UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

FINAL DEGREE PROJECT

Bachelor's degree in Mechanical Engineering

SIMULATION OF THE HUMAN GAIT AS AN OPTIMAL CONTROL PROBLEM USING AUTOMATIC DIFFERENTIATION AND COMPLEX CONTACT MODELS

Report

Author: Joan Badia Torres
Director: Gil Serrancolí Masferrer

Call: June 2022

Abstract

In recent years, there has been a considerable effort to enhance treatments or surgical procedures by adapting them to individual patients. The idea of personalized medicine has influenced many developments in multiple fields, mainly in areas that ultimately trace back to diagnostics and the design of rehabilitation treatments.

The goal of this project is the improvement of a model of the human gait so that it can correctly describe its motion and dynamics. Additionally, the data extracted from the previous program is used to analyse the behaviour of a knee prosthesis, with the goal of calculating and visualizing the pressure distribution on its surface.

To reach the objectives laid out above, the patient's gait is studied as an optimal control problem (OCP), making use of a complex contact model and CasADi, a tool that uses algorithmic differentiation to accurately and efficiently solve OCPs. In the case of the prosthesis, a mesh-based contact model is used to obtain the pressure map which is displayed on a 3D model of the object using Blender.

The result is a tool that can adapt to multiple cases and support the work of healthcare professionals, providing them with the insight they need in a concise manner. In doing so, it should also contribute to advancing both personalized treatment and the use of tools like algorithmic differentiation.

Date of last modification: June 15th, 2022.

Resum

En els darrers anys hi ha hagut un esforç considerable per millorar tractaments i cirurgies adaptant-los individualment als pacients. El concepte de medicina personalitzada ha influït sobre una multitud d'avenços en diversos camps, principalment en àrees relacionades amb tècniques diagnòstiques i el disseny de tractaments de rehabilitació.

L'objectiu d'aquest projecte és la millora d'un model de la manera de caminar d'un humà per poder descriure correctament les característiques cinemàtiques i dinàmiques del cicle. A més, la informació produïda pel programa s'utilitza per estudiar el comportament d'una pròtesi de genoll i calcular i representar la distribució de pressions sobre la superfície de la peça.

A fi d'aconseguir els objectius proposats, la manera de caminar del pacient s'analitza com un problema de control òptim (OCP) utilitzant un model de contacte complex i CasADi, una eina que empra derivació algorítmica per resoldre OCPs de manera precisa i eficient. Pel que fa a l'estudi de la pròtesi, es fa servir un altre model de contacte basat en una malla per obtenir el mapa de pressions, que es visualitza amb Blender.

El resultat és una eina de suport a professionals de la salut capaç d'aportar la perspectiva que poden necessitar de manera clara i directa. Alhora, aquesta tecnologia també hauria de contribuir a expandir tant el concepte de medicina personalitzada com l'ús d'eines com la derivació automàtica.

Darrera modificació d'aquest document: 15 de juny de 2022.

Resumen

En los últimos años se ha realizado un esfuerzo considerable para mejorar tratamientos y cirugías adaptándolos individualmente a los pacientes. El concepto de medicina personalizada ha influido sobre una multitud de avances en varios campos, principalmente en áreas relacionadas con técnicas diagnósticas y el diseño de tratamientos de rehabilitación.

El objetivo de este proyecto es la mejora de un modelo de la andadura del ser humano para poder describir correctamente las características cinemáticas y dinámicas del ciclo. Además, la información producida por el programa se utiliza para estudiar el comportamiento de una prótesis de rodilla y calcular y representar la distribución de presiones sobre la superficie de la pieza.

Para lograr los objetivos propuestos, el caminar del paciente se analiza como un problema de control óptimo (OCP) utilizando un modelo de contacto complejo y CasADi, una herramienta que usa derivación automática para resolver OCPs de manera precisa y eficiente. Por lo que se refiere al estudio de la prótesis, se emplea otro modelo de contacto basado en una malla para obtener el mapa de presiones, el cual se visualiza en Blender.

El resultado es una herramienta de apoyo a profesionales de la salud capaz de aportar la perspectiva que pueden necesitar de manera clara y directa. A la vez, esta tecnología también debería fomentar el concepto de medicina personalizada y el uso de herramientas como la derivación algorítmica.

Última modificación de este documento: 15 de junio de 2022.

Acknowledgements

I'd like to thank my family and friends for your encouragement and support, and for putting your faith in me not just these last few months, but during the last four years.

I'd also like to thank the people I've met since I began attending classes at the EEBE, I'm not confident I would've made it this far without you all.

And last but certainly not least, I'd like to thank Gil Serrancolí. Thank you for your invaluable guidance, for pushing me to challenge myself and learn more, and for the great effort you have invested in this project. Thank you.

Glossary

AD: Automatic Differentiation, Algorithmic Differentiation.

Adduction: Movement of a body part toward the midline of the body (*Definition adapted from the NCI thesaurus of the NIH* [1]).

API: Application Programming Interface.

Design variable: In numerical optimisation, a variable the value of which is set by the user.

DoF: Degree of Freedom.

Flexion: Bending movement between body parts (*Definition adapted from the NCI thesaurus of the NIH* [2]).

GRF: Ground Reaction Forces.

GRM: Ground Reaction Moments.

Internal rotation: Rotational movement of a body part inward, toward the body (*Definition adapted from the NCI thesaurus of the NIH* [3]).

JRM: Joint Residual Moments.

KCF: Knee Contact Force.

NCSRR: National Center for Simulation in Rehabilitation Research.

NIH: National Institutes for Health.

Lateral: Relative to or toward the side of the body.

Medial: Relative to or toward the middle of the body.

Mocap: Motion capture.

OCP: Optimal Control Problem.

RMSE: Root-Mean-Square Error.

UPC: Universitat Politècnica de Catalunya

Table of contents

1. Preface

1.1. Origin of the project

The project presented in this report is related to a set of papers concerning the use of optimal control problems (OCP) to create a detailed model of the human gait. These papers have involved the labour of many people representing a broad range of universities and other institutions. Some of the names that have showed up the most in the literature are: Gil Serrancolí, from the Department of Mechanical Engineering, Universitat Politècnica de Catalunya; Antoine Falisse and Friedl De Groote, from the Department of Movement Sciences, KU Leuven; and Joris Gillis, from the Department of Mechanical Engineering, KU Leuven.

In the months leading to the start of this project, researchers at the Simulation and Movement Analysis (SIMMA) Lab at the UPC had been working on applying the previously mentioned simulation of the human gait to a model of a knee joint prosthesis. As will be further explained, the goal of this project is to improve on the initial state of the simulation to make it more precise and efficient.

1.2. Motivation

Initially, as a student of the bachelor's degree in mechanical engineering, I did not care much for biomechanics and probably would not have chosen to do this project had it not been for the Movement Simulation course I took last semester. While taking this class, I was introduced to dynamic and nonlinear optimisation, as well as briefly exposed to CasADi. Possibly because of what may be an unhealthy obsession with precision, the concept of optimising the motion of a system to such an extent appealed to me.

Another relevant aspect of this project that drew me to it was the skills that I anticipated I would need to develop to achieve the goals put forward, such as more complex coding with MATLAB. This would end up being realised, and also would extend to python, which I had only minimal knowledge of.

2. Introduction

As the average age and life expectancy of the world increases, societies are bound to face the growing challenges of an older population. This is especially true in the case of countries that already have high life expectancies and are beginning to feel the demographic impact of the baby boomer generation reaching retirement age. This tendency is likely to influence most aspects of the future functioning of communities around the globe, healthcare being no exception.

There are many conditions and ailments that predominantly affect the elderly, one of which is osteoarthritis. As much as healthier lifestyles and other preventative approaches may mitigate the coming rise in cases of osteoarthritis, the reality is that the prevalence of this disease jumped from 247.5 million cases in 1990 to 527.8 million cases in 2019 and there are no signs of a change in the trend anytime soon [4].

Osteoarthritis is the main cause behind most knee replacement surgeries, as such, this procedure is predicted to be even more commonly performed in the future. According to an article in the Journal of Bone and Joint Surgery, 1.52% of the US population had had a knee replacement surgery by the year 2010, amounting to around seven million people. If the focus is put on people over eighty years of age, the prevalence of this type of surgery is 10.38% [5], suggesting that there is a correlation not only between age and arthritis, but also between age and cases of arthritis requiring this type of treatment.

To deal with some of the consequences of an aging population, new ways of handling this issue will be needed. This may include both radical new ways of tackling this matter, as well as gradual improvements of proven techniques that are being used today. One such way to improve existing strategies involves developing tools to better understand the nuances of the established methods.

The approach chosen to attempt to ameliorate patient outcomes in this assignment is the modelling of the contact forces of a knee joint prosthesis using a model of the human gait. This method also has implications for personalized medicine, as it opens the door to adapting treatments and modifying procedures to individual patients without unreasonably high costs.

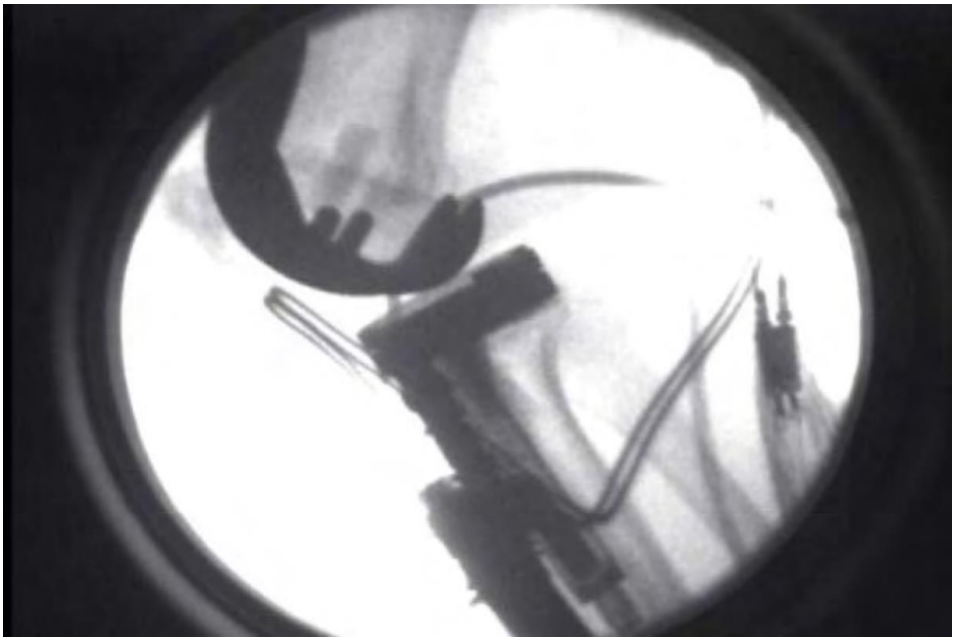

Figure 2. 1. Prosthesis relevant to the analysis, incorporating sensors for the measurement of contact forces [6].

As mentioned in the abstract of the project, the way this is to be done is through the improvement of a program so that it can accurately predict how the human body achieves locomotion and how the signals, forces, torques, and other aspects related to the musculoskeletal system behave during motion. The final goal of the whole endeavour is to accurately predict the distribution of pressures on the surface of the knee joint prosthesis.

2.1. Goals

This project has three main aims, some of which include secondary objectives. The amount that are completed and manner in which they are will serve as the basis for evaluating the results at the end of the report. The goals are:

- To review literature, understand the concepts involved in the formulation of the problem, learn to use the tools needed to solve the problem, including:
    - Automatic differentiation: how to implement it, its strengths, and weaknesses.
    - CasADi: how to build and solve optimal control problems using the MATLAB API.
    - Blender: how to perform the basic operations using the user interface, and how to automate them using python.

- To improve on the original program, with an emphasis on enhancing:
    - Precision, so that the simulation can better predict the real behaviour of the gait cycle.
    - Efficiency, if possible, so that the software can be used in more complex problems involving similar systems. The final program should still be suited to running on a single computer and require a comparable amount of time to solve the problem.
- To build a useful environment for visualizing the results in blender.

2.2. Scope of the project

Given the near-infinite ways in which the task at hand could be made to reach its goals, there are multiple aspects of the assignment that have been deemed off limits.

First, although it is a fundamental part of the statement of the optimal control problem, the contact model has not been modified. This is to avoid an unmanageable degree of complexity, especially when there are already abundant avenues for improvement that do not involve altering a file written in yet another programming language.

Second, the lower half of the body is at the focal point of the study. Therefore, the modelling and discussion of the functioning of the musculoskeletal system is mostly centred around the muscles, bones, tendons, and joints of this section, even though the rest of the body is taken into account in the simulation of the physics of the gait cycle.

The third point to consider is the set of constraints imposed by the available resources. Although the computer provided by SIMMA Lab is modern and capable (boasting a 3.90 GHz Intel Xeon W-2123 CPU and 32 GB of memory), it is still just one desktop computer, and this was kept in mind when testing and developing the program. This is on top of the goal to preserve a reasonable computation time for the final form of the program.

3. State of the art

3.1. Precedents

3.1.1. Early evolution

The human gait, as a fundamental part of the life of all human species and cultures, has been observed since humanity itself has existed. Bipedalism has made possible much of what characterises humans today, such as the ability to carry objects and see over obstacles. However, it has also come at a cost, including arthritis in hips and knees [7].

One of the earliest studies of human locomotion came from 17th century Italy, where physician Giovanni Alfonso Borelli documented the various phases of human motion, showing how the positioning of the centre of gravity along the cycle allows for the body to maintain balance [8].

