UNIVERSITAT DE BARCELONA

Final Degree Project

Biomedical Engineering Degree

“Contribution to development of a fully predictive Simulation Framework for Crutch-assisted Gait“

Barcelona, 05th of June of 2024

Author: Benet Fité Abril

Director: Míriam Febrer Nafría

Tutor: Jordi Colomer Farrarons

Abstract

After a traumatic event such as a spinal cord injury or a stroke, patients need to undergo a recovery process to maximize their post-injury mobility, which usually involves using an assistive gait device. This thesis is part of a broader effort to develop a tool to aid clinicians in creating highly personalized rehabilitation plans, using individualized multibody simulations of the human body and trajectory optimization techniques.

Among the different assistive devices that are available, this thesis focuses on the forearm crutches, more specifically on the three-point and swing-through crutch gait patterns. It builds on a previously developed optimal control formulation for a similar gait pattern, increasing its complexity by adding muscle torque generators (MTGs) to model muscle actuation, aiming towards a 3D fully predictive simulation framework. For that purpose, MATLAB is used as the programming interface, the optimal control problem is solved with GPOPS-II and OpenSim is used to host the multibody model. The developed problem formulation, explained in detail in the thesis, produces simulations that show realistic behaviours and the joint variables predicted are within reasonable values. The results demonstrate that incorporating MTGs in this kind of simulations is feasible; however, they also show that finding an optimal solution for such a complex problem is difficult, as it was not fully achieved during this thesis. Nevertheless, various hypotheses are presented to explain this outcome, along with proposed approaches to achieve convergence in future projects.

Acknowledgments

This Bachelor's thesis would have not been possible without the help and support of a number of people. Firstly, I would like to give my most sincere thanks to Míriam Febrer, this thesis' director, who has gone above and beyond any expectations I had to make sure this thesis was successful. Her continuous effort and her encouragement when I got frustrated because simulations did not work as expected have shaped this thesis, and the frequent meetings to discuss new ideas and implementations have allowed me to explore many things under the scope of this project. It has been a pleasure to work with you!

Secondly, I would like to acknowledge the work of Carlos Pagès, a master's student working in the lab at the same time as me, for allowing me to use part of his thesis for the development of this project, and for his support in implementing it to the crutch gait simulation. We have shared happiness when a simulation converged to an optimal solution, and pain when it did not (too often). Without his work this thesis would have not been possible, so thanks! I would also like to extend my gratitude to my colleagues at BIOMEC, because from day one I felt included. I am glad to have met people from many different nacionalities, and to have been able to collaborate in your research.

Thirdly, I would like to acknowledge the support of Jordi Colomer, the tutor of this thesis, for his help in defining the structure of the project, and for his feedback on the first thesis drafts and the timeline of execution.

We are in 2024, the age of AI, and it would not be fair to not acknowledge the support of LLM tools in the writing and development, specially ChatGPT for writing tips, coding suggestions and debugging ideas, and Bing Copilot for his fantastic ability to find an academic paper to cite practically anything. This tools are still quite new and are definitely a game changer, becoming more and more present in the academic and professional world.

And last but not least, I would like to express a huge thanks to my fellow engineering friends, specially to Anna, for providing a way to escape the stress of the final degree thesis, as well as valuable feedback at the final stages of the project.

Contents

List of Figures

List of Tables

List of Abbreviations

GPOPS General Purpose OPtimization Software

MSK Musculo-skeletal

MTGs Muscle torque generators

SCI Spinal cord injury

ROM Range of motion

DOF Degree of freedom

OCP Optimal Control Problem

CGCM Crutch ground contact model

FGCM Foot ground contact model

PWB Partial weight bearing

NLP Non-linear programming

IPOPT Interior Point OPTimizer

SNOPT Sparse Nonlinear OPTimizer

1 Introduction

1.1 Motivation

Reaching as far back as the ancient Egyptians, crutches have been used for thousands of years to overcome gait disorders [1]. Despite society having transformed since the time of the Pharaohs, crutches still remain nowadays the primary choice for walking assistance when there is impaired use or injury of one or both legs. Even if a crutch user might decide to use other devices such as wheelchairs or scooters, crutch use forces an upright posture, an active lifestyle and grants more independence, all of which are highly beneficial for the wellbeing and long-term health of the patients.

To look at some statistics, in Europe around 4.2% of women and 3.4% of men have a walking disability [2], many of which use crutches. A wide variety of diseases can lead to crutch use, such as Parkinson, multiple sclerosis, strokes or spinal cord Injuries (SCI). Furthermore, the use of crutches is becoming more common as life expectancy increases and society ages. The European comission estimates that the amount of people over 65 in Europe will increase from 17.5% in 2011 to almost 30% in 2060 [3], potentially raising the number of age-related disease and the amount of crutch users.

When crutch use comes from a traumatic event such as a stroke or SCI, rehabilitation is key to recover as much of the patient's mobility as possible. This rehabilitation needs to be planned taking into account the specific functional status of each patient, because it is widely accepted that the major improvements after a traumatic neurological event come during the first six months [4].

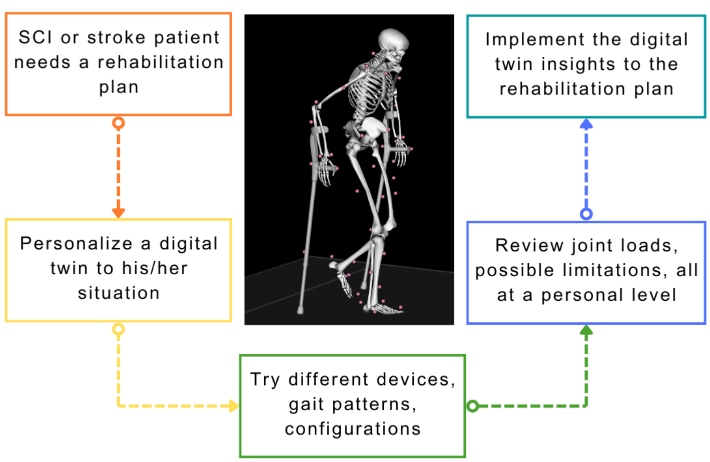


Figure 1: Diagram showing what a future tool using this thesis' advancements could do.

This Bachelor's thesis is encompassed in a bigger project that aims to build a digital twin personalized to a specific patient, like the one in Figure 1. On this twin, you could try different assistive devices, different configurations, different gait patterns, and it would provide valuable insights on how rehabilitation would turn out, identifying factors that promote or hinder the functional improvement of the patient. This way, the doctor could show him/her to walk in the most beneficial way or include/exclude specific devices from this rehabilitation plan, all before the patient even tries to walk. With a tool like that, physicians could increase the accuracy of the treatment plan for each patient and better choose the assistive devices. This is specially important with crutches, which completely change the kinematic chain of the upper body, and small misalignments can cause lumbar or shoulder injuries, specially to older people [5]. My contribution to this bigger project revolves around improving of the crutch-gait prediction simulation framework.

In the context of biomechanical research, investigating the impact of different walking aids on the biomechanics of the upper trunk requires expensive equipment and a considerable amount of time to prepare patients for measurements in the laboratory. These measurement sessions can cause fatigue due to repetitive movements, especially considering that patients may have altered energy levels and a high risk of falls. For this reason, before taking measurements with real patients, it would also be beneficial to have a digital twin to conduct tests without putting anyone at risk.

Finally, on a personal note, my motivation for doing this project comes from mainly two sides: first, an interest in doing a bachelor's thesis that included a strong mathematical component, to compensate the lack that the Biomedical Engineering degree has when compared to other Engineering degrees; and second, a curiosity for the field of biomechanics and multibody simulation after enjoying Dr. Ciaran Simms course on Multibody Dynamics during my Erasmus in Trinity College Dublin. Finally, having used crutches myself a few times in my life after sports injuries, I was interested in the topic and the potential impact that this project could have on the rehabilitation of patients that are forced to use crutches in a daily basis.

1.2 Institutions and work period

This project has been developed in the Biomechanical Engineering Lab (BIOMEC, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Av. Diagonal, 647, Les Corts, 08028 Barcelona) of the Universitat Politècnica de Catalunya (UPC), under the supervision of Dr. Míriam Febrer Nafría. BIOMEC is a research group within the Research Centre for Biomedical Engineering (CREB). The mission of the lab is to develop multibody models to simulate human movement dynamics, and to design robotic devices to assist human locomotion. It is a multidisciplinary and international environment, where I'vs had the opportunity to work with people from different backgrounds and countries and have been integrated into weekly team meetings and seminars, as well as focus sessions on musculoskeletal modelling and simulation.

The work period for this thesis has been from the 22nd of January to the 5th of June of 2024, dedicating around 20h per week to the thesis.

1.3 Objective

The objective of my final degree thesis is to develop a customizable Muscle Torque Generator (MTG) driven simulation framework for three-point crutch gait pattern, capable of predicting movement without tracking experimental data. Once and if this step is finished, the second objective is to virtually test different levels of muscle weakness in a small set of simulated patients.

This simulation framework will be used in the lab for a future project involving real SCI patients and multiple simulation frameworks for different scenarios, so it needs to be well documented and prepared for that purpose.

1.4 Project scope

It is important to define what the scope of the project is, to better direct the resources and time invested in the project and to avoid going too far away from the project objective. For this project, the following limits apply:

- There are many types of crutches, such as axiliary crutches or strutter crutches. For this project, only forearm crutches will be considered. The contact hand-crutch will not be modelled due to its complexity, assuming instead that the crutch and the hand are rigidly connected.
- The contact models involved in the simulations will only be changed if necessary, as they are not the main focus of this thesis.
- The only gait patterns studied in this project will be three-point gait and, during development stages, swing-through.
- The MTGs used in this project are developed by another student in the scope of his thesis, my job is understanding and implementing them in the crutch gait simulation framework.
- The softwares used are predefined by previous work in the lab, they are MATLAB, GPOPS-II and OpenSim, and this project will not change them. However, in Concept Engineering (Section 3) some alternatives are exposed to understand the strengths and weaknesses of the chosen simulation environment.
- The simulations will only be validated qualitatively and against the literature. The quantitative experimental vaildation is left outside of this project due to it being very time-consuming for the objectives of this Bachelor's thesis, but it will likely be done in the future in the scope of another project.

To achieve the project objectives, I will follow these steps:

1. **Model development**: to make this task more feasible, I have taken various simpler formulations of the problem at hand and experimented with them first, building intuition and knowledge towards solving the more complex MTG-driven 3D simulation.
   - Familiarize myself with the optimal control problem formulation and biomechanic simulations.
   - Develop a "simple" version of the crutch control problem: a 2D ideal torques simulation based on the previous work from [6].
   - Add MTGs to the simulation framework.
   - Progressively increase the DOF to reach a 3D version of the three-point crutch gait.
2. **Result evaluation**
   - Compare the MTG-driven and the ideal torque models to see which one performs better.
   - Analyse the outputs of the simulations.
   - Design a few virtual patients, reflecting on which variables in the problem formulation define weakness.
   - If the final problem formulation allows it, test the effects of muscle weakness on the simulations.

1.5 State of the art

Biomechanics is a field of research that is growing all over the world, motivated by the aging of the general population and the migration to personalized medicine techniques. Computationally predicting human movement can be a great tool to aid doctors in decision making, so there are many teams working on that.

Several studies have significantly contributed to the field of human motion prediction. Notably, Fournier et al, 2018 [7] and Mouzo et al 2018 [8] both developed 3D 4 point gait models, which were used for tracking experimental data and not for prediction of new motions. Other teams have tried prediction of gait patterns, like Ackermann et al, 2012 [9] and Liu at al, 2011 [10], but they used very simple 2D models and only predicted the swing-through gait pattern. Only very recently, Falisse et al, 2019 [11] developed an optimal control simulation framework for predicting healthy gaits, opening the door to using it for pathological gait patterns. Although they are not the only ones, they are the most advanced in terms of efficiency, detail and personalizations of the simulations. For that reason, Falisse's team and BIOMEC have been in contact ever since to share advancements and help each other.

Furthermore, to have a bigger picture on the topic, this two reviews give intersting analysis and insights on the current state and future of the field: the most recent one, from Febrer-Nafría et al [12], provides a comprehensive overview of the research regarding predictive multibody dynamic simulation of human movements; and this other one from De Groote et al [13] presents the different simulation approaches to predict walking, and exposes what are the different challenges that future works such as this one will face to increase the accuracy of the simulations. Throughout the next section (theoretical background), more specific examples of previous investigations from teams all around the world will be commented on, formatted in an way that is easy for the reader to understand and set the theoretical foundation necessary for understanding this project.

Motivated by the lack of complex predictive models in the literature, in the BIOMEC lab there have been many theses before this one about prediction of human motion. Pallarès-López's thesis [14] developed an optimal control formulation to predict dynamically consistent walking motions in 2D lower body simplification, and later Dr. Febrer-Nafría's PhD thesis expanded this analysis to 3D motion, and included assistive devices in the simulation such as crutches and orthoses, making them potentially personalizable to a patient. She also developed a 3D torque-driven crutch gait simulation framework for a specific gait pattern [6], which is the starting point for this thesis as will be explained later.

Regarding the Muscle Torque Generators (MTGs) that will be used in this project, Lasierra's thesis [15] explored the possibilities and the reliability of MTGs, comparing the performance against torque driven simulation in simple scenarios; and Vilanova-Badosa's thesis [16] performed a sensitivity analysis on the effects of the MTG parameters on the solutions of optimal control prediction problems, aiming to clarify which MTG parameters should be personalized for musculoskeletal simulations. Finally, in Carlos Pagès's thesis, which is still not finished, the calibration of the MTG parameters to a specific subject is performed, to be able to run full body simulations. This calibrated parameters are used in this project's simulation framework.

2 Theoretical background

In this section I will provide the reader with the necessary background information to understand the work done in this Bachelor's thesis. I will start by explaining the biomechanics of human motion, focusing on the biomechanics of crutch walking. Then, I will introduce the concept of modelling the human body as a multibody system, explaining the skeletal and muscle modelling requirements. Finally, I will introduce the concept of Muscle Torque Generators (MTGs) and summarize the basics of optimal control simulations and the output they provide.

In every section, the reader will find also some relation of the theoretical concepts with the work done in this project, to help him/her understand the importance of the concepts explained in the context of the project.

2.1 Biomechanics of human motion

By definition, biomechanics combines principles from mechanics with biology to analyze the forces, motions, and structures involved in human movement [17]. More specifically, it looks at everything from the forces acting on the body to the internal forces generated by muscles and joints. This section will focus on the biomechanics of crutch walking, defining the gait patterns that will later be explored in the project.

2.1.1 Anatomical planes

Anatomical planes are the standard divisions used in anatomy to describe the human body's orientation and structure (Figure 2). They are:

1. **Sagittal plane**: Divides the body into left and right sides.
2. **Coronal (frontal) plane**: Divides the body into front and back sections.
3. **Transverse (horizontal) plane**: Divides the body into upper and lower parts.

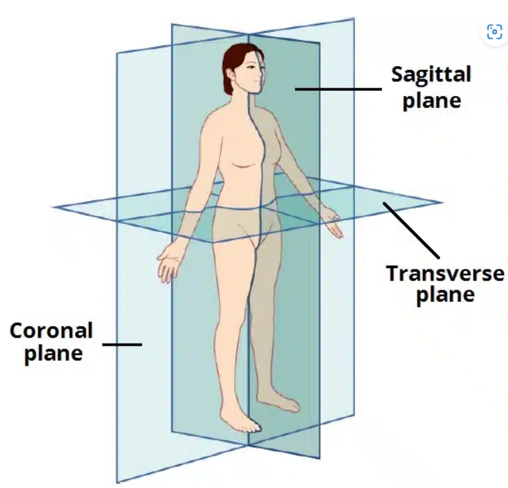


Figure 2: Standard anatomical planes of the human body. Source: [18]

In the case of gait analysis, a common simplification [19] is to only take into account the motion in the sagittal plane and ignore the other two, because it is the plane where most of the motion occurs. In more complex situations, such as injury analysis in crutch walking, a 3D model might be necessary to really capture the complexity of the movement, but a 2D model can provide some nice insights and guidance towards solving the more complex 3D approach, which is why part of this project is centered on a 2D sagittal version of crutch walking.

2.1.2 Healthy gait cycle

Healthy gait is a complex, coordinated process involving the musculoseletal and CNS system working in syncrony to achieve efficient and stable locomotion [20]. As seen in figure 3, the cycle has two defined sections: the stance phase, where the foot is in contact with the ground, and the swing phase, where the foot is in the air. For a healthy subject, this phases usually divide in 60% stance and 40% swing, but this can vary depending on the speed of the gait [19].

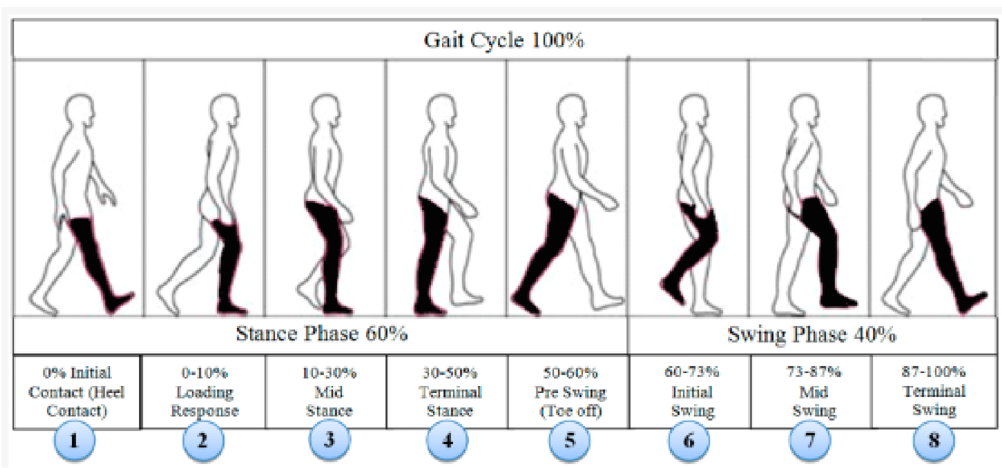


Figure 3: Healthy gait cycle. Source: [21]

2.1.3 Crutch gait cycle

Walking with crutches significantly changes the gait pattern, most notably the arms stop oscillating, the upper body muscles start producing force and the trunk becomes more flexed to allow the crutches to provide better support. This causes the joint angles and loads to be quite different from normal gait, something that can cause problems if too much load is put on a joint that cannot manage it.

