	Example: catch a ball
Grasping, holding and release		
	Maintaining digit extension with weight bearing	
	Grasp, hold and release containers	With gravity minimized (on table)
	Pick up and move/release small object	On table
	Pick up and move/release large objects	Without proximal support
	Incorporate key and pinch grips in hold and release	Including stacking, lifting and overhead activity
Upper extremity ADL		
	Dressing, grooming	
	Carrying objects	With bilateral upper extremities
	Opening bottles	Stabilizing with paretic extremity for reaching
	Writing, drawing, manipulating small objects	
	Folding towels, vacuuming	

ROM: range of motion; **ADL**: activity of daily living

2.4 Current FES technologies

Functional Electrical Stimulation (FES) is a technique that uses electrical currents to stimulate muscle contractions in individuals with neurological or musculoskeletal impairments. It is a popular technology used for upper limb rehabilitation. Here is a summary of some current FES technologies for upper limb rehabilitation and whether they are EMG controlled or not:

- NESS H200 and H200 Wireless - These are two FES devices for the upper limb. The NESS H200 is EMG controlled, while the H200 Wireless is not. They are designed to assist with grasping and releasing objects and for stimulating the nerves of the forearm, wrist, and hand.
- Bioness StimRouter - This FES device is not EMG controlled and is designed for use in individuals with upper limb paralysis caused by stroke or traumatic injury. It uses a small implantable stimulator to stimulate the peripheral nerves in the upper limb.
- MyoPro - This is a myoelectric orthosis that uses EMG signals to control FES stimulation. It is designed to help individuals with paralysis or weakness of the arm and hand due to conditions such as spinal cord injury or stroke.
- Freehand System - This FES device is EMG controlled and is designed to help individuals with quadriplegia or upper limb paralysis due to spinal cord injury. It uses a small implanted stimulator to stimulate the nerves that control the muscles of the hand and forearm.
- SaeboStim Micro - This FES device is not EMG controlled and is designed for individuals with neurological or musculoskeletal impairments affecting the upper limb. It uses electrical stimulation to activate the muscles and assist with grasping and releasing objects.
- OmniHi5 - The OmniHi5 is a wearable robotic exoskeleton that combines advanced robotics and electrical stimulation to assist individuals with upper limb disabilities in performing functional activities. The OmniHi5 is highly customizable and programmable to provide a range of stimulation patterns to fit individual needs and preferences.

Overall, FES has gained popularity in upper limb rehabilitation, and there are several products available in the market for personal use. EMG signals are increasingly being used as part of product control. However, detailed information about the type of EMG control used by these devices is scarce, but they appear to be threshold-based, which means that the motor response is initiated when the EMG signal crosses a specified threshold value. Although the threshold can be adjusted to fit individual needs and preferences, this type of control may be considered somewhat rudimentary and not automatically adaptable to each case. Nevertheless, FES technologies for upper limb rehabilitation offer a promising avenue for individuals with upper limb disabilities to regain or improve their motor functions.

2.5 FES Device Limitations

Exclusion Criteria

2.5.1 State of the Art

The task of control design for Functional Electrical Stimulation (FES)-enabled reaching motions in the upper extremities is challenging. The difficulties stem from the goal-directed character of reaching motions, which demand unique simulation patterns for each reach, in contrast to the cyclic movements of the lower extremities. This is because tasks involving the upper extremities are goal-directed, which means that the amplitude, speed, and direction of the motions are always changing. Therefore, a more sophisticated controller is needed, one that can continuously calculate the stimulation patterns for a range of user-commanded motions. [9].

Several studies have been conducted on the development and control of intelligent assistive exo-gloves, functional electrical stimulation (FES) systems, and robot-assisted rehabilitation systems. It is one of the major difficulties to provide a adaptable, functioning and precise controller design. This is because the external disturbance itself is subjected to another unsolved controller, the human control.

Vast majority of upper-limb stroke patient trials using ES employ open-loop or triggered controllers which can lead to imprecise control of movement. The controller system should have the ability of learning, generalizing and adaptation as human brain controls the body. This is a very novel topic and is currently being studied.

Many different control strategies have been proposed to accurately treat the arm as a complete system. A summary of them is presented below:

Feedforward Control: This approach is often used in clinical practice and relies solely on user command without considering system performance into account (13, 15). A drawback is that it cannot be adjusted if the actual movement differs from the desired movement, despite the fact that it is favorable because sensors are not needed to measure the system output. [9].

Feedback Control: Feedback control uses sensed information of the current output of the system to correct for deviations. It can make up for fatigue and disturbances. It enables optimum tracking performance. However, the system's response time have some inherent delays that could be problematic, especially for fast actions. [9].

Combined Feedforward and Feedback Control: Some FES system designs have opted for a feedforward and feedback control combination producing excellent results. Methods like training an Artificial Neural Network (ANN) with time-delayed inputs have been used for the feedforward part, which is often an inverse-dynamic model.

Neural Network Control: Another strategy is to create a non-modeling control mechanism, such as the neural network control theory.In the research [10], traditional PID was combined with Neural Networks to establish the control parameters. The neural network is trained to detect input/output relationships with the least amount of error, and the PID controller component fine-tunes the current. They were able to successfully design the correct input of electrical stimulating current signals for the patient's foot and establish relationships between stimulating current and ankle joint angle.

Iterative Learning Control (ILC): The recurring aspect of the rehabilitation process is taken advantage of by the innovative approach known as ILC. Patients repeatedly attempt the same task, and ILC gradually improves accuracy by leveraging information from earlier attempts to modify the FES for the subsequent execution. In [11] an ILC algorithm

adjusted the amount of stimulation applied to the triceps and anterior deltoid muscles to improve accuracy and maximize voluntary effort. In another study, [12], an input-output linearization (IOL) structure was developed. It presented few parameters and was shown to be robust. ILC systems are best used for systems that repeatedly track a fixed reference signal over a limited time period since they are susceptible to disruptions and modeling errors.

Fuzzy system with emotional learning control: A more state of the art study implemented a fuzzy system equipped with emotional learning to control force and position separately in the exo-glove. The performance of this system was compared to that of a linear controller, and it was found that the overshoot decreased by up to one-third and the settling time declined to one-fifth [13].

In conclusion, it is challenging to establish control parameters for highly dynamical and complex systems, such as upper-body extremity,since new parameters must be redefined for every subject and might not hold true in all situations. Classical control theory—which calls for a mathematical model of the controlled system—is ineffective. FES-control systems is intricate and demands creative solutions to ensure optimal operation on day-to-day or rehabilitation-oriented assistive devices. It is an ongoing field of research with various approaches to determine the most effective methods to improve outcomes.

3 Design

The overall design to achieve the goal on section XX is presented in this chapter. The prerequisites and criteria are firstly examined to replicate the rehabilitation scenario.

Table 3.1: Comparison of Two Simulation Studies

Developing a Quasi-Static Controller for Paralyzed Human Arm	**Developing an EMG-Based Quasi-Static Controller for Stroke Reaching Rehabilitation**
Focused for spinal cord injury	Focused on stroke survivors
Controlling 9 muscles	Controlling 2 muscles
Surgical FES at 13Hz	External FES stimulation
Exoskeleton Arm Support	No Exoskeleton Arm (enough mobility provided by the patient)
Wrist Position Control	EMG and Wrist Position Control
Multiple movements in the workspace	Same target, different initial positions. Repetition of movement for rehabilitation purposes.

3.1 Design Criteria

To pinpoint the crucial factors that would direct our choice of design and tools, a thorough criterion analysis was carried out. The following important standards came to be recognized as essential elements in the search for an efficient simulation solution:

1. **Real Arm Simulator** In order to correctly reproduce real-world dynamics and movements, an exact simulation of an arm was required.
2. **Accurate Dynamics Representation** To ensure that the simulated arm's behavior and responses adhere to the fundamentals of human biomechanics.
3. **Neural Stimulation Control** An essential criteria to provide more realistic interaction between the simulated arm and the brain or the FES.
4. **Tracking** Ability to track the arm´s movement either in joint space or task space. This will provide flexibility in the control system.
5. **EMG Reading** It should present the capability or a possibility to read real or fake neural excitation data. This data will be used as the main driven input to control the arm's stimulation.
6. **Control Model Integration** A key criterion is that the tool presents the possibility to develop and evaluate various control methods.
7. **External Forces Addition** In order to simulate realistic interactions and enabling more thorough and accurate simulation of arm dynamics.

3.2 Block Diagram

3.3 Simulation Experiment Setup

Spasticity (velocity-dependent stiffness) in stroke typically produces resistance to arm extension due to overactivity of biceps, wrists and finger flexors and loss of activity of triceps,

anterior deltoid, wrist and finger extensors (reference 24). [12] Triceps and anterior deltoid are hence selected for stimulation to align with the clinical need to increase muscle tone and restore motor control of weakened muscles. The relationship between muscle stimulation adn subsequent movement are well explored. However, simplification opens up routes for both aprameter identification and controller derivation that have not yet been possible for more complex models. [12] It is assumed that applying FES to the anterior deltoid produces a moment about an axis that is fixed with respect to the shoulder. Applying FES to the triceps produces a moment about an axis orthogonal to both the forearm and upper arm.[12]

As discussed in [26] the most prevalent form of muscle representation is the Hill-type muscle. (Explain the Hill-Type muscle) Functional Electrical Stimulator (a DSP-based FES) was developed in our laboratory. 23,35,50Hz are available ofr the pulse frequency. The pulse width ranges from 0 to 300 μs. The output current is from 0 to 100 mA. The stimulation wave can be modified into two models: single phase, and biphasic phase. [10]

4 Tools

4.1 Dynamic Simulation of the Arm: A Comprehensive Model for Musculoskeletal Dynamics

The study of human mobility has benefited greatly from the development of musculoskeletal modeling, which is now a crucial tool for comprehending the dynamics of the human body. Multibody models have historically proved helpful in resolving inverse dynamic issues, particularly in the analysis of human gait and arm movement (please add paper citation for more information). The main disadvantage of these inverse dynamic techniques is the need to gather data from participants who are actively doing the required movement.

Moving away from conventional techniques, muscle-driven forward dynamics have allowed the simulation of innovative and fictional movements that would resemble impaired movements, such as those of stroke survivors. These simulations have practical applications in fields such as rehabilitation and motor control research. In [14] the DAS allowed to test controllers and command interfaces with a human user in the loop. For this project, forward dynamics simulation will be used to test the hypothesis for EMG-inspired PD control of Functional Electrical Stimulation (FES) in stroke rehabilitation.

A summary of the musculoskeletal dynamics, including topics like multibody dynamics, muscle contraction dynamics, and muscle activation dynamics, will be covered in this section. Then, an explanation of the forward-dynamic implicit formulation will be given. This formulation has been proven to improve the computational time required to model dynamics with only an RMS error of 0.11 degrees in joint angles when running at real-time speed for a gait simulation with a prosthetic foot and ankle [14].

Overall, the Dynamic Arm Simulator (DAS) offers a robust framework for the simulation of human arm movements. It is implemented as a MATLAB Mex function that contains the system dynamics and other functions, making it accessible via a MATLAB function interface, flexible, and adaptable for various applications.

An extended overview of the DAS model is provided in the following sections.

Table 4.1: Nomenclature

Nomenclature	
q	Generalized coordinates vector
$\dot{q}$	Generalized angular velocities vector
$M(q)$	Mass matrix
τ	Generalized forces vector/ Joint moments
$B(q, \dot{q})$	Coriolis, centrifugal and passive forces matrix
L_M	Muscle-tendon length
L_{CE}	Muscle contractile element length
ϕ	Pennation angle
s	Muscle contraction dynamics state variable
a	Activation state
x	State vector $x = (q, \dot{q}, s, a)^T$
u	Neural excitations for each muscle

4.1.1 Structural Overview

Links and Degrees of Freedom

- **7 Links**: Includes the thorax, clavicle, scapula, humerus, ulna, radius, and hand. Figure 4.1 shows a colour-coded of the different links.