The next major step materialised with the spread of photography in the late 19th century. This allowed, for the first time, to take precise pictures of a subject at regular intervals (as shown in Figure 3. 1), a practice known as chronophotography [8]. One of the pioneers of this kind of data gathering was Eadweard Muybridge, mainly known for his work *Horse In Motion.*

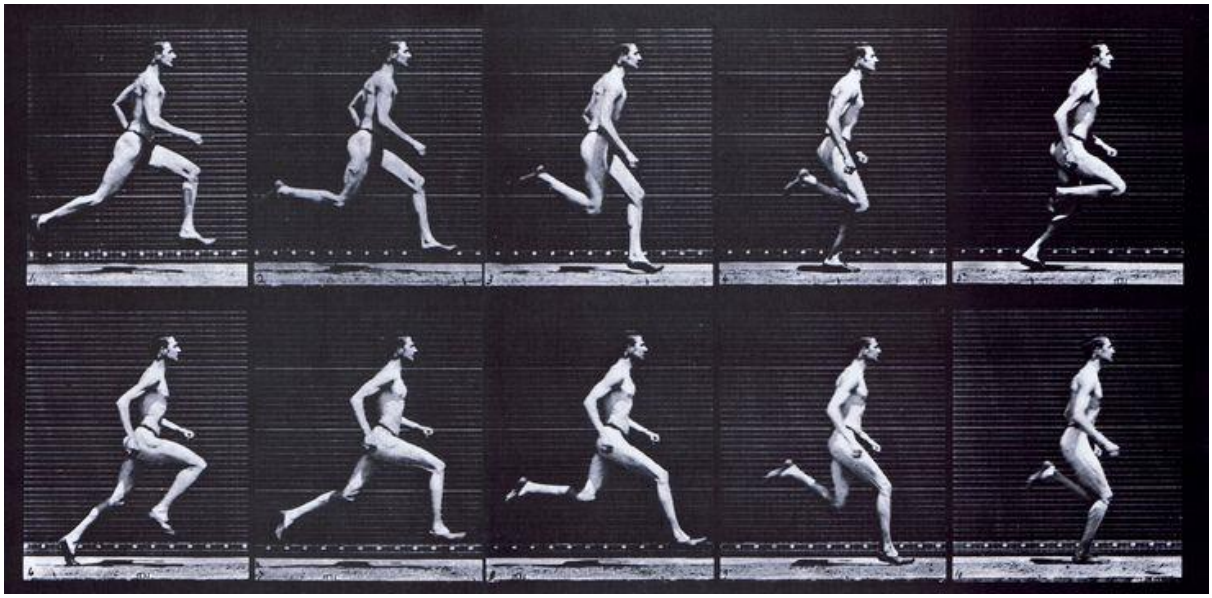

Figure 3. 1. Pictures from Eadweard Muybridge's Animal Locomotion (1887).

Also in the late 19th century, C. W. Braune and O. Fischer built a system of light emitters and cameras that produced three-dimensional data, which was used to successfully determine joint angles during motion (Figure 3. 2) [9].

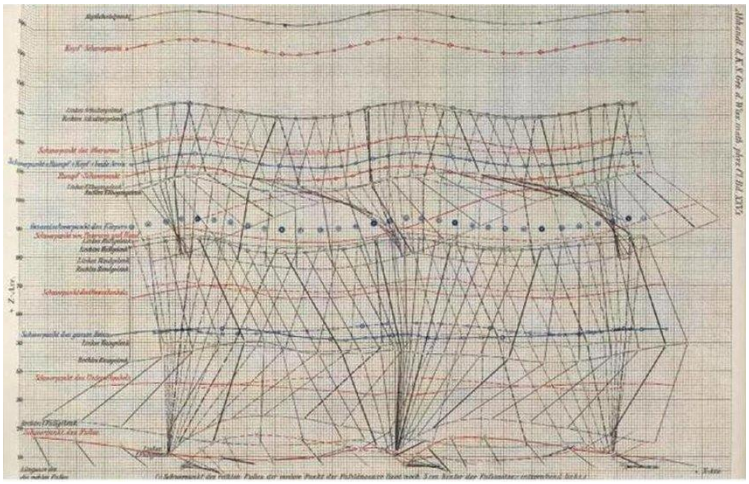

Figure 3. 2. Representation of the data extracted by Braune and Fischer in Der Gang des Menschen (1895).

3.1.2. The age of computational analysis

Advancements in mathematics during the earlier part of the 20th century made it possible to obtain more information from slowly improving data recording systems. Beginning in the 1940's, multiple new inventions made their way into biomechanics labs, most notably: electromyographic sensors, which measure the electric activity in muscles; force plates, which quantify reactions and moments in the ground under the feet of the subject; and accelerometers.

By 1976, SELSPOT, the first system that automatically digitized measurements from images of a test subject walking, was being used in studies in the United States. In the decades that followed, techniques that were easier to use and more precise reached universities around the world. Some of these are VICON (Vicon, Yarnton, United Kingdom), which uses reflective markers; and CODA (Codamotion, Rothley, United Kingdom), which uses LEDs. Versions of both VICON and CODA are still in use as of 2022 [10] [11].

As this technology became more available worldwide, specialized open-source software, such as OpenSim, emerged to answer the necessities of a growing market. The knowledge acquired in the field of biomechanics analysis also spilled over to entertainment as a way of generating realistic 3D graphics involving people, mostly in film and videogames.

3.2. Recent developments

In the last few years, progress in efficient hardware and methods for solving complex problems has opened new avenues for research. Some of these avenues are the use of detailed contact models and the formulation of human locomotion as an optimal control problem with the goal of predicting the behaviour of nerves and muscles.

The contact models used in these papers simulate the human body as a multi-body system of bones and joints that are actuated by muscles. This interpretation of the musculoskeletal system uses state variables to track the condition of the bodies and control variables to influence their state.

One of the first papers to propose the OCP formulation was *Dynamic Optimization of Human Walking*, by F. Anderson and M. Pandy in 2001 [12]. The model put forward in the article assumed that the motion was symmetrical, reducing the degrees of freedom of the body to 23, the number of muscles to 54, and the control variables to 810. The five terms that were minimised were all related to the energy consumed during the gait cycle and produced an average walking speed consistent with previous studies. The problem with Anderson and Pandy's 810-variable program was the computational cost. It took almost 10,000 hours of CPU time to find the solution on a Cray T3E, one of the most advanced supercomputers at the time. This made small scale studies infeasible and limited the applications of the findings.

Another study, *Implicit methods for efficient musculoskeletal simulation and optimal control*, published by van den Bogert et al. in 2011 [13], also used the OCP formulation to predict the gait of simplified human models. Additionally, this analysis utilised a direct collocation scheme which allowed for precise results at a reasonable cost.

The latest set of notable papers combine the OCP approach and the direct collocation method with automatic differentiation to greatly reduce the time and cost of the computation. Complex contact models are also used to increase the accuracy of the results. Some of these articles are: Falisse et al. [14], Serrancolí et al. [15], Falisse et al. [16], Bishop et al. [17], and Serrancolí et al. [18]; all published between 2019 and 2021.

One of the papers most relevant to this project, *Algorithmic differentiation improves the computational efficiency of OpenSim-based trajectory optimization of human movement* [16], published in 2019, achieved an optimisation of the human gait with 61318 variables and 65250 constraints without requiring a supercomputer.

Another of the previous articles, Bishop et al. [17] suggests that a model obtained from the optimisation of the gait of a species of bird (*Eudromia elegans*) can be extrapolated to similar species, including extinct ones. Seeing how the problem is formulated similarly for humans in the other papers, this would indicate that the same methods can be used to predict changes in the manner in which different people walk. This would also open the door to studies taking into account the height, age, possible disabilities, and other characteristics of subjects by adjusting the model to more accurately understand how these attributes influence their movement.

4. Theoretical framework

In august 1964, R. E. Wengert published an article by the name of *A Simple Automatic Derivative Evaluation Program* [19] on the monthly journal Communications of the ACM. In this paper, the author laid out what he called "A procedure for automatic evaluation of total/partial derivatives of arbitrary algebraic functions". He described the technique that would eventually be known as automatic differentiation as a process that simplifies the analysis of complex functions through their decomposition.

A core element of the program, automatic differentiation (AD), also called algorithmic differentiation, is a method of obtaining derivatives of a function by breaking down its basic operations and applying the chain rule. The key advantage of this approach compared to the main alternatives, finite differences (FD) [16] [20] and symbolic differentiation [20], is an important increase in efficiency when used appropriately. Additionally, like symbolic differentiation and deriving by hand, the result that this process produces is exact and does not suffer from the cumulative error that characterises FD.

To understand the basics of this method and how it accomplishes faster computation in suitable applications, it is best to exhibit the sequence of actions required accompanied by an example. The function presented for this demonstration contains three independent x_i variables that are used to obtain *y* as shown in Eq. 4. 1:

$$y = x_1 x_2 - x_2 \cdot cos(x_3) \quad \text{(Eq. 4. 1)}$$

As Wengert defined it, the first step of the method is to separate all variables and individual operations of the equation. Then, a diagram, also known as expression graph, can be drawn to represent the step-by-step process that would be involved in building the equation as can be seen in the Figure 4.1.

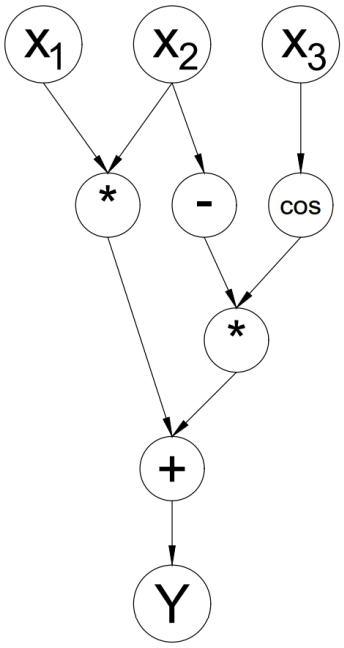

Figure 4. 1. Diagram showing the operations of equation 4. 1.

To keep track of the process, intermediate variables, tagged w_i, are created and assigned to each operation:

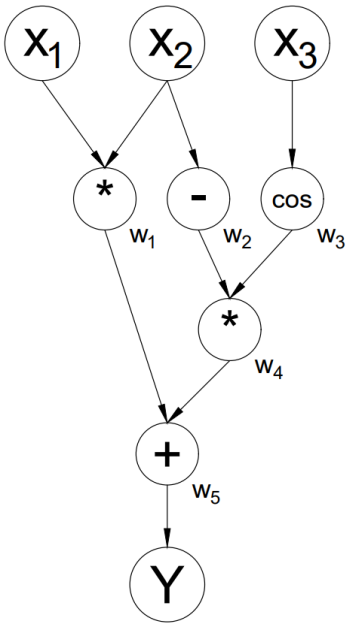

Figure 4. 2. Identity of every intermediate variable.

Intermediate variables are defined in relation to the variables that directly precede them; that is, a given w_i is a function of the other variables which directly feed into its corresponding node in Figure 4.2 as defined in table 4.1. From here, there are two ways of obtaining the partial derivatives of $Y(x_1, x_2, x_3)$: through the use of the forward mode and using the reverse mode.

Table 4. 1. Explicit definition of the intermediate variables.

Variable	Definition
w_1	$x_1 * x_2$
w_2	$-x_2$
w_3	$\cos(x_3)$
w_4	$w_2 * w_3$
w_5	$w_1 + w_4$

4.1. Forward mode

The procedure first proposed by Wengert exploited the chain rule as shown in the following equations:

$$\frac{\partial y}{\partial x_k} = \frac{\partial y}{\partial w} \cdot \frac{\partial w}{\partial x_k} \text{ (Eq. 4. 2)}$$

Where y is the dependent variable, x_k is an independent variable, and w an intermediate variable dependent on x_k. In essence, this is just the fundamental form of the chain rule. To fully exploit the potential of AD, a variable must be created for each operation in the way from x_k to y.

$$\frac{\partial y}{\partial x_k} = \frac{\partial y}{\partial w_N}\frac{\partial w_N}{\partial w_{N-1}} \cdot \ldots \cdot \frac{\partial w_1}{\partial x_k} \text{ (Eq. 4. 3)}$$

With the subindex N being the amount of nodes in the path from *y* to the independent variable of interest in a diagram like the one in Figure 4.1.

The next step is to get the partial derivatives of all intermediate variables with respect to the previous variable. The result of this set of operations is shown in table 4.2. Since the list of possible basic operations is short, the derivatives of these are most often repeated.

Table 4. 2. Derivatives of each of the middle variables.

Variable	Derivative
w_1	$\dot{x}_1 * x_2 + x_1 * \dot{x}_2$
w_2	$-\dot{x}_2$
w_3	$-\sin(x_3) * \dot{x}_3$
w_4	$\dot{w}_2 * w_3 + w_2 * \dot{w}_3$
w_5	$\dot{w}_1 + \dot{w}_4$

Up to this point, only a handful of the simplest derivatives has been laid out. These are combined to produce the value $\dot{y}$. In the forward mode, a single sweep provides the partial derivative of all functions in the system with respect to a chosen variable.