A variety of crutch gait patterns exist [22] and using one or the other depends on the specific patient requirements, such as limb strength, balance, laterality of impairment... [2] For example, if the patient has balance issues but no weight bearing problems, a 2-point contralateral pattern might be chosen. If he has only one specially weak leg, he might use the 3-point partial weight bearing pattern, or even the swing-through. In figure 4 you can see the most common patterns for crutch gaits that require some type of assymetric weight bearing. For the scope of this project, only swing-through non weight bearing (WB) and 3-point partial weight bearing (PWB) gaits have been studied. For the reader to picture them better, swing-through is the crutch gait used when an injury makes it impossible to bear weight on one foot, and three-point is the gait pattern used when one foot/leg does not have full strength and needs support for weight bearing.

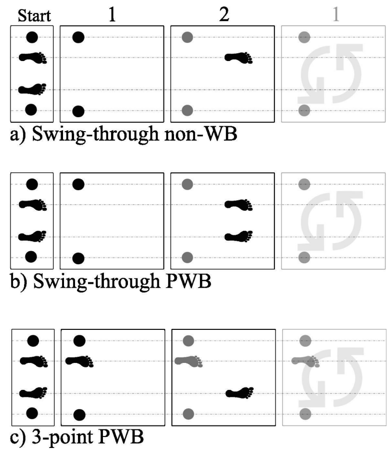


Figure 4: Crutch gait patterns studied in this project. Source: Adapted from [2]

2.2 Human body modelling: a multibody system

Modelling the human body is key to analyse the intricacies of human motion and is becoming more important in the context of personalized medicine. Some applications include refined diagnostics, prognostic prediction and treatment plan definition for a variety of neuromusculoskeletal diseases [23]; and also design and testing of medical devices like crutches or exoskeletons. For these musculoskeletal (MSK) models, the body is treated as a multibody system in which the bones are treated as rigid bodies connected by joints and muscles are modelled in diverse ways (see next sections). The central nervous system (CNS) is much more complex to model and the intricacies of it are outside the scope of this project, but a brief summary of the CNS modelling in this thesis is included.

2.2.1 Skeletal modelling

The skeletal system is normally modelled as an open kinematic chain with rigid bodies representing bones and joints linking a rigid body to the next one [24]:

- The **rigid bodies** are defined by a series of parameters that capture all the physical information necessary for a kinematic and dynamic analysis, called Body Segment Parameters (BSP). Some of these parameters are total mass, tensor of inertia, length and center of mass.

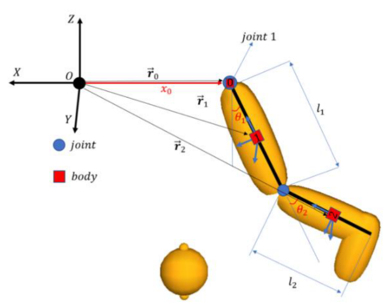


Figure 5: Multibody model of a leg. Source: Multibody Dynamics Lecture, TCD, Prof. C. Simms

- The **anatomical joints** are usually modelled as ideal joints and they can be revolute (or hinge) joints, spherical joints or pivot joints depending on the type of motion and the DOFs [6]. They are restricted to only allow physiological ROMs. Another type of joint, called *weld joint*, is used when two rigid bodies should not move relative to one another, such as the hand-crutch joint in this thesis.

In Figure 5 you can see a multibody model of 2 bodies, the thigh and the lower leg, with revolute joints allowing movement between the segments. Each of these segments would have all BSP defined so, via equations of motion, a software can calculate the motion of the bodies when a torque is applied at the hip for example. Depending on the objectives of the study at hand, we can have 2D or 3D models, and represent the whole body or only a part of it (like in Figure 5).

2.2.2 Actuation modelling

The actuation refers to the forces that cause the body to move. In the human body without any active assistive device, this is done by muscles, and they are modelled in diverse ways:

Ideal torque generators

This approach, used by [6], is one of the simplest way to model muscle actuation. It can be easily understood by imagining each joint has a servomotor inside that can produce force in both the positive and negative directions of each DOF. That way, you model all muscles actuating in a joint with a single parameter, the torque generated by this motor. This approach has some limitations, because it does not capture any of the muscle modelling requirements that will be explained in the next section. For this reason, one of the aims of this project is to implement MTGs to the crutch model to see if they can improve the realism of gait pattern prediction, to bring it closer to real life clinical applications.

Individual muscle modelling

The mechanical properties of muscle tissue can be captured using passive elements like springs and dampers [25]. Various models in the literature combine the features of these mechanical components differently. The most widely used individual muscle model, due to its relative complexity and ease of computational implementation, is the Hill-type muscle model [26], which can be seen in Figure 6.

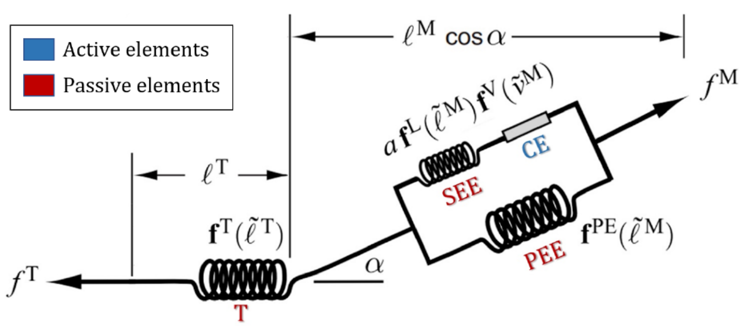


Figure 6: Representation of the Hill-type muscle model. Taken from [15]

This model comprises four main components: the contractile element (CE), the serial elastic element (SEE), the parallel elastic element (PEE), and the tendon (T) [27][28]. The CE represents the active (or contractile) properties of the muscle, while SEE, PEE, and the tendon represent passive nonlinear stiffness. More specifically, SEE represents the elasticity of actin-myosin crossbridges (that is why it is connected in series to the contractile element), PEE indicates the passive elastic properties of muscle fibers, and the tendon encapsulates the elastic properties of the tendon [26].

These components contribute to overall force production in a quite intuitive way, depicted graphically via four characteristic curves:

1. The **active force-length curve**, f_a, illustrates the active tension generated by the CE of the sarcomere, showcasing the interaction between myofibrils. Peak tension occurs at the sarcomeres' resting length, l_{opt}, aligning actin and myosin filaments optimally. An example can be seen in the first plot of Figure 7(a), where a decline in force with shortening and lengthening is observed. This occurs because as myofibrils overlap or become out of range from each other, the ability to produce force is reduced [29].

2. The **passive force-length curve**, f_{PE}, represents the passive tension from PEE, not contributing to tension generation when the muscle is in a shortened position. As sarcomere lengths increase, passive tissues reach full length, providing resistance to further lengthening. The length were this resistance appears is called slack length l_0, and usually aligns with l_{opt} [29]. It can be seen in the first plot of Figure 7(a), in a dashed line.

3. The **force-velocity curve**, f_v, accounts for the muscle's force generation capacity, influenced by contraction velocity and type (eccentric or concentric). Force generation increases during eccentric contraction but decreases during concentric contraction [26]. This curve can be seen in the second plot of Figure 7(a).

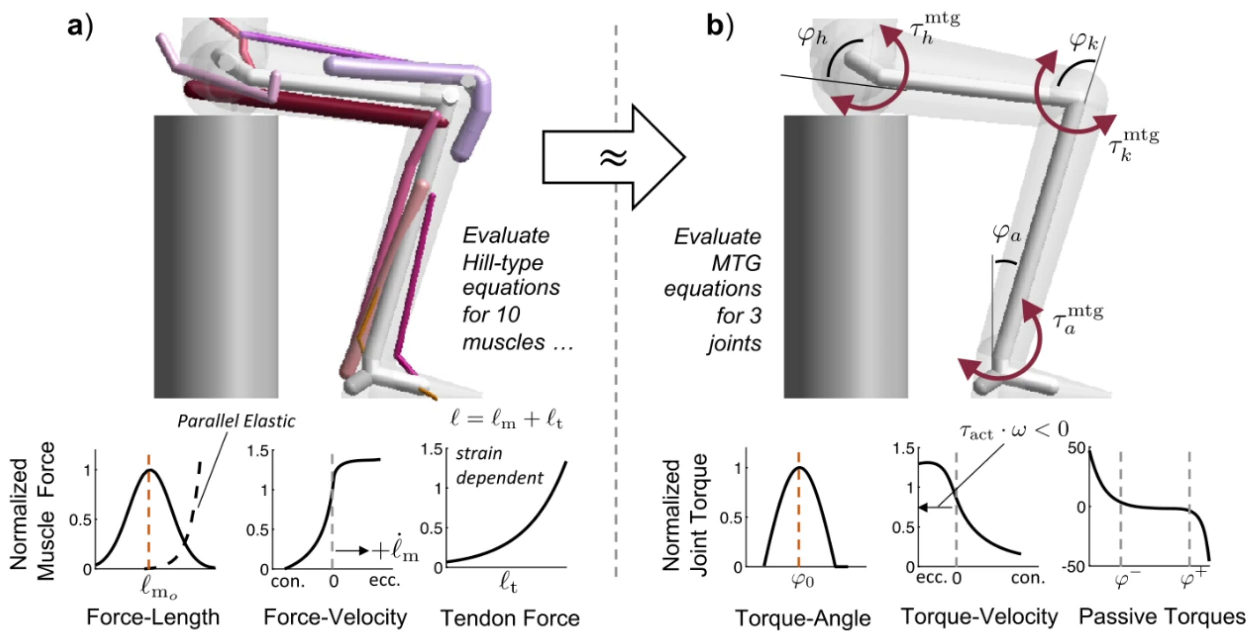


Figure 7: Leg model actuated by individual Hill-type muscles (a) and MTGs (b). Interesting insights about the modelling potential of MTGs can be seen. Source: [30]

4. The **tendon force-length curve**, f_T, depicts forces transferred from CE and PEE to bones via the tendon. The tendon's nonlinear elasticity includes a slack region with no force generation, gradually becoming more linear with higher strain. It can be seen in the third plot of Figure 7(a).

Muscle Torque Generators (MTGs)

The MTG approach has a complexity that is between individual muscle modelling and ideal torque generators. It has been used by Inkol et al [30] to simulate optimal sports performance and by Milliard et al [31] to predict the motions and forces of wearable robotic systems, and studied thoroughly in the BIOMEC as explained in the state of the art (Section 1.5).

Since each joint's movement is driven by an agonist group and an antagonist group working against one another, the MTG approach assumes each joint to have a pair of actuators that can produce torque in opposite directions. Each MTG component (in Eq. 1, τ_+ and τ_-) represents a muscle group, so for example in the elbow joint the agonist would be the biceps and the antagonist the triceps. The MTG formula is:

$$\begin{gathered} \tau_+, \tau_- = act \cdot \tau_{\text{max}} \cdot f_{FA}(\theta) \cdot f_{FV}(\omega) \\ \tau = \tau_+ - \tau_- + \tau_{\text{passive}} \end{gathered} \tag{1}$$

In this formula there are terms representing:

- **Activation** (act): describes how much the muscle is activated. It is a value between 0 and 1, where 0 means the muscle is not activated and 1 means the muscle is fully activated. In this project, the activation is defined as a control and is calculated by the optimization algorithm.
- **Maximum isometric torque** (τ_{max}): describes the maximum torque that the muscle can produce. It is a parameter that needs to be calibrated (see Section 4.3.3).
- **Force-angle relation** ($f_{FA}(\theta)$): what fraction of its maximum force can a muscle group produce at a given angle. The angle implicitly represents the length of the attached muscles in the joint. In the figure 7 (b), first plot you can see the evolution of this curve along the length axis. Note the optimal angle (θ_0), where the muscle can produce the most force, and note the similarities and differences with the Hill-type model.
- **Force-velocity relation** ($f_{FV}(\omega)$): what fraction of its maximum force can a muscle group produce at a given velocity. In the figure 7 (b), second plot you can see the evolution of this curve along the velocity axis. The curve is flipped with respect to the Hill-type model, but this is due to a sign convention.
- **Passive curve** (τ_{passive}): how much force is generated by the passive effect of tendons, collagen, etc. It can be seen in the figure 7 (b), third plot. Note how one pair of MTGs shows both the slack length of the positive and the negative torque generators, showing the model complexity reduction capabilities of this model.

All these terms allow MTGs to capture muscle behaviour in a quite realistic way showing the antagonist behaviour of flexor-extensor groups, something that is not possible with the more simple ideal torques approach. When compared to individual muscle modelling, they reduce notably the amount of equations to be evaluated as you can see in figure 7: from 10 to just 3, one per joint. All of this curves have several parameters that need to be calibrated in order to personalize the model to a specific subject. This calibration is explained in section 4.3.3.

2.2.3 Central nervous system actuation

The central nervous system (CNS) consists of the brain and spinal cord, acting as the body's control center, processing sensory information and directing responses. It controls both voluntary activities, such as movement, and involuntary ones, including breathing and heartbeat. In neuromusculoskeletal (NMSK) simulations, the CNS has been modelled with various methods:

- **Muscle synergy**: this hypothesis [32] suggests that the CNS activates groups of muscles together rather than individually. Muscle synergies are represented by modules that include a neural command (NC), indicating the time activation of a set of muscles, and a vector that acts as a weighting factor, representing how activated a muscle within the group gets activated with a NC. Since the number of NCs is lower than the number of muscles, using muscle synergies reduces the complexity of simulations.
- **Stretch-reflex-based control**: This idea incorporates biologically-motivated stretch-reflex-based control rules to the NMSK model, activating muscles based on current kinematics, muscle states, and environmental conditions [33]. For example, Geyer and Herr [34] proposed a reflex-based control strategy for directing a 2D model with Hill-type muscles, and demonstrated its robustness in generating walking simulations that could handle ground irregularities and slope changes.
- **Trajectory optimization**: some predictive simulation studies have explored the optimality principles underlying human gait. Recent findings [11] suggest that an objective function combining squared terms of metabolic energy, muscle activations and joint accelerations can produce a human-like walking pattern. Due to its simplicity and ease of implementation, this is the approach taken in this project.

2.2.4 Contact models

In a multibody simulation it is crucial to define the way the multibody system interacts with the environment. In the problem at hand, this translates to how the feet and the crutches interact with the ground. There are many ways that this contact can be modelled, and in this project two will be explored. The contact models are very important because they can have a major role in the outcome and realism of the simulation, so in BIOMEC a lot of energy has been spent exploring them [12][35]. Trying to perfect a contact model could be a bachelor's thesis by itself, so that is why I have only developed the model until I had a working prototype, but without diving deep into the calibration of the parameters or the physical justification behind the contact model.

2.3 Optimal control prediction problems

Optimal control theory aims to find the best control input to drive a dynamic system from an initial state to a desired final state while optimizing a certain performance criterion. This section briefly explains the fundamental components and concepts involved in optimal control problems, because understanding them is crucial to comprehend the work done in this Bachelor's thesis.

2.3.1 General problem formulation

The general formulation of an optimal control problem is given by the following mathematical expression:

$$
\begin{aligned}
\text{Minimize} \quad & J = \int_{t_0}^{t_f} f(x(t), u(t), t)\, dt \\
\text{Subject to} \quad & \dot{x}(t) = g(x(t), u(t), t), \quad t \in [t_0, t_f] \\
& h_{\text{min}} \leq h(x(t), u(t), t) \leq h_{\text{max}}, \quad t \in [t_0, t_f] \\
& x_0^{\text{min}} \leq x(t_0) \leq x_0^{\text{max}} \\
& x_f^{\text{min}} \leq x(t_f) \leq x_f^{\text{max}} \\
& u_{\text{min}} \leq u(t) \leq u_{\text{max}}, \quad t \in [t_0, t_f]
\end{aligned}
\tag{2}
$$

This formulation has the following components:

- **States**: States represent the dynamic variables that evolve over time and describe the system's behavior. They can include physical quantities such as position, velocity, temperature, etc. In this project, they are variables like angle, angular velocity, angular acceleration or torque. In Eq. 2, they are denoted by $x(t)$, and x_0^{min}, x_0^{max}, x_f^{min} and x_f^{max} are the lower and upper bounds of the initial state $x(t_0)$ and final state $x(t_f)$, respectively.

- **Controls**: Controls are the inputs applied to the system to influence its behavior and drive it towards the desired state. Controls can be continuous or discrete variables and may represent forces, torques, voltages, etc. In this project, they are variables like the derivative of torque, ground reaction forces, MTG activations... They are represented by $u(t)$ in Eq. 2, and u_{min} and u_{max} are the lower and upper bounds.

- **System Dynamics**: The system dynamics describe how the states evolve over time in response to controls and external influences. They are typically represented by differential equations or difference equations that capture the system's behavior and constraints. In Eq. 2, the system dynamics are denoted by $\dot{x}(t) = g(x(t), u(t), t)$.

- **Objective Function**: The objective function, also known as cost function, quantifies the performance criteria that the optimal control problem aims to optimize. It is typically defined as a function of states, controls, sometimes time, and can include terms that penalize deviations from desired states, control effort, or other specific criteria. In this project, one example could be minimizing joint power, which is directly related to metabolic cost [36]. The objective function is represented by J in Eq. 2, and $f(x(t), u(t), t)$ is the integrand of the cost function, representing the instantaneous cost at each time step.

- **Path Constraints**: Path constraints are restrictions imposed on the states and controls, that are enforced throughout the system's evolution. In this project, it could be enforcing both crutches to move together.

- **Endpoint Constraints**: Endpoint constraints specify conditions that must be satisfied at the initial and/or final time instants. They ensure that the system reaches a desired state or trajectory at the end of the control process. Both path and endpoint constraints are represented by $h(x(t), u(t), t)$ in Eq. 2, with lower and upper bounds h_{min} and h_{max}.