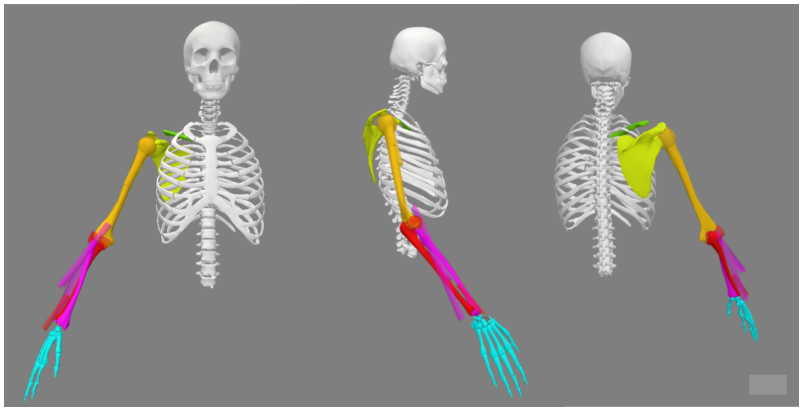

Figure 4.1: Dynamic Arm Simulator 7 Links. Green: Clavicle, Yellow: Scapula, Orange: humerus, Red: Ulna, Purple: Radius, Blue: hand

- **11 Degrees of Freedom**: These consist of 2 orthogonal hinges at specific joints, including the sternoclavicular, acromioclavicular and glenohumeral joints, elbow flexion-extension, and forearm pronation-supination (see Figure 4.2 for reference). Initially, the model was focused on planar movement but was improved by adding independent control of the shoulder girdle. These additions allowed for more flexibility and stabilisation when designing the neuroprosthetic control algorithm [15]. The range of motion of the different DoFs is shown in Table 4.2.

Inspired by [15] throughout this project the arm configuration is described in 5 DoFs: plane of elevation, angle of elevation, internal rotation, elbow flexion and forearm pronation following the recommendation in [16]. These angles were restricted in order to ensure correct wrapping of muscles in the model, the final values are show in Figure 4.3.

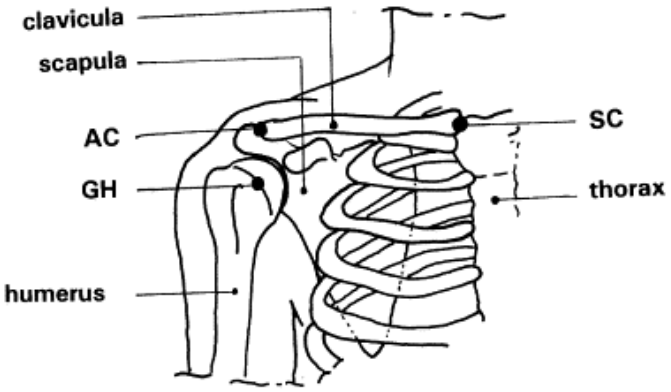

Figure 4.2: Shoulder mechanism of skeletal entities: thorax, clavicula, scapula and humerus; and the joints: sternoclavicular (SC) joint, acromioclavicular joint (AC) and glenohumeral joint (GH) [17].

Table 4.2: *Angular limits for each degree of freedom (in degrees)*: The arm mechanism consists of the following skeletal entities SC: Sternoclavicular joint, AC: Acromioclavicular joint, Gh: Glenohumeral joint, EL: Elbow, PS: Forearm Pronation and Supination.

Degree of Freedom	Min Angle (degrees)	Max Angle (degrees)
SC y	-65.00	-19.00
SC z	5.00	30.00
SC x	0.00	83.00
AC y	33.00	69.00
AC z	-22.00	20.00
AC x	-17.00	18.00
GH y	-170.00	175.00
GH z	-30.00	84.00
GH yy	-177.00	179.00
EL x	5.00	141.00
PS y	5.00	160.00

TABLE I
ANGULAR LIMITS FOR EACH DEGREE OF FREEDOM (IN DEGREES)

Degree of freedom	Min angle	Max angle
Plane of elevation	-90	90
Angle of elevation	5	90
Internal rotation	-55	70
Elbow flexion	5	140
Forearm pronation	5	160

Figure 4.3: Angular limits for each degree of freedom (in degrees) for the arm configuration following [16]. Table from [15]

- **22 Muscles and 138 Muscle Elements:** The real-time model consists of 22 muscles and muscle components in total. In order to accurately represent the mechanical line of action of each member, muscles are modeled using the fewest amount of parts possible. The table in Figure 4.4 displays the amount of elements utilized for each muscle as well as the degrees of freedom that each muscle crosses

TABLE II
MUSCLES INCLUDED IN REAL-TIME MODEL, SHOWING JOINTS CROSSED BY EACH MUSCLE (GLENO-HUMERAL: GH, HUMERO-ULNAR: HU, OR RADIO-ULNAR: RU) AND NUMBER OF ELEMENTS USED TO MODEL MUSCLE

Muscle	Joints	No. of elements
Deltoid, scapular part	GH	11
Deltoid, clavicular part	GH	4
Coracobrachialis	GH	3
Infraspinatus	GH	6
Teres minor	GH	3
Teres major	GH	4
Supraspinatus	GH	4
Subscapularis	GH	11
Biceps, long head	GH, HU and RU	1
Biceps, short head	GH, HU and RU	2
Triceps, long head	GH and HU	4
Latissimus dorsi	GH	6
Pectoralis major, thoracic part	GH	6
Pectoralis major, clavicular part	GH	2
Triceps, medial head	HU	5
Brachialis	HU	7
Brachioradialis	HU and RU	3
Pronator teres	HU and RU	2
Supinator	HU and RU	5
Pronator quadratus	RU	3
Triceps, lateral head	HU	5
Anconeus	HU	5

Figure 4.4: Muscle elements and degrees of freedom crossed by the muscles. Table from [15]

4.1.2 Musculoskeletal Dynamics

The dynamics governing the musculoskeletal system must be thoroughly understood in order to research it. This analysis is divided into three interconnected components: multibody dynamics, muscle activation dynamics, and muscle contraction dynamics. Multibody dynamics allows us to represent the system as a collection of interconnected rigid or flexible bodies, focusing on the kinematic constraints that guide motion. Muscle contraction dynamics describes the complex interactions of muscle fibres and surrounding elements using the three-element Hill-type model, while muscle activation dynamics explores how neural signals translate into muscle actions.

The thorough analysis of the musculoskeletal dynamics is described in [14]. In this section, a summary of each most relevant section is presented with a specific focus on the project's application.

State variables

It is necessary to define the state variables before examining the various dynamics of the multibody system.

The multibody model has 298 state variables. It is represented with the letter ***x***:

- 11 angles, q
- 11 angular velocities, $\dot{q}$
- 138 muscle contractile element (CE) lengths, L_{CE}
- 138 muscle active states, a

Multibody dynamics

Multibody dynamics is used to study the motion and forces acting on the musculoskeletal system as it is identified as a system composed of interconnected rigid or flexible bodies.

Generalised coordinates are used to simplify the description of the system. As a result, the configuration of the system can be represented using the minimum number of coordinates to describe the position and orientation.

Generalized coordinates, are often denoted by *q*. They are a set of independent parameters that define the configuration of the system. In the DAS *q* represents the 11 joint angles.

The equations of motion for a multibody system can be expressed in the following matrix form ([18]):

$$M(q) * \ddot{q} + B(q, \dot{q}) = \tau \tag{4.1}$$

Where,

- $\ddot{q}$ is the vector of generalized accelerations
- $M(q)$ is the mass matrix that describes the inertia of the system. It relates the acceleration of *q* to the generalized forces.
- $B(q, \dot{q})$ is the matrix that describes Coriolis and centrifugal forces that arise from the motion of the bodies in the system. Moreover, it contains gravity and other passive forces.
- τ is the generalised force vector. In the case of DAS it represents the forces generated by muscles according to the muscle contraction and activation dynamics.

Forward dynamics is used to predict the motion of a multibody system based on its initial conditions and known forces. As a result Equation 4.1 is solved to the generalized accelerations $\ddot{q}$.

$$\ddot{q} = M(q)^{-1}(\tau - B(q, \dot{q})) \tag{4.2}$$

The inverse of the mass matrix, M, is a critical component, as the simulation will not be possible if its singular. This singularity is quite common in musculoskeletal systems due to small mass and moment of inertia of body segments.

Moreover, elements of B can present high stiffness and damping, leading to high eigenvalues of the Jacobian matrix that correspond to fast-changing components of the system. These high stiffness and high damping lead to oscillatory behaviours within the system. The presence of stiffness originates from the ligaments and tendons having highly nonlinear mechanical properties (zero stiffness when unloaded, high stiffness when maximally loaded).

To counteract the stiff system, standard numerical methods, such as Euler method, require small time steps.

Muscle contraction dynamics

The muscle contraction dynamics is based on the three-element Hill-type model. It represents the muscle fibers (CE: contractile element), the tendon and force transmitting tissue (SEE: series elastic element), and the passive elastic tissue surrounding the muscle fibers (PEE: passive elastic element).

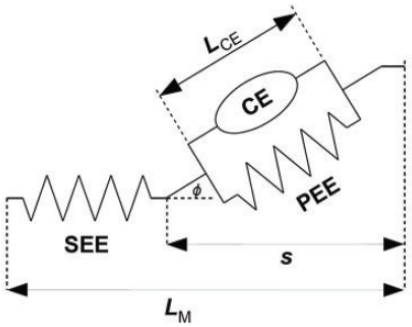

Figure 4.5: Three-element muscle model. CE: contractile element, SEE: series elastic element, PEE: parallel elastic element, Pennation angle ϕ: Angle between the muscle fiber in the CE and the line of action of the muscle, s state variable for the muscle contraction dynamics [14]

The contractile force is calculated by the multiplicative interaction of the maximal isometric force, the activation, the fiber length and the fiber length velocity, respectively represented in the formula below:

$$F_{CE} = F_{max} * a * f_{FL}(L_{CE}) * f_{FV}(\dot{L_{CE}}) \tag{4.3}$$

The series and parallel elastic elements are represented by passive force-length relationships.

$$F_{SEE} = f_{SEE}(L_M - L_{CE} * cos(\phi)) \tag{4.4}$$

$$F_{PEE} = f_{PEE}(L_{CE}) \tag{4.5}$$

The force balance equation following Figure 4.4 is equal to:

$$(F_{CE} + F_{PEE}) * cos(\phi) - F_{SEE} = 0 \tag{4.6}$$

The differential equation for the state variable L_{CE} is defined by Equation 4.6.

Muscle activation dynamics

The nervous system sends a neural excitation $u(t)$ as a control signal to the muscle that changes the activation through a first-order non-linear activation-deactivation process. This is because the activation a, also called the *active state* cannot be directly controlled by the nervous system. The muscle activation of a muscle, a, is formulated as:

$$\dot{a} = (u - a)(c_1 u + c_2) \tag{4.7}$$

,

where $c_1 + c_2$ is the rate constant for activation and c_2 is the rate constant for deactivation, typically in the range of 20-50 s^{-1} [14].

Combining Equation 4.1, Equation 4.6 and Equation 4.7 an implicit state equation for musculoskeletal dynamics can be formulated.

4.1.3 Forward Dynamics for implicit state equation

The forward dynamic simulation of the musculoskeletal system model is a complex process that involves determining the state trajectory $x(t)$, starting from an initial state at $t = 0$ and using a control function u that contains the neural excitations for all muscles.

As mentioned in section 4.1.2 the state vector x is equal to:

$$x = (q, \dot{q}, s, a)^T \tag{4.8}$$

q and $\dot{q}$ for each degree of freedom and s and a for each muscle.

All dynamic equations for the model, which include equilibrium of forces and activation for muscle control (Equation 4.1, Equation 4.6 and Equation 4.7) are combined into one single equation in terms of the state vector x and control input u.

$$f(x, \dot{x}, u) = 0 \tag{4.9}$$

This equation is implicit as there are relationships among the variables. In contrast with explicit formulation, they can not be solve directly for each variable. Solving this implicit equation in a conventional can be hard to solve numerically. This usually requires small steps in simulation, making it time-consuming, computationally expensive and less accurate.

The DAS uses an alternative approach where by leaving the equation in its implicit form is possible to calculate the Jacobian of f with respect to x, $\dot{x}$ and u. [14].

To predict how muscles and joints mover time based on an initial condition and neural excitations controls, the differential algebraic equation (DAEs) must be solved (Equation 4.9).

For real time simulation, the DAS uses Rosenbrock method as it requires quick and predictable computations. The Rosenbrock method is a numerical technique used to solve ordinary differential equations (ODEs). The ordinary differential equations describe how the arm motion changes over time. The Rosenbrock methods involves predicting the state at the next time step. It iterates until finds the right guess. The main advantages of Rosenbrcok is that it produce realistic values if the Jacobian matrices are known.