To begin the process, the desired differentiation is defined as:

$$\frac{\partial y}{\partial x_2} = \dot{w_5} \text{ (Eq. 4. 4)}$$

Once the time derivative of the desired independent variable is fixed, the derivatives of each intermediate step can be obtained with the chain rule. The time derivative of the fixed variable, also called seed, takes a value of 1; all other independent variables take 0:

$$\frac{dx_2}{dx_2} = 1 \text{ (Eq. 4. 5)}$$

$$\frac{dx_1}{dx_2} = \frac{dx_3}{dx_2} = 0 \text{ (Eq. 4. 6)}$$

Table 4. 3. Sequence for obtaining the partial derivative with respect to x2.

Derivative	Definition	Complete derivative
$\dot{x}_1$	0	0
$\dot{x}_2$	1	1
$\dot{x}_3$	0	0
$\dot{w}_1$	$\dot{x}_1 * x_2 + x_1 * \dot{x}_2$	x_1
$\dot{w}_2$	$-\dot{x}_2$	-1
$\dot{w}_3$	$-\sin(x_3) * \dot{x}_3$	0
$\dot{w}_4$	$\dot{w}_2 * w_3 + w_2 * \dot{w}_3$	$-\cos(x_3)$
$\dot{w}_5$	$\dot{w}_1 + \dot{w}_4$	$x_1 - \cos(x_3)$

In accordance with Eq. 4.4 and table 4.3, the result is the following:

$$\frac{\partial y}{\partial x_2} = x_1 - cos(x_3) \text{ (Eq. 4. 7)}$$

Through the previous exercise, it has been proven that this method is capable of as much precision as symbolic differentiation. However, little has been accomplished in the way of efficiency; to achieve accurate results without consuming an unreasonable amount of memory [21] a slightly different approach is needed.

If a numerical value is assigned to the independent variables, the exact result of each step can be calculated with a small computing cost. For instance, table 4.4 shows the process of finding the derivative of y with respect to x_2 assuming the value of the former is required at a point defined as x_1=2, x_2=4, x_3=6:

Table 4. 4. Numerical values of every step of the forward sweep.

Variable	Value	Derivative	Value
x_1	2.0000	$\dot{x}_1$	0.0000
x_2	4.0000	$\dot{x}_2$	1.0000
x_3	6.0000	$\dot{x}_3$	0.0000
w_1	8.0000	$\dot{w}_1$	2.0000
w_2	-4.0000	$\dot{w}_2$	-1.0000
w_3	0.9602	$\dot{w}_3$	0.0000
w_4	-3.8407	$\dot{w}_4$	-0.9602
w_5	4.1593	$\dot{w}_5$	1.0398

A small error is introduced every time a result is truncated or rounded. Nevertheless, the error that arises from successive operations can be controlled by preserving a reasonably high number of significant figures.

Note that, since the paths that are followed are those which ultimately can be traced back to the seed, the forward mode will provide the partial derivative with respect to the variable chosen for all functions in the system. The same is not true for the reverse mode.

4.2. Reverse mode

In 1976, more than a decade after Wengert's original paper, the Finnish mathematician Seppo Linnainmaa published *Taylor Expansion of the Accumulated Rounding Error*. This paper created the basis for an alternative approach to algorithmic differentiation [22].

What eventually became known as the reverse mode of automatic differentiation makes use of the adjoints of variables to obtain the value of all partial derivatives of a function at a set point. The adjoint of a variable can be defined as:

$$\overline{w}_i = \frac{\partial f}{\partial w_i} \quad \text{(Eq. 4. 8)}$$

$$\overline{w}_i = \sum \overline{w}_j \frac{\partial w_j}{\partial w_i} \quad \text{(Eq. 4. 9)}$$

Applying the chain rule to equation 4. 8 as described in Eq. 4. 2, equation 4. 9 is obtained: where w_j refers to all the variables directly and immediately dependent on w_i.

To maintain consistency with the previous equations, the variables x_1, x_2, and x_3 can be referred to as w_{-2}, w_{-1}, and w_0.

To begin the sweep, in a similar manner as in the forward mode, a seed is chosen. This time, however, the seed is a function and not an independent variable.

Table 4. 5. Sequence for obtaining all partial derivatives according to the reverse mode.

Adjoint	Definition	Full definition
$\overline{w}_5$	1	1
$\overline{w}_4$	$\overline{w}_5$	1
$\overline{w}_3$	$\overline{w}_4 * w_2$	$-x_2$
$\overline{w}_2$	$\overline{w}_4 * w_3$	$\cos(x_3)$
$\overline{w}_1$	$\overline{w}_5$	1
$\overline{x}_3$ ($\overline{w}_0$)	$-\overline{w}_3 * \sin(w_0)$	$x_2 * \sin(x_3)$
$\overline{x}_2$ ($\overline{w}_{-1}$)	$\overline{w}_1 * w_0 - \overline{w}_2$	$x_1 - \cos(x_3)$
$\overline{x}_1$ ($\overline{w}_{-2}$)	$\overline{w}_1 * w_{-1}$	x_2

Once again, table 4. 5 proves that this mode of AD produces exact partial derivatives of the proposed function. Those are:

$$\overline{x}_1 = \frac{\partial y}{\partial x_1} = x_2 \text{ (Eq. 4. 10)}$$

$$\overline{x}_2 = \frac{\partial y}{\partial x_2} = x_1 - cos(x_3) \text{ (Eq. 4. 11)}$$

$$\overline{x}_3 = \frac{\partial y}{\partial x_3} = x_2 \text{ (Eq. 4. 12)}$$

Note that equation 4. 11, the partial derivative of y with respect to x_2 obtained through reverse AD, is identical to 4. 7, the same derivative produced by forward AD; further confirming the validity of both methods.

To truly exploit this technique's benefits, the numerical value of each step and adjoint is calculated as shown in table 4. 6. Just like in forward AD, this avoids the memory overhead of using a symbolic system.

Table 4. 6. Numerical values of every step of the reverse sweep.

Variable	Value	Adjoint	Value
w_5	4.1593	$\bar{w}_5$	1.0000
w_4	-3.8407	$\bar{w}_4$	1.0000
w_3	0.9602	$\bar{w}_3$	-4.0000
w_2	-4.0000	$\bar{w}_2$	0.9602
w_1	8.0000	$\bar{w}_1$	1.0000
x_3	6.0000	$\bar{w}_0$	-1.1177
x_2	4.0000	$\bar{w}_{-1}$	1.0398
x_1	2.0000	$\bar{w}_{-2}$	4.0000

To recapitulate, the main difference between the two modes of algorithmic differentiation is how the sweep is conducted. The forward mode starts from a variable and provides the value of the partial derivatives of all outputs with respect to that variable, whereas the reverse mode starts from a function and generates the value of every partial derivative of the chosen function. A key aspect of this method is selecting appropriately the kind of sweep, especially when analysing a complex system with large quantity of inputs and outputs.

Generally, the forward sweep will be best suited to systems with far more outputs than inputs, since the number of sweeps will be equal to the number of inputs. The reverse sweep will perform better when the number of inputs is greater (such as evaluating the gradient of a cost function with one output and several design variables), considering that the quantity of sweeps is the same as the quantity of inputs.

4.3. Limitations of automatic differentiation

From what has been presented about AD so far, one might wonder why it is not used more often; it produces exact results and is often faster than methods reliant on finite differences. Despite its advantages, there are barriers to the implementation of this technology.

Non-differentiable equations are not compatible with the most basic versions of AD, such as the one described previously. That said, there are workarounds for most cases [23], such as avoiding the exact points where an equation might be non-differentiable. For functions including conditional expressions, there are also ways to obtain a valid result, such as the ones described in Beck and Fisher (1994) [24] and Baker (1996) [25], although they can come at the cost of a noticeable loss in efficiency. This is a situation where FD is usually better suited, as it does not matter if a function is differentiable when using this method because only the result of the computation is considered.

Another common obstacle is the involvement of external functions. Such functions cannot receive the same treatment as functions whose definitions are accessible to the program. To mitigate this issue, tools like casADi provide the option of using FD to approximate the value of a derivative.

Finally, many applications simply do not require a tool as powerful as AD. For instance, simple problems with very short computation times usually benefit more from symbolic differentiation, with the added benefit that the user can gather more information about the system [26]. In rudimentary uses, the precision and efficiency benefits of algorithmic differentiation might be more than offset by the time and effort involved in implementing the package, especially in cases in which complications such as non-differentiable equations require that the user hand-tune the program.

4.4. Automatic differentiation applied to optimisation

AD is a valuable tool in the mathematical optimisation of large problems because of its previously discussed properties. An optimisation problem is defined by the function to optimise, the variables on which it is dependent, and the constraints imposed on the variables. A basic form of such a problem can be formulated as follows:

$$min\ f(x) \quad \text{(Eq. 4. 13)}$$

Subject to:

$$g_i(x) = 0 \text{ (Eq. 4. 14)}$$

$$h_i(x) \geq 0 \text{ (Eq. 4. 15)}$$

Where *f(x)* is the cost function, x an independent variable, Eq. 4. 14 an equality constraint, and Eq. 4. 15 an inequality constraint.

If an optimisation problem comprises a linear cost function and only linear constraints, the solution can be found easily by the simplex method developed in the late 1940's by George B. Dantzig [27]. In the case of the model studied, the problem includes nonlinear elements in the cost function and constraints; therefore, it is nonlinear programming (NLP) problem. Some NLPs are solved with minimal resources; others present difficulties, such as incorporating non-differentiable functions.

Differentiation plays an important role in the process of solving the problem using gradient-based algorithms. The optimization step is chosen such that in the next iteration design variables decrease the cost function value, while satisfying (or being closer to satisfy) the constraints. The computation of the gradient is crucial to ensure that the cost function value is decreased at the next step, indicating progress toward a minimum. If the function is convex, the minimum to which the process converges will be the global minimum.

If the problem is not convex, gradient-based methods are susceptible to find local minima, which are not necessarily the global minimum. To tackle this, multiple initial guesses can be made to attempt to find a set of minima and pick the best candidate, at the cost of longer computation times.

The problem can be formulated as a lagrangian function to be solved using the method of lagrangian multipliers:

$$\mathcal{L}(x) = f(x) - \lambda^T g_i(x) - \mu^T h_i(x) \text{ (Eq. 4. 16)}$$

Where λ and μ are the aforementioned multipliers. To find the minimum value of *f(x)* that is compatible with the constraints, equation Eq. 4. 16 is derived with respect to each variable involved. Then, all the solutions are found that result in Eq. 4. 17 being fulfilled.

$$\frac{\partial \mathcal{L}}{\partial x} = 0 \text{ (Eq. 4. 17)}$$

A useful tool to deal with issues arising from working with inequality constraints is the interior point method (IPOPT). To implement IPOPT, the inequality constraints $h_i(x)$ of the problem are reformulated as a part of the first order necessary conditions (Karush-Kuhn-Tucker conditions) so that:

$$\nabla f(x) - \tau \sum_{i=1}^{q} \frac{1}{h_i(x)} \nabla h_i(x) = 0 \text{ (Eq. 4. 18)}$$

In the previous equation, q is the amount of active inequality constraints and τ is a positive scalar that is decreased as the root is iterated. As τ tends to 0, the approximation of the result becomes more accurate [28]. This way, Eq. 4. 18 is a smooth function and the problem is solvable finding the root using Newton's method.

Once the solutions of Eq. 4. 17 are found, every point obtained is tested to check if the constraints are satisfied and to find out if they are minima or maxima.

The user can add as many constraints as necessary, while being careful not to create a situation in which there is no solution that can satisfy all constraints. If a point that is consistent with the constraints imposed cannot be found, the problem is considered unfeasible.

A jacobian matrix is used to organize the differentiation of the constraints *c(x)* as shown in Eq. 4. 19:

$$\begin{pmatrix} \frac{\partial c_1(x)}{\partial x_1} & \cdots & \frac{\partial c_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial c_m(x)}{\partial x_1} & \cdots & \frac{\partial c_m(x)}{\partial x_n} \end{pmatrix} \text{ (Eq. 4. 19)}$$

Looking back at the explanation of how each mode of AD propagates through a system, it is easy to see how different jacobians can benefit from forward or reverse modes, depending on their shape.

Generally, a large number of elements of the jacobian hold a value of zero independently of the point that is taken to evaluate the system. This is usually because most constraints are affected by few variables and thus do not possess a nonzero partial derivative for the most of them, resulting in a sparse matrix. Many programs that deal with these matrices, such as CasADi, include algorithms that save time and resources by avoiding the treatment of most of these empty items.

4.5. Optimal Control Problems

CasADi is specialized in optimal control problems such as the one presented in the introduction. An optimal control problem (OCP) is a specific type of optimisation problem that seeks to minimize a cost function across a given time frame. There are multiple methods used to solve OCPs, mainly: single shooting, multiple shooting, and direct collocation; the technique most relevant to this project being the direct collocation method.

Dynamic optimisation is concerned with managing a system using control variables so that the goals of the optimisation are reached. The fundamental assumption of OCPs is laid out in Eq. 4. 20. The system is time-continuous and the derivative of a state ($\dot{x}$) is a function of the state itself (x), time (t), and control (u):

$$\dot{x} = f(x, t, u) \quad \text{(Eq. 4. 20)}$$

Since the OCP is basically an optimisation along a period of time, the is an integral over time, called functional (J):

$$J = \Phi\big(x(t_0), x\big(t_f\big), t_0, t_f\big) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt \quad \text{(Eq. 4. 21)}$$

Here, Φ is a function of the initial and final time and the states of the system at those points. The other term of the functional is the integral of the elements that are to be minimised along the course of the event.