2.3.2 Optimal control simulation output

When an optimal control simulation is run, it can finish in different ways. The most common outcomes are:

- **Optimal Solution Found**: The solver has found a solution that satisfies all the constraints and optimizes the objective function. This is the desired outcome.
- **Converged to an infeasible Solution**: The solver has found a solution that minimizes the objective function but does not satisfy all the constraints. This can happen if the problem is not well formulated or if the constraints are too restrictive, for example.
- **Maximum number of iterations exceeded**: The solver has reached the maximum number of iterations allowed without finding a solution. When this happens the evolution of the variables should be checked to see if more iterations are needed or the problem is unlikely to converge in the current formulation.
- **Solver error**: The solver has encountered an error and has stopped the simulation. The worst outcome, because it causes MATLAB to crash and lose the simulation data.
- **Restoration failed!**: The solver has tried to run restoration mode, which tries to get the current solution back to the feasible space, and failed. It can happen for a variety of reasons, and it has been a major problem during development as will be explained throughout the document.

Apart from the EXIT messages, while a simulation is running the following variables are given to explain the evolution of the optimization process. In Figure 8 you can see the solver output when an optimal solution is found.

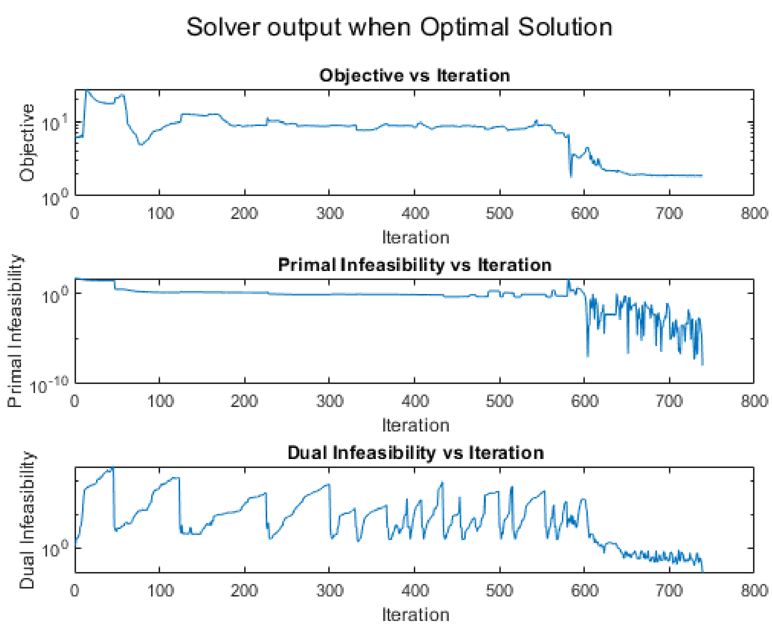


Figure 8: Example of solver output when an optimal solution is found.

It is important for the reader to know a little bit what they represent and what expected values should they have to understand later explanations throughout the document:

Objective: The final value of the objective function. The absolute value has no meaning, but a declining trend should be seen in the iteration log, as seen in Figure 8.

Primal infeasibility: Measures how much the current solution violates the problem constraints. Low values are desired.

Dual infeasibility: Dual infeasibility measures how far the current solution is from satisfying the optimality conditions with respect to the dual variables (Lagrange multipliers). The Lagrange multipliers are additional variables introduced in optimization problems that represent the sensitivity of the objective function to the different constraints. To put it simple, they measure how much the objective function would change with a slight change of a variable that is very near the constraint boundary.

In the Lagrange equation below (Eq. 3) you can see where they come from, where λ and μ are the multipliers, m and p are the number of equality and inequality constraints respective, and the rest of terms follow the nomenclature described above. Low values of Dual infeasibility are desired.

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x) \tag{3}$$

Complementarity measures how well a solution follows the **Karush-Kuhn-Tucker (KKT)** conditions and should be as low as possible. The KTT conditions define when a solution is considered optimal and are:

1. Stationarity: The gradient of the Lagrangian (Eq. 3) with respect to x must be zero at the optimum (nomenclature described above):

$$\nabla_x L(x, \lambda, \mu) = \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) + \sum_{j=1}^{p} \mu_j \nabla h_j(x) = 0 \tag{4}$$

2. Primal Feasibility: The solution x must satisfy the original constraints.

3. Dual Feasibility: The Lagrange multipliers associated with the inequality constraints must be non-negative.

4. Complementary Slackness: For each inequality constraint, if the constraint is not active (the value constrained is comfortably inside the bound) then the lagrange multiplier associated is zero. If the condition is active (the value constrained is exactly or near the boundary) the corresponding Lagrange multiplier is not zero, to reflect the importance it is playing in determining the solution.

Overall Non-Linear Programming (NLP) error: a summary measure that combines the primal and dual infeasibilities. It gives an overall indication of how close the current solution is to being feasible and optimal. Lower values are desired.

3 Concept engineering

The aim of this section is to explore the possible approaches to solve our problem: finding an optimal control problem formulation for MTG-driven three-point gait that produces realistic results without tracking experimental data. For that, first the initial setting from which this thesis is started will be defined, then chosen options for the simulation environment, which came predefined in the scope of the project, will be justified. Afterwards, the starting development objective, one simpler that the problem at hand, will be stated and finally a possible approach to build virtual patients will be discussed.

3.1 Initial problem formulation

The starting point for this thesis was a part of my director's PhD [6], which was a 3D torque-driven four-point gait simulation. While that formulation found an optimal solution, it was not very realistic and had room for improvement. In order for the reader to have a notion of what the initial setting was, in Table 1 you will find a summary of it, with all the constraints, variables, objective terms... Note that q is the joint position, $\dot{q}$ the joint velocity, τ the joint torque, $\ddot{q}$ the joint acceleration, $\dot{\tau}$ the derivative of the joint torque, and GRF the ground reaction forces. The detail of the path and endpoint constraints is explained in Detail engineering (section 5).

Table 1: Initial problem formulation

States	$q, \dot{q}, \tau$
Controls	$\ddot{q}, \dot{\tau}$, GRF
Path constraints	• $-0.1\,\text{N} < F_{\text{residuals}} < 0.1\,\text{N}$ • $\tau_{\text{ida}} \approx \tau$ • GRF_control $\approx$ GRF_contactModel • SlideVelocity ≈ 0 • Crutch_top $\approx$ forearm
Endpoint constraints	• LeftFoot(x) - RightFoot(x) $\approx L_{\text{stride}}/2$ • RightCrutch(x) $\approx$ LeftCrutch(x) • RightFoot(x) $\approx$ LeftFoot(x) • Crutch&Feet$(x_f - x_0) \approx$ LStride • All -10° $< q_f - q_0 <$ 10°
Dynamic constraints	$\frac{d}{dt}\begin{pmatrix} q & \dot{q} & \tau \end{pmatrix} = \begin{pmatrix} \dot{q} & \ddot{q} & \dot{\tau} \end{pmatrix}$

Table 2: Components of the objective function in the initial formulation

Cost function	
Term	**Mathematical expression**
Mechanical power	$\sum_{i=1}^{N}(\tau_{\text{sc}}^{i} \cdot v_{\text{sc}}^{i})^2$
Lumbar & shoulder torque	$\sum_{i=1}^{N}(\tau[\text{lumbar,shoulder}]^{i})^2$
Derivative of torque	$0.1 \cdot \sum_{i=1}^{N} \lVert\dot{\tau}^{i}\rVert^2$
Equalize crutch forces	$0.1 \cdot \sum_{i=1}^{N}(\text{CRT\_contr}_2^{i} - \text{CRT\_contr}_5^{i})^2$
Joint acceleration	$0.01 \cdot \sum_{i=1}^{N} \lVert\text{a}^{i}\rVert^2$

The problem was formulated as to make the solver's job as easy as possible in a problem of this complexity:

- The torques are stated as controls, but can also be calculated from the states via Inverse Dynamics. These two instances of the same variable are enforced to be as equal as possible to keep the solution physically realistic.

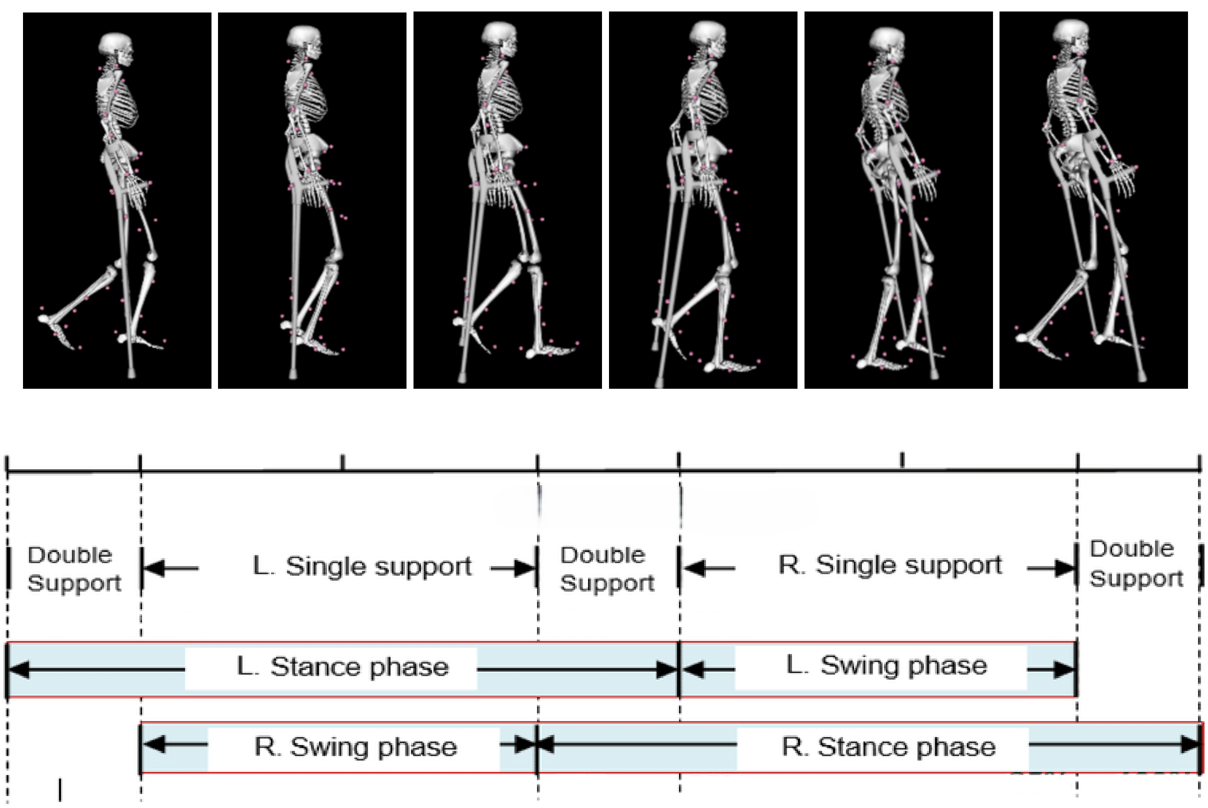


Figure 9: Three-point crutch gait cycle diagram. The diagram below shows the temporal sections of the gait cycle. The right leg is the healthy one, and the left leg is the injuried one.

- The GRF are stated as controls, but are also calculated from the states (x,v) using the contact models. The solver is then enforced to make these two instances of the same variable as equal as possible. This intermediate step seems to help the solver algorithm, because not including the GRF as controls and just hoping that the optimizer can capture the complex state $\rightarrow$ GRF relation has not worked well in different experiments during development.
- The gait pattern is enforced by setting the temporal sections of the feet and crutch GRF vectors corresponding to the swing phase to zero, incentivising the solver to follow the desired pattern. This swing-stance cycle for three-point gait can be seen in Figure 9, and in the 6 frames you can see how each step of the cycle looks like.

Additionally, to solve an optimization problem, you have to provide an initial guess of all the states and controls, for the solver to start iterating from somewhere. Ideally, it needs to be as close as possible to the desired solution. In this initial formulation, the data came straight from an experimental measure of healthy gait.

3.2 Simulation environment

3.2.1 Coding language

The initial code was written in MATLAB R2019a [37], which is usually the chosen option when solving numerical problems. Another option known for its flexibility is Python, which also can be used to solve optimal control problems. Let's explore briefly the pros and cons of each software option, and justify why MATLAB is the chosen option.

MATLAB

MATLAB is a high-performance programming language developed by MathWorks, widely used in academia, research, and industry, specially for numerical computing. Its pros and cons are summarized here:

+ **Integrated environment**: MATLAB provides a robust integrated development environment with extensive built-in functions for numerical calculations and plotting.
+ **Toolboxes**: MATLAB offers specialized toolboxes highly optimized for control applications.
+ **Performance**: Libraries are optimized for numerical computations, offering high performance.
- **Cost**: MATLAB is commercial software and can be expensive depending on the plan. For an individual academic license, the cost can go up to 262€ per year.
- **Less flexibility**: MATLAB can be less flexible compared to open-source alternatives when it comes to integrating with other programming environments and tools.

With these points in mind, it is clear why MATLAB was chosen in [6] to solve the optimal control problem, for the high performance, specialized add-ons and ease of use in the IDE. The difficulty integrating with other sofwares will need to be overcome, but luckily most optimal control softwares are designed in MATLAB or prepared to interact with it.

Possible alternative: Python

Python is a high-level, interpreted programming language known for its simplicity and readability. While its main use is not numerical simulations and optimal control problems, it can be used for that. Python's pros and cons are:

+ **Open source:** Python and its libraries are free and open-source, making it accessible to everyone.

+ **Flexibility and integration:** Python can easily integrate with other programming languages and software, offering great flexibility for complex workflows.

+ **Libraries:** Python has powerful libraries such as CasADi, GEKKO, SciPy, and Pyomo, which provide robust solutions for optimal control problems.

- **Performance:** Python can be slower than MATLAB for certain numerical computations due to its interpreted nature.

- **Steeper learning curve:** While Python itself is easy to learn, the learning curve for mastering optimal control libraries and integrating them into complex workflows can be steeper compared to MATLAB.

- **Fragmentation:** The variety of libraries and tools available in Python can lead to fragmentation, requiring users to make decisions about which libraries and frameworks to use and how to integrate them effectively.

3.2.2 Biomechanical modelling

The multibody model representing the human body, with all the parameters representing the bones and the joints, needs to be hosted in a specific program. In BIOMEC, the OpenSim software [38] is used to manage these multibody models, and this project is no exception. In this section I will explain the advantages of OpenSim and a possible alternative for a similar project.

OpenSim

OpenSim is a biomechanical modellin software that is free to use, developed by a team at Stanford. The software has some integrated functions that are very useful in musculoskeletal simulations. These functions are:

- **Inverse dynamics**: This function obtains the net joint torques from kinematic data (joint position, velocity, acceleration), ground reaction forces and the equations of motion [19].

- **Point kinematics**: This function receives the time, joint positions, and velocities of all joints as input. It performs a change in the reference frame to calculate the linear position and velocity of a point based on its local position within a body (for example, position [0.2 0 0] in the thigh body).

- **Motion loading**: After running the optimal control problem, the software is able to show the motion in a nice visual way from a file containing the joint trajectories and time, helping the evaluation of results in a more anatomical way rather than just looking at the evolution of each variable in linear plots.

- **Scaling**: OpenSim provides a scaling tool to change the size of a model to match a specific subject. The scaling can be done very simply via a scaling parameter or more precisely using a set of markers in specific body locations, that tell OpenSim how much to scale each body segment.

It also integrates very well with MATLAB and the optimal control problem solver via MEX files, so it is an optimal choice for this project.

Possible alternative: AnyBody Modelling System

AnyBody Modeling System [39] is a human body simulation software for Windows that allows advanced biomechanical modeling. Its features are more advanced than OpenSim, allowing for muscle forces calculations, joint contact interaction, integrated metabolism estimates and antagonistic muscle interactions. However, it is a paid software, it is not open-source, and can be more difficult to integrate with other softwares than OpenSim.

3.2.3 Optimal control problem solver

In this thesis, GPOPS-II has been used because it is the option that BIOMEC usually works with and the initial code was designed to fit with its structure. However, it is not the only software available for solving OCPs, so in this section I will explain why GPOPS-II is the chosen option and briefly introduce one possible alternative.

GPOPS-II

GPOPS-II stands for *General Pseudospectral Optimization Software*, it was developed by the University of Florida [40] and requires a paid license to operate. Its main features are:

1. **Pseudospectral methods**: GPOPS-II employs pseudospectral methods, which are numerical techniques for solving OCPs. These methods discretize the time domain into a set of nodes (or points) and approximate the system dynamics and control inputs over these nodes. Pseudospectral methods are known for their accuracy and efficiency compared to other more traditional discretization methods like direct collocation [40].

2. **Wide variety of problem formulations**: GPOPS-II can handle a wide range of optimal control problems, including both continuous and discrete-time systems, single-phase and multi-phase problems, as well as problems with path constraints, boundary constraints, and mixed integer components [40].

3. **Flexible interface**: It provides a flexible interface for users to define their optimal control problems. Users can specify the system dynamics, objective functions, constraints, and other problem parameters using high-level mathematical expressions.

4. **Wide application range**: GPOPS-II finds applications in various fields such as aerospace engineering (trajectory optimization, spacecraft guidance), robotics (motion planning), biomedical engineering (optimal control of drug administration, MSK simulations), and many others [40].

GPOPS-II incorporates efficient solvers capable of handling large-scale OCPs, even if they are highly nonlinear and complex. The software can use the Interior Point OPTimizer (IPOPT), suitable for large sparse problems like the one at hand; and Sparse Nonlinear OPTimizer (SNOPT), suited for problems with smooth nonlinear functions in the objective and constraints [40]. While both are a nice fit for this project, SNOPT requires another paid license on top of GPOPS-II, so it was decided to use IPOPT.

Possible alternative: Algorithmic differentiation

When solving a trajectory optimization problem, evaluating derivatives is a key and time consuming step in the simulations [41]. GPOPS uses Finite Diferences (FD), which works but is quite time consuming when compared to Algorithmic Differentiation (AD), which was proven [41] to be between 1.8 and 17.8 times faster. However, it has the limitation that the existing software that easily integrates with GPOPS, ADiGator [42], is not compatible with the biomechanical model host OpenSim. Another option is CasADi [43], which has been demostrated to integrate with OpenSim but would require to change the optimal control solver, which is a very time-consuming step. For a future project, this migration to algorithmic differentiation should be considered.