The DAS implements a first order method with constant step size h. This means that if the the step size is halved, the error will also be halved.

$$\begin{aligned} \Delta x = (\frac{\partial f}{\partial x} + \frac{1}{h}\frac{\partial f}{\partial \dot{x}})^{-1}(\frac{\partial f}{\partial \dot{x}}\dot{x}_n - f(x_n, \dot{x}_n, u_n) - \frac{\partial f}{\partial u} * (u_{n+1} - u_n)) \\ x_{n+1} = x_n + \Delta x \\ \dot{x}_{n+1} = \frac{\Delta x}{h} \end{aligned} \tag{4.10}$$

4.1.4 Functionalities

The DAS can be downloaded from the simTK.org website (https://simtk.org/projects/das). It includes the **main MEX file**, some files to test the correct functionality and installation of the simulation environment and a basic manual for understanding its different functionalities.

Loading the model

During the initialization the MEX functions reads in the parameters from the model. When initializing the model the 13 joints and 138 muscle elements are loaded.

```
load model_struct;
das3('Initiliaze',model);
```

Joint Moments

The input is the states of the model and the output presents the moment of the arm for each degree of freedom (N m)

```
moments = das3('Jointmoments', x) %(ndof x 1)
```

Muscle Forces

The input is the states of the model and the output presents the forces of the arm for each muscles (N)

```
forces = das3('Muscleforces', x)
```

Muscle Length and SEE Slack

The length of each muscle (meters) and the slack length of series elastic element (m) are given using the *'Musclelengths'* and *'SEEslack'* functions respectively.

When a muscle contracts, it shortens its length to produce force. Before being applied to the load, this force passes via the series elastic element (SEE). The SEE is in a slack or non-stretched state when a muscle is in rest and not exerting any force. Because the SEE is not applying any further tension to the muscle fibers at this time, the length of the contractile element (CE) is equal to the length of each muscle fiber. However, when the force is transmitted through the SEE when the muscle contracts and generates force, the SEE begins to stretch and the CE shortens.

In general, the CE length is equal to the length of each muscle fibre when the muscle is relaxed and does not generate any force. When a force is applied, i.e., during muscle contraction, the CE shortens and the SEE stretches, enabling efficient force transmission to the load. This is why to calculate the correct length of the muscle in an initial state where the muscle is contracted, the CE value is equal to the formula below.

$$LCE = length - SEEslack \tag{4.11}$$

The function below initializes the muscle values for the state correctly when knowing only the angle values for the different degrees of freedom.

```
function x = initialize_state(position, nstates, iLce)
    x=zeros(nstates,1);
    LCEopt=das3('LCEopt');
    lengths = das3('Musclelengths', x);    % only the first 11
       elements of x (the joint angles) will be used
    SEEslack = das3('SEEslack');
    Lce = (lengths - SEEslack);
    x(iLce)=Lce;
    x(1:11)=position;

end
```

Go to Section 4.1.2 for more information on the three-element model used to model muscle contraction dynamics.

Dynamics

This function is to evaluate the dynamics of the model in the implicit form f(x,$\dot{x}$,u) = 0. The inputs are:

- x (nstates x 1) : the model states.
- $\dot{x}$ (nstates x 1): the derivatives of the state of the model.
- u (nmus x 1): muscle excitations.

Some optional inputs include

- M (5x1): moments applied to the thorax-humerus YZY and the elbow flexion and supination axes.
- exF (2x1): external vertical force of amplitude exF(2) applied at exF(1) meters from the elbow.
- handF (3x1): force applied to the center of mass of the hand. The positive value of handF (1) represents the lateral movement far from the thorax, the positive value of handF (2) represents the upward movement, and the positive value of handF (3) represents the posterior movement.

The outputs are as follows:

- f (nstates x 1): dynamic imbalance.

Optional outputs include

- $\frac{\partial f}{\partial x}$ (nstates x nstates): Jacobian of f with respect to x.
- $\frac{\partial f}{\partial \dot{x}}$ (nstates x nstates): Jacobian of f with respect to $\dot{x}$.
- $\frac{\partial f}{\partial u}$ (nstates x nmus): Jacobian of f with respect to u.
- qTH (3x1): angles between the thorax and humerus (YZY sequence). This angles are calculate following the standardization on [16]

```
[f, dfdx, dfdxdot, dfdu, ~,~,qTH] = das3('Dynamics',x,xdot,
   step_u,M,exF,handF);
```

The das3step function is used to calculate the next time step following the Rosenbrock method (Equation 4.10) previously explained in Section 4.1.3.

```
function [x, xdot, step_u, qTH] = das3step(x, u, tstep, xdot,
    step_u, M, exF, handF)
     [f, dfdx, dfdxdot, dfdu, ~,~,qTH] = das3('Dynamics',x,xdot,
        step_u,M,exF,handF);

  % Solve the change in x from the 1st order Rosenbrock formula
  du = u - step_u;
  dx = (dfdx + dfdxdot/h)\(dfdxdot*xdot - f - dfdu*du);
  xnew = x + dx;

  % update variables for the next simulation step
  xdot = dx/h;
  step_u = u;
end
```

Visualization

The model can be visualised in OpenSim as shown in the image below. However, for user-friendliness the DAS presents a function named *das3stick(x,model)*. The inputs are the model state and the Matlab model structure, and it outputs a Matlab figure with a more simplified drawing of the model.

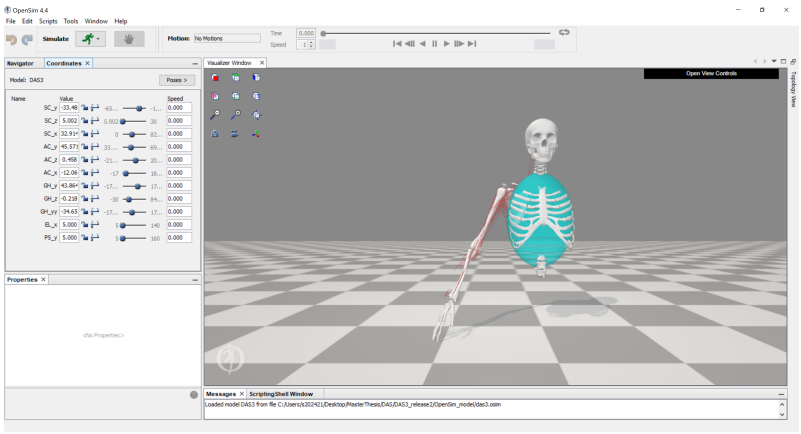

Figure 4.6: Open Sim DAS

```
das3stick(xneutral,model);
```

Output of the *das3stick*

The *das3stick* function utilizes the *'Visualization'* functionality that outputs a 13x12 matrix d that contains the orientation matrix and position vector information for each bone.

```
d = das3('Visualization',x);
p = d(j,1:3)';           % position vector of bone
R = reshape(d(j,4:12),3,3)';  % orientation matrix
```

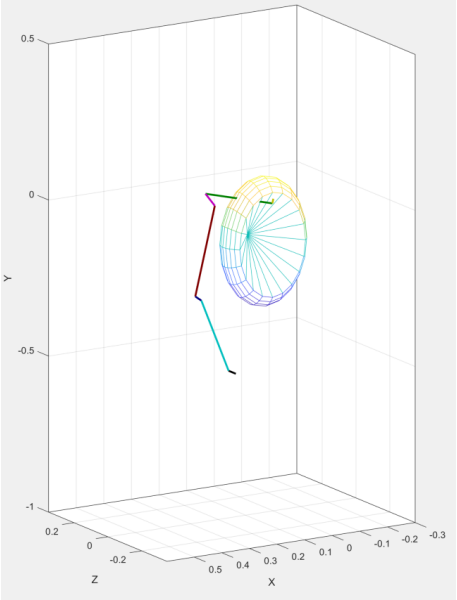

Figure 4.7: Matlab Visualization Output

5 Model Design and Development

Background

Model-based methods for controlling the arm as a unified system and determining real-world arm dynamics have been suggested in earlier studies as a solution for precise control. For instance, physics-based models have demonstrated some success in controlling two muscles for rehabilitation [12]. However, a major challenge lies in the identification of the physical parameters of the entire arm, which demands a significant amount of data.

To address this issue, black-box model-based control methods have been developed. An example of these methods successfully utilized an artificial neural network (ANN) to create a map of task space configuration in relation to the forces that muscles can generate, thereby achieving control of planar arm tasks [19]. Moreover, Lyapunov-based methods have been employed to create a data-driven Deep Neural Network (DNN) based adaptive control method used for Functional Electrical Stimulation (FES)-induced leg extension rehabilitation [20].

Another difficulty lies in establishing control parameters for these systems, with new parameters needing to be set for each subject. Even for the same subject, these parameters can vary under different conditions, making the traditional control theory that requires a mathematical model of the controlled system unsuitable [10].

Despite the potential of model-based FES control in providing necessary accuracy, few approaches have found their way into clinical practice. This shortfall can be attributed to challenges in deriving an accurate model given the limited identification time due to factors such as onset of fatigue and time constraints of the patient, carer, physiotherapist, or engineer [12]. Furthermore, due to time-varying physiological effects, models must be re-identified at the commencement of each treatment session.

Previous work has demonstrated that a semi-parametric Gaussian Process Regression (GPR) model can form the foundation for a controller achieving three-dimensional dynamic trajectories. This model-based method [21] circumvents parametric modelling challenges owing to the difficulty in defining the parameters. The semiparametric GPR combines the generalization potential of a parameterized model with the ability of a nonlinear function approximator that represents the diversity of the musculoskeletal dynamics.

In this project, we developed our model identification in a manner akin to the approach outlined in the paper [21], as it was found to be successful in circumventing the challenges intrinsic to parametric modeling. The GPR model provides a non-parametric, data-driven approach, allowing for an analytical representation of complex patterns without necessitating rigid assumptions regarding the model's structure.

Design

A two part modeled is developed consisting of:

- **Inverse Statics**. It calculates the joint torque needed to hold a desired static arm configuration.
- **Muscle Torque Production**. It maps the arm configuration with the muscle activation to the joint torque produced.

To gather data for the models a PI controller is designed to compute the force applied at the wrist in order to hold the arm in a static position, both with and without neural excitation. This force is subsequently converted into torque using the kinematic Jacobian. The process is detailed in sections 5.1, 5.2 and 5.3 and a flow diagram is shown in Figure 5.1.

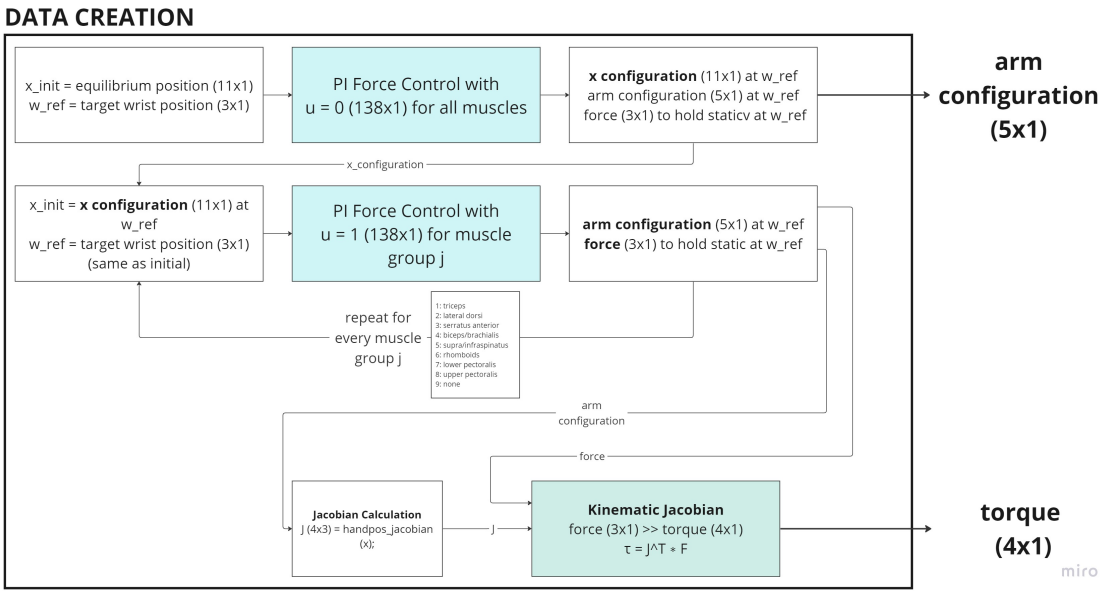

Figure 5.1: Data Creation Flow Diagram

Finally, the collected data is used to train the two semiparametric GPR models which are used to predict τ_j for a given configuration. The process is detailed in section 5.4 and a flow diagram is shown in Figure 5.2.