Optimal control problems are defined by multiple kinds of constraints, which can be divided into boundaries and dynamic and path constraints [29]. Boundaries are the constraints that condition the initial and final states and their place in time. They can be characterised as follows:

$$x(t_0)_{lower\ bound} \leq x(t_0) \leq x(t_0)_{upper\ bound} \text{ (Eq. 4. 22)}$$

$$x(t_f)_{lower\ bound} \leq x(t_f) \leq x(t_f)_{upper\ bound} \text{ (Eq. 4. 23)}$$

$$t_{0,\,lower\ bound} \leq t_0 \leq t_{0,\,upper\ bound} \text{ (Eq. 4. 24)}$$

$$t_{f,\,lower\ bound} \leq t_f \leq t_{f,\,upper\ bound} \text{ (Eq. 4. 25)}$$

In problems that require a fixed state or time, the lower and upper bound can be the same and the constraint then becomes an equality constraint. Dynamic constraints are equality constraints that ensure that equation 4. 20 is satisfied:

$$\dot{x}(t) = f(x(t), t, u(t)) \text{ (Eq. 4. 26)}$$

Path constraints may limit the state, control variables or a combination of the former and time and can be generally formulated as:

$$b(x(t), u(t), t) \leq 0 \text{ (Eq. 4. 27)}$$

The direct collocation method discretizes states and controls at instants in time known as collocation points. To achieve continuous derivatives throughout the whole trajectory, states are interpolated using Lagrange polynomials.

5. Original program

5.1. Functional

The specific program (TrackSim_3D_GC.m) that has been modified to meet the goals put forward was created by Gil Serrancolí in 2021 and has been updated since. The original formulation that was used to produce the muscle moment arms, initial guess, and tracking of the kinematics of the skeleton is similar to Falisse et al. 2019 [16]. The main difference is the use of a complex knee contact model, published in Serrancolí et al. 2020 [30].

The subject who provided the data was identified as an 88-year-old male with a weight of 65 kg and a height of 166 cm. The patient was also implanted with a prosthesis in his right knee capable of recording contact forces on the lateral and medial directions. The model includes 94 muscles and 34 degrees of freedom (DoFs). Kinematics, ground reaction forces, knee contact forces and fluoroscopy data were available sourced from Fregly et al. 2012 [31]

The optimal control problem that this program uses to predict the gait of a given subject can be described as the optimisation of a functional subject to a series of constraints. The functional is a combination of eleven terms:

$$J_{Total} = \int_{t_0}^{tf} \sum_{n=1}^{11} j_{nt} \cdot dt \quad \text{(Eq. 5. 1)}$$

Here j_{nt} represents one of the eleven terms of the functional evaluated at a given time t.

$$j_{1t} = W_Q \cdot \left(\frac{1}{I}\sum_{i=1}^{I}\left(\widehat{Qr_{it}} - Qr_{it}\right)^2\right) \quad \text{(Eq. 5. 2)}$$

The first term is the contribution of the squared difference between the experimental and predicted positions of 28 DoFs (all degrees of freedom except the secondary DoFs of the knee). W_Q is a scalar used to adjust the weight given to the term; the value of W_Q is 10 in this case. In a similar manner as

the first, the second term minimises the values of the ground reaction forces recorded by force plates in the lab with the ones estimated. This time, I has a value of 6, since the force is measured in three directions for each foot. The weight assigned to W_{GRF} is 1.

$$j_{2t} = W_{GRF} \cdot \left(\frac{1}{I}\sum_{i=1}^{I}\left(\widehat{GRF_{it}} - GRF_{it}\right)^2\right) \text{ (Eq. 5. 3)}$$

The third term is almost identical to the second in formulation, comparing the moments around 3 axes per foot. The weight assigned to W_{GRM} is 1.

$$j_{3t} = W_{GRM} \cdot \left(\frac{1}{I}\sum_{i=1}^{I}\left(\widehat{GRM_{it}} - GRM_{it}\right)^2\right) \text{ (Eq. 5. 4)}$$

The fourth term is similar to all the terms seen up to this point, but it just minimises the error between the knee contact forces (KCF) produced by the model and the ones captured in the instrumented prosthesis in the right knee of the subject. $KCF_{L\,t}$ refers to the lateral contact force in the knee at the time instant *t*; $KCF_{M\,t}$ references the medial contact force. W_{KCF} is assigned a weight of 10.

$$j_{4t} = W_{KCF} \cdot \left(\frac{(\widehat{KCF_{L\,t}} - KCF_{L\,t})^2 + (\widehat{KCF_{M\,t}} - KCF_{M\,t})^2}{2}\right) \text{ (Eq. 5. 5)}$$

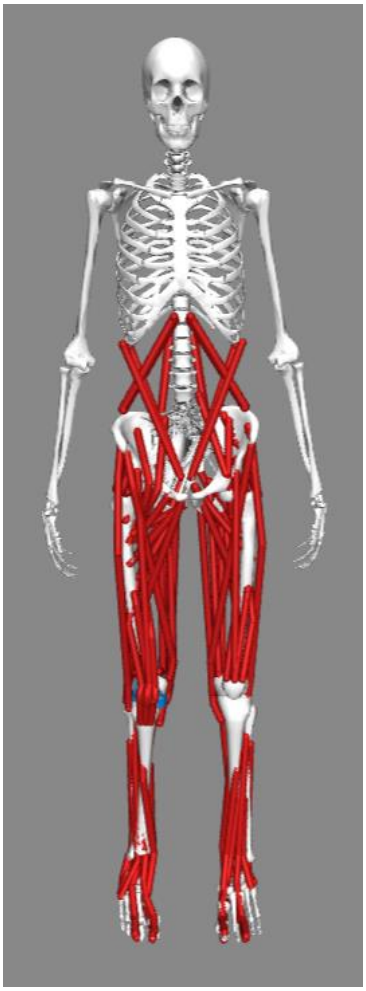

Figure 5. 1. Modelled muscles.

The purpose of the fifth term is to track the inverse dynamics loads of the model to the ones calculated by OpenSim. This is done by comparing the experimental and modelled torques of 22 degrees of freedom, leaving out the pelvis, right knee, and ankles. The weight assigned to W_{ID} is 1.

$$j_{5t} = W_{ID} \cdot \left(\frac{1}{I}\sum_{i=1}^{I}\left(\widehat{T_{ID_{it}}} - T_{ID_{it}}\right)^2\right) \text{ (Eq. 5. 6)}$$

From this term on, the goal is to minimise the value of the variable. The first term of this kind aims to minimise the control variable representative of the activation of each of the 94 modelled muscles. The value of W_a is set at 1.

$$j_{6t} = W_a \cdot \left(\frac{1}{I}\sum_{i=1}^{I}(a_{it})^2\right) \text{ (Eq. 5. 7)}$$

$$j_{7t} = W_u \cdot \left(\frac{1}{I}\sum_{i=1}^{I}\left(\ddot{Q_{it}}\right)^2\right) \text{ (Eq. 5. 8)}$$

$$j_{8t} = W_u \cdot \left(\frac{1}{I}\sum_{i=1}^{I}(\dot{a_{it}})^2\right) \text{ (Eq. 5. 9)}$$

$$j_{9t} = W_u \cdot \left(\frac{1}{I}\sum_{i=1}^{I}\left(\dot{Ft}_{it}\right)^2\right) \text{ (Eq. 5. 10)}$$

The previous three are regularisation terms. They have the primary objective of reducing local infeasibilities and local minima produced by the stiffness of the OCP. j_{7t} minimises the angular acceleration $\ddot{Q}$ of all 34 degrees of freedom. j_{8t} reduces the derivative of the muscle activations with respect to time. j_{9t} decreases the derivative of the tendon forces for each of the 94 muscles. W_u is assigned a value of 0.001 so as to not interfere unnecessarily with the problem.

The last two terms minimise the value of the residuals of the KCF and the joint residual moments (JRM). How these residuals impact the motion will be explained later. Both W_{KCFres} and W_{JRMres} have a value of 1.

$$j_{10t} = W_{KCF_{res}} \cdot \left(\frac{1}{I}\sum_{i=1}^{I}\left(KCF_{res_{it}}\right)^2\right) \text{ (Eq. 5. 11)}$$

$$j_{11t} = W_{JRM_{res}} \cdot \left(\frac{1}{I}\sum_{i=1}^{I}\left(JRM_{res_{it}}\right)^2\right) \text{ (Eq. 5. 12)}$$

All estimators and their experimental counterparts (if involved) are scaled to keep all terms in the same order of magnitude prior to assigning a weight variable to it. For the scaling of the degrees of freedom in Eq. 5. 2, the experimental data is interpolated using a spline. Then, the maximum and minimum value of each DoF is found. The scaled position vector is defined as:

$$Q_i = \frac{Q_{i_{unscaled}}}{max\{|max(Q_i)|, |min(Q_i)|\}} \text{ (Eq. 5. 13)}$$

The result of the previous operation is a set of vectors, one for each DoF, that hold values between 1 and -1.

The scaling for equations Eq. 5. 3, Eq. 5. 4, and Eq. 5. 8 follow the same structure as that of Eq. 5. 2. Eq 5. 5 uses the same scaling as the ground reaction forces.

For equations 5. 6, 5. 9, 5. 10, and 5. 12, instead of finding the scaling factor for each DoF, the torques are simply divided by 150, 100, 100, and 100; respectively. The scaling of the residuals of KCF, Eq. 5. 11, is set up similarly, although each vector has its own assigned value.

Since the muscle activations in Eq. 5. 7 range from 0 to 1 already, they do not need to be scaled.

Finally, the integral with respect to time of each term of the functional is approximated numerically. Most terms will only be evaluated at the collocation points as shown in Eq. 5. 14:

$$J = \int_{t_0}^{t_f} j_t \cdot dt = \sum_{n=1}^{N} j_{t_n} \cdot h \text{ (Eq. 5. 14)}$$

Here N is the number of intervals, the spaces between collocation points. The scalar h is the time step, the length of the interval in seconds; it is the same for all the intervals and obeys the following formula:

$$h = \frac{t_f - t_0}{N} \text{ (Eq. 5. 15)}$$

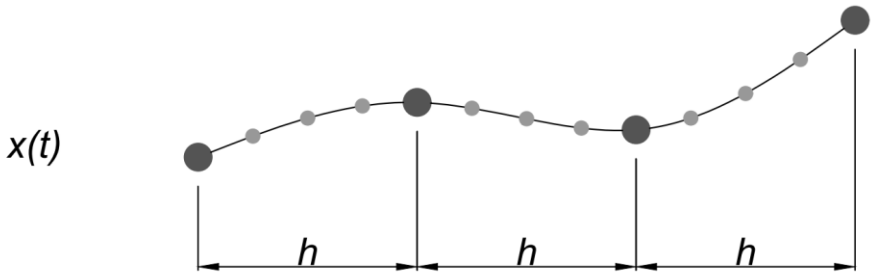

Figure 5. 2. Direct collocation scheme, showing the value of a variable at collocation points in dark grey and interpolated values in lighter grey.

The term that aims to minimise muscle activations, j_6, is the only part of the functional that uses third degree Lagrange polynomials to interpolate between collocation points, being evaluated $4 \cdot N+1$ times (once for every point in figure 5. 2) instead of N times like the others.

5.2. Constraints

This program includes three main types of constraints: dynamic, continuity, and path constraints.

5.2.1. Dynamic constraints

The first type, dynamic constraints, ensures that variables are continuously differentiable. For the derivative of the scaled muscle activations, the dynamic constraints are the following:

$$\dot{a_{it}} - u_{a_{it},\, scaled} = 0 \text{ (Eq. 5. 16)}$$

This constraint also defines the control variable $u_{\dot{a}}$ as the first derivative of the muscle activations and links it to model through a state variable.

The same set of constraints is imposed on other control variables, the derivative of the tendon forces and the angular acceleration of all DoFs.

$$\dot{Ft}_{it} - u_{\dot{Ft}_{it},\, scaled} = 0 \textit{ (Eq. 5. 17)}$$

$$\ddot{Q}_{it} - u_{\ddot{Q}_{it},\, scaled} = 0 \textit{ (Eq. 5. 18)}$$

Additionally, the first order derivative of the activations of the arm muscles are controlled by the arm excitations, another control variable. There is a 35 ms delay between the excitation and the activation to simulate the real behaviour of signals not being transmitted instantly [16].

$$a_{arms} - (e_{arms} - \dot{a}_{arms} \cdot 0.035) = 0 \text{ (Eq. 5. 19)}$$

5.2.2. Continuity constraints

The role of continuity constraints is to ensure that the state variables are continuous by making their value at the end of an interval be equal to that of the beginning of the next.

$$x_{end}^{t} - x_{0}^{t} = 0 \text{ (Eq. 5. 20)}$$

In the previous equality, the same state variable is evaluated at the same instant twice. The first term is the final state of a variable at interval n, while the second term is the state variable at the start of interval n+1.

The state variables that must satisfy this constraint are the angles of the model's DoF and their angular velocities, *Q* and *Q̇*; the tendon forces, *Ft*; the muscle activations, *a*; and the activations of the arms, a_{arms}. Because of their purpose, they are evaluated at every collocation point.

5.2.3. Path constraints

Constraints concerned with limiting the possible values of variables to, for instance, keep an object between the shoulders of a road, are called path constraints. In the case of this problem, one of the ways this kind of constraint is used is to maintain consistency with the physics of the model and aid in convergence.