3.3 Initial model complexity

To tackle the development of a simulation framework of this complexity, it is a good idea to start as simple as possible. An approach commonly used in biomechanical simulations is to remove degrees of freedom (DOFs), from the 3D model in the initial formulation down to 2D. A 2D model is defined by only allowing movement in the sagittal plane, not by flattening a human in 2D as one might funnily imagine. This means that the ground reaction forces are still 3D, but the ones acting in directions other than the sagittal plane have to balance out.

To be more specific, a 2D crutch gait model has: **3 residual joints**, which connect the model to the ground's reference frame and allow the model to move and rotate with respect to it; **4 joints per leg**: hip flexion, knee flexion, ankle flexion and toe flexion; **lumbar flexion**; and **3 joints per arm**: shoulder flexion, elbow flexion and wrist flexion. When compared with the 3D model it has the following pros and cons:

+ **Simplicity**: Since most of the movement happens in 2D, the model might be able to capture the motions and joint torques involved.

+ **Less computational cost**: A 2D model has 18 DOF and a 3D one has 37. This reduction in DOFs makes computation significantly faster, which is good for development when you need to perfom many simulations.

+ **Easier convergence**: Since in 2D the movement is confined to the sagittal plane, there are less spatial constraints to enforce, which might ease convergence.

- **Model positioning**: Since the 2D model has all joints involving movement out of the sagittal plane locked in place (shoulder, hip, ankle, lumbar), they must be carefully positioned manually to allow the necessary motion and not cause any infeasibilities.

To further simplify development, even if the final objective is to develop a model for three-point gait it was decided to start with a swing-through gait, which guided by the experience of my colleagues in the lab, it was thought to be easier to develop.

3.4 Virtual experiments

As stated in the objective of this thesis, the secondary goal was to execute a series of virtual experiments and compare the ability of the simulation framework to handle them, and to explore how the output joint loadings and body positions changed in the three-point gait in function of the simulated patient condition. This gait pattern is common in patients with assymetrical injuries [44], so this needs to be accounted when designing virtual patients.

In an ideal clincal setting, a patient would come into the clinic and the parameters defining how ”strong” the digital twin is would be calibrated. The idea was to tweak the necessary parameters in a pattern mimicking real SCI or stroke patients' functionality. The parameters defining how ”strong” the OpenSim multibody model is are:

- The **maximum isometric torque** of each muscle group, which comes from experimental data when calibrating the MTG functions. To weaken the model then, this maximum torques need to be reduced to decrease the amount of force the model can produce with that joint. Since MTGs are separated in flexor and extensor groups for each joint, this weakening can be very precise and customizable to each patient.

  To determine the reduction amount of these parameters, maximum isometric tests could be done and the results inserted into the MTG functions. Since the force-angle and force-velocity curves are normalized and they are intrinsic of the anatomy of the joint, they would not need to be calculated again, but further research should be done to support this simplification.

- The **weight bearing percentage (PWB)** for each foot and crutch. This parameter is key in the simulation environment, and could be easily measured with a force plate by telling the patient to support as much weight as he/she feels comfortable. It is a major director on how the simulation will look like, because varying the amount of weight that goes to the foot or to the crutches changes the joint loadings and model positioning throughout the motion.

3.5 Proposed roadmap

After analysing the options that are available to tackle a problem of this magnitude, a starting point can be defined for model development. The decided route is to attempt to start developing a simulation framework with less DOF, the swing-through gait pattern and ideal torque actuation. Later, depending on how experimentation and model development advance, progressive steps will be made towards the a three-point gait pattern and towards adding the MTG functionality, all while exploring different variations in the formulation to achieve more realistic results.

The simulation environment chosen to solve the optimal control problem, keeping into account that some of them came determined by the previous development in [6], is MATLAB, with the GPOPS-II solver using IPOPT as the nonlinear programming solver and OpenSim managing the multibody model.

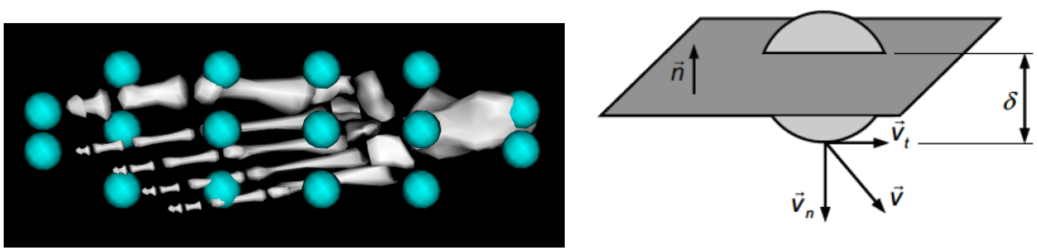
4 Model and Optimal Control Problem development

In this section, the development of the simulation framework is explained. First the contact models used in the simulation are explained, then the scaling approach and its importance is shown and finally the different experiments in the search for an optimal solution to the problem are explained, detailing the different approaches that have been tried. An emphasis is put in the varying degrees of complexity of the problem as it increases from a 2D torque driven model to a 3D MTG actuated model, and also in the different problems that have arisen in each step.

4.1 Contact models

4.1.1 Foot ground contact model (FGCM)

The code from which my development started had a FGCM implemented, one that was studied, implemented and calibrated by David Civantos in 2020 [45], a past student, and is used in many simulation projects in the lab. Since it does not produce errors or unrealistic behaviors, it was left with the parameters that came from his previous work. Below is a brief explanation of the contact model.



(a) (b)

Figure 10: Visualization of the FGCM, in (a) the location of the 16 spring-like spheres is shown and in (b) you can see an example of spherical indentation. Taken from [45].

This FGCM consists in 16 springs placed all around the foot sole, as you can see in Figure 10 (a). Each one of them produces a vertical force F_N, depending on how deep and fast into the ground it goes, as you can see exemplified by a sphere in Figure 10 (b). The tangential force F_T is generated depending on the vertical force produced, a friction parameter and the hyperbolic tangent of a ratio that depends on the linear velocity of that spring relative to the floor.

$$F_N = k\delta(1 + c\dot{\delta}) \tag{5}$$

$$F_T = F_N \mu \tanh\left(\frac{v_{\text{slip}}}{v_{\text{thresh}}}\right) \tag{6}$$

The equations modelling this contact can be seen in Eq. 5 and 6, where δ is the distance each spring sinks into the floor, v_{slip} is the slip velocity and v_{thresh} is an arbitrary parameter that, in combination to the $tanh$, it controls how the tangential velocity responds to changes in slip velocity. k, c and μ are the classic stiffness, damping and friction parameters. Finally, to simplify calculations for the inverse dynamics analysis, all 16 forces are transported to a specific point of the foot, compacting them in a vector of 3 forces and 3 moments: $[F_x, F_y, F_z, M_x, M_y, M_z]$.

4.1.2 Crutch-ground contact model (CGCM)

The CGCM that came in the original code used a similar approach than the FGCM. However, during the first trials in development it was causing a lot of problems: the crutches produced force while sliding on the ground (unrealistic), difficulting convergence. This motivated me to design a different approach, more simple and intuitive, that worked good enough for the purposes of the crutch simulation. In the following lines both the discarded option and the final version for the CGCM will be covered.

Sphere - Flat plane contact model

The contact that came in the code was a sphere-flat contact model, where the vertical force was calculated from how deep and fast into the ground the crutch is going (see Eq. 7) and the tangential force was calculated in exactly the same way as the FGCM (Eq. 6). It is quite similar to the FGCM, with the difference that there was only one ”spring”, so no moments were applied, and the equation calculating the vertical force was slightly different.

$$F_{ax} = k|\delta^{3/2}| + c|\delta^{3/2}|\dot{\delta} \tag{7}$$

Similar as before, this model calculates the tangential force from the dynamic friction, and while this works in the feet, in the crutches it caused massive problems. This could be caused by the difference in contact surface: while the foot can rock from heel to toes, the crutch either touches the ground or not.

Static friction axial contact model (new version)

This model makes two assumptions: that all force is produced from static friction, which means that the dynamic friction limit is higher than the force that a person can produce leaning on it on regular ground; and that the forced produced by the crutch is in the axial direction of the crutch, not vertically.

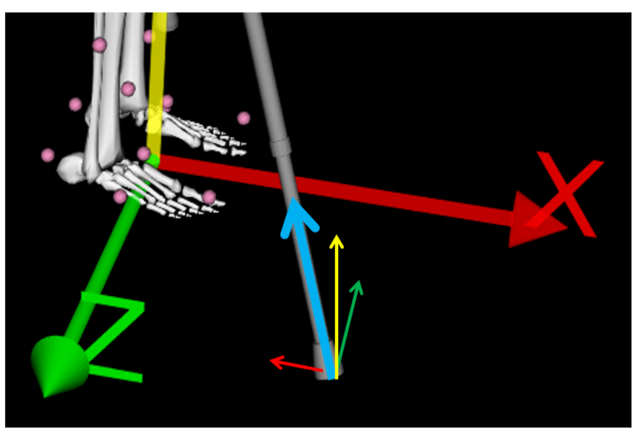


Figure 11: Crutch-ground contact model. The axial force (blue) is calculated from the vertical displacement of the crutch tip, and then projected onto the global axes to get the individual components (in red, yellow and green).

To get the axis decomposition of the forces, it first calculates the modulus of the axial force with Eq.7 and then, using the global positions of the tip and another point 40 cm higher in the crutch frame, it gets the direction. With these 2 components, MATLAB projects the vector on each axis to get the decomposition (see figure 11) using Eq. 8, where $\vec{c}$ is the vector from the tip to the higher point in the crutch frame, and $c_{\vec{norm}}$ is the modulus of $\vec{c}$.

$$F_{x,y,z} = F_{\text{axial}} \cdot \frac{\vec{c}_{x,y,z}}{c_{\vec{norm}}} \tag{8}$$

The parameters were manually calibrated up to a point where the crutch tip did not slide nor sink too much into the ground in OpenSim, and further refined during model development. The final parameters are $k = 45000$ and $c = 50000$.

4.2 Scaling

In optimal control problems, scaling the states, controls, constraints and objective function is key, because the solver performs best when all variables have a similar order of magnitude. If not, it is difficult for it to decide which constraints are more important that others, and magnitude can mask the relative importance of the variables in the objective function. Also, it can lead to numerical instabilities and errors. In GPOPS-II, scaling can be done:

- **Automatic-bounds**: scales the problem based on the bounds defined by the user. It is more robust that other options, but if the bounds are very permissive it can scale the variables to very small values [40].
- **Automatic-guess**: scales the problem based on the initial guess provided by the user. Its quality depends vastly on the quality of the initial guess [40].
- **Manual scaling**: GPOPS does not scale the data and the task is left to the user. This approach allows more flexibility for complicated problems in which the general methods provided by the software do not work properly. For the *dynamic constraints* of the problem to be accurate, the scalings of the different variables need to be related. For example, the scaling factor for velocity needs to be related to displacement and time scaling factors.

In the initial problem formulation described in section 3.1, manual scaling was implemented. At the beggining of development some attempts were made to change it to the options defined above and to tweak the manual scaling but they were unsuccessful, so scaling was left unchanged until the last stages of development, as explained later in this section.

4.3 Road to convergence

In this section a summary of the development steps and experiments is presented. The full detail of my work, comprising more than 100 simulations, is explained in detail in separate logs, which I have made available to the lab for further development or to get ideas for their own projects. These logs have not been included as an Annex due to their extensive length and the mixture of Catalan and English. If despite that the reader wishes to see them, please ask for them at `bfite2020@gmail.com`.

4.3.1 Ideal torques - 2D

As stated in Concept engineering (Section 3), it was decided to start as simple as possible. Building on the initial formulation, the first objective was to build a 2D ideal torque driven swing-through gait simulation

Swing through gait pattern

At the start of development, the gait pattern imposition was changed so the feet swinged together, effectively acting as one foot in the normal gait cycle; and the crutches did the role of the other foot. It was also imposed that the injuried foot could not produce any force. With this adaptations, the initial simulations were not converging at all. It was producing very unrealistic results in OpenSim, and the solver yielded high constraint violation and dual infeasibility, suggesting that the constraints, the initial guess or the objective function were not properly defined. To fix it, I tried multiple approaches:

- **Bound adjustment**: The variable bounds were taken from an experimental normal gait measurement. Since the displacements, velocities... are different in crutch gait than normal gait, they were adjusted to more realistic values. For the lower body, the bounds were taken from the experimental healthy gait measurement, adding a tolerance margin to account for possible differences; and for the upper body the bounds were increased widely to make the model ”as strong as it needs”, and then they were tightened progressively.
- **Refactor path constraints**: review the path constraints and refactor constraint enforcing slide velocity to be 0 to only account for the antero-posterior direction. Also include a term to make the crutches symmetrically in the sagittal plane.
- **Refactor endpoint constraints**: review the endpoint constraints, and attempt to enforce periodicity to the movement.
- **Change initial guess**: try to use a previous simulation as initial guess instead of experimental data from normal gait.
- **Increase parameter tolerance**: in the formulation the stride length and the speed are treated as hyperparameters which are decided by the user. Attempts over a range of speeds and stride lengths were done to ensure that this was not a limitation. This approach is repeated often throughout the development of the simulation framework.

Table 3: Components of the Objective function added or modified (in green) or removed (in red).

Modification	Mathematical Expression
Add linear slide velocity	$\sum_{i=1}^{N} (dx(\text{feet})/dt)^2$
Add local angular momentum	$0.001 \cdot \sum_{i=1}^{N} \|\|\text{normGKv2\_sc}^i\|\|^2$
Remove lumbar & shoulder torque	$\sum_{i=1}^{N}(\tau[\text{lumbar,shoulder}]^i)^2$
Add healthy knee and ankle to avoid hopping on one leg	$\sum_{i=1}^{N}(\tau[\text{knee,ankle}]^i)^2$
Allow crutch forces to be different	$0.1 \cdot \sum_{i=1}^{N}(\text{CRT\_contr}_2^i - \text{CRT\_contr}_5^i)^2$
Increase weight of acceleration	$\cancel{0.01} \cdot \sum_{i=1}^{N} \|\|\text{a}^i\|\|^2$

- **Change the objective function**: progressively add and remove various terms from the cost function. They are summarized in Table 3. Out of those, only the removal of lumbar and shoulder torque and the addition of knee and ankle torques where kept, the rest were reverted to the original state.

At this point, even if some of this changes achieved little improvements, it was clear that there was a problem with the contact model, because the forces looked very unrealistic and the model was twitching, which means that it was moving the crutch tip when it was ”inside” the ground, which should not happen. To improve it, the contact model explained in Section 4.1.2 was implemented. The results were much better, but still not converging to an Optimal Solution. To improve it, the following approaches were attempted:

- **Lock the wrist joints**: In the model configuration, the wrist and crutch act as a multiplier of movement, meaning that if the wrist moves a few degrees, the crutch tip moves a significant distance. Since in crutch gait it can be considered locked in place (it is undergoing a large force transfer from the crutch) the wrist joints were welded in the multibody model, effectively removing that degree of freedom.

- **Model arm positioning**: as mentioned before, since it is a 2D approximation the shoulder rotation and adduction of the multibody model had to be manually chosen. The final placement was chosen to be as close as possible to the real arm placement in crutch gait and allow full range of motion in the sagittal plane.

- **Remove periodicity constraints**: Since the simulations are for a full gait cycle, the movement should be periodic. However, the constraints enforcing periodicity were the only ones blocking convergence at some point, so they were removed.

All this changes were implemented in about three weeks, and they lead to an optimal solution. However, an error in the bound definitions was found in further stages of development, so the results were rendered useless, and convergence was not reached with the correct bound. Nevertheless, in Section 6.1 you can see the final result for swing-through, which provides interesting insights.

This mistake remarked that even being a simplified version of crutch gait, the problem is very complex. Since the final objective was to reach the three-point gait pattern and the swing-through pattern was just part of the exploration, further development was stopped and focus shifted to the three point gait pattern.

Three point gait pattern

Three-point and swing-through gait are similar in the sense that they are both assymetric patterns, with the difference that swing-through does not support any weight with the injuried foot and three-point gait does. To account for this, the crutches were attached to the injuried foot so they moved together in the sagittal plane, and the gait pattern imposition was adapted so that the healthy foot swinged in half of the gait cycle and the injured foot + the crutches swinged in the other half (see Figure 9 in Section 3.1).

Similar approaches as the ones described in the swing-through gait pattern were attempted to make the problem converge, to no success. This efforts are not worth documenting in the project memory, but can be found in the logs. At this point, it was decided that the issue might be not in the formulation, but at the dimensionality of the problem, so it was decided to change the formulation from 2D to a 2.5D three point gait pattern: 3D for the arms and torso and 2D for the legs.

The main reason for this change was the difficulty in finding the exact arm placement that produced a plausible crutch gait, specially because when walking on crutches the arm rotates and adducts apart from swinging in the sagittal plane. The change to 3D was not made directly because preliminary trials in 3D were not promising and it took significantly longer to run simulations.

4.3.2 Ideal torques - 2.5D

To increase dimensionality, lumbar rotation and bending, shoulder adduction and rotation, and forearm rotation were added to the model. The code was also adapted to manage this change in dimensionality, and bounds were added to the new joints associated angular velocity and acceleration guided by experimental healthy gait data, plus a tolerance to account for the differences between that and crutch gait.

Already on the first trials, the 2.5D formulation visually improved the results, both in the plots and the Opensim visualization (details later in Section 6.2). This further confimed the hypothesis that crutch gait is not simplifiable to 2D. Even if the results were good, an optimal solution had not yet been found, so some more attemps to fix it were done, the most rellevant being:

- **Reduce crutch length**: The new DOFs allowed the model to position the arms in a way that minimizes effort, and the simulated behaviour suggested that the crutches needed to be shortened. It provided good results so they were kept 3 cm shorter (3.8% less) than the initial model.
- **Harden the ground**: With the increase in DOF the arm movement changed and the stiffness of the crutch-ground contact had to be increased to avoid the crutches sinking too deep into the ground (sinking translates to more force as defined by the contact model in Section 4.1.2).
- **Include jerk in the problem formulation**: it was decided to include the derivative of acceleration (jerk) and treat acceleration as a state to mimic other simulations being developed in the lab and see if it helped the solver converge to a solution. This change increased the average time per iteration from $1.6s$ to $2.9s$, which is an 81% computational cost increase, but since it made the states trajectories and velocities more smooth, it was kept.