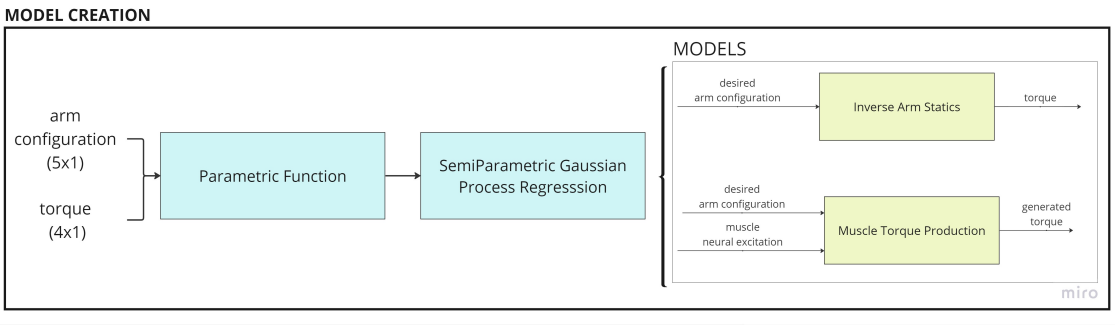

Figure 5.2: Model Creation Flow Diagram

5.1 Target Positions and Initial Configuration

In the paper [21] the simulated experiment resembles the one performed in a real-case scenario. A total of 27 different positions were analyzed where an external robot held the wrist of the arm into the different static positions. The robot was equipped with a three-dimensional force sensor at its end-effector [22].

Since no real-world situation has been evaluated for this project, the simulation's goal is to represent a reaching motion beginning from an equilibrium posture. Additionally, the stroke survivor's arm is not being held by an external robot.As detailed in Section X, this approach stems from the assumption that the stroke survivor retains a degree of arm stability, sufficient to facilitate a limited reaching movement.

For this, it is decided to find calculate the forces in 64 different positions. This 64 positions are automatically generated using the function ***create_grid()***.

The ***create_grid()*** function generates a grid of target points by defining the starting and ending coordinates for the x, y, and z dimensions and then linearly spacing these coordinates. If a model is provided as an input, the function also plots the wrist references, offering a visual representation of these coordinates.

Table 5.1: Start and End Positions for x, y, and z Coordinates

Coordinate	Start Position	End Position
x	0.3	-0.08
y	0	-0.3
z	-0.1	-0.35

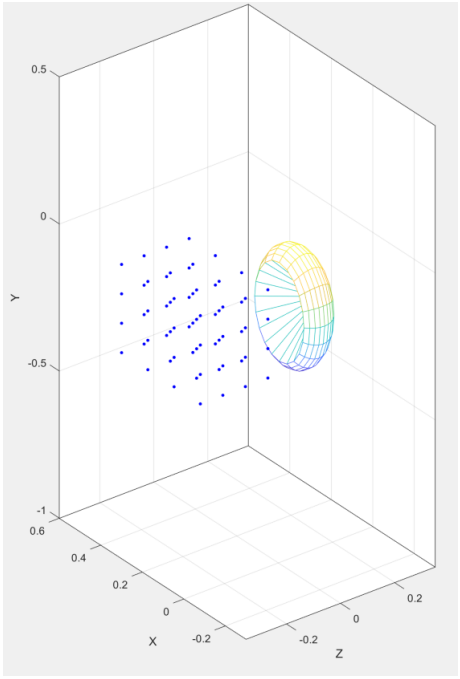

Figure 5.3: *create_grid* function output. 64 target positions used to calculate the required force to hold a this reference static positions with and without neural excitation.

5.2 PI Force Controller

The goal of the static force simulation is to determine the necessary force to keep the wrist in a specific static position. It involves two distinct scenarios. Both scenarios use a PI controller to exert the required force at the wrist, using the input *handF* in the ***das3_step(...)*** function.

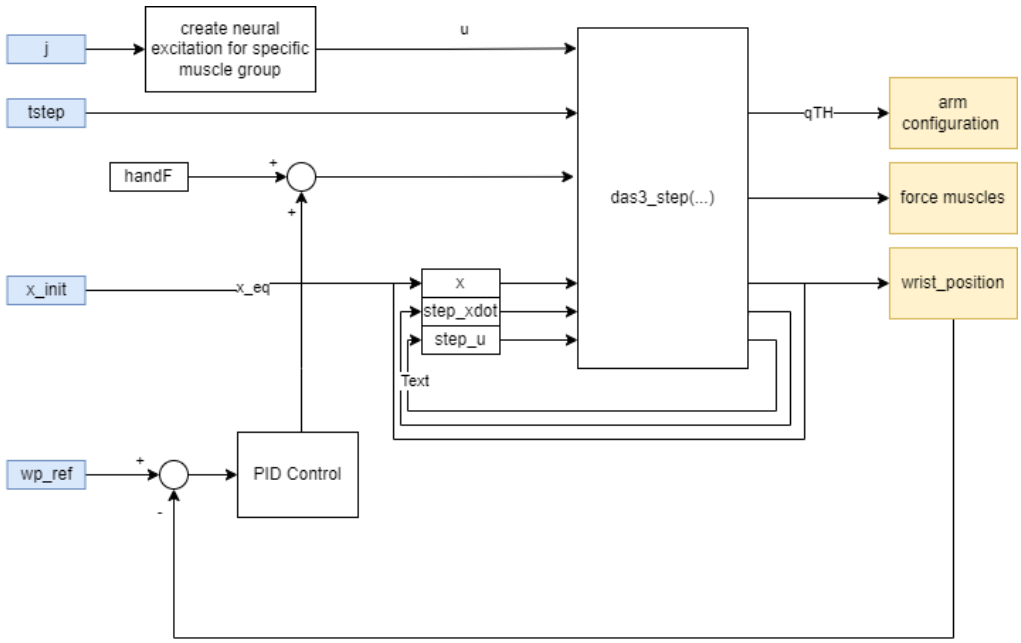

Figure 5.4: Block Diagram for PI Force Controller

In the first scenario, no muscles are excited. The simulation starts from the equilibrium position and continues for 3 seconds, with a time step of 0.001 seconds. The focus is to calculate how much force is needed to apply to maintain a specific arm configuration static.

In the second scenario, the simulation begins with the arm configuration that was determined after running the previous no-excitation simulation. Here, one muscle at a time is stimulated at 100%, i.e, u_j. The index j represents the muscle, this values goes from 1 to 9. Although the muscles may deviate from the starting position, the PI controller continuously adjusts to keep the hand in its initial position. This simulation is shorter, running for 0.5 seconds, but still utilizes a time step of 0.001 seconds.

After each time step,in both scenarios, the difference between the wrist's current position and the target is computed. This error is then multiplied by the Proportional (P) coefficient (Kp) of the PI controller. Concurrently, the accumulated error is updated and multiplied by the Integral (I) coefficient (Ki). The sum of both the proportional and integral terms dictates the *handF* for the next step (Equation 5.1). In the context of the PI controller, the proportional segment is responsible for creating an output in direct relation to the present error, while the integral portion systematically reduces steady-state error by considering the history of past errors.

$$\begin{aligned} error_{pos} &= hand_{goal} - hand_{current}; \\ error_i &= error_i + error_{pos} * t_{step}; \\ hand_F &= K_p * error_{pos} + K_i * error_i; \end{aligned} \tag{5.1}$$

where $error_{pos}$ is the difference between the target goal and the current position, the $error_i$ represent the cumulative error from every time step and $hand_F$ is the calculated force that will be exerted on the hand to achieve the goal position.

The specific values for T, t_{step}, PI parameters for both scenarios are detailed in Table 5.2.

The force necessary to reach each point is calculated by averaging the last 10% of the data collected during the simulation.The resulting information, including the forces exerted, mean force, and the arm's configuration, is saved for each point in the simulation.

Table 5.2: Values for Period (T), Time step (t_step), and PI Gains for PI controller with no neural excitation ($u_j = 0$) and with neural excitation for each muscle ($u_j = 1$). The index j represents each muscle $j \epsilon [1, 9]$.

Parameter	$u_j = 0$	$u_j = 1$
T (Period)	3	0.5
t_{step} (Time Step)	0.001	0.001
K_p (Proportional Gain)	2000	2000
K_i (Integral Gain)	100	100

Below is a MATLAB code snippet that demonstrates the calculation of PI force, inclusive of the step simulation.

```
hand_current = wrist_position(x)
error_pos = hand_goal - hand_current
error_int = error_int + error_pos*tstep;

% PI calculation for time step
handF = K*error_pos+I*error_int;

[x, xdot, step_u] = das3step(x, u, tstep, xdot, step_u, M, exF,
   handF);
```

The function ***wrist_position(x)*** calculates the global position of the wrist using transformation techniques. Initially, the *das3('Visualization',..)* is wrist position vector (p) and the orientation matrix (R). Constants and position vectors are defined, including the local coordinates of bone points.

Next, the function transforms the local coordinates into global ones. By utilizing the orientation matrix **R** and the position vector **p**, it applies a transformation to the local coordinates, yielding the global coordinates **x**, **y**, and **z**. The process is a straightforward application of the principles of rotation and translation, aligning the local wrist coordinates with a global frame of reference.

$$\begin{aligned} p_{rotated} &= R * p_{local} \\ p_{global} &= p_{rotated} + t \end{aligned} \tag{5.2}$$

where $p_{rotated}$ is the rotated position vector, R is the orientation matrix, p_{local} is the local position vector, and t is the translation vector representing the shift from the local to the global coordinate system.

5.3 Kinematic Jacobian and Torque Calculation

5.3.1 Background

The Kinematic Jacobian is a critical part of the transformation of the recorded robot controller force to the joint torque τ_j. The index j represents the muscle group being activated, when no muscle is being activated j is equal to 0.

In robotics, the Jacobian matrix provides essential information about the relationship between the velocities of the joints and the end-effector of a robot manipulator. This relationship can be expressed using the equation

$$\dot{x} = J \cdot \dot{q} \tag{5.3}$$

where $\dot{x}$ is the end-effector velocity and $\dot{q}$ is the joint velocity.

Understanding the relationship between how the robot joint's velocities correlate to the end-effector's velocities is vital for planning trajectories in different spaces.

Furthermore, the kinematic Jacobian can be use to determined the necessary forces in joint space to achieve specific forces in end-effector space. In the musculosketal system it is assumed that the 11 degrees-of-freedom have no friction at the joint. The joint torques ($\tau \epsilon R^n$) required to bear an endpoint/hand force ($F \epsilon R^{6x1}$), are given by,

$$\tau = J^T * F \tag{5.4}$$

where the Jacobian matrix ($J \epsilon R^{6xn}$) relates the infinitesimal joint displacement dq to the infinitesimal end-effector displacements dp. The theorem is proven using the Principle of Virtual Work [23].

$$dq = J * dp \tag{5.5}$$

Calculating the kinematic Jacobian in robotics involves defining the structure and parameters of the robot's joints and links, often using conventions like Denavit-Hartenberg (D-H) parameters. Transformation matrices are found for each joint, and the end-effector position is obtained by multiplying these matrices.

The Jacobian matrix is then constructed by differentiating the end-effector's position with respect to each joint variable, considering whether the joint is revolute or prismatic. The resulting 6×n matrix, where n is the number of joints, maps the joint velocities to the linear and angular velocities of the end-effector.