One way to facilitate convergence is through the introduction of the joint residual moments (JRM_{res}). JRMs are implemented as motors that provide small amounts of torque to compensate for some of the simplifications of the model. Most DoFs have slightly different formulations, the example provided is for the internal rotation of the left hip:

$$T_{ID} - \sum_{n=1}^{NMuscles_{hip,rot,L}}(MA_n \cdot F_n) - T_{pass} - JRM_{res} = 0 \text{ (Eq. 5. 21)}$$

T_{ID} is the torque calculated by the contact model created in OpenSim and imported into MATLAB; the second term is the sum of the torque generated exclusively by muscles, defined as the product of the moment arms and the forces of each muscle involved. T_{pass} accounts for friction and other sources of passive torque with data gathered by Anderson & Pandy [12]. JRM_{res} ensure that there is a margin of error in the previous terms. If they were not used, the other control variables would have to counteract those errors and the accuracy of the model would suffer.

Similarly, all the degrees of freedom of the right knee, except for flexion, use another kind of residual, residual knee contact forces (KCF_{res}).

$$T_{ID} - \sum_{n=1}^{NMuscles}(MA_n \cdot F_n) - T_{pass} - KCF_{res} = 0 \text{ (Eq. 5. 22)}$$

Another use given to path constraints in this program is the modelling of muscle activation dynamics. As muscles are activated, the speed at which muscles are activated or deactivated is affected. The minimum and maximum velocities at which a muscle in this model can be activated are limited by the following constraints:

$$\dot{a} \geq -\frac{a}{0.06} \text{ (Eq. 5. 23)}$$

$$\dot{a} \leq \frac{1-a}{0.015} \text{ (Eq. 5. 24)}$$

By plotting the constraints as done in Figure 5. 3, a simple map of the permitted activation velocities can be built.

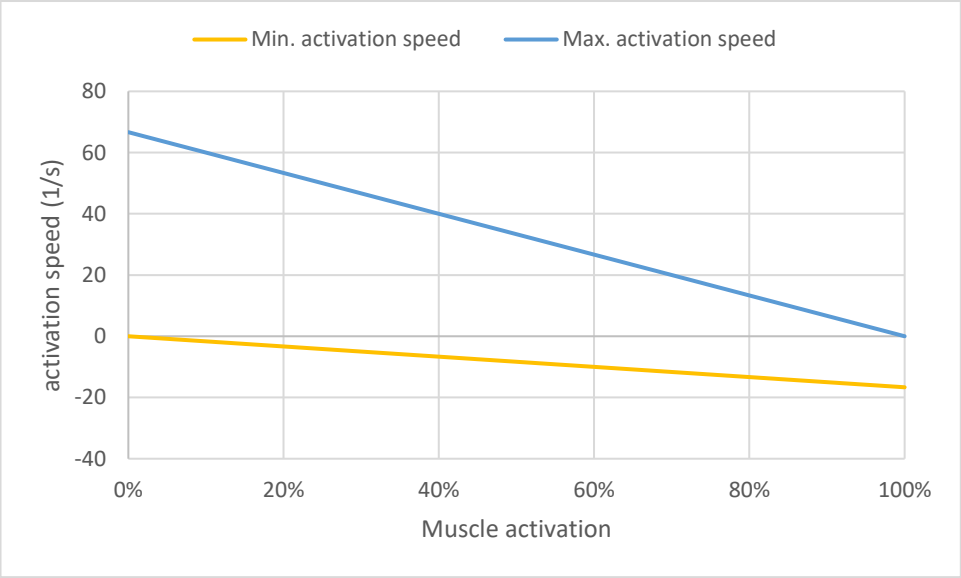

Figure 5. 3. Maximum and minimum speed for the activation/deactivation of a muscle depending on its activation state.

The final path constraint concerns the relation between muscle activations, tendon forces and geometric state of the muscles and tendons using Hill's muscle model as shown in De Groote and Falisse (2021) [32].

$$f_{dHill}\left(a, Ft, \dot{Ft}, L_{muscles,\ tendons}, \dot{L}_{muscles,\ tendons}\right) = 0 \text{ (Eq. 5. 25)}$$

The equality imposes that the force of every muscle is defined by their activation, the tendon forces and their derivatives, and the length and speed at which muscles and tendons are expanding or contracting.

5.2.4. Bounds

There is one more type of constraint, bounds. In CasADi, they are treated differently to other constraints, as they only limit the possible values of one variable at a time. Most of the 65882 variables have some sort of bound, some examples are:

$$0 \leq a \leq 1 \text{ (Eq. 5. 26)}$$

As mentioned earlier, the bounds enforced on the activations of muscles limit their possible values between 0 and 1.

The bounds on most state and control variables, such as the positions of the DoFs and their derivatives, are the extreme values of the interpolated experimental data plus the value of the difference between minimum and maximum in each direction. For example, the bounds for DoFs such as the adduction of the right hip are the following:

$$min(Q_i) - |max(Q_i) - min(Q_i)| \leq \hat{Q}_i \leq max(Q_i) + |max(Q_i) - min(Q_i)| \text{ (Eq. 5. 27)}$$

5.2.5. Implementation of the contact model

One of the features that distinguishes the original program from similar studies is its contact model. Contact models generally are representations of systems comprising multiple bodies that interact with each other through physical touch.

The specific contact model used in this simulation utilises spheres to estimate the forces transferred from one body to the next. As spheres intersect with solids, the penetration determines the magnitude of the force involved. This contact model was created using the OpenSim/Simbody API and Visual Studio.

The model is imported into MATLAB in two functions. The first function, F1, takes the state variables of the problem and outputs the position and velocity of the calcaneus and the toes with respect to the ground. To do so, each foot has six contact spheres that interact with the ground.

The second function, F2, takes the position and velocity of the feet calculated by F1 and the states and controls of the problem to determine the GRFs, GRMs, and joint torques of all DoFs. This formulation is essential to the calculation of the forces internal to the knees, which F2 determines using meshes to estimate the pressures applied on the internal surfaces of the knee.

The model is divided in two functions to accommodate it being imported from a .dll file to MATLAB. If it were not divided, it would need to be called during every iteration of the optimisation.

6. Methodology

6.1. Programs used

During this project, a range of programs and other tools have been used to implement the proposed techniques and advance the goals described in the introduction. They are presented following a rough order from the most central to the least significant.

MATLAB R2020b

According to its own website [33], MATLAB (Mathworks, Natick, MA, United States of America) is a programming platform specialized in the treatment of matrices (as its name suggests) and arrays more broadly. Although this environment was designed with a focus on numeric computation, it also supports symbolic systems. MATLAB is also the name of the programming language that serves as the basis for the software.

MATLAB is a powerful tool that allows for the manipulation of large sets of data with a variety of built-in functions. It also allows for the visualisation of complex problems using 2D and 3D plots. Another of MATLAB's core strengths is the ability to incorporate toolboxes created by either Mathworks or any user [34].

This program was chosen because of its compatibility with CasADi as well as the experience acquired in multiple previous courses.

CasADi v3.5.5

This is an open-source tool that uses automatic differentiation to solve optimisation problems using the methods discussed in the theoretical framework. CasADi is an especially powerful resource in

optimal control problems that incorporate nonlinear elements in the functional and/or constraints of the problem [35]. This tool can be implemented in Python, MATLAB, Octave, and C++.

CasADi was built with engineering in mind, although it is also used in biology and the development of artificial intelligence (most notably machine learning), among others. Work on this project began in 2009, and it has been built mainly by Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl [36].

OpenSim 4.3

This is an open-source software that specialises in biomechanics analysis and simulation [37]. OpenSim has been a necessary part of this project since the inverse dynamics of the model are solved in the OpenSim environment and later submitted to the MATLAB script [38].

This program has also been useful for visualizing the results of tests and assessing the general accuracy of the motion visually.

OpenSim is being actively developed by the National Center for Simulation Rehabilitation Research (NCSRR) at Stanford University with the support of the National Institutes of Health [39] [40].

Blender 3.1.0

Although typically used for creating 3D art in the fields of advertising, film, videogames, and many others [41]; Blender (Blender Foundation, Amsterdam, Netherlands) is also useful to animate and visualize simulations. This software provides an extremely wide assortment of tools, most of which have niche applications at best when it comes to engineering.

Having this quantity of options means that, contrary to the rather austere animations that can be created in MATLAB, the user is less constrained and is able to present their work in a manner more akin to their vision. Additionally, there are many add-ons that allow for even more variety and customisation. Another notable advantage is the included python API that allows for any operation available in the UI to be performed through code [42] .

The reasoning behind the choice of this specific platform is discussed further in chapter 9, which also explains how it has been used.

Github

Github (Github, San Francisco, CA, USA) is an online platform that offers free tools for collaboratively developing software. It is also particularly useful for distributing open-source projects [43]. Github uses Git, a distributed version control software, to decentralise the development process and avoid overwriting previous versions of a project [44].

Some of the characteristic functionalities of Github are commits, pulling and pushing. Commits record the state of an ASCII file and stores it on the local repository. Pulling updates branches in the local repository with new commits. Pushing uploads commits to the remote repository.

Github has been key to updating and keeping updated the files used in this project without losing progress.

6.2. Performance evaluation

An essential part of any program of this kind is the assessmentand validation of results. To do so, special attention was paid to a few aspects of the results and how they were obtained. The main metrics used to guide the modification process have been:

- The **coefficient of determination** between the results of the model and the experimental data, with a focus on the angles of each degree of freedom of the motion and knee contact forces. This measure is sensitive to the similarity of the shape of the data sets and gives a number from 0 to 1; 1 is indicative of the two samples being proportional, although not necessarily close in terms of their absolute value.

- The **root-mean-square error (RMSE)** of the results of the analysis with respect to the experimental data, also with the focus set on the state of each DoF at every collocation point. The RMSE quantifies the average error in a series independently of wether a result is greater or smaller than it was expected to be. In contrast to the coefficient of determination, this tool rates how far, on average, the individual points of two sets are. This is particularly useful if there is an issue with unit conversion or if the results are otherwise similar in shape to what they should be but not in value, as using only the previous metric might paint the picture of the two series being alike when in reality they are very different.

- **Computation time** is also important to determine how well suited to the proposed purposes the program is. As a part of the cost-benefit analysis of the application, the time and resources needed to produce results have to be taken into account. The time required to calculate the results of the unmodified program were set as a benchmark of a reasonable value at around four hours and twenty minutes.

- Another method used was direct **visual examination** of the results. This was done mainly in two ways: using OpenSim to visualize the motion of a model of the human skeleton and visually comparing the graphed series of data that is also evaluated using the first two methods, the coefficient of determination and the RMSE. This is by far the most intuitive technique used to estimate how accurate the preliminary results are. To compare the results of multiple variations of the program comfortably, a script was created in MATLAB.

- Lastly, the **terms of the functional** presented in chapter 5, specifically equations 5. 2 to 5. 6, are useful in locating sources of potential disturbances that might have an impact on most quantifiable attributes of the solution.

6.2.1. Results comparison program

As mentioned in the previous section, a script was created in MATLAB to compare the solutions of multiple configurations of the program. It was made specifically for the analysis of the results of tests

of this project and not designed to be publicly released, which means that functionality was prioritised over user-friendliness.

The comparison script takes multiple Results_3D.mat files produced by the simulation program (Results_3D.mat is the default name, the name of the file can be changed as long as the structure it contains remains unaltered). To keep track of the multitude of files that can be compared at one same time, two structure arrays are created. As shown in Figure 6. 1, one structure array, “Exp”, holds the numerical values of the experimental data; the other, “Model”, contains the results of all the tests being studied.

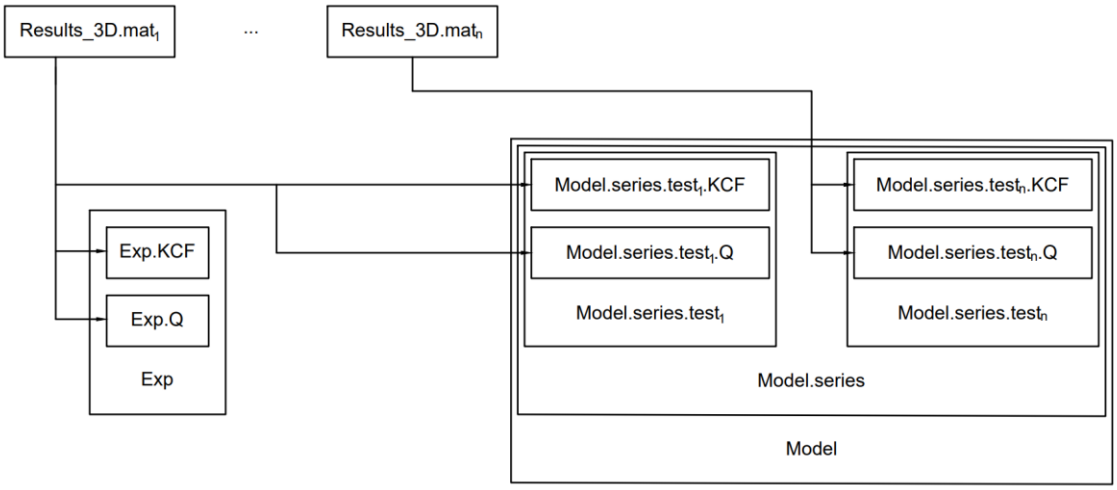

Figure 6. 1. Data storage scheme of the comparison program.

Figure 6. 1 also shows that the results of each test are stored in a field specific to their test, and the field of the test is contained within a field that contains all the tests that the user wants to compare to each other.