After this point the issues and development decisions were done in parallel with the MTG formulation, but since the focus of the project was the latter they are explained there (Section 4.3.4).

4.3.3 MTG calibration

One of the objectives of this project was to introduce Muscle Torque Generators (MTGs) to the crutch gait prediction problem and see if the simulations became more similar to real-life crutch gait. MTGs are implemented as a series of functions that take in the joint angles and velocities and return the torque for every joint, and in an ideal clinical scenario this functions would be calibrated to a specific patient, to accurately represent his functional situation.

To make development as close to reality as possible, the MTG functions have been calibrated [1] with experimental data from the University of Waterloo in Canada. This data was taken by the director of this thesis from a young healthy male performing a series of isometric, isokinetic and passive tests with a BIODEX machine (see Figure 12), and from almost all joints in the body. From this data, another student in the lab has fitted

[1] *For the calibration of the MTG parameters and functions I have used part of Carlos Pagès' work, a research assistant working in the lab at the same time as me on his Master's thesis. It is not possible to cite his work because it has not been published at the moment of writing this thesis, but it has been crucial to this thesis. Thank you Carlos!*

diverse curves to the isometric and isokinetic measurements, to get the f_{FA} and f_{FV} equations for each MTG pair (see Section 2.2.2).

Figure 12: BIODEX machine with two possible motions exemplified. Data extracted from the BIODEX website.

For the passive curves, it was decided to use data from *Yamaguchi et al, 2001* [46], that described the passive curves and provided parameters for the lower body, and from *Brown et al, 2018* [47], who following the equations from *Yamaguchi et al, 2001* provided parameters for the upper body passive curves.

4.3.4 MTG - 2.5D

Once a working version of the MTGs was developed, it was implemented into the crutch gait simulation framework and the initial trials, while not converging to an optimal solution, were very promising. At this stage of development, the biggest problems standing between the simulation framework and convergence were shared with the ideal torque formulation, and they were starting to get related to the solver itself rather than the violation of a specific constraint or visual issues in OpenSim. These issues were:

- **Simulations running in restoration mode**: this mode is called when the iterations have strayed to far from the feasible region and the solver attempts to return to it. The simulations ran for hours in this mode without converging anywhere, even if the simulation outcomes were visually and graphically acceptable, and the initial guess provided was feasible.
- **MATLAB Crash error**: This error occurred far too often, after the problem had run for a while in *restoration mode*. It most likely meant that the problem formulation had an issue that, in combination with how the solver is programmed, caused MATLAB to crash. The exact reasons behind this are very hard to discover, because there is little documentation and the code behind GPOPS-II is quite complex and unaccessible.

Solver settings changes

To improve this situation, the solver settings were changed:

- **NLP linear solver:** `'mumps'` $\rightarrow$ `'ma57'`.

  The linear solver is a key component of IPOPT, in charge of solving linear systems of equations that arise during the optimization process. `MUMPS` (MUltifrontal Massively Parallel sparse direct Solver) is a solver that is particularly effective for large sparse systems [48], and `ma57` is a sparse symmetric solver known for its efficiency in solving large-scale optimization problems [49].

- **Derivative dependency calculation:** `'sparse'` $\rightarrow$ `'sparseNaN'`.

  The derivative dependency calculation refers to determining how the states, controls and their derivatives depend on each other. Since this dependency is often sparse, meaning it has many zeros, this parameter controls the treatment of this sparse dependency matrix [50].

- **Radau Pseudospectral Method:** `'RPM-Integration'` $\rightarrow$ `'RPM-Differentiation'`.

  Change the method to discretize the continuous optimal control problem from integration to differentiation.

From the first trials it was clear that there was an improvement: the solver was no longer entering *restoration mode* and MATLAB stopped crashing. However, a new problem arose: instead of an optimal solution, the solver was exiting the optimization process and giving **EXIT: Restoration Failed!**. This exit message meant that the solver attempted to call the restoration mode and failed.

Avoiding "Restoration Failed"

It is quite hard to diagnose exactly why **EXIT: Restoration Failed!** happens, because the output of the solver is quite limited, but in an attempt to fix it, the following approaches were tried:

- **Refine the mesh collocation points**: The mesh collocation points can sometimes help a problem converge. In this case, an increase from 5 $\rightarrow$ 10 was attempted, but it did not yield any benefit and it significantly increased computational cost, from 2.9 to 4 seconds per iteration (38% increase).

- **Improve initial guess**: further process the experimental data to have an initial guess that is closer to an optimal solution, and also reattempt to use different previous simulations as an initial guess.

- **Gait pattern imposition changes**: As explained before in Section 3.1, to impose the gait pattern an interval of the GRF of each foot is set to zero. This creates an interval of variables that, since they are forced to zero, can take any value within the bounds of the problem, and constraints will not apply.

  This unused variables could be causing problems, because any time the solver changes them it produces no effect in the cost function nor in the constraint evaluation. To see if this was the cause, two approaches were tried: forcing this unused variables to zero, and removing the GRF from the controls altogether. None of this options yielded any benefit so they were reverted.

- **Remove jerk from problem formulation**: To reduce problem complexity and ease the solver's job, acceleration was moved to controls and jerk was removed. No convergence and visible degradation of the quality of the simulation and the plots.

- **Add residuals to cost function**: As explained before, residual forces represent the forces applied on the joints that connect the model to the ground reference frame, which allow the model to move but should not have any forces associated. They behave erratically at the endpoint of the simulation, so an attempt to reduce them was made to see if they were the cause of the non-convergence. It also did not achieve convergence but since it managed to reduce the residual forces it was left in the formulation.

None of these attempts managed to avoid the **EXIT: Restoration Failed!** situation. Some advice was given to me that improper scaling could cause the solver to output this message and stop searching for an optimal solution, so scaling was explored in depth to see if it was the source of the problem.

Scaling exploration

Up to this point in the framework development, manual scaling had been used, as explained in earlier in Section 4.2. In order to determine if scaling could fix non-convergence, the two internal GPOPS scalings previously discarded, **automatic-bounds** and **automatic-guess**, were tried. Since both approaches produced similar results, automatic-bounds was chosen because it is more robust.

The internal scaling successfully avoided the *Restoration Failed!* error, but it failed to make the problem converge, instead making it run until the iteration limit (up to 8000 iterations). Of the simulation logs, the dual infeasibility was quite high, and it did not show a declining trend as the iterations increase. This could mean that the solver had issues deciding the importance of the constraints relative to the objective function, maybe because their magnitude was too different and this was causing numerical infeasibilities. This pointed at more scaling problems.

Since neither manual nor automatic scalings showed better performance, it was decided to try and refine both scalings with additional modifications of the *objective function* and the *path/endpoint constraints* to keep them all in a comparable range, keeping in mind that for the *dynamic constraints* of the problem to be accurate, the scalings of the different variables need to be related (displacement, velocity, time...). To do so, personalized scaling factors were added to each constraint and each minimization term in an attempt to bring all of them between 10-0.1. None of this approaches managed to show an improvement or to make the problem converge.

Conclusions

At this point in development, the time allocated for the Bachelor's thesis ran out, leaving me unable to find a formulation that made the problem converge. In the next section the simulation framework is explained in detail, the achieved results are shown and, most importantly, some hypotheses as to why convergence is not achieved are presented.

5 Detail engineering

In this section, the detail of the final problem formulation achieved is presented. Firstly, the MSK simulation starter pack, a repository for new students and researchers in the BIOMEC lab is introduced, as well as my contribution to it. Later, the detailed problem formulation is explained and finally, the OpenSim multibody model final designs are briefly presented, as well as an overview of the functionalities of the simulation framework.

5.1 MSK simulation starter pack

There is an internal repository in the BIOMEC lab that explains the fundamentals of musculoskeletal simulations and shows simple examples of the code structure that is used for the problem formulation in GPOPS. It is used by new students and researchers to adapt to this challenging code. As part of my contribution to the lab, I have expanded it for the next students, including a detailed step-by-step explanation of how the optimal control formulation in GPOPS structurally works, and some modifications to the simple pendulum that is included in the starter pack.

In figure 13 (left) you can see the MTG actuated pendulum. The muscles are just for visualization purposes. Since it is very simple, it is designed to show how the MTG functions work. In this case, notice how as the pendulum angle changes, the length of each muscle (implicitly modelled by the MTG) changes. Notice also how the muscle in the right acts as a flexor and the muscle in the left as an extensor, behaviour that will be captured by the force-velocity relationship.

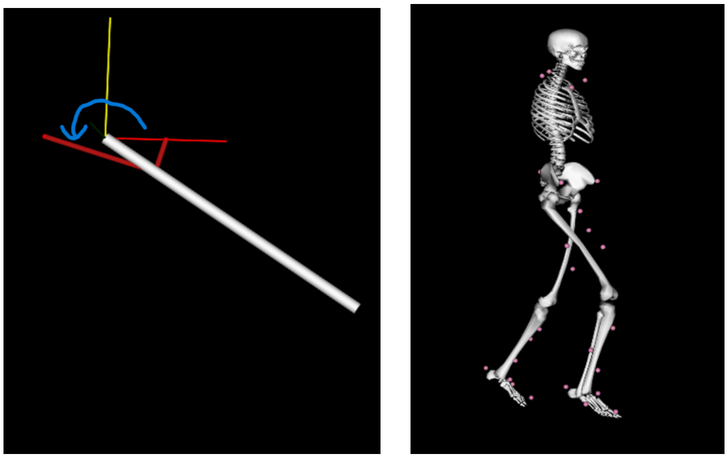


Figure 13: Simulation starter pack components. Left: Actuated pendulum with MTG functions, the muscles are there for visualization purposes only. Right: 2D HAT model used for the starter pack.

This example is useful to explore what each parameter represents and to allow the learner to play and modify the code, for example setting one MTG way stronger than its antagonist, tweak the passive resistances, make the force-velocity curves more or less abrupt...

In figure 13 (right) you can see the 2D HAT model that is also included in the starter pack. This model is used to show a more complex optimization problem and to train the eye to predict the optimization results. It is a good starting point to understand the problem formulation and the code structure that is used later for the crutch-assisted gait problem.

5.2 Problem formulation

5.2.1 Detailed problem formulation

In table 4 you can see in detail the problem formulation, specifying the states, controls and diverse path, endpoint and dynamic constraints that are enforced on the solver. Note that q is the joint position, $\dot{q}$ the joint velocity, $\ddot{q}$ the joint acceleration, $\dddot{q}$ the joint jerk (derivative of acceleration), act are the MTG muscle group activations, and GRF the ground reaction forces.

Table 4: Problem formulation for final MTG-driven simulation framework.

States	$q, \dot{q}, \ddot{q}$
Controls	$\dddot{q}, act,$ GRF
Path constraints	• $-5\,\text{N} < F_{\text{residuals}} < 5\,\text{N}$ • $\tau_{\text{ida}} \approx \tau_{\text{MTG}}$ • GRF_control $\approx$ GRF_contactModel • SlideVelocity ≈ 0 • Crutches(x) $\approx$ InjuredFoot(x) • $0.2\,\text{m} <$ Crutches(z) $< 0.4\,\text{m}$ • RightCrutch(x,y) $\approx$ LeftCrutch(x,y)
Endpoint constraints	• LeftFoot(x) - RightFoot(x) $\approx L_{\text{stride}}/2$ • RightCrutch(x,y) $\approx$ LeftCrutch(x,y) • Crutches(x) $\approx$ InjuredFoot(x) • Crutch&Feet$(x_f - x_0) \approx$ LStride • All -10° $< q_f - q_0 <$ 10°
Dynamic constraints	$\frac{d}{dt}\begin{pmatrix} q & \dot{q} & \ddot{q} \end{pmatrix} = \begin{pmatrix} \dot{q} & \ddot{q} & \dddot{q} \end{pmatrix}$

Path constraints explanation:

- $-5\,\text{N} < F_{\text{residuals}} < 5\,\text{N}$: Residuals are forces that are applied at the joint that acts as the reference between the model and the global frame. They ideally should be zero, so this constraint keeps them within a defined small range to ease the job of the solver.

- $\tau_{\text{ida}} \approx \tau_{\text{MTG}}$: Ensure joint torque from inverse dynamics matches joint torque from the MTG functions (necessary to ensure following physical laws).

- GRF-control $\approx$ GRF-contactModel: Ensure the ground reaction force (GRF) calculated from the contact model matches the desired GRF coming from the controls.
- SlideVelocity ≈ 0: Avoid feet or crutches sliding during stance phase.
- Crutches(x) $\approx$ InjuredFoot(x): Ensure crutches advance in sync with the injured foot.
- $0.2\,$m $<$ Crutches(z) $< 0.4\,$m: Prevent the crutches from colliding with the foot or leg or opening too far out.
- RightCrutch(x,y) $\approx$ LeftCrutch(x,y): Maintain the crutches at a similar position in the x-y plane (within a tolerance).

In table 5 you can see the details of the final version of the objective function. The weights of each term have been adjusted to make the solver reach a solution that is physically realistic and that resembles the desired gait pattern.

Table 5: Components of the objective function. N is the number of joints, and R is the number of residuals.

Cost function		
Term	**Mathematical expression**	**Justification**
Mechanical power	$\sum_{i=1}^{N}(\tau^i_{\text{sc}} \cdot v^i_{\text{sc}})^2$	Minimizes the effort
Local angular momentum	$0.001 \cdot \sum_{i=1}^{N} \|\|\text{normGKv2\_sc}^i\|\|^2$	Ensures smooth joint movement
Minimize residuals	$\sum_{i=1}^{R}(f^i_{\text{residu\_sc}})^2$	Reduces external fictional forces
Positive muscle act	$0.01 \cdot \sum_{i=1}^{N} \|\|\text{act-pos}^i\|\|^2$	Encourages efficient muscle use
Negative muscle act	$0.01 \cdot \sum_{i=1}^{N} \|\|\text{act-neg}^i\|\|^2$	Encourages efficient muscle use
Equalize crutch forces	$0.01 \cdot (CRT_{y,l} - CRT_{y,r})^2$	Ensures even crutch load distribution
Joint jerk	$0.001 \cdot \sum_{i=1}^{N} \|\|\text{j}^i\|\|^2$	Reduces abrupt joint movements, minimizing effort
GRF periodicity	$GRF_f - GRF_0$	Ensures ground reaction force periodicity

Endpoint constraints explanation:

- LeftFoot(x) - RightFoot(x) $\approx L_{\text{stride}}/2$: The starting separation of the feet is approx. half of the stride length in the x-axis.
- RightCrutch(x,y) $\approx$ LeftCrutch(x,y): The starting position of the crutches is similar in the x-y plane.

- Crutches(x) $\approx$ InjuredFoot(x): The starting position of the crutches is similar to the injured foot in the x-axis.
- Crutch&Feet($x_f - x_0$) $\approx$ L-Stride: Both the feet and the crutches advance *LStride* during the gait cycle.
- All -10° $< q_f - q_0 <$ 10°: All joints end position is within 10 degrees of the starting position, to help with periodicity.

5.3 OpenSim model final design

The OpenSim model that was provided to me initially came from Rajagopal et al [51], and was scaled and further adapted by Dr. Febrer-Nafria in [6]. After many design modifications, there is a final version both in 2.5D and 3D for:

- A 1.60 woman, corresponding to the scaled model from [6].
- A 1.85 male corresponding to the subject on which the MTGs were calibrated.

It is worth noticing that the crutches are rigidly attached to the wrist and the wrist joint is welded, which means it allows no movement in any direction. This is a reasonable approximation because when walking on elbow crutches the high amount of force that needs to be transferred through the wrist joint keeps it locked in the most anatomical place to transfer weight. Furthermore, through a combination of shoulder and elbow rotation any crutch position can be achieved. As explained before, this change was done because the crutch acts as a very big lever, and a very small wrist rotation translates to a very big displacement of the crutch tip, which can pose difficulties to the solver and lead to an ill-conditioned problem.

5.4 Code structure and flexibility

The code structure has been designed to allow flexibility, so it can accomodate most of the different changes that have been made to the model during development. This structure not only eases future further development, but also allows the developer to go back in time, rerun an older simulation mixed with some new changes and see the results.

This flexibility is also useful for when new objective functions, gait patterns or other unforeseen variables have to be included, because there are specific sections defined all around the code for implementing new parameters. As a summary, here are the different hyperparameters that can be tweaked when running a simulation:

- **Model dimension**: to specify if you want to run a 2D, 2.5D or 3D simulation. This is important because the amount of states, controls... directly depends on how many joints there are, and the code is formatted in a way that can accomodate this.
- **Initial guess**: a variety of functions for initial guess refinement have been created during development. The most useful ones are *simulated*, that takes as initial guess a previous simulation, and *exp_preprocessed*, which takes experimental data from normal gait and preprocesses it to make the arm movement and GRF more similar to crutch gait.

- **Injury foot**: Allows to choose the injured foot, right or left, which in the swing-through gait will not touch the ground and in the three-point gait will touch the ground with the crutches, allowing for partial weight bearing.
- **Gait pattern**: Allows to choose between gait patterns, at the moment *swing-through* and *three-point*.
- **Actuation**: Allows to choose between the model actuation from *MTG*, *torque_jerk* (torque-driven with jerk in the problem formulation), and *torque_tau* (torque-driven with the derivative of the torque in the formulation).
- **MTG parameters**: Choose wherever the default MTG parameters or a specific set for a virtual patient should be used.
- **Stride length and speed**: The stride length and speed of gait can be choosen also as a hyperparameter.