5.3.2 Implementation

The equation 5.5 is applied as shown in the MATLAB code below to calculate the torque values from the mean forces previously calculate using the PI controller.

```
1 [dPhand_dx, ~, ~] = handpos_jacobian(x);
2 torque = dPhand_dx'*hand_F;
```

Due to the complex system, to calculate the kinematic Jacobian of the arm, the concept of infinitesimal displacement presented in Equation 5.5 is used. The function ***handpos_jacobian(x)*** (MATLAB code below) take the state and perturbs each degree of freedom by a small mount h. For each perturbation, the old and the new hand positions are

subtracted and divided by h. This describes how the changes in the degrees of freedom affect the hand´s position at that particular state.

```
function [dPhand_dx, Phand, Phand_new] = handpos_jacobian(x)
    dPhand_dx = zeros(3,11);
    stick = das3('Visualization', x);
    Phand = stick(13,1:3)';
    h = 1e-7;
    for i = 1:11
        tmp = x(i);
        x(i) = x(i) + h;
        stick = das3('Visualization',x);
        Phand_new = stick(13,1:3)';
        dPhand_dx(:,i) = (Phand_new - Phand)/h;
        x(i) = tmp;
    end
end
```

The shoulder and elbow torques are computed using the transpose of the kinematic Jacobian. They are denoted by τ_j, where j symbolizes the muscle group being activated, with 0 signifying no active muscles. The torque concerning elbow pronation is disregarded as it does not influence the wrist's positions.

If no muscles are activated, the torque needed to hold the the wrist in static position is equal to,

$$\tau_0 = p(q) \tag{5.6}$$

where $p(q)\epsilon R^{4x1}$.

The torque produced by the muscle's group j activations is calculating by subtracting the torques recorded when no muscles are activated and the torques recorded with muscle group j active. This torque is represented by $M(q)\alpha$ where $\alpha\epsilon M^{8x1}$ is the vector of muscle group activations and $M(q)\epsilon M^{4x8}$ is the mapping from muscle group activation to joint torque. It is assumed that the torques scales linearly with activation [24].

$$M_j(q) = \tau_0 - \tau_j \tag{5.7}$$

Each column of $M(q)$ describes the torque generated by each muscle group at full activation. It is assumed that individual muscle torques add linearly without interference from adjacent muscles or connective tissue interactions.

5.4 Model Creation: Semi-Parametric Gaussian Process Regression

5.4.1 Comparative Analysis of Model Types

A comparative accuracy analysis was conducted between non-parametric, semiparametric, and parametric models in [24]. The semiparametric GP model resulted as the most accurate, combining the flexibility of a black-box function approximation with the generalization strength of a parameterized model, as indicated in [24]. This semiparametric model estimated torques during muscle stimulation with less than 20% error relative to the total muscle torque and passive torque required to move the arm.

Details on the quality of the model can be found in [24]. A summary of the three models is described in the table 5.3.

5.4.2 Background

Gaussian Process Regression (GPR) is a powerful and flexible tool used in machine learning for fitting a function to data. It is rooted in the mathematical foundation of Multivariate Gaussian Distributions, where each random variable is distributed normally, and their joint distribution is also Gaussian. GPR can be expressed as

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')) \tag{5.8}$$

where μ is the the mean vector and $k(x, x')$ is the covariance or kernel function. The mean vector μ describes the expected value of the distribution, while the kernel models the variance along each dimension and determines how the different random variables are correlated. The covariance matrix is always symmetric and positive semi-definite.

GPR creates a model of the function it is trying to learn, using a process that considers an infinite number of dimensions and includes an element of randomness. It encapsulated initial beliefs about the function in the *prior distribution*, defined by the mean and kernel functions. Observations are used to update these beliefs through the *likelihood function*, leading to the computation of the *posterior distribution*. This posterior reflects updated beliefs about the possible functions that best fit the data.

Predictions are made by calculating the mean and variance of the predictive distribution at new inputs, allowing for both point estimates and uncertainty estimates.

In summary, GPR operates on the foundational principles of Bayesian inference to model a function from given data. This enables GPR to deliver both point predictions and associated uncertainties. The inherent flexibility and adaptability of GPR make it a powerful and widely applicable tool for function approximation in machine learning and statistical analysis. [25]

5.4.3 Implementation

Based on the results of the aforementioned study, the data for the set of 64 training positions is used to train semiparametric GPR models which are used to predict τ_j for a given configuration. The models are used to determine the static arm torques $p(q)$ and the muscle force mapping $R(q)$ for a desired arm configuration. The kernel function used in the GPR was the squared exponential function using the distance metric for rigid bodies.

A MATLAB script, ***compute_GPR_model***, is designed to compute the model performing Gaussian Process Regression to fit the data. This code loads the data and does GPR for each muscle where the inputs are the arm configuration angles and the output are the joint torques.

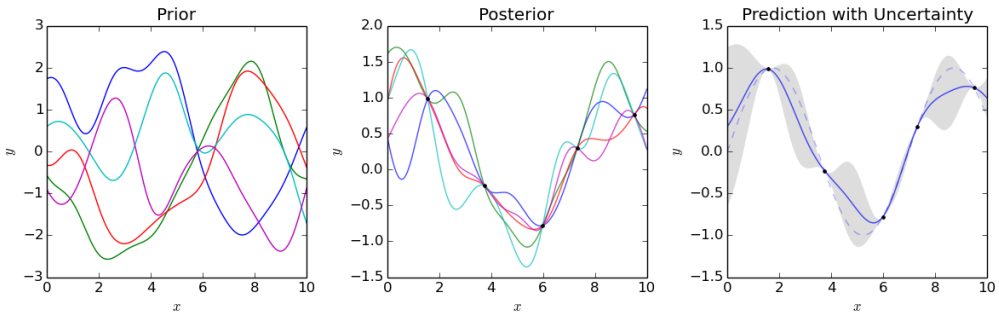

Figure 5.5: Gaussian Process Regression [26]

The flow diagram shows the steps of the code.

A more thorough description is presented below.

After initializing the prior parameters and the covariance matrices the data containing the information related to each model is loaded. The script sets up 10 specific muscles and 64 target positions.

A loop iterates through each muscle, fexecuting the following steps:

- **Data Extraction**. The angle and torque data to each muscle is extracted.
- **Model Fitting** The script applies both parametric and semiparametric fitting methods to the five different configurations: elevation plane, shoulder elevation, shoulder rotation, elbow flexion and elbow pronation. Custom function ***parametricfunctionReach*** and ***semiparametricfunctionReach*** are called to perform the fitting process. These functions incorporate GPR to build models that capture the underlying relationship between joint angles and torques.
- ***Output Model***. Once the analysis is completed the **modeldata** structure containing the details of the models for all muscles is saved into a MATLAB file.

PARAMETRIC FUNCTION

For the **parametricfunctionReach**, Bayesian Regression ([27]) is used to calculate the mean (μ) and the covariance (Σ) for the prior parameter distribution for the semiparametric model. Bayessian Regression is a statistical method that estimates the unknown parameters of a linear model by considering them as random variables with prior distributions. It takes into account prior knowledge of the parameters to provide a more precise understanding.

In the following paragraphs, the code and the corresponding mathematical implications will be detailed. Refer to figure 5.6 for a detailed flow diagram.

First the basis function or design matrix ,Φ, is evaluated for each arm configuration torque. It is used to represent the relationship between the input variables and the target outputs in a compact and mathematical form.

Then a residual squared error computation following Equation 5.4.3 is calculated , where the model parameter, β, to predict the outputs is calculated by isolating it from the linear model. y is the target/output vector, Φ is the design matrix, β is the vector of coefficients,$\hat{y}$ is the predicted output vector and $sqres$ is the residual error. This residual error will act as the covariance matrix noise.

$$\begin{aligned} y &= \Phi\beta \\ \beta &= \Phi^{+}y \\ \hat{y} &= \Phi * \beta \\ sqres &= mean((\hat{y} - y)^2) \end{aligned} \tag{5.9}$$

Finally, the mean μ and covariance Σ of the parameter β distribution that will define the prior distribution for the semiparametric model are calculated as a posterior distribution following the Bayesian linear regression.

The prior distribution mean μ_0 and covariance Σ_0 are given as input to the function. A Gaussian prior distribution describes the parameters β distribution.

$$p(\beta) = \mathcal{N}(\beta|\mu_0, \Sigma_0) \tag{5.10}$$

where μ_0 is the prior mean, and Σ_0 is the prior covariance matrix.

Given the data, the distribution is updated using Bayes' theorem to find the posterior distribution over the parameters:

$$p(\beta|\Phi, y) \propto p(y|\Phi, \beta) \cdot p(\beta) \tag{5.11}$$

Assuming Gaussian noise, the likelihood term is also Gaussian:

$$p(y|\Phi, \beta) = \mathcal{N}(y|\Phi\beta, S) \tag{5.12}$$

where S is the covariance matrix of the noise.

The posterior distribution for β given y and Φ is also Gaussian, and its mean μ and covariance matrix Σ can be found as follows:

$$\begin{aligned} \mu &= \Sigma(\Phi^T\Sigma^{-1}y + S_0^{-1}\mu_0) \\ \Sigma &= \left(\Sigma_0^{-1} + \Phi^T S^{-1}\Phi\right)^{-1} \end{aligned} \tag{5.13}$$

In summary, the prior mean μ_0 and covariance Σ_0 encapsulate our beliefs about the coefficients before seeing the data. The design matrix Φ and the response vector y represent the data we observe. The posterior mean μ and covariance Σ encapsulate our updated beliefs about the coefficients after seeing the data.

SEMI-PARAMETRIC FUNCTION

In the following paragraphs, the code and the corresponding mathematical implications will be detailed. Refer to figure 5.6 for a detailed flow diagram.

First the basis function or design matrix ,Φ, is evaluated for each arm configuration torque. It is used to represent the relationship between the input variables and the target outputs in a compact and mathematical form.

A Gaussian Process is fully specified by the **mean function** and a **covariance function**. Following [24], the mean function is the parametric model and the covariance function is the sum of squared-exponential covariance (Equation 5.15) function and a term that takes into account the uncertainty in the parameters of the parametric model.

$$\begin{aligned} \mu &= \Phi * \beta \\ \Sigma &= k(x, x') + \Sigma_{uncertainty} \end{aligned} \tag{5.14}$$

$$k(x, x') = \sigma^2 * \exp - \frac{(x - x')^2}{2l^2} \tag{5.15}$$

The length scale (l) were chosen a priori to be small which means that the correlation between observations drops off quickly as their separation increases. Leading to a more flexible model.

The vertical scale (σ^2) controls the overall variance of the function values. Higher values means more significant variation from one point to another, allowing GP to capture strong fluctuations.

Moreover the likelihood function chosen is the Gaussian Likelihood. The likelihood describes the noise model, in this case assuming normally distributed noise.

Next, hyper-parameters GPR are defined. Hyperparameters are essential parameters that define how the model behaves. There are three main hyperparameters:

- Mean. The mean previously calculated in the parametric function.
- Covariance. It includes, the length scale parameters, the vertical scale hyperparameters and the covariance previously calculated in the parametric function.
- Likelihood function. It is set to be the $log(10)$ which suggests a specific setting on the noise variance in the Gaussian likelihood.

The hyperparameters are optimized to the GP model to best explain the training data. The optimized hyperparameters are used to make predictions using Gaussian Process model. All Gaussian process computations are made using the GPML toolbox for MATLAB ([28]).

```
1 covGP = {@covSEard};
2 likfunc = @likGauss;
3 covfunc1 = {'covSum',{covGP,covUNCERT}};
4
5 % optimize the hyperparameters and then make predictions with
     the optimized
6 % parameters
7 [hypOPT fX iter] = minimize(hyp1, @gp, -100, @infExact,
     meanfunc1, covfunc1, likfunc, trainingInputs, trainingOutputs
     (:,1));
8 [meantest covtest] = gp(hypOPT, @infExact, meanfunc1, covfunc1,
     likfunc, trainingInputs, trainingOutputs(:,1), testInputs);
9 %prediction: [ymu ys2 fmu fs2   ] = gp(hyp, inf, mean, cov, lik,
      x, y, xs);
```

The prediction outputs are y_{mu} (meantest) for test output mean and y_{s2} (covtest) for test output covariance.

The output of the semiparametric model are the **hyperparameters** that will be used later for future predictions.

	Description	Advantages	Disadvantages
Parametric	Linear model used to represent torque in a single degree of freedom, accounting mainly for joint stiffness and damping.	Simplicity and efficiency due to linear nature. It avoids solving nonlinear optimization problems.	Inaccuracy with full complexity. Limited application in simpler version as it does not account for physicial properties.
Nonparametric	Utilizes Gaussian process regression to predict arm joint torques, defining mean and covariance functions. It relies on training samples to make predictions.	Adaptability, less overfitting, avoids identifiability issues as it does not need rich enough data to guarantee identifiability of parameters.	Computational complexity and demanding.
Semiparametric	Combines aspects of both parametric and nonparametric models. It uses a Gaussian process where the mean function is derived from the parametric model and the covariance is the sum of a squared-exponential covariance function and an term for uncertainty in the parameters of the parametric model.	Flexibility, generalization, local relevance (dominate predictions for inputs close to training data) with complex dynamics.	Complexity in understanding, implementing, and tuning.