To create the plots and calculate the coefficient of determination and the RMSE, the program goes through the tests in each series and plots them in two figures, one for the knee contact forces and another for the positions of the degrees of freedom. As it is building an array of plots for each series, the program also records the coefficient of determination and RMSE of each subplot.

For example, in the case of a series of four tests conducted altering the scaling of the joint residual moments, two figures and four matrices containing the relevant indicators are generated. The studied

series comprises four tests: the first, shown in figure 6.2, keeps the default value of the scaling at 100, the second reduces the value to 10, the third to one, and the last disables JRM_{res} entirely. The two DoFs chosen for the motion analysis presented here are the hip flexion on both legs.

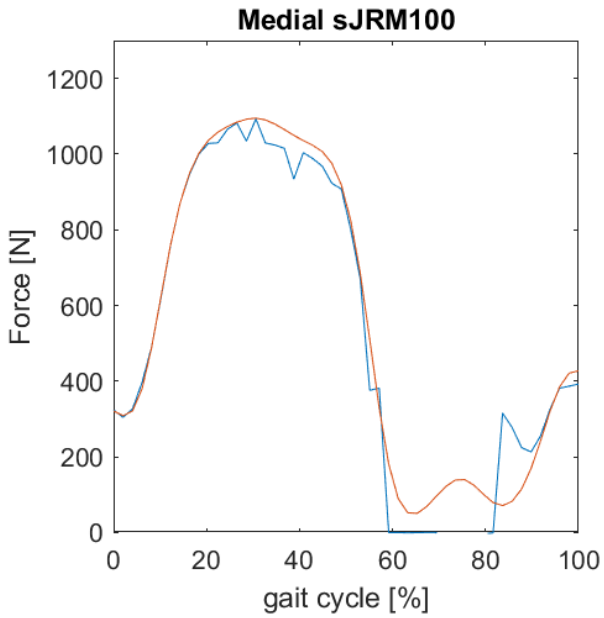

Figure 6. 2. First plot of the series, depicting experimental medial knee contact forces in orange and the model's prediction in blue.

The same plot is created for the rest of the medial and lateral contact forces tests of the series in Figure 6. 3.

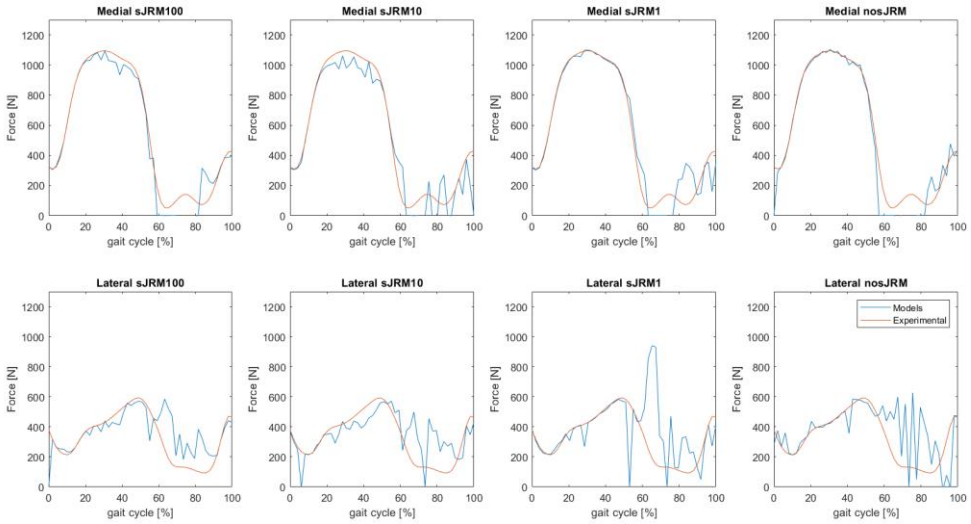

Figure 6. 3. Array of the KCF graphs of the complete series.

As the code went through the data of every test, the coefficient of determination and RMSE of every comparison were also being recorded. The result is shown in tables 6. 1 and 6. 2:

Table 6. 1. Determination coefficient of each KCF comparison of the example.

R^2	Default scaling	Scaling 10	Scaling 1	JRM_{res} disabled
Medial KCF	0.9601	0.9300	0.9405	0.9499
Lateral KCF	0.3968	0.3974	0.0974	0.1992

Table 6. 2. RMSE in N of each KCF comparison in the example.

RMSE (N)	Default scaling	Scaling 10	Scaling 1	JRM_{res} disabled
Medial KCF	88.0594	115.1799	98.3964	112.6329
Lateral KCF	130.0323	127.4393	216.7938	177.5052

The procedure used to obtain the contrast of experimental and predicted knee contact forces is repeated for the position of the degrees of freedom of the flexion of the hips. The result, shown in Figure 6. 3, has the same structure as Figure 6. 2.

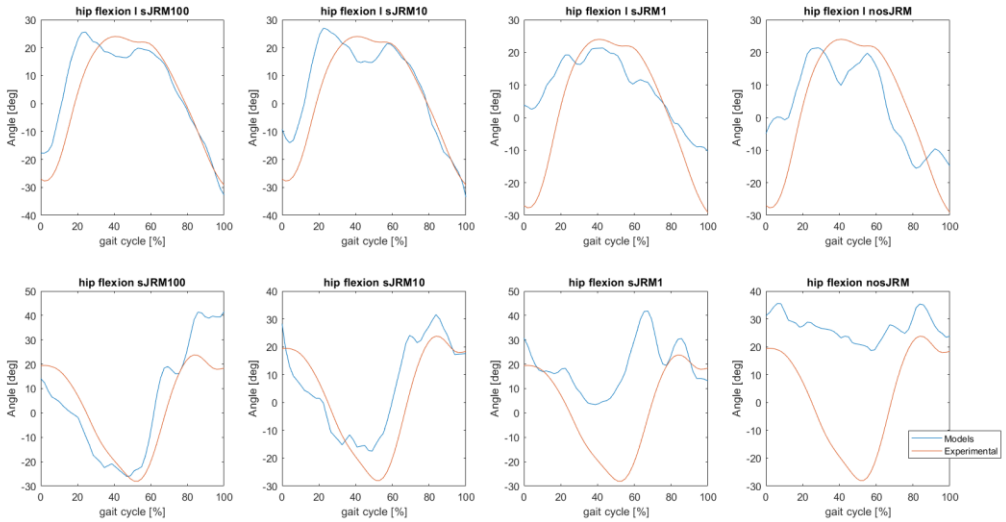

Table 6. 3. Array of the hip flexion position graphs for the complete series.

The determination coefficient and RMSE of the DoF position plots of each of the tests are calculated in the same way as they were for the KCFs. The results are shown in tables 6. 4 and 6. 5.

Table 6. 4. Determination coefficient of each DoF comparison of the example.

R^2	Default scaling	Scaling 10	Scaling 1	JRM_{res} disabled
Hip flexion (Left)	0.8042	0.7741	0.6550	0.5339
Hip flexion (Right)	0.7143	0.6953	0.1342	0.6025

Table 6. 5. RMSE of each DoF comparison of the example.

RMSE	Default scaling	Scaling 10	Scaling 1	JRM_{res} disabled
Hip flexion (Left)	8.7501	9.4787	13.7740	12.9955
Hip flexion (Right)	12.3641	10.7600	24.0998	29.0366

In this example, the reasonable conclusion of the analysis is that reducing the scaling coefficient in this version of the simulation program does not improve the results.

There are multiple advantages to using this system over directly generating the plots and metrics in the MATLAB command window. On the one hand, the process is streamlined, as the only code that needs to be written is the name of the results file. On the other hand, the graphs and data produced are stored so that they can be studied again without having to solve the whole problem again, as it took several hours to compute on most occasions.

7. Modifications to the original program

7.1. Initial assessment

The first step in the process of upgrading the model was the evaluation of the results produced by the default version of the program. The computation time required for the opening set of tests was 4 hours and 19 minutes. Using the tool presented in chapter 6, the results produced by the standard version of the program led to the results shown in Figures 7. 1 and 7. 2.

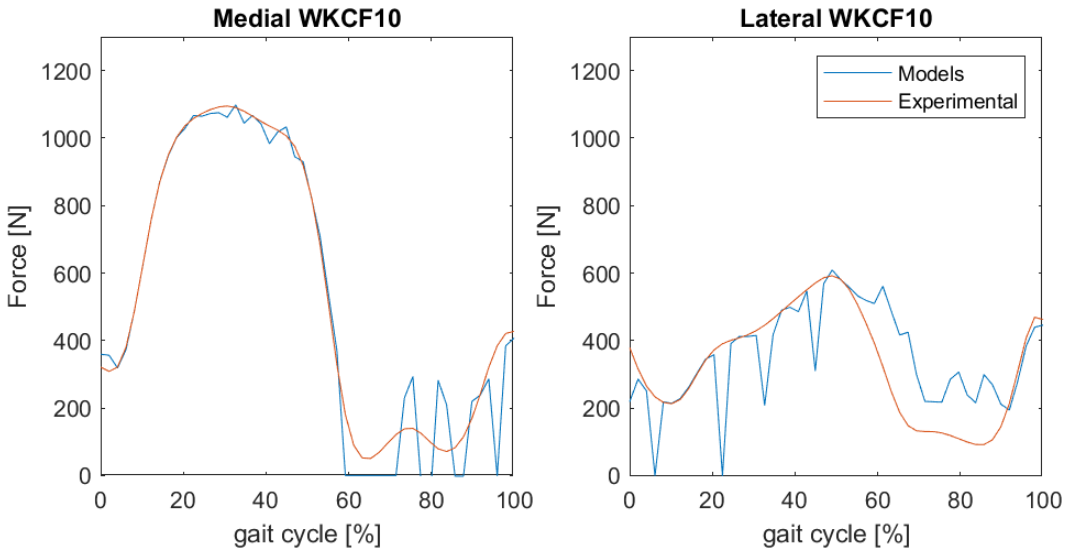

Figure 7. 1. Medial and lateral knee contact forces as calculated by the original program.

The values of the coefficient of determination between experimental and predicted contact forces for the medial and lateral compartments are 0.9558 and 0.4046, respectively. The RMSEs of the predicted medial and lateral forces with respect to the experimental data are 91.4 N and 130.8 N, respectively.

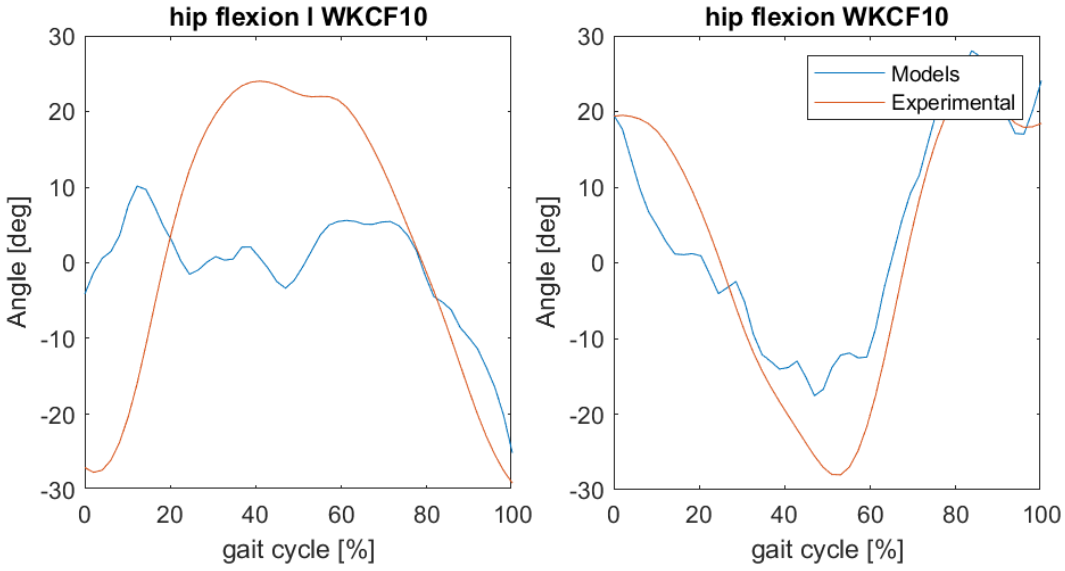

Figure 7. 2. States of the flexion of the left and right hip, respectively, during motion as calculated by the original program.

The analysis of the comparison of the motion reveals less accuracy than the previous test. The determination coefficients are 0.2095 and 0.8560 for the left and right hips, respectively. The RMSEs are 17.2° and 7.5°.

On the one hand, the results of the contact forces are by no means bad, the range of forces that are generated is consistent with the expectations and its distribution across time roughly corresponds to the real phenomenon. However, the modelled lateral contact force (and, to an extent, the modelled medial contact force) presents steep changes that deviate substantially from the anticipated solution. These disturbances have a negative effect on the metrics used to rate the similarity between the estimation and real functioning of the system. Moreover, the lower the value of the force, the more susceptible it seems to be to these interferences.

On the other hand, the graphs of the flexion of the hips show a less-than-accurate picture of the motion. Although the source of the error is not clear, there is an apparent relation with the experimental data and further testing is required to determine the cause of the pattern of the inexactitude of the simulation.

7.2. Determination of the sources of the discrepancy

As discussed in chapter 6, the value of the cost function terms of the optimal control problem can be a useful clue. In the analysis of this iteration of the program, the components of the term function display a clear imbalance; as shown in table 7. 1, term J1 is by far the greatest.

Table 7. 1. Terms of the cost function, default version of the program.