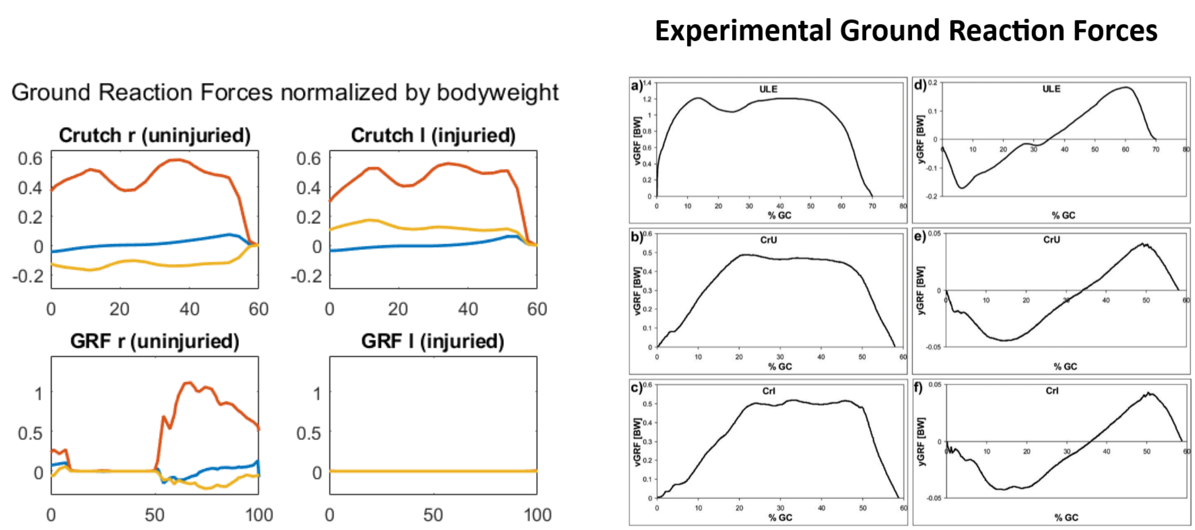
Regarding readibility, the code has been documented and divided into functions whenever possible to make it easier to understand and also to find a section/parameter. It is easier to find a 5 line specific version of the objective function if there is a file named *torque_jerk_2_5DFull.m* than searching for it in a 500 line long script.

6 Results and discussion

Even though an Optimal Solution to the problem has not been found, the solutions achieved by the solver before triggering the *Restoration failed!* are quite good. In this section, the different achieved solutions are compared and discussed, and since the simulations are videos, they are shared via links.

6.1 2D torque driven swing-through

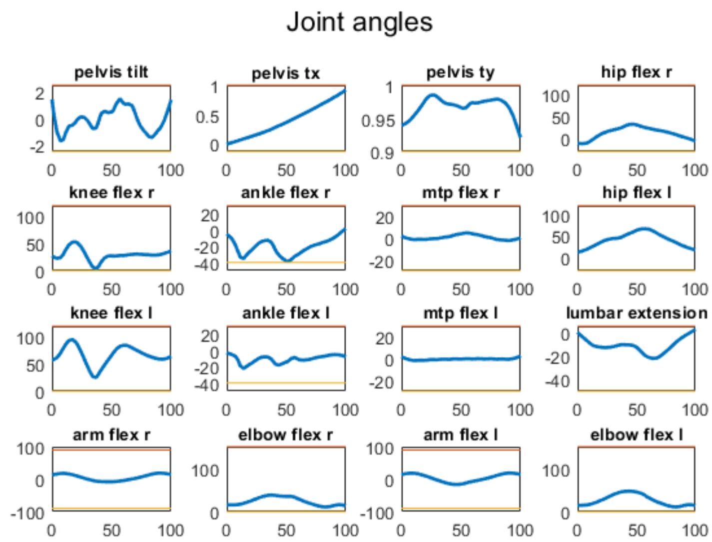
As explained in Model Development (Section 4), a 2D swing through crutch gait simulation was achieved as part of the development of the more complex three-point MTG-driven simulation. The results, while not converging to an optimal solution, pointed in the right direction. In this link you can see a video of how the simulation looks like. Notice how since it is a 2D simulation, the arms cannot move in any direction other than the sagittal plane, causing unrealistic behaviour when looking the simulation from the front. The sagittal view, however, is quite realistic.



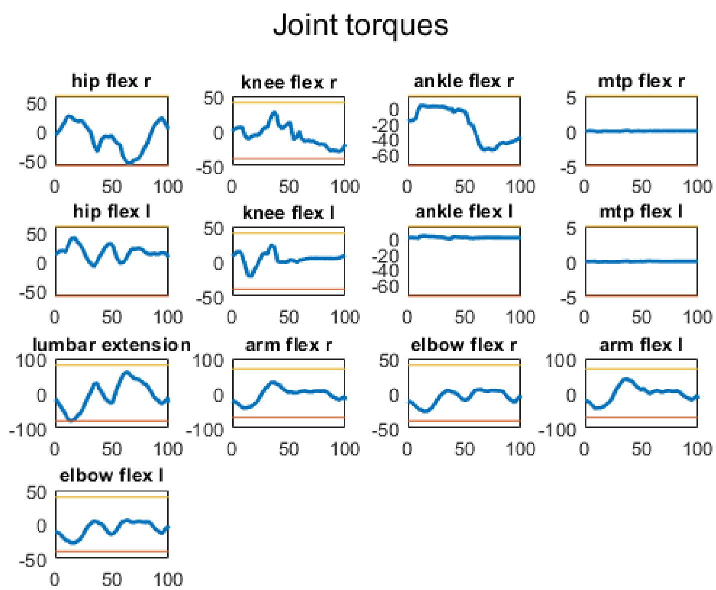
(a) **(b)**

Figure 14: Plots for GRF in swing-through motion. In figure (a) red corresponds to the vertical axis, blue to the antero-posterior and yellow to the medio-lateral. Figure (b) is taken from [52], in which the axis are the same scale as figure (a).

Figure 14 shows the GRF in the motion you just observed. In Rzepnicka2020 et al [52] they measured experimentally the forces involved in swing-through gait, you can see the results in Figure 14, plot (b). Their figure is very small, but the axis are on the same scale as plot (a) so they can be compared. The first column is the vertical GRF for the healthy foot (which only shows the non-zero part of the plot), the right crutch and the left crutch. It can be seen that while the shapes are not exact, the maximum magnitudes are quite close. Both crutches move around 0.6*bodyweight in both simulated and experimental data, and the max foot GRF is around 1.1*bodyweight. The difference in shape is probably due to the contact model, that is very simple. This similarity with experimental data suggests that the simulated framework is at least pointed in the right direction.



(a)



(b)

Figure 15: Plots for joint angle and joint torque evolution in the swing-through motion.

In Figure 15 you can see the plots that describe the motion you observed in link. Notice how the *pelvis ty* in Joint Angles (plot (a)) shows a double bump, something that generally seen experimentally. Regarding the torque values in plot (b), notice how the shoulder and elbow joint are producing quite high torques of around 60N, more than what is possible by an average human, taking into account that to hold a 3kg weight with your arm (assume 0.7m) fully horizontal you need around 20N of force. Reducing the bounds caused issues to other variables, so this analysis of the plots also points towards the need of increasing dimensionality, as justified in section 4.3.1.

6.2 MTG vs ideal torque driven 2.5D three-point gait pattern

The final formulation for the ideal torque driven model is very similar to the one specified in Section 5.2.1, with the difference that instead of activations it has torques as controls, and instead of minimizing activation it minimizes torque in the objective function. To find the best possible scenario to compare, combinations of stride length $(0.8, 1, 1.2)$ and speed $(0.6, 0.8, 1)$ were tried with the final formulation, finding the best pattern to be $Lstride = 1m$ and $speed = 0.6m/s$, much slower than normal gait as is common in crutch gait [2]. Since the formulation is almost identical, it allows for excellent comparison without confounding variables. In this link you can see the video for the ideal torque driven simulation and for the MTG driven one, side by side so it is easy to compare. I encourage the reader to pause and replay the video, try to look for differences, and even use the speed reduction tool the video player offers if desired.

When looking at the motions in OpenSim the first thing that is noticed is that the MTG version walks more upright, something that is always recommended when using crutches to avoid lumbar injuries. This suggests that the more complex MTG model captures better the forces in the trunk and pelvis. The other thing that is noticed is that the torque-driven version keeps the upper body still, while the MTG one swings it more. This detail however is minor because in an optimal solution, since the objective function minimizes joint power $(\tau \cdot v)$, this motion would likely be reduced.

In Figure 16 you can see the GRF that are associated with the motions you just visualized. They are quite similar, even if the torque driven version manages to use the left feet a little bit more. This dented shape is not reallistic but with tighter bounds in an optimal solution it would likely be fixed. It is logical once again that the crutches produce around 200N of force and the feet also produces around 200N, totaling at a little bit more than bodyweight (588N).

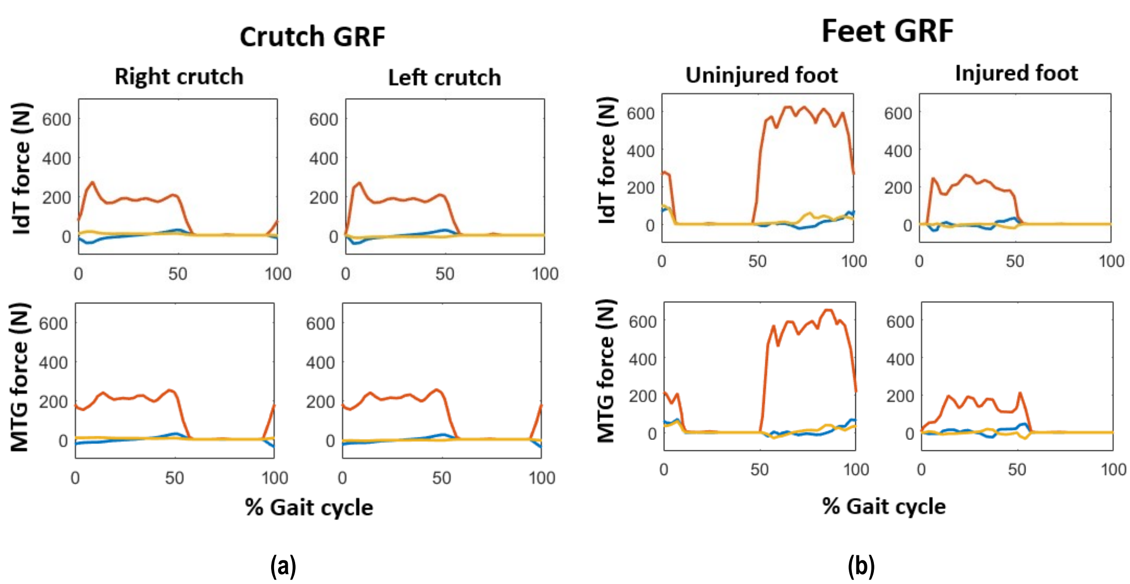
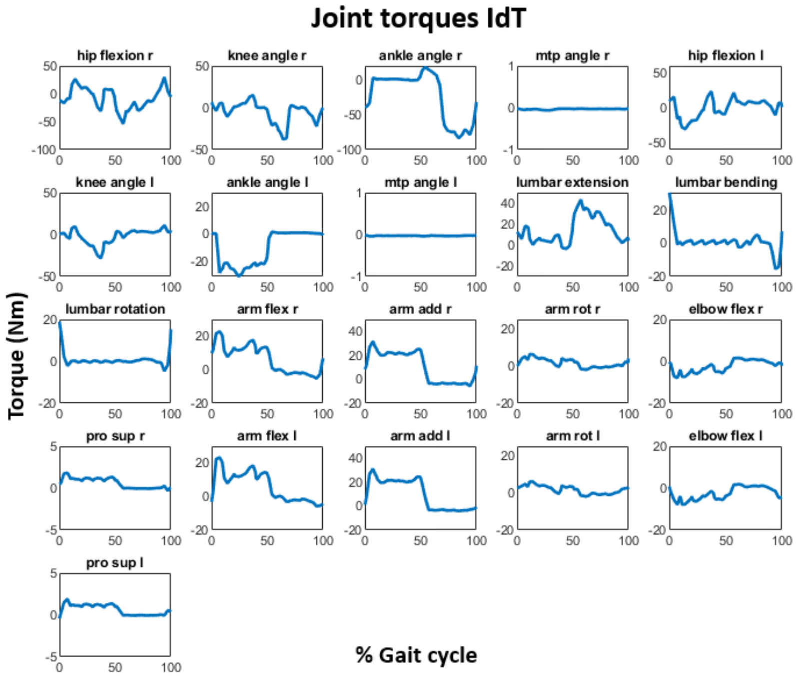
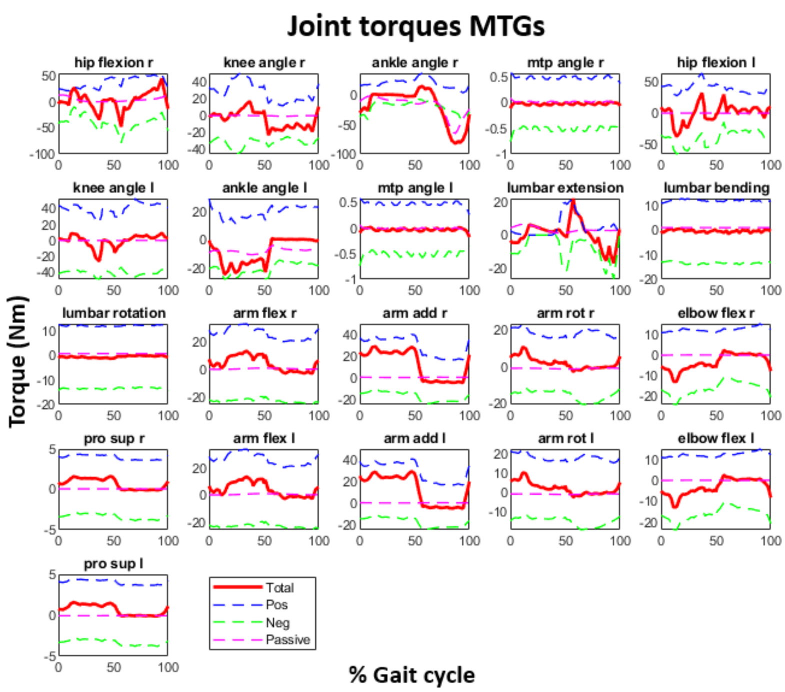


Figure 16: Comparison between the Ground Reaction Forces of the crutches (a) and the feet (b). The upper row corresponds to the torque-driven simulation and the bottom row corresponds to the MTG-driven simulation. Red corresponds to vertical axis, blue to antero-posterior and yellow to medio-lateral.



(a)



(b)

Figure 17: Comparison between joint torques under a torque driven (a) and MTG driven simulation (b).

When comparing the torque plots in Figure 17, the first clear observation is that they look very similar. The antagonistic balance between the flexor and extensor groups would be expected to provide smoother torque curves with less oscillation, but there is not a significant difference. As seen in the video, the only significant magnitude difference is in the lumbar extension joint, which is two times higher in the ideal torque-driven simulation. Regarding the realism of the simulations, the knee and ankle plots have the typical shape seen in normal gait [19], and the magnitudes of force produced in the shoulder and elbow joints are within reasonable ranges, unlike the swing-through version.

Having checked both 2.5D simulations, it can be concluded that adding MTGs to the crutch simulation framework does not make the simulation worse, but it also does not make it significantly better. Since the results are not optimal solutions, optimality conditions could be slightly different, but the preliminary findings point towards a realistic prediction of crutch gait, both in the visualization and in the specific joint variable analysis. For further validation, comparison against experimental measures should be performed, which will hopefully be done in the future in the scope of another project.

Regardless of the results, one clear advantage of the MTGs over the ideal torques simulation is the explainability and customizability of the MTG functions: they allow for specific modification of the force-angle, force-velocity and passive response of the muscles, for example in patients suffering from a muscle stiffening condition. This allows for further subject personalization, which is one of the aims of the bigger project that encompasses this thesis.

6.3 3D MTG driven three-point

Since development got stuck at the 2.5D version of the model, not a reduced amount of time was devoted to the 3D version. However, some initial simulations point in the right direction, as it can be seen in this link. The only difference with the 2.5D version is the hip and ankle extra DOFs, and the only strange behaviour can be seen at the endpoint with the left leg. This can most likely be fixed with some additional constraints, but adding constraints to a problem that already does not converge might not be the way to go. This is why I preferred to fix the convergence issue before attempting to clean the 3D simulation.

The main conclusion that can be extracted from the 3D simulations is that the 2.5D version is a nice approximation for the crutch gait model, because it produces very similar results while avoiding extra constraints at the lower body and, since it has less DOFs, it simulates faster. Also, the similarity between both simulations suggests that adding DOFs to the formulation is not a problem, it is robust in that way, so probably finding and fixing the cause of non-convergence in 2.5D would solve the problem in both formulations.

6.4 Hypotheses for non-convergence

As explained in Model Development (Section 4), I was unable to make the model converge to an optimal solution. After much investigation, a few hypotheses as to why the problem is not converging arise:

- **Ill-conditioned problem**:

  An ill-conditioned problem is one where a very small change in the input leads to a very large change in the output. This behaviour ”confuses” the solver and is a common cause of Restoration Failed. Since the difference with working formulations are the crutches, the problem might be that a small change in arm angle or shoulder rotation implies a big movement of the crutch tip and a big change in crutch GRF.

- **Inefficiency of crutch walking: OVERCONSTRAINING**:

  In an optimization problem we aim to minimize an objective function, in this case joint power and jerk are the decided metrics to represent effort and efficiency. Since crutch walking is by default a very inefficient way of motion (just think on how exhausting it is compared to normal gait) this ”traditional” minimization terms might not be the way to go, because there need to be many constraints for them to lead to a motion that is realistic. Many constraints end up defining an increasingly small feasible region, which makes the solver's job very difficult. In the lab, simulations with a walker or with normal gait have been achieved, which are intrinsically more efficient than crutch gait, but the methodology used in their code has not worked for this project.

  Maybe a different objective function would yield better results, but I have been unable to find one in this past months of experimentation.