Table 5.3: Comparison of Parametric, Nonparametric, and Semiparametric Models

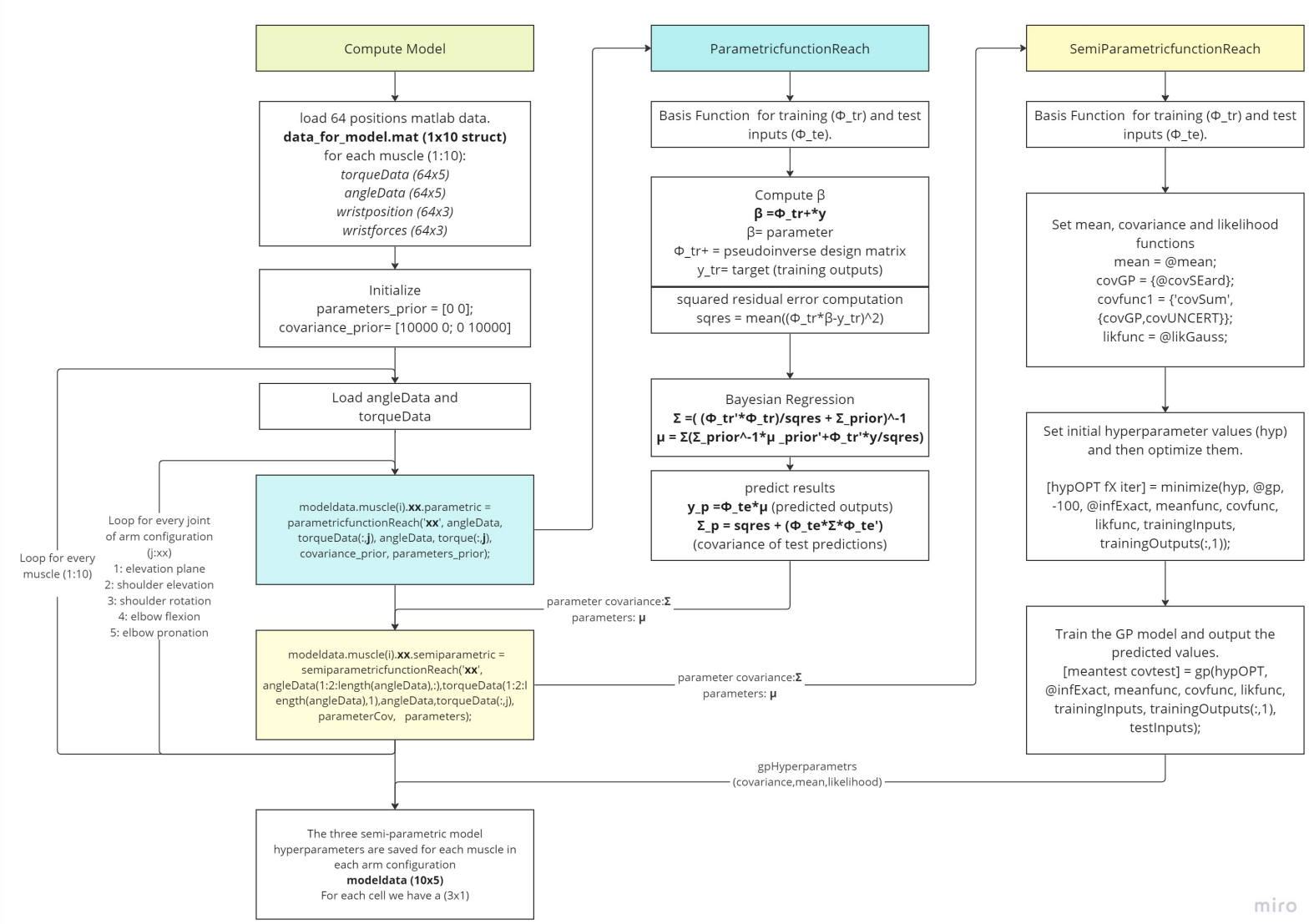

Figure 5.6: Flow Diagram for Model Computation of Semi-Parametric GPR

6 Control Design and Development

6.1 Static Controller

The development of the controller, inspired by the methodologies proposed in the paper [21], aims to determine the optimal muscle excitation to stabilize the wrist's position. The architectural layout of this controller is portrayed on Figure 6.1.

The input of the controller is the desired arm configuration. Using the Inverse of the Arm Statics model, described in Section 5, the required torques to maintain the arm in a static position are computed. These torques are subsequently given as an input into the Muscle Torque Production model, elaborated in Section 5. This model outputs the corresponding muscle excitations which will define the neural excitation input for the Dynamic Arm Simulator.

The Dynamic Arm Simulator will then determine the arm's dynamics in accordance with the Rosenbrock equation, as explained in Section 4.1.4. Post these dynamic computations, the wrist's actual position is deduced. The error between the desired and the actual wrist position will drive a PID Controller, which will calculate the corrective force needed to match the desired wrist position. The kinematic Jacobian, Section 5.3, will compute the requisite corrective torque. This torque, when combined with the desired torque from the Inverse Arm Statics, provides the total torque which is then input to the Inverse Muscle Torque model, closing the control loop.

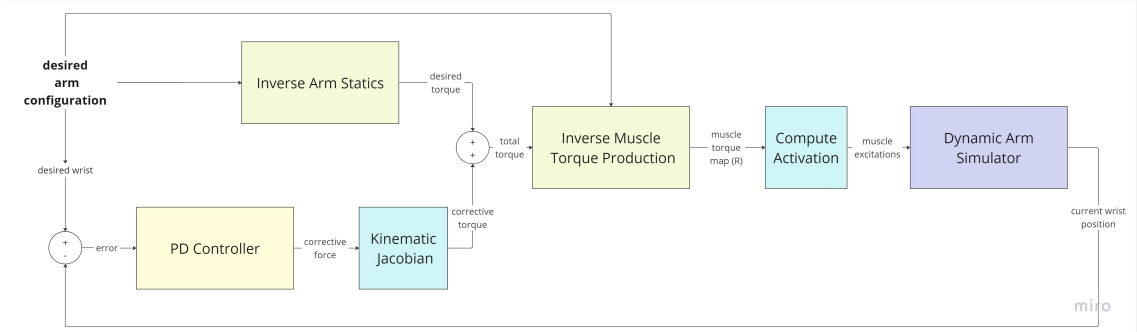

Figure 6.1: Controller Block Diagram inspired in [21]

PD-Tuning

To assure the PID Controller's performance, an exhaustive hand-tuning was undertaken across 20 different positions, concluding in values of $K_p = 300$ and $K_i = 100$. $K_d = 0$ resulting in only a PD.

6.1.1 Predicting Static Torque

To predict the static torque values for an arm configuration, the hyperparameters found for the Gaussian Process Arm Statics model described on section 5 are used. The resultant torque represent the torque needed to hold the arm static in that configuration.

6.1.2 Predicting Muscle Excitations

In muscle-driven systems, accurately determining muscle activation patterns is crucial for understanding force generation. The provided functions aim to compute optimal muscle activations under given constraints using the penalty method and a quasi-Newton optimization approach. This method is inspired by [21].

The objective is to compute the muscle activation `act` for given muscle-torque relationships `M`, desired torques `tau`, and initial guesses for activations `alpha0`.

The muscle-torque relationships `M` the Muscle Torque Production model hyperparameters calculated following the steps explain in Section 5 (see Formula 5.7). The model gives out a matrix each column represents the torque for the desired shoulder elevation plane, shoulder elevation, shoulder rotation and elbow flexion produced by each muscle group.

The objective is to solve the equation:

$$\tau_{desired} = M * \alpha_0 \tag{6.1}$$

In other words how much activation the different muscle groups need to generate the desired torque.

The **quasi-newton method** is the iterative method used to find this function´s minimum. As computing the Hessian tends to be computationally expensive, the quasi-newton method build an approximation of it iteratively. In other words, it uses and approximation of the information of the function's curvature (Hessian) to find the shortest way to the minimum.

The function is supported by the Armijo rule and a penalty-based approach, where the muscle activation are effectively optimized.

The **armijo rule** is a method to decided how big of a step to take when trying to find he lower point of a curve or surface. The step is halved every time a step is not taken into a direction where the value of the function is reduced as expected.

The **penalty function** makes sure that the muscle activations are withing the range of [0,1].

Together, they are used to determine the step size of the quasi-Newton method and to include constraints in the optimization problem.

By employing, the quasi-Newton methods, the function efficiently converges to the optimal solution.

6.1.3 Feasible Points

With this tuned controller, a grid, encompassing 960 points, was generated using the **create_grid** function (refer to Section 5.1). Each point's arm configuration was calculated using the PI force controller, as detailed in Section 5.2. The feasibility of a point was determined by the ability of the controller to sustain for 1 second its position with an error margin of 0.1. For clarification purposes, the start and the goal position of the controller are set to be the same. For seamless and accurate operations, a time step (t_{step}) of 0.001 is employed. To improve the controller's stability an auxiliary force is exerted on the arm in the first 100 steps. This eliminates potential oscillations and assures a better control performance.

The figures (gif if opened in Adobe Acrobat) below present the generated grid. Points in red signify positions where the controller failed to find the neural excitation needed to sustain for 1 second. In contrast, green points indicate successful stabilization by the controller. These feasible points will be used in the following section (Section 6.2) to generate paths for the arm to follow.

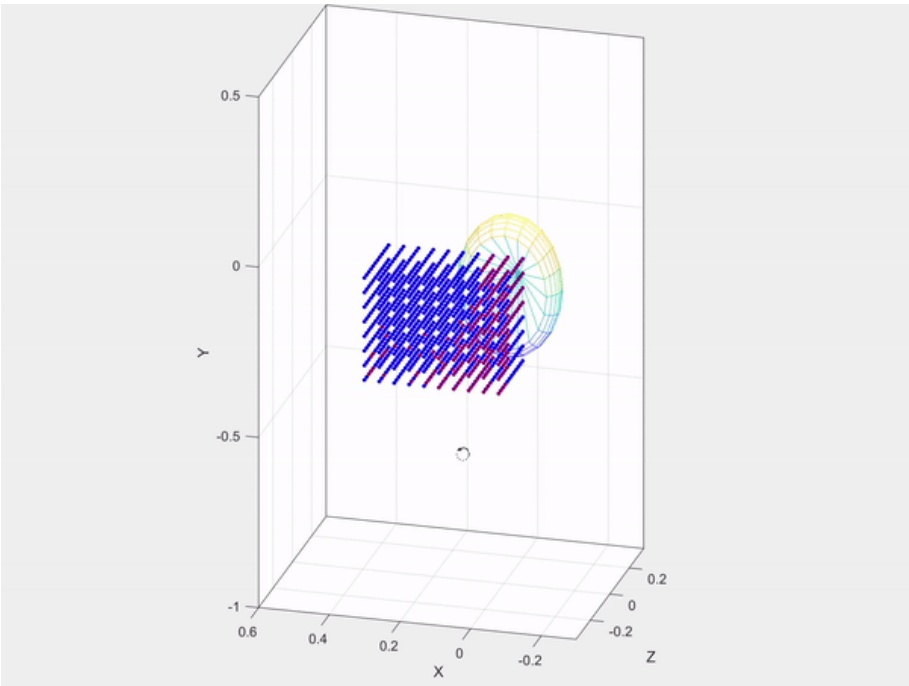

Figure 6.2: Feasible points. Open in Adobe Acrobat to see gif movement.