Term	Value
J1	0.980
J2	0.023
J3	0.020
J4	0.077
J5	0.124
J6	0.013
J7	0.000
J8	0.000
J9	0.000
J10	0.000
J11	0.021
Total	1.259

J1 is the term that references the differences between the real and predicted position of the model along the motion. To try to understand the source of the disparity between the expected motion and the one that the program was producing, the constraints and terms of the functional were gradually disabled until reaching a point at which the only remaining terms were the term that was supposed to minimise the difference with the experimental kinematics (J1) and the regularisation terms. The result was a series of tests in which the path constraints introduced in chapter 5 were disregarded and the only significant term was J1, yet the result (shown in Figure 7. 3) was among the worst yet.

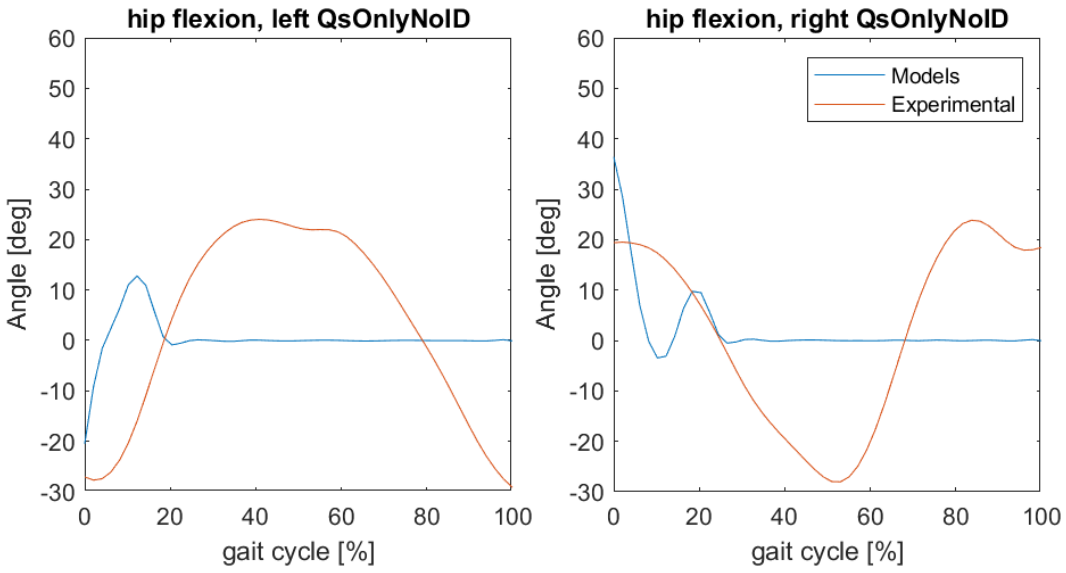

Figure 7. 3. Flexion of the left and right hip without any inverse dynamics constraints and only J1 active in the functional.

The value of R^2 was 2.074e-06 for the left hip and 0.0680 for the right hip; the RMSEs amounted to 19.5° and 17.5°, respectively. This test strongly suggested the existence of an issue in the way the term J1 was implemented in the code. Further testing also showed that, even though the coefficient of determination and RMSE were worse, the internally stored optimal value of the functional had gone down to 0.2665.

The bug was found in the definition of the functional, the cause was the data of the experimental motion not reaching J1 and being substituted by 0 past the first collocation point. After fixing the issue, the result improved substantially, as shown in Figure 7. 4.

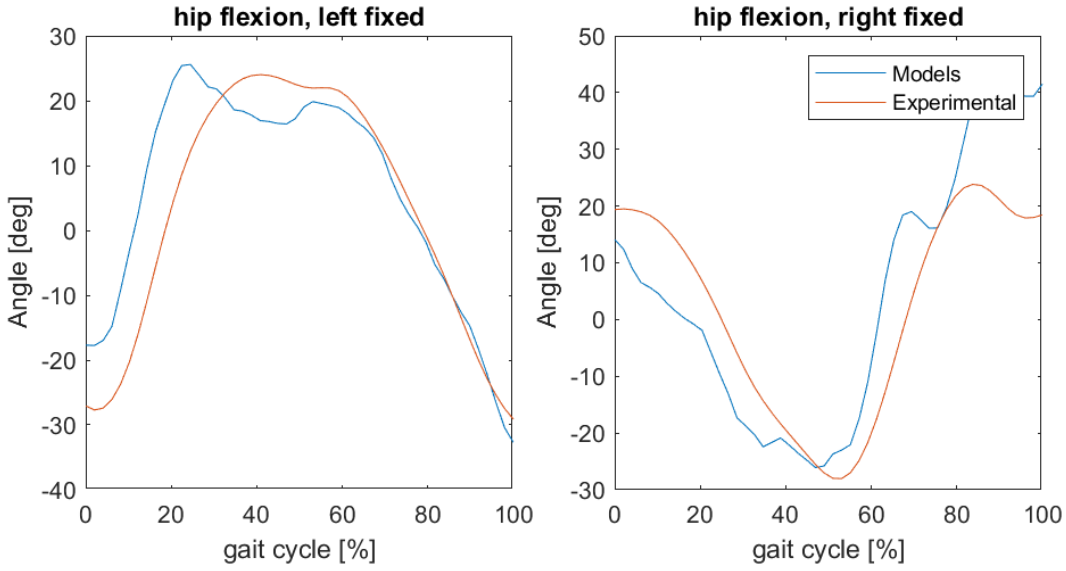

Figure 7. 4. Flexion of the left and right hip after the bug fix.

Even though the coefficient of determination of the right hip has declined slightly, the same metric for the left side has improved remarkably. The new R^2 of the left and right hips are 0.8042 and 0.7143, respectively. The value of the RMSE of the left side has decreased from 17.2° to 8.8°, while the RMSE of the right side has increased from 7.5° to 12.4°. The overall change is positive, but not game changing. The analysis now turns to the knee contact forces presented in Figure 7. 5.

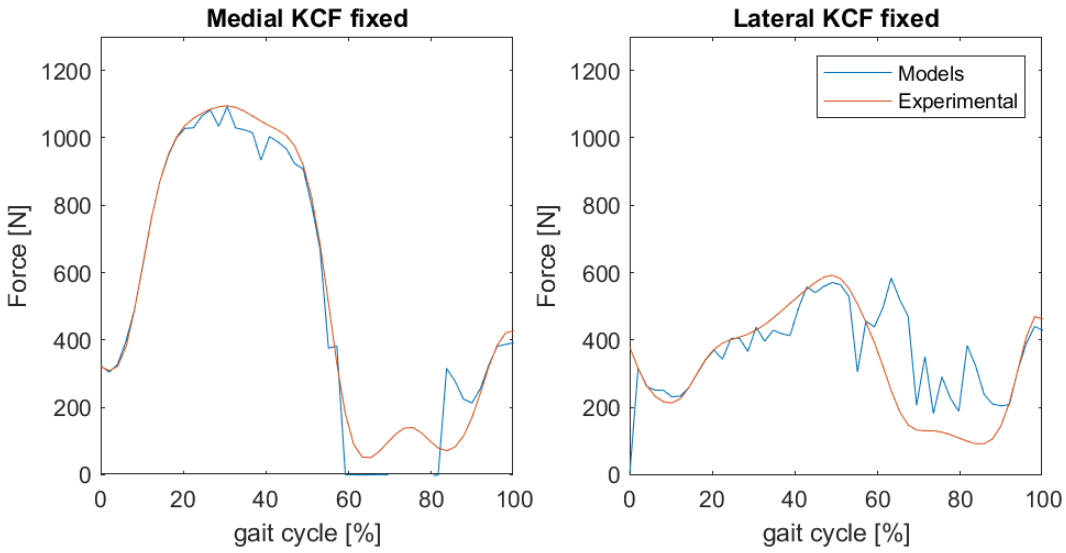

Figure 7. 5. Medial and lateral KCFs of the right knee after the bug fix.

The changes to the coefficient of determination are mixed and not significant; the measurement has very slightly increased for the lateral contact force, to 0.9601. The value of R^2 has seen a similarly small reduction to 0.3968. The fix has done little for the RMSEs of the contact forces as well, the error has decreased a marginal amount for both the medial (88.1 N) and lateral (130.0 N) directions.

7.3. Further refinement of the model

Even if the early tests of the amended version of the program suggest little improvement in some areas, the fix allows for more adjustments. There is an unlimited amount of possible tweaked configurations, and the CPU time required to test each one puts a limit on how many useful combinations can be found. The best two candidates for the most accurate version of the program use different approaches to produce more fitting results.

The first of the pair lifts some of the constraints on the residuals of the knee contact forces by disabling the so-called "KCF constraints as bounds" option. In a practical sense, this widens the range of possible values of the residuals. The other variant that stands out for the validity of its solution increases the weight of the regularisation terms to lessen the stiffness of the optimal control problem. When combining the two strategies, the solver fails to find a solution, converging to a point of local infeasibility. Tables 7. 2 and 7. 3 compare the metrics of both options to the corrected program.

	R^2			RMSE (N)		
	Fixed	**Wu = 0.01**	**KCFcab disabled**	**Fixed**	**Wu = 0.01**	**KCFcab disabled**
Medial	0.960	0.965	0.967	88.1	90.5	84.7
Lateral	0.397	0.547	0.921	130.0	112.1	48.4
Average	0.678	0.756	0.944	109.0	101.3	66.6

Table 7. 2. Comparison of the simulation of the KCFs by the previously discussed candidates.

	R^2			RMSE (°)		
	Fixed	**Wu = 0.01**	**KCFcab disabled**	**Fixed**	**Wu = 0.01**	**KCFcab disabled**
Hip rotation, left	0.8042	0.7264	0.8246	8.8	10.5	8.6
Hip rotation, right	0.7143	0.8889	0.8855	12.4	6.6	6.4
Average	0.7593	0.8077	0.8551	10.6	8.6	7.5

Table 7. 3. Comparison of the simulation of the rotation of the hips by the previously discussed candidates.

Of the three candidates, the option that disables the "KCF constraints as bounds" comes out on top on every single test. The second candidate, increasing the weight of the regularisation terms, appears to be somewhere in between the other two on all tests. Although the variant that lifts restrictions on the KCF residuals is chosen as the final iteration of the program, the runner-up is promising and probably could be adapted to upgrade the program further.

8. Results of the final program

After the changes made, the best-performing version of the software was chosen, and its solution was submitted to the program introduced in chapter 6.

The specific version that has led to results most similar to the experimental data was the one where the residual knee contact forces had more freedom. Subjecting this build to the same initial assessment as the first iteration of the program in section 7. 1, the indicators of interest are determined to have substantially improved. This can be clearly seen on Figure 8. 1.

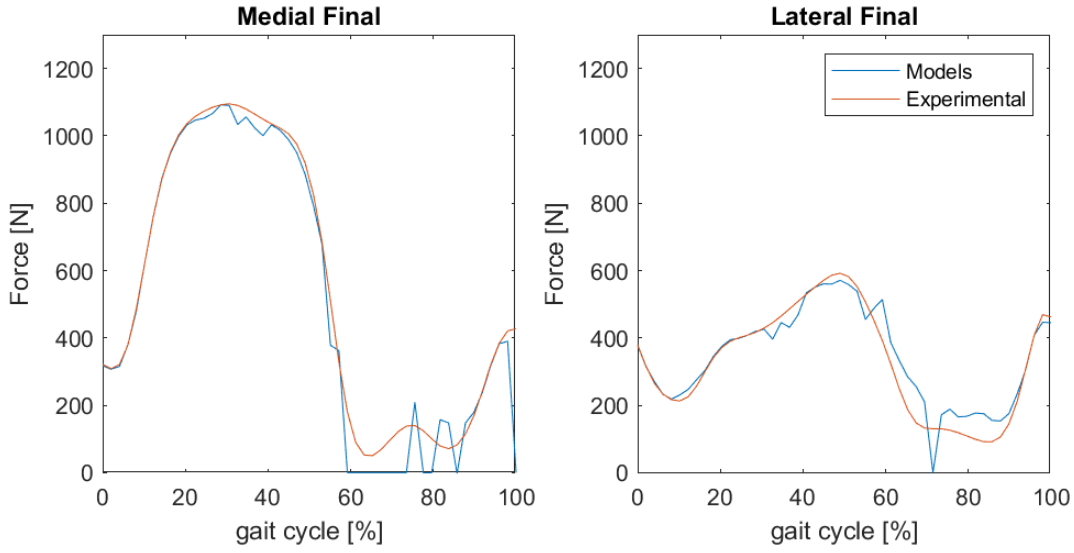

Figure 8. 1. Medial and lateral knee contact forces as calculated by the final modification of the program.

In the case of the knee contact forces, the determination coefficient between the prediction and the experimental data is 0.9673 for the medial forces and 0.9214 for the lateral forces. This is a considerable increase over the initial values of 0.9558 and 0.4046, mainly as it relates to the lateral contact forces. The RMSEs of both contact forces also have greatly decreased, from 91.4 N to 84.7 N (a 7.3% reduction) for the medial KCF and from 130.8 N to 48.4 N (a 63.0% drop) for the lateral KCF.