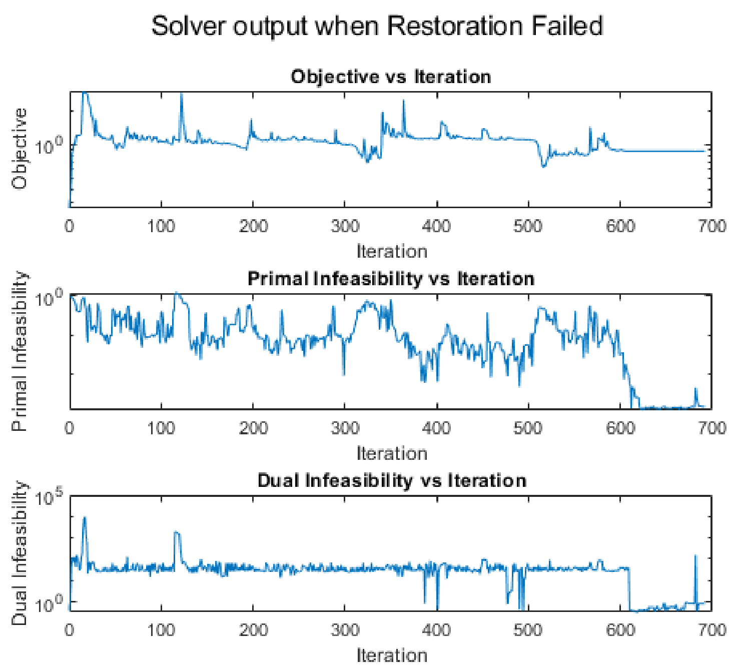
- **Solver issues**:

  For two weeks during development, the solver was not behaving as expected, it ran for hours in restoration mode until MATLAB crashed, as explained in Model Development (section 4) before. This problem was fixed by changing the settings of the solver, which lead to the *Restoration failed!* but at least the solutions found were quite good.

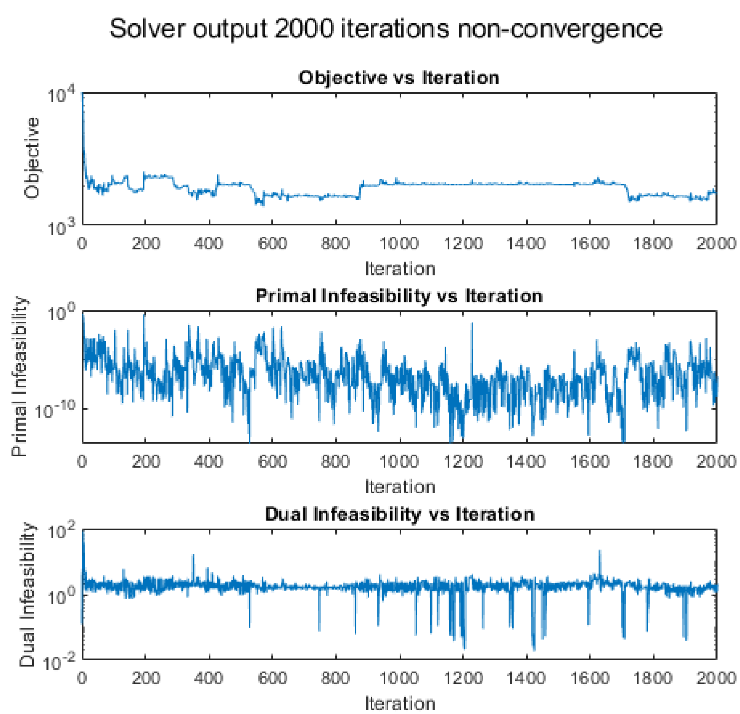
  In Figure 18 you can see two examples of solver output. The one on the left corresponds to a *Restoration failed!*, while the one on the right corresponds to one of the GPOPS scaled formulations that run for hours without converging anywhere. This big change in the solver behaviour just by changing a few settings parameters makes me think that changing the solver entirely might provide good results. However, this hypothesis was formulated very late in the project's timeline and, for this reason, it has remained out of the scope of this Bachelor's thesis.

Notice in Figure 18 how the **Primal Infeasibility** takes low values in both situations, but while in the non-convergence case it just keeps oscillating up and down forever, in the *Restoration failed!* case it drops various orders of magintudes about a 100 iterations before failing. Regarding **Dual Infeasibility**, which is I believe the indicator that shows the biggest issues, it just stays consistently at values greater than 1, oscillating and never decreasing, with the exception of the end period of the *Restoration failed!* simulation where it drops before failing.

Also, in most of the simulations and in both of the plots in Figure 18 the solver focuses on decreasing primal and dual infeasibility rather than minimizing the objective function, that does not behave at all as it would be expected, rising steadily at the first iterations of optimization and then reaching a plateau. Mathematically, this could mean that the solver has problems deciding which constraints are important and which ones are not. These behaviours are common in most of the simulations that end in *Restoration failed!* and should be studied in depth in further studies, because they most likely hide the problem keeping this formulation from converging to an optimal solution.



(a)



(b)

Figure 18: Comparison of two optimization solver outputs: (a) is a Restoration Failed and (b) is an infinite non-convergence.

7 Timeline of Execution

In this section the timeline of execution is described. Firstly, the work breakdown structure is exposed, with a description of all tasks involved, then the PERT analysis is shown and finally the GANTT diagram of this thesis is discussed.

7.1 Work breakdown structure (WBS)

The WBS is a hierarchical decomposition of the project into smaller and more manageable parts, which allows for a better understanding of the project and helps with planning and tracking the project's progress. In Figure 19 you can see the Work Breakdown Structure (WBS) of the project. The project is divided into 6 main tasks (in green), and each of these tasks is divided into subtasks (in yellow).

In Table 6 the different tasks are detailed, specifying the objectives and the expected results of each one. This tasks will be used in the PERT/GANTT diagram in the next section.

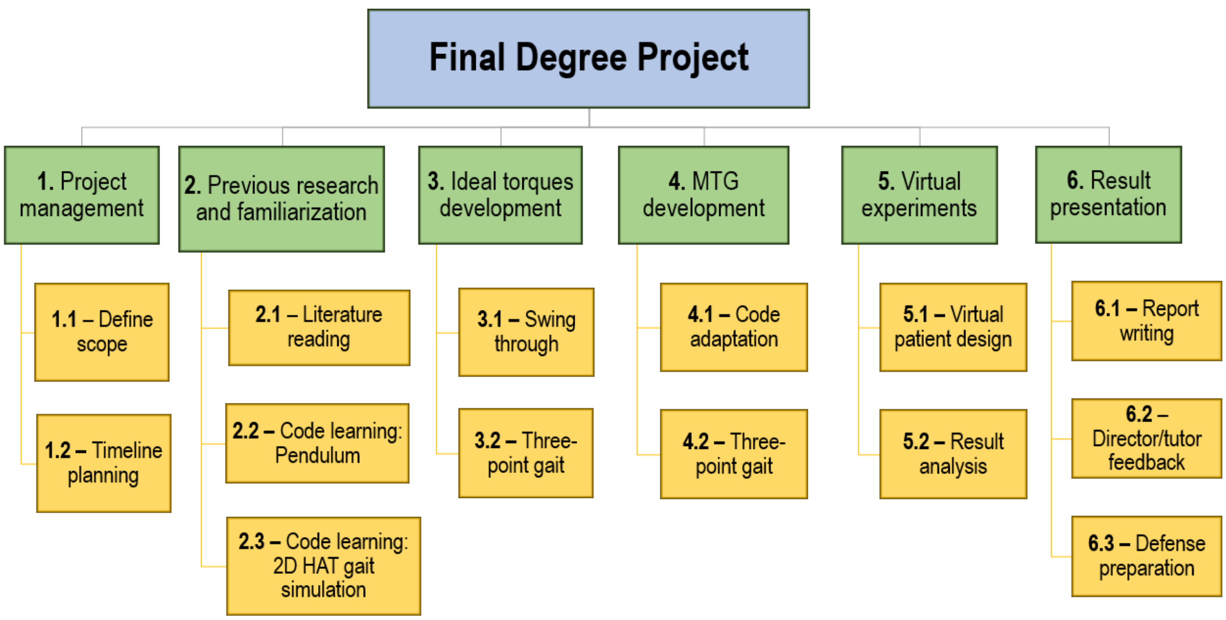


Figure 19: Work breakdown structure (WBS) of the project

Table 6: Task dictionary of the project

1. Project Management		
1.1	Define scope	Establish the project boundaries and objectives.
1.2	Timeline planning	Develop a schedule for project milestones and deadlines.

2. Previous Research and Familiarization		
2.1	Literature reading	Review relevant academic and technical literature, as well as the work of previous students in the lab
2.2	Code learning: Pendulum	Study the Optimal Control Problem code for the simple pendulum simulation to learn how GPOPS works.
2.3	Code learning: 2D HAT gait simulation	Understand the more complex code for 2D HAT gait simulation. Also improve the starter pack (see Section 5.1)

3. Ideal Torques Development		
3.1	Swing through	As an initial exploration of the crutch problem, develop ideal torques for swing through motion.
3.2	Three-point gait	Develop ideal torques formulation for the three point gait.

4. MTG Development		
4.1	Code adaptation	Adapt the code to the MTG functions, and the OpenSim model to the new reference.
4.2	Three-point gait	Develop MTG driven three-point gait.

5. Virtual Experiments		
5.1	Virtual patient design	Design virtual patient models for experiments.
5.2	Result analysis	Analyze the results of the virtual experiments.

6. Result Presentation		
6.1	Report writing	Write the final project report.
6.2	Director/tutor feedback	Incorporate feedback from the director/tutor.
6.3	Defense preparation	Prepare for the project defense presentation.

7.2 PERT diagram

The PERT diagram shows the dependencies between the different tasks of the project. The tasks are represented by arrows, and the time limitations are represented inside the nodes. In table 7 you can see the detail of the PERT and in figure 20 you can see the PERT diagram.

Table 7: PERT table of the project

WBS ID	PERT ID	Title	Requirements	Planned Duration (weeks)
1.1	A	Define scope	None	1
1.2	B	Timeline planning	A	1
2.1	C	Literature reading	B	2
2.2	D	Code learning: Pendulum	B	1
2.3	E	Code learning: 2D HAT gait simulation	D	2
3.1	F	TD swing through	C,E	3
3.2	G	TD three-point gait	F	4
EXT	EXT	MTG Calibration (Carlos)	None	9
4.1	H	Code adaptation	EXT,E,C	2
4.2	I	MTG three-point gait	H	4
5.1	J	Virtual patient design	C	1
5.2	K	Result analysis	G,I,J	1
6.1	L	Report writing	K	1
6.2	M	Director/tutor feedback	L	1
6.3	N	Defense preparation	M	1

As explained throughout the thesis, one of the initial sub-objectives of the project has not been reached (Task 5.2), which in experimentation and research is common, experiments do not always go the way we expect. This does not mean that significant conclusions could not be made, but since the virtual patient result analysis required a fully functional model for crutch gait, they were impossible to perform. In terms of the real execution of the project, node 16 has been skipped, with task L (Report writing) connecting directly node 13 to node 15.

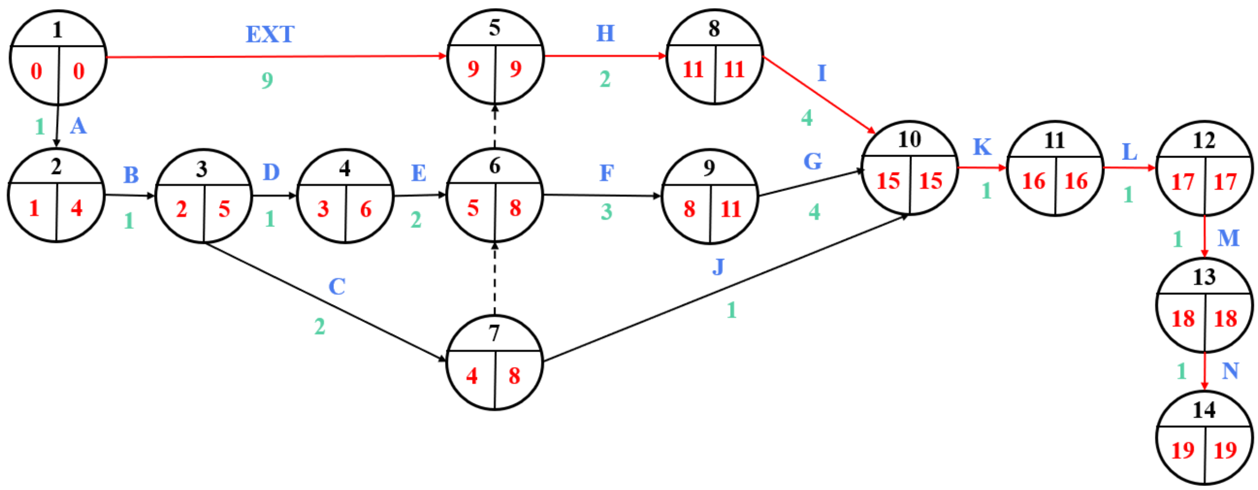


Figure 20: PERT diagram of the project. Inside the nodes, the left number is the earliest time of completion of the incoming tasks and the right number is the latest start time of the outgoing tasks.

7.3 GANTT diagram

It is worth noting that this project was planned with 2 weeks of buffer time to account for possible delays in the tasks. This buffer time was not enough to account for the delays in the MTG development, which became infinite in the sense that the problem was not fully solved within the time frame of the project. In figure 21 you can see the GANTT diagram of the execution of the project.

Milestone description	Start	Weeks
1.1 - Define Scope	22/01/2024	1
1.2 - Timeline planning	29/01/2024	1
2.1 - Literature reading	12/02/2024	2
2.2 - Code learning: Pendulum	05/02/2024	1
2.3 - Code learning: 2D HAT	12/02/2024	2
3.1 - TD Swing through	26/02/2024	3
3.2 - TD three-point gait	18/03/2024	4
EXT - MTG calibration (Carlos)	29/01/2024	9
4.1 - Code adaptation	01/04/2024	2
4.2 - MTG three-point gait	15/04/2024	6
5.1 - Virtual patient design	27/05/2024	1
5.2 - Result analysis	-	-
6.1 - Report writing	03/06/2024	1
6.2 - Director/tutor feedback	10/06/2024	1
6.3 - Defense preparation	17/06/2024	1

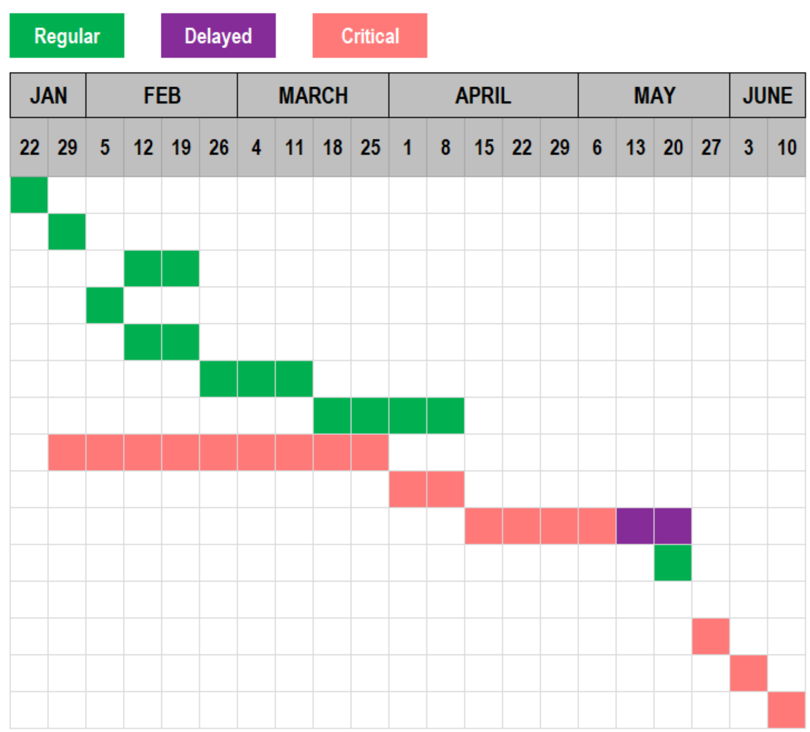


Figure 21: GANTT diagram of the project

8 Project viability

In this section the project viability is analysed. For the technical viability, a SWOT analysis is performed; and for the economical viability the different costs incurred in this project are documented.

8.1 Technical viability

Table 8: SWOT Analysis for the Project

Strengths	Weaknesses
• Strong theoretical foundation based on previous work by Febrer-Nafria et al. • Comprehensive approach covering both 2D and 3D simulations. • Integration of more complex MTG functions into ideal torque models. • Virtual testing allows for safe and controlled experimentation. • Output is a well documented code prepared for future use in the lab.	• Dependence on qualitative validation, which may not be as robust as quantitative validation. • Complexity of crutch gait may introduce unforeseen challenges. • Limited initial knowledge of control theory. • Inability to capture CNS dependant stimulus like fear of falling, pain or equilibrium.
Opportunities	**Threats**
• Potential for future use in real SCI patient projects. • Opportunity to refine and validate the model in future projects. • Possible publication of findings and contributions to the field. • Potential to extend the model for other crutch gait patterns.	• Solver code is unaccessible so solver issues might be very hard to solve. • Future projects may encounter difficulties if the model is not robust. • Dependence on the accuracy of previous work and existing OpenSim models. • Unforeseen challenges during model development.

As you can see, the SWOT analysis has a nice balance between strengths and opportunities, and weaknesses and threats. The project has a strong theoretical foundation, a comprehensive approach, and the potential for future use in real SCI patient projects. However, it also has some weaknesses and threats, such as dependence on qualitative validation and solver issues that may be hard to solve. Overall, the project is technically viable and has the potential to make significant contributions to the field of computational biomechanics.

8.2 Economical viability

This project has a diversity of costs associated with it:

- **OpenSim**: OpenSim is free to use.
- **GPOPS-II**: it has a license costing 750$ for the whole department. During my 5 month stay there were more than 10 people with a GPOPS simultaneously, so a cost estimation of 75$/pp/year is reasonable.
- **MATLAB**: it has a license cost of 69$ per year for students, which is covered by the university.
- **Laptop**: the laptop used for the project was an ASUS ZenBook with a cost of 750€. Its repeated use for simulations day and night, constant overheating and RAM overuse have caused a significant amount of deterioration, which I estimate to be around a third of the original price, so 250€.
- **Hours of work**: the project has taken 21 weeks of part-time work, which is 420 hours. The cost of the hours of work is estimated to be 10€/hour, which is a reasonable salary for a student. This gives a total of 4200€. The director has invested around 1h per week to this project. Estimating a rate of 20€/hour, this gives a salary of 420€.