6.2 Path Following Quasi-Static Controller Development

The goal is to develop a controller that allows the dynamic arm to navigate or follow a specific trajectory or path. This will simulate the rehabilitation reaching task to be performed. For more information on rehabilitation tasks refer back to Section 2.3. Quasi-static control allows to design a safe, precise and movement-controlled therapy, as it focuses on slow, steady and controlled operations. It facilitates the decomposition of the entire reaching path trajectory into smaller static positions. Dividing the task into smaller steps offers some benefits under the rehabilitation lens:

- **Gradual Progression**. Rehabilitation often demands gradual approach towards exercises, specially for recovering patients. progress.
- **Progress Tracking** Breaking the movement into smaller steps helps not only patients to progress at a comfortable and safe pace but allows to keep track of the progress.
- **Minimized Fatigue** Continuous motions can take a toll in patients with limited strength and endurance. Smaller tasks reduce muscle and join strain preventing fatigue. Using a quasi-static controller can help modify the parameters to adjust to the patient's abilities.

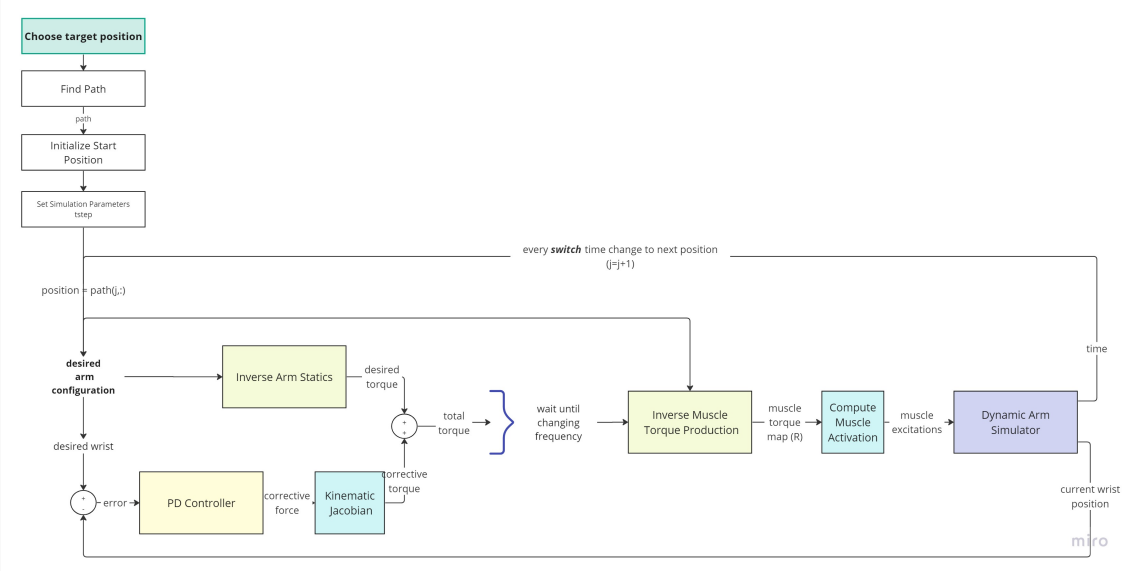

Figure 6.3: Flow Diagram Quasi-Static Path Controller

For a clearer understanding of the quasi-static control designed, refer to the flow diagram included in this chapter (Figure 6.3). Here's a descriptive step-by-step breakdown of the process of path controller:

1. **Selection of Target Position**: From a set of feasible points, a target position is chosen. This serves as the end-point or goal for the system.
2. **Path Discovery via the 'find path' Function** (see Section 6.2.1, for a more detailed explanation of the function): The selected target position is input into the 'find path' function. This function works by first identifying the nearest neighbors from the feasible points workspace. Among the first 200 nearest neighbors of the starting point, it selects one that:
   - Is at a distance $d = 4$ units away from the starting point. This distance recommendation comes from the paper [21].
   - Lies in the direction of the end-point or target position.

   Through this method, a path consisting of multiple positions is generated.
3. **Setting Initial Simulation Parameters**: At the beginning of the simulation, the initial position from the path is defined. Moreover the simulation parameter are set: $t_{step} = 0.003$, and a $frequency$ of 30Hz. The variable $frequency$ refers to the rate at which the FES (Functional Electrical Stimulation) device can change its input. This is a criteria detailed on the design Section 3 limited by the FES devices.
4. **Control Loop**: The control loop operates as previously outlined in the Section 6.1 , with a few distinct modifications:
   - **Neural Excitation**: This only takes place at set intervals based on the defined $frequency$.
   - **Position Switching**: At every $time_{switch}$ the position in the control system is updated to the subsequent one in the predefined path array.

     If the hand has not reached the vicinity of the targeted position within the set

time frame of $time_{switch}$, the system will attempt to reach the position a maximum of two more times before potentially moving to the next point or terminating.

In conclusion, the proposed controller is designed to guide a dynamic arm through specified trajectories, mimicking the rehabilitation reaching tasks outlined in Section 2.3. Quasi-static control is used to emphasize safety, precision and control, three key components for a successful rehabilitation. Key components of the control system presented include a ***find_path*** function that uses KNN to find the best positions that will define the path to the desired target. Moreover the control loop, adapted from the one on Section 6.1, operated with timed neural excitation (controlled by the variable $frequency$) and position switching to guide the arm accurately along the predefined path (controlled by variable $time_{switch}$).

6.2.1 Find Path Function

The `FindPath` function calculates a path between two positions within a predefined set of feasible wrist points. The function takes three input parameters: d, $startPos$ and $endPos$.

The code will load a file that contains all the feasible wrists positions to calculate a path from. Section 6.1 explains how the feasible positions are determined.

Once all the initial parameters are set, the function begins to construct the path. K-Nearest Neighbours is the supervised machine learning algorithm used to calculate the most optimal path. The Matlab function **knnsearch** is used to seek the predefined number of nearest neighbours to the current position within the feasible positions. Then, another **knnsearch** is performed to identify which one of the closest neighbours is closer to the end position. The selected point should be within a distance d from the current position but closet to the end position.

This process is repeated until the path reaches the desired end position. As a result, find path outputs an array called path will all the positions.

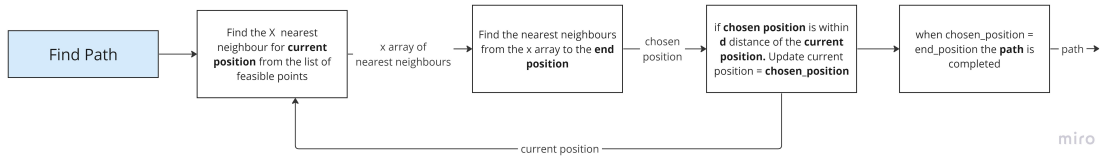

Figure 6.4: Flow Diagram Find Path Function

6.3 EMG-Influenced Controller Development

Building on top of the controllers previously designed and discussed in this section, we further extend our investigation into the intricate challenges faced by stroke patients. An after-math of stroke is the muscular imbalances, most specifically muscle spasticity, which refers to muscle stiffness that influences movement and speed. As a result, they present difficulty extending the arm. The biceps tend to be over-activated and the triceps and anterior deltoid under-activated [12]. To counteract these muscular imbalances, this controller has being design to simulate an effective and targeted electric stimulation.

Two key additions are included in the following controller design:

1. a component that emulates the over activity of the biceps post-stroke
2. the addition of FES to activate the triceps

A complete flow diagram is portrayed on Figure 6.5

6.3.1 Emulating Biceps Over-Activity

The code segment initiates by calculating muscle activation levels using the ***computeActivations*** function, which factors in muscle force models and desired torques (refer to Section 6.1). Then, it proceeds to iterate through the set of muscles. When the loop reaches the fifth muscle, the biceps, the activation level of the biceps is modulated by a factor termed ***stroke***, representing the over-activity induced due to the stroke condition. This modulation is subsequently bounded: if the resultant activation surpasses a threshold of 1, it is capped at this upper limit. In scenarios where the activation is null and the stroke factor exceeds 1, a minimal activation level of 0.3 is set to ensure a base activation. For each iteration, the modified activation values are assigned to the corresponding muscle index in the control input vector u. It is important to highlight that the values are only updated with respect to the FES frequency (concept explained in Section 6.2).

6.3.2 FES Stimulator for the Triceps

Considering the biomechanical implications, this project focuses on applying the Functional Electrical Stimulation the triceps. The control examines the triceps activation levels. This information comes from the Dynamic Arm Simulator and in a real case scenario it will come from EMG sensors situated in the triceps of the human.

If no activation is detected, a default minimal value is assigned. However, in cases with existing activation, a sigmoid-based control system is adopted.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{6.2}$$

This methods adapts the gain in a smooth manner between predefined boundaries, allowing for a nuanced control response that remains sensitive to the velocity-driven nature of spasticity.

For correct simulation it is ensured that the triceps activation remains bounded between [0.2, 1].

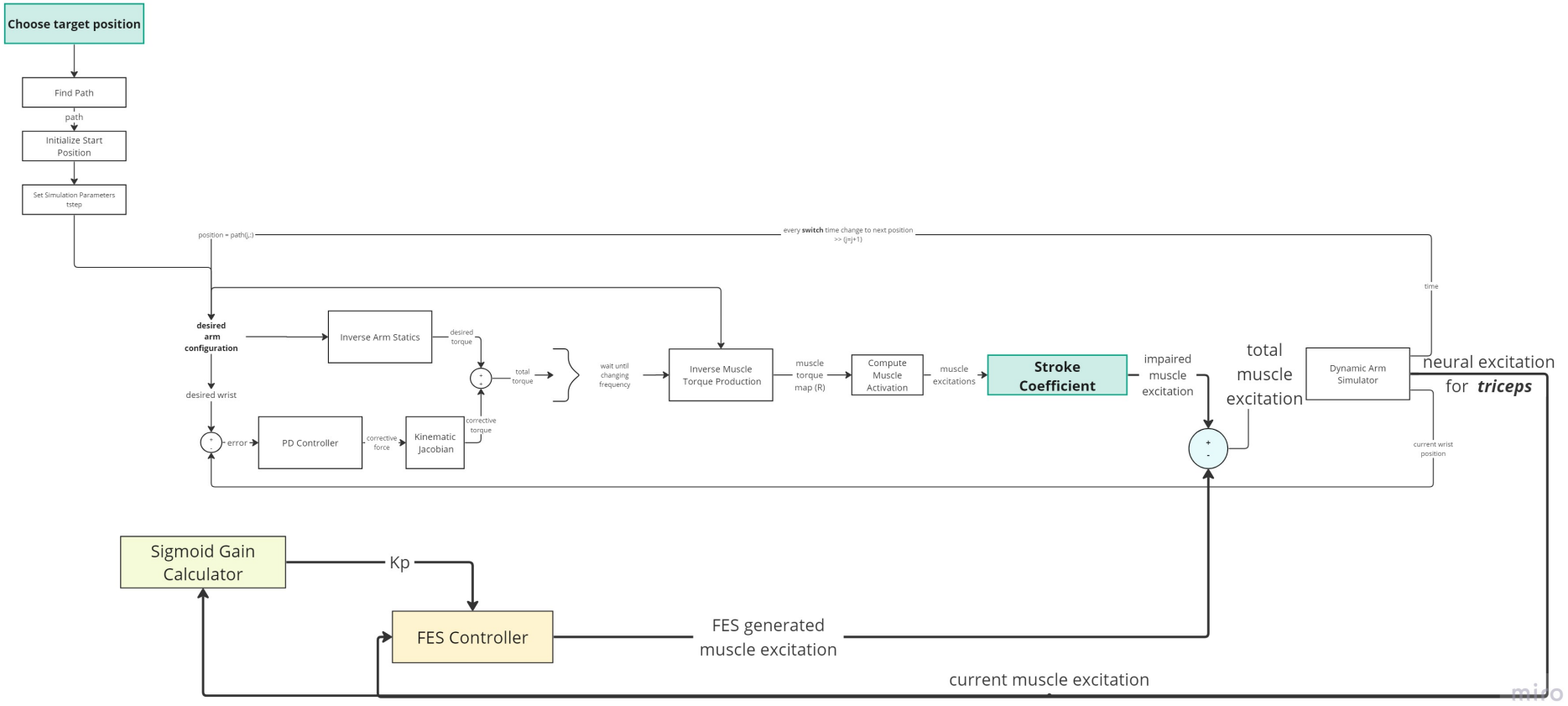

Figure 6.5: Flow Diagram for EMG-influenced FES Controller

6.3.3 PD Controller Tuning: Simulated Annealing

Building on the insights from the study by [29], their approach was adopted to refine the performance of a proportional derivative controller for a two-segment planar arm model.