If we add all this costs up and do the necessary currency exchanges using a rate of 1€ = 1.08$, we get a total of $\sim$ **5003 €**, a reasonable cost for a project of this complexity.

9 Regulations for clinical decision support systems

While the current scope of the project may not immediately trigger extensive regulatory requirements, it is important to be aware of existing laws and standards that may become relevant in future phases. Given that the project aims to guide clinical decisions, it will be subject to various regulations and standards designed to ensure patient safety, data privacy and system reliability. The following regulations are particularly relevant:

9.1 General data protection regulation (GDPR)

Compliance with data privacy regulations, particularly the General Data Protection Regulation (GDPR) [53], is essential. Since the GDPR sets the standards for data protection across the European Union, it is important to establish robust data handling practices that prioritize privacy and security from the beggining. This includes:

- **Data minimization**: Ensuring that only data necessary for the intended purpose is collected and processed.
- **Consent**: Obtaining explicit consent from patients for data collection and processing.
- **Data security**: Implementing robust security measures to protect data from breaches.
- **Transparency:** Providing clear information to patients about how their data will be used.

9.2 Medical device regulations (MDR)

In the European Union, software intended for medical purposes, including clinical decision support systems (CDSS), is regulated under the Medical Device Regulation (EU) 2017/745 (MDR) [54]. Under MDR, such software is classified based on its intended use and risk. Ensuring compliance with MDR involves:

- **Classification**: Determining the appropriate classification of the software (Class I, IIa, IIb, or III) based on its intended purpose and the level of risk associated with its use.
- **Conformity assessment**: Undertaking a conformity assessment procedure, which may require involvement from a Notified Body, particularly for higher-risk classes.
- **CE marking**: Achieving CE marking to indicate that the software meets EU safety, health, and environmental protection requirements.

9.3 Clinical evaluation and validation

To avoid the software becoming a *black box*, it is essential to go thorough clinical evaluations and validation studies, as well as to keep this threat in mind during the development steps of the software, documenting every decision and inner working. This involves:

- **Clinical evidence**: Collecting clinical evidence to demonstrate the software's performance, safety, and benefits.

- **Transparency**: Ensuring that the decision-making process of the software is transparent and explainable. This may involve using explainable algorithms to make the software's recommendations understandable to clinicians.
- **Post-market surveillance**: Establishing a system for continuous monitoring and reporting of the software's performance in real-world settings.

9.4 Health insurance portability and accountability act (HIPAA) (if applicable)

If the project involves collaboration with entities in the United States or deals with data of US citizens, compliance with the Health Insurance Portability and Accountability Act (HIPAA) [55] is necessary. HIPAA sets standards for the protection of health information and applies to entities that handle such data.

9.5 Ethical standards

As the project involves research and potential applications in the field of biomechanics, it is important to adhere to ethical standards governing research involving human subjects. While direct human experimentation may not be part of the current scope, any future studies involving human participants must comply with relevant ethical guidelines and obtain appropriate approvals from the university's ethical committees.

10 Conclusions

After all these months working in this project, it is now time to look back and extract some conclusions. Looking at the objective definition, it is clear that a lot of progress has been made towards a functioning MTG-driven crutch gait simulation. In the Detail Engineering section a new formulation for the problem has been explained, one that has been studied for both swing-through and three-point gait, allows a variety of hyperparameters and works for two subjects of quite different height and weight.

Furthermore, when compared to the previous code from were the project started, the simulation framework has been changed toward a more readable and organized format, dividing the code into functions whenever possible and adding comments and documentation where needed. This effort has been done with the aim of easing the work of future colleagues further developing, experimenting and integrating this simulation framework in a bigger project.

The main issue encountered during the project was the lack of convergence of the solver despite reaching solutions that are realistic and acceptable. This unforeseen difficulties have forced me to dig inside the inner workings of optimization solvers to find the causes, and point the next steps in the right direction. They have also removed focus from the secondary objective of developing virtual patients and shifted it back to the primary aim of developing the framework. Nevertheless, the preliminary results shown are very promising and show that achieving an optimal solution for this problem is not only possible, but it is not far from the formulation presented in this project. In the next steps section below I point out the different directions and experiments that could be done as a follow-up of this project.

From an academic point of view, during this project I have explored the world of optimization and numerical simulations, all from a biomechanical point of view, which I believe is a very original and beautiful application of a technology commonly used to guide rockets into orbit or for motion planning in robotics. Apart from the theory behind the project, I have developed an ability to navigate very complex problems and creativity for, when a problem seems like an unclimbable wall, finding new gripping points to keep getting closer to a solution. These skills are exportable to other widely different projects, and will for sure be useful during my professional career.

On a more personal note, during this project I have experienced the frustration of a research idea not going as initially planned. This is a common feeling in the academic world, where the questions you are trying to solve might not have an answer, and it has taught me that sometimes the absence of perfect results is a result itself, providing key insights and opening new lines for future research.

10.1 Future steps

In order to further develop the simulation framework towards a working optimal solution that can be personalized and used for patients in the scope of a bigger project, the following next steps are suggested:

Mathematic exploration of the formulation

The major limitation in achieving an optimal solution in this thesis has been the *Restoration Failed* error. In Results and Discussion (Section 6) a few hypotheses have been presented as to why this EXIT situation might be happening. To check if there is a hidden problem with the constraints, a thorough mathematical analysis should be made.

For example, if the determinant of the constraints matrix is zero, it indicates that some of the constraints are linearly dependent. This linear dependency can significantly narrow the solution space, akin to having a plane in a three-dimensional space where two lines are dependent. Such ill-conditioning makes it challenging for the optimal control problem to converge, as the solution space becomes very restricted in certain directions. By identifying and addressing these dependencies, the conditioning of the problem and the likelihood of convergence could be improved.

Also, evaluating the Jacobian matrix, which is the partial derivative matrix of the constraints with respect to the states and controls, would provide insights on the sensitivity of constraints to changes in these variables. If the Jacobian reveals that certain constraints have high sensitivity (are ill-conditioned), it may indicate potential issues with the problem formulation or constraints. It could also help identifying redundant constraints, which could be difficulting convergence. Analysing the Hessian matrix, which is the partial derivative of the Jacobian, could also provide insights related to why the problem is not converging, uncovering the nature of the diverse critical points of the solution space and helping determine if the solver algorithm is good or needs to be changed, because the Hessian matrix plays an important role on deciding how the problem's variables are modified in each iteration.

A good idea would be to repeat this analysis on other formulations developed in the lab that have converged to an optimal solution and compare the results to this one, to hopefully find the weaknesses of this formulation and improve them towards achieving convergence.

Changing the solver used

GPOPS-II is a powerful solver but, as shown in this project, it has some limitations that might be difficulting the formulation to reach an optimal solution. To overcome this limitations, the problem formulation could be translated to another OCP solver, implement algorithmic differentiation or other unexplored options, and see if this new approach avoids the *Restoration Failed* early EXIT message, if it provides better solutions…This was left outside the scope of this thesis because it is very time consuming, but could be a good starting point for another one.

Get experimental measures

To validate the results of this thesis and further advancements made in the future, experimental data will be needed. Measuring experimental data is very time-consuming and requires specialized equipment, but can be crucial in developing predictive models like the one in this thesis, and should definitely be done in the scope of a future project.

Enforce partial weight bearing (PWB)

Partial Weight Bearing is a parameter used to define the gait pattern in assymetrical gaits like the ones studied in this thesis. In the final problem formulation, it was enforced as a maximum in the feet and crutches GRF, but further exploration could be made in this aspect to find a better way to enforce it. A good objective could be to leave the GRF capped at healthy values and let the solver decide which is the best PWB for a patient with a specific injury, building towards the greater goal of this software being part of a clinical decision aid system.

References

[1] Sigmund Epstein. Art, history and the crutch. *Annals of medical history*, 9(4):304, 1937.

[2] Fatemeh Rasouli and Kyle B Reed. Walking assistance using crutches: A state of the art review. *Journal of Biomechanics*, 98:109489, 2020.

[3] EC Eurostat. Population structure and ageing. *European Commission: Luxembourg*, 2014.

[4] L Brewer, F Horgan, A Hickey, and D Williams. Stroke rehabilitation: recent advances and future therapies. *QJM: An International Journal of Medicine*, 106:11–25, 5 2012.

[5] Ranita H K Manocha, Megan K MacGillivray, Mehdi Eshraghi, and Bonita J Sawatzky. Injuries associated with crutch use: A narrative review. *PM and R : the journal of injury, function, and rehabilitation*, 13:1176–1192, 10 2021.

[6] Míriam Febrer-Nafría, Roger Pallarès-López, Benjamin J Fregly, and Josep M Font-Llagunes. Prediction of three-dimensional crutch walking patterns using a torque-driven model. *Multibody System Dynamics*, 51:1–19, 2021.

[7] Brandon N Fournier, Edward D Lemaire, Andrew J J Smith, and Marc Doumit. Modeling and simulation of a lower extremity powered exoskeleton. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 26:1596–1603, 8 2018.

[8] Francisco Mouzo, Urbano Lugrís, Rosa Pàmies-Vilà, and Javier Cuadrado. Skeletal-level control-based forward dynamic analysis of acquired healthy and assisted gait motion. *Multibody System Dynamics*, 44:1–29, 2018.

[9] Marko Ackermann and B. A. Taissun. A computational study of the swing phase of the gait with standard and spring-loaded crutches. *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1476–1481, 2012.

[10] Guangyu Liu, Sheng-Quan ShaneXie, and Yanxin Zhang. Optimization of spring-loaded crutches via boundary value problem. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 19:64–70, 2 2011.

[11] Antoine Falisse, Gil Serrancolí, Christopher L Dembia, Joris Gillis, Ilse Jonkers, and Friedl De Groote. Rapid predictive simulations with complex musculoskeletal models suggest that diverse healthy and pathological human gaits can emerge from similar control strategies. *Journal of the Royal Society, Interface*, 16:20190402, 8 2019.

[12] Míriam Febrer-Nafría, Ali Nasr, Mahdokht Ezati, Peter Brown, Josep M Font-Llagunes, and John McPhee. Predictive multibody dynamic simulation of human neuromusculoskeletal systems: a review. *Multibody System Dynamics*, 58:299–339, 2023.

[13] Friedl De Groote and Antoine Falisse. Perspective on musculoskeletal modelling and predictive simulations of human movement to assess the neuromechanics of gait. *Proceedings of the Royal Society B: Biological Sciences*, 288:20202432, 2021.

[14] Roger Pallarès López. Optimal control prediction of dynamically consistent walking motions. *ETSEIB - UPC*, 2017.

[15] David Lassierra. Development of muscle torque generators for optimal control tracking of human walking. *ETSEIB - UPC*, 2022.

[16] Eduard Vilanova Badosa. Prediction of human motion using musculoskeletal models driven by muscle torque generators. *ETSEIB - UPC*, 2023.

[17] Duane Knudson. *Introduction to Biomechanics of Human Movement*, pages 3–18. Springer International Publishing, 2021.

[18] Oliver Jones. Anatomical planes, 2022. Available at https://teachmeanatomy.info/the-basics/anatomical-terminology/planes/. Accessed 30/03/2024.

[19] T.K. Uchida, S.L. Delp, and D. Delp. *Biomechanics of Movement: The Science of Sports, Robotics, and Rehabilitation*. MIT Press, 2021.

[20] Vladimir Medved, Rodolfo Vastola, Daniele Albano, and Marko Pećina. *Gait Analysis*, pages 219–255. Springer International Publishing, 2021.

[21] Arif Reza Anwary, Hongnian Yu, and Michael Vassallo. An automatic gait feature extraction method for identifying gait asymmetry using wearable sensors. *Sensors*, 18, 2018.

[22] Gary L Smidt and M A Mommens. System of reporting and comparing influence of ambulatory aids on gait. *Physical Therapy*, 60:551–558, 5 1980.

[23] David J Saxby, Bryce Adrian Killen, C Pizzolato, C P Carty, L E Diamond, L Modenese, J Fernandez, G Davico, M Barzan, G Lenton, S Brito da Luz, E Suwarganda, D Devaprakash, R K Korhonen, J A Alderson, T F Besier, R S Barrett, and D G Lloyd. Machine learning methods to support personalized neuromusculoskeletal modelling. *Biomechanics and Modeling in Mechanobiology*, 19:1169–1185, 2020.

[24] Benjamin J Fregly. A conceptual blueprint for making neuromusculoskeletal models clinically useful. *Applied Sciences*, 11, 2021.

[25] F Romero and F J Alonso. A comparison among different hill-type contraction dynamics formulations for muscle force estimation. *Mechanical Sciences*, 7:19–29, 2016.

[26] Serdar Göktepe, Andreas Menzel, and Ellen Kuhl. The generalized hill model: A kinematic approach towards active muscle contraction. *Journal of the mechanics and physics of solids*, 72:20–39, 12 2014.

[27] Darryl G Thelen. Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *Journal of Biomechanical Engineering*, 125:70–77, 2 2003.

[28] Matthew Millard, Thomas Uchida, Ajay Seth, and Scott L Delp. Flexing computational muscle: Modeling and simulation of musculotendon dynamics. *Journal of Biomechanical Engineering*, 135, 2 2013.

[29] Dan Robbins. *Chapter 7 - Muscle biomechanics*, pages 121–135. Academic Press, 2022.

[30] Keaton A Inkol, Colin Brown, William McNally, Conor Jansen, and John McPhee. Muscle torque generators in multibody dynamic simulations of optimal sports performance. *Multibody System Dynamics*, 50:435–452, 2020.

[31] Matthew Millard, Manish Sreenivasa, and Katja Mombaur. Predicting the motions and forces of wearable robotic systems using optimal control. *Frontiers in Robotics and AI*, 4, 2017.

[32] Andrew J Meyer, Ilan Eskinazi, Jennifer N Jackson, Anil V Rao, Carolynn Patten, and Benjamin J Fregly. Muscle synergies facilitate computational prediction of subject-specific walking motions. *Frontiers in bioengineering and biotechnology*, 4:77, 2016.

[33] Tim W Dorn, Jack M Wang, Jennifer L Hicks, and Scott L Delp. Predictive simulation generates human adaptations during loaded and inclined walking. *PloS one*, 10:e0121407, 2015.

[34] Hartmut Geyer and Hugh Herr. A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 18:263–73, 6 2010.

[35] Josep M Font-Llagunes, Ana Barjau, Rosa Pàmies-Vilà, and József Kövecses. Dynamic analysis of impact in swing-through crutch gait using impulsive and continuous contact models. *Multibody System Dynamics*, 28:257–282, 2012.

[36] Anne D Koelewijn, Dieter Heinrich, and Antonie J van den Bogert. Metabolic cost calculations of gait using musculoskeletal energy models, a comparison study. *PloS one*, 14:e0222037, 2019.

[37] The MathWorks Inc. Matlab version: 9.6.0 (r2019a), 2019.

[38] Scott L Delp, Frank C Anderson, Allison S Arnold, Peter Loan, Ayman Habib, Chand T John, Eran Guendelman, and Darryl G Thelen. Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on bio-medical engineering*, 54:1940–50, 11 2007.

[39] AnyBody technology. Anybody modelling system website. Online. Available at https://www.anybodytech.com/.

[40] Michael A Patterson and Anil V Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Trans. Math. Softw.*, 41, 10 2014.

[41] Adriaan Falisse, Gabriel Serrancolí, Christopher L Dembia, Jonathan Gillis, and Friedl De Groote. Algorithmic differentiation improves the computational efficiency of opensim-based trajectory optimization of human movement. *PLoS ONE*, 14(10):e0217730, 2019.

[42] Matthew J. Weinstein, Anil V Rao, Matthew J. Weinstein, Rao, and Anil V Rao. Algorithm: Adigator, a toolbox for the algorithmic differentiation of mathematical functions in matlab using source transformation via operator overloading distribution a. approved for public release; distribution unlimited. author's addresses. Online. Available at https://api.semanticscholar.org/CorpusID:262398217.

[43] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[44] Rui Lima, Andreia Monteiro, António J Salgado, Susana Monteiro, and Nuno A Silva. Pathophysiology and therapeutic approaches for spinal cord injury. *International Journal of Molecular Sciences*, 23, 2022.

[45] David Civantos. Study of the foot-ground contact model for the prediction of human gait. *ETSEIB - UPC*, 2020.

[46] Gary Tad Yamaguchi. *Dynamic Modeling of Musculoskeletal Motion*. Springer US, 2001.

[47] Colin Brown. Predictive forward dynamic simulation of manual wheelchair propulsion. *University of Waterloo*, 2018.

[48] MUMPS. Multifrontal massively parallel solver (mumps 4.10.0) user's guide. Online, 2011.

[49] HSL Mathematical Software Library. Ma57: Sparse symmetric system: multifrontal method. Online. Available at https://www.hsl.rl.ac.uk/catalogue/ma57.html.

[50] Michael A. Patterson and Anil V. Rao. Gpops user guide. Online, 2016. Available at https://www.gpops2.com/resources/gpops2UsersGuide.pdf.

[51] Apoorva Rajagopal, Christopher L. Dembia, Matthew S. DeMers, Denny D. Delp, Jennifer L. Hicks, and Scott L. Delp. Full-body musculoskeletal model for muscle-driven simulation of human gait. *IEEE Transactions on Biomedical Engineering*, 63(10):2068–2079, 2016.

[52] Agata Rzepnicka, Jarosław Kabaciński, Michał Murawa, Anna Fryzowicz, Małgorzata Syczewska, and Lechosław Bogdan Dworak. Biomechanical evaluation of swing-through crutch gait in patients with lower extremity injury. *Acta of Bioengineering and Biomechanics*, 22, 2020.

[53] European Parliament. General data protection regulation. Online, 2018. Available at https://gdpr-info.eu/.

[54] European Parliament. Regulation 2017/745 (medical device regulation). Online, 2017. Available at https://eur-lex.europa.eu/eli/reg/2017/745/oj.

[55] US Department of Health and Human Services. Health insurance portability and accountability act. Online, 1996. Available at https://www.hhs.gov/hipaa/index.html.