The core idea was to identify the best controller gains for the sigmoid function (Equation 6.2) by minimizing the positional error over time. This error was quantified by the formula:

$$f_{error} = \sqrt{\frac{1}{2TN_m} \sum_{i=1}^{N_m} \sum_{j=1}^{3} \int_0^T (\varphi_{ij}(t) - \varphi_{ij}^{target})^2 dt} \tag{6.3}$$

Here, $\varphi_{ij}(t)$ represents the wrist's current position in either x, y, or z coordinates, while φ_{ij}^{target} indicates the desired wrist position. The term T specifies the movement's time span, and N_m denotes the 10 arbitrary movements carried out during the simulation.

To calculate the optimal values, simulated annealing algorithm [30] was utilized. Simulated annealing is a probabilistic technique for approximating the global optimum of a given function. It seeks to find the best solution.

It begins with a initial solution which is set to an initial high temperature. This temperature will control the likelihood of accepting worse solutions as the algorithm progresses. At each iteration a neighbouring solution is chosen randomly. If the neighbouring solution is better than the current one it is accepted as the new current solution. If it is worsen it may accept it depending on the temperature and the difference in quality of accepting the solution or not. After each iteration the temperature is cooled down following the cooling schedule.

The Matlab function **simulannealbn** is used with the default annealing function *annealingfast*. This function represents how the new points are generated for the next iteration. In this case it has a step with the length temperature (higher temperature higher step) and with a direction uniformly at random. The temperature function used is *temperatureexp*, an exponential decay where the temperature is multiplied by a factor between 0 and 1. The algorithm stops when the temperature reaches a predetermined minimum value, after a fixed number of iterations.

Simulated annealing avoids getting trapped in local minima, as early on when the temperature is higher the algorithm is more willing to accept worse solutions. In other words, as the temperature cools down the algorithm becomes more selective.

For this project purposes, maximum of 100 iterations is stipulated, a fixed a function tolerance at 1e-3, and an initial estimate of [10 1.3] for the sigmoid function's two gains is provided.. The bounds for the controller gains were set between [1.2 3] and [3 15].

The simulated annealing successfully calculated the proportional value to be equal to [2.99 14.478].

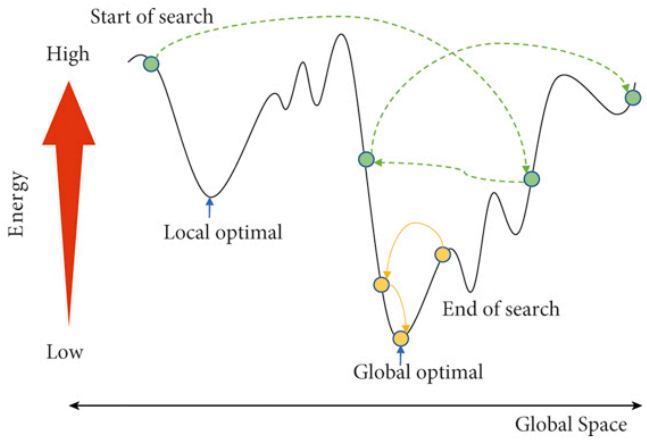

Figure 6.6: Simulated Annealing Explanatory Diagram [31]

7 Results

7.1 PI Force Controller

7.1.1 Support vs Non-Support

7.1.2 No Neural Excitation

7.1.3 Neural Excitation

7.2 Torque

7.3 Semi-Parametric GPR Models

7.3.1 Parametric

7.3.2 Semi-Parametric

7.4 Controller

7.4.1 Static Control

PID tuning

7.4.2 Path Following Quasi-Static Control

7.4.3 EMG-Influenced Control

8 Discussion

9 Future Work

10 Conclusion

Bibliography

[1] World Health Organization. “WHO methods for life expectancy and healthy life expectancy.” In: *Global Health Estimates Technical Paper WHO/HIS/HSI/GHE* (2014).

[2] Maurizio Corbetta et al. “Common behavioral clusters and subcortical anatomy in stroke”. In: *Neuron* 85 (5 Mar. 2015), pp. 927–941. ISSN: 1097-4199. DOI: 10.1016/J.NEURON.2015.02.027. URL: https://pubmed.ncbi.nlm.nih.gov/25741721/.

[3] “Prevention of cardiovascular disease : guidelines for assessment and management of total cardiovascular risk”. In: (). URL: https://apps.who.int/iris/handle/10665/43685.

[4] Saba Anwer et al. “Rehabilitation of Upper Limb Motor Impairment in Stroke: A Narrative Review on the Prevalence, Risk Factors, and Economic Statistics of Stroke and State of the Art Therapies”. In: *Healthcare 2022, Vol. 10, Page 190* 10 (2 Jan. 2022), p. 190. ISSN: 2227-9032. DOI: 10.3390/HEALTHCARE10020190. URL: https://www.mdpi.com/2227-9032/10/2/190/htm%20https://www.mdpi.com/2227-9032/10/2/190.

[5] Thinni Nurul Rochmah et al. “Economic Burden of Stroke Disease: A Systematic Review”. In: *International journal of environmental research and public health* 18 (14 July 2021). ISSN: 1660-4601. DOI: 10.3390/IJERPH18147552. URL: https://pubmed.ncbi.nlm.nih.gov/34299999/.

[6] “Investigation of upper arm muscle activation for the progress monitoring in stroke rehabilitation”. In: *AIP Conference Proceedings* 2045 (1 Dec. 2018), p. 20028. ISSN: 15517616. DOI: 10.1063/1.5080841/817075. URL: https://pubs.aip.org/aip/acp/article/2045/1/020028/817075/Investigation-of-upper-arm-muscle-activation-for.

[7] Shailesh Kantak, Steven Jax, and George Wittenberg. “Bimanual coordination: A missing piece of arm rehabilitation after stroke”. In: *Restorative neurology and neuroscience* 35 (4 2017), pp. 347–364. ISSN: 1878-3627. DOI: 10.3233/RNN-170737. URL: https://pubmed.ncbi.nlm.nih.gov/28697575/.

[8] Gad Alon, Alan F. Levitt, and Patricia A. McCarthy. “Functional electrical stimulation enhancement of upper extremity functional recovery during stroke rehabilitation: A pilot study”. In: *Neurorehabilitation and Neural Repair* 21 (3 Sept. 2007), pp. 207–215. ISSN: 15459683. DOI: 10.1177/1545968306297871.

[9] Dimitra Blana, Robert F. Kirsch, and Edward K. Chadwick. “Combined feedforward and feedback control of a redundant, nonlinear, dynamic musculoskeletal system”. In: *Medical Biological Engineering Computing* 47 (5 May 2009), pp. 533–542. ISSN: 0140-0118. DOI: 10.1007/s11517-009-0479-3.

[10] Yu-Luen Chen et al. “Development of the FES System with Neural Network+PID Controller for the Stroke”. In: *2005 IEEE International Symposium on Circuits and Systems* (), pp. 5119–5121. DOI: 10.1109/ISCAS.2005.1465786.

[11] Katie L Meadmore et al. “Functional electrical stimulation mediated by iterative learning control and 3D robotics reduces motor impairment in chronic stroke”. In: *Journal of NeuroEngineering and Rehabilitation* 9 (1 2012), p. 32. ISSN: 1743-0003. DOI: 10.1186/1743-0003-9-32.

[12] Christopher T. Freeman. “Upper Limb Electrical Stimulation Using Input-Output Linearization and Iterative Learning Control”. In: *IEEE Transactions on Control Systems Technology* 23 (4 July 2015), pp. 1546–1554. ISSN: 1063-6536. DOI: 10.1109/TCST.2014.2363412.

[13] Arash Abarghooei, Hassan Salarieh, and Mehrdad Boroushaki. “Development and control of an intelligent assistive exo-glove via fuzzy controller and emotional learning system”. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 235 (16 Aug. 2021), pp. 3058–3070. ISSN: 0954-4062. DOI: 10.1177/0954406220959100.

[14] Antonie J. Van Den Bogert, Dimitra Blana, and Dieter Heinrich. “Implicit methods for efficient musculoskeletal simulation and optimal control”. In: vol. 2. 2011. DOI: 10.1016/j.piutam.2011.04.027.

[15] Edward K. Chadwick et al. “A real-time, 3-D musculoskeletal model for dynamic simulation of arm movements”. In: *IEEE Transactions on Biomedical Engineering* 56 (4 2009). ISSN: 00189294. DOI: 10.1109/TBME.2008.2005946.

[16] Ge Wu et al. “ISB recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion - Part II: Shoulder, elbow, wrist and hand”. In: *Journal of Biomechanics* 38 (5 2005). ISSN: 00219290. DOI: 10.1016/j.jbiomech.2004.05.042.

[17] Mary D. Klein Breteler, Cornelis W. Spoor, and Frans C.T. Van Der Helm. “Measuring muscle and joint geometry parameters of a shoulder for modeling purposes”. In: *Journal of Biomechanics* 32 (11 Nov. 1999), pp. 1191–1197. ISSN: 0021-9290. DOI: 10.1016/S0021-9290(99)00122-0.

[18] Bruno Siciliano et al. *Robotics: Modelling, planning and control*. 2009. DOI: 10.5860/choice.46-6226.

[19] Reza Sharif Razavian et al. “Feedback Control of Functional Electrical Stimulation for 2-D Arm Reaching Movements”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26 (10 2018). ISSN: 15344320. DOI: 10.1109/TNSRE.2018.2853573.

[20] Emily J. Griffis et al. “Closed-Loop Deep Neural Network-Based FES Control for Human Limb Tracking”. In: IEEE, Dec. 2021, pp. 360–365. ISBN: 978-1-6654-3659-5. DOI: 10.1109/CDC45484.2021.9683206.

[21] Derek N. Wolf and Eric M. Schearer. “Developing a quasi-static controller for a paralyzed human arm: A simulation study”. In: vol. 2019-June. 2019. DOI: 10.1109/ICORR.2019.8779381.

[22] Derek N. Wolf and Eric M. Schearer. “Holding Static Arm Configurations with Functional Electrical Stimulation: A Case Study”. In: *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society* 26 (10 Oct. 2018), p. 2044. ISSN: 15344320. DOI: 10.1109/TNSRE.2018.2866226. URL: /pmc/articles/PMC6284830/%20/pmc/articles/PMC6284830/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6284830/.

[23] H.Harry Asada. “Introduction to Robotics”. In: *MIT Education* (2011).

[24] Eric M. Schearer et al. “Semiparametric Identification of Human Arm Dynamics for Flexible Control of a Functional Electrical Stimulation Neuroprosthesis”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 24 (12 2016). ISSN: 15344320. DOI: 10.1109/TNSRE.2016.2535348.

[25] Distill. *Visual Exploration of Gaussian Processes*. 2019. URL: https://distill.pub/2019/visual-exploration-gaussian-processes/.

[26] *Gaussian Process Regression*. URL: https://ru.m.wikipedia.org/wiki/%5C%D0%5C%A4%5C%D0%5C%B0%5C%D0%5C%B9%5C%D0%5C%BB:Gaussian_Process_Regression.png.

[27] Carlos Eduardo Cancino Chacon, Maarten Grachten, and Gerhard Widmer. “Bayesian Linear Basis Models with Gaussian Priors for Musical Expression”. In: *Austrian Research Institute for Artificial Intelligence* 1.1 (2014), p. 23. DOI: doi.

[28] *Documentation for GPML Matlab Code*. URL: http://gaussianprocess.org/gpml/code/matlab/doc/.

[29] Kathleen M. Jagodnik and Antonie J. van den Bogert. “Optimization and evaluation of a proportional derivative controller for planar arm movement”. In: *Journal of Biomechanics* 43 (6 2010). ISSN: 00219290. DOI: 10.1016/j.jbiomech.2009.12.017.

[30] William L. Goffe, Gary D. Ferrier, and John Rogers. “Global optimization of statistical functions with simulated annealing”. In: *Journal of Econometrics* 60 (1-2 1994). ISSN: 03044076. DOI: 10.1016/0304-4076(94)90038-8.

[31] Li Yang. “Research on Logistics Distribution Vehicle Path Optimization Based on Simulated Annealing Algorithm”. In: *Advances in Multimedia* 2022 (2022). ISSN: 16875699. DOI: 10.1155/2022/7363279.

A Title

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Technical
University of
Denmark

Brovej, Building 118
2800 Kgs. Lyngby
Tlf. 4525 1700

www.byg.dtu.dk