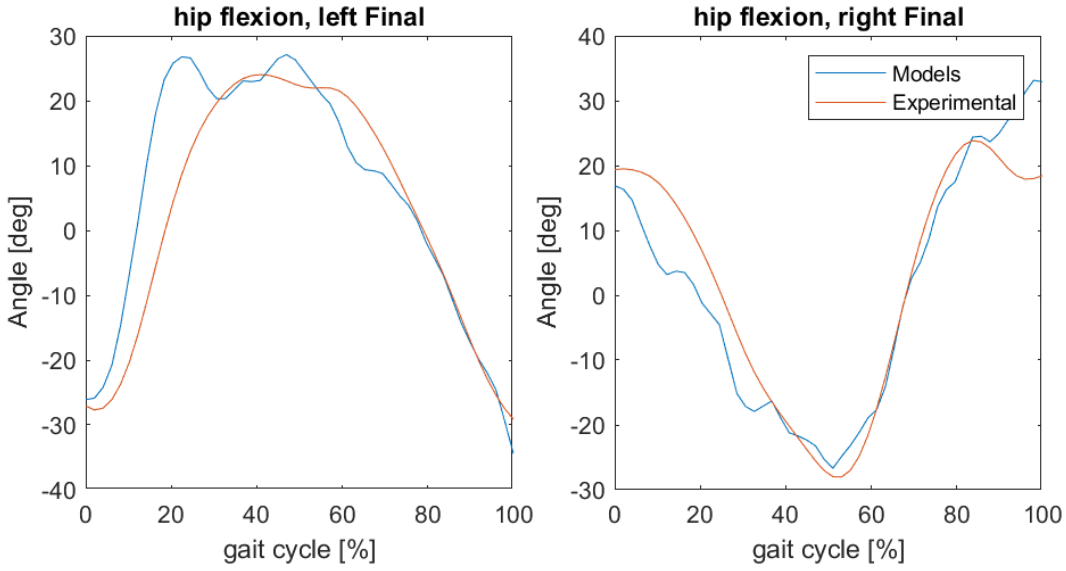

Figure 8. 2. Left and right hip flexion angles during motion as calculated by final modification of the program.

The two indicators for the kinematics of the model also show a marked improvement. The coefficient of determination of the flexion of the left hip has increased from 0.2095 to 0.8246; the same value for the right hip has also increased, from 0.8560 to 0.8855. The RMSEs of the selected degrees of freedom also show a considerable decrease, going from 17.2° to 8.6° (a 50.0% reduction) in the case of the rotation of the left hip and 7.5° to 6.4° (a 14.7% reduction) in the case of the right one.

A similar exploratory analysis of the other predicted characteristics of the motion indicates modest but noticeable improvements for the ground reaction forces, suggesting a broad (even if limited) improvement beyond the attributes that have been at the centre of the project. The first of these secondary items to be compared to the experimental benchmark are the ground reaction forces on the right foot (Figure 8. 3).

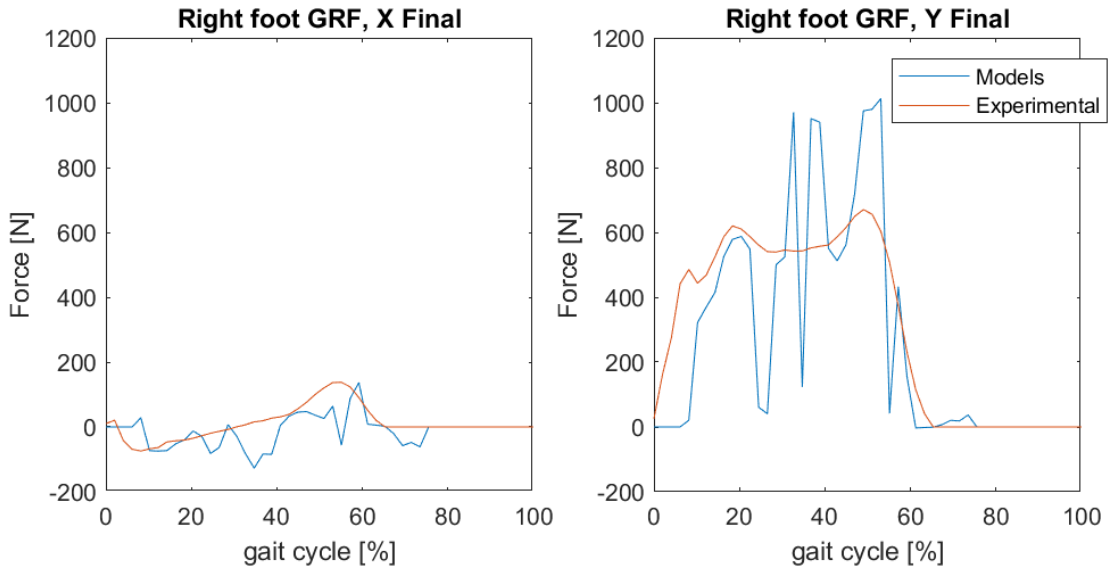

Figure 8. 3. Ground reaction forces endured by the right foot along the X and Y axes as calculated by the final modification of the program.

For the ground reaction forces on the right foot, the coefficient of determination has gone from 0.2194 to 0.2324 in the case of the force in the direction of the general motion of the body (X), the RMSE associated to the prediction has decreased from 66.8 N to 54.9 N (a 17.8% reduction). The coefficient of determination of the vertical reaction forces (Y) suffered by the same foot displays a similarly trivial increase, from 0.5908 to 0.6090; the RMSE, has similarly decreased from 238.6 N to 216.67 N (a 9.2% improvement in accuracy over the initial predictions).

The same analysis is extended to the ground reaction moments of the right foot, as can be seen in Figure 8. 4. This time, the moments around the X and Z axes have been picked, as their role in the gait cycle is more relevant and the values are expected to be overall higher and less susceptible to minor disturbances.

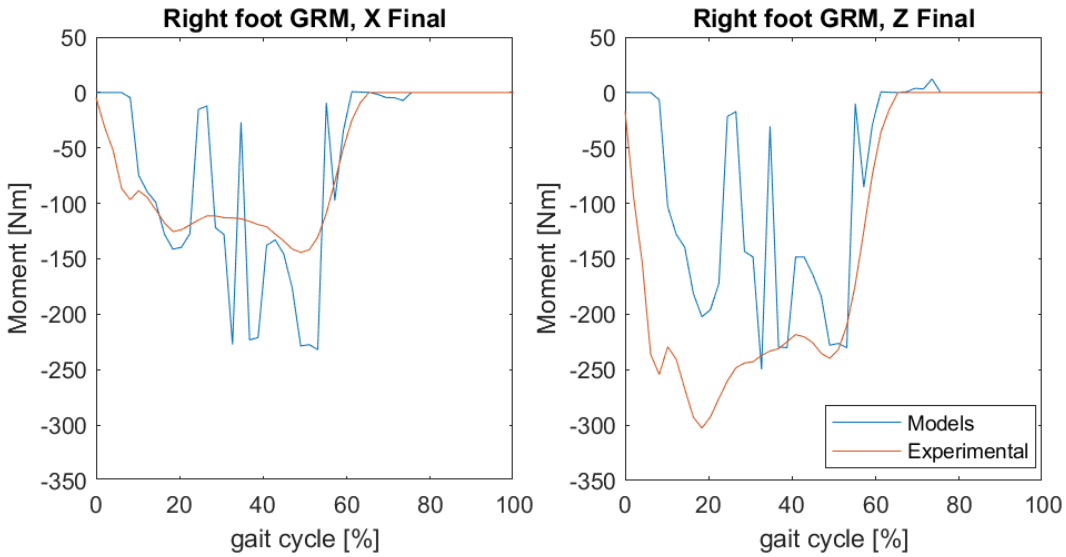

Figure 8. 4. Ground reaction moments endured by the right foot along the X and Z axes as calculated by the final modification of the program.

As a result of the changes in the code, the value of the coefficient of determination of the ground reaction moments around the X axis has gone up from 0.5995 to 0.6436; the same metric in the case of the Z axis has seen a tiny increase from 0.5751 to 0.5842. The RMSE of the new prediction is slightly worse in the case of the moment around the X axis; the value of the measure has increased from 43.6 Nm to 49.2 Nm (a 12.8% growth). This is not the case for the moment around the Z axis, as its RMSE has decreased from 123.3 N to 95.6 N (a 22.5% reduction).

It is to be expected that the terms with less weight on the functional will benefit less from the bug fixing and other changes aimed at enhancing the more important parts of the solution. This explains why the improvements in the fit of KCFs and the position of DoFs do not translate as well to the GRFs and GRMs.

Another issue that has been solved is the slipping or dragging motion of the feet during the simulated event. A series of frames of the original animated prediction of the gait are presented in figure 8. 5. In these images, which are 100 ms apart, both the left and right foot advance simultaneously. In fact, humans move one foot at a time and use the other to propel themselves using the static friction of the non-moving foot with the ground.

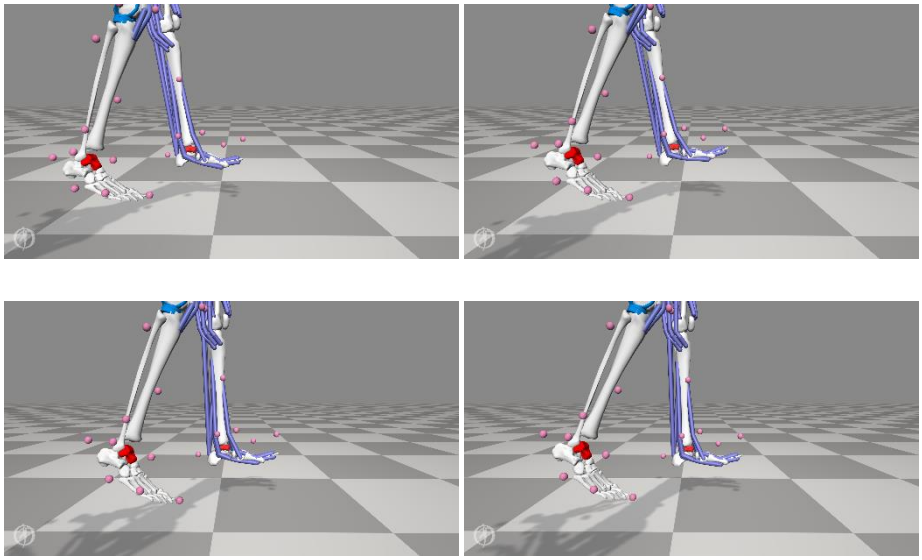

Figure 8. 5. Motion from t=1.850s to t=2.150s according to the original program.

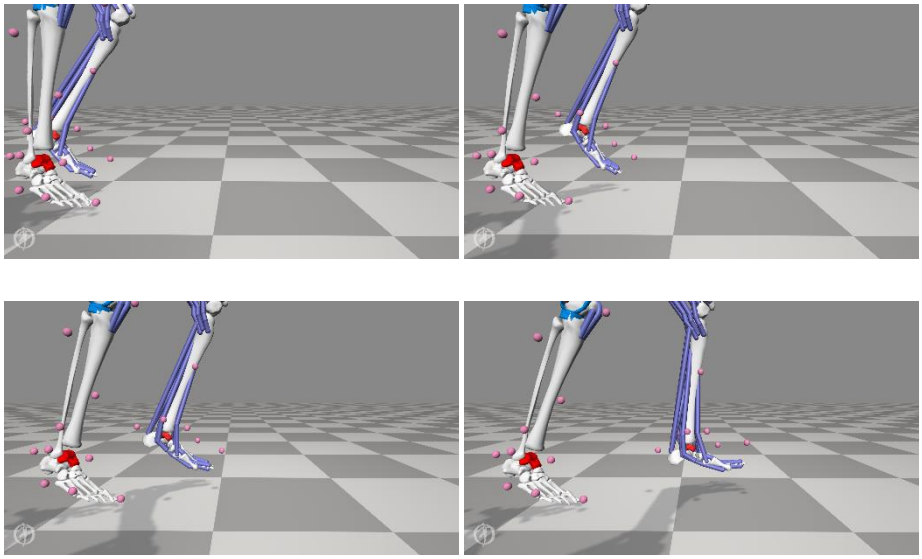

Figure 8. 6. Motion from t=1.850s to t=2.150s according to the modified program.

Figure 8. 6, however, shows the solution found by the final version of the program, in which the right foot is used to thrust the rest of the body forward as the left leg prepares to change roles. Figures 8. 5 and 8. 6 show the same interval of time.

To prove that the improvement is not only limited to selected joints, the coefficient of determination and RMSE for each degree of freedom that was tracked in the laboratory has been calculated and is displayed in table 8. 1.

Table 8. 1. Comparison of the change in R^2 and RMSE for every degree of freedom of the model.

	R^2		RMSE (°)	
	Default	**Improved**	**Default**	**Improved**
Pelvis tilt	0.010	0.814	5.72	0.51
Pelvis list	0.042	0.982	3.41	0.29
Pelvis rotation	0.138	0.984	4.69	0.35
Pelvis translation along X	0.858	0.982	24.45	24.35
Pelvis translation along Y	0.000	0.022	54.81	54.82
Pelvis translation along Z	0.053	0.779	7.36	7.27
Hip flexion, left	0.209	0.825	17.16	8.56
Hip adduction, left	0.003	0.975	6.27	0.74
Hip internal rotation, left	0.001	0.960	6.56	1.16
Hip flexion, right	0.856	0.886	7.51	6.45
Hip adduction, right	0.526	0.963	9.77	1.67
Hip internal rotation, right	0.008	0.900	8.64	1.36
Knee flexion, left	0.339	0.667	34.49	12.58
Knee flexion, right	0.571	0.605	18.92	19.80
Knee translation along Y, right	0.003	0.000	2.36	2.36
Ankle flexion, left	0.243	0.498	10.61	6.23
Ankle flexion, right	0.245	0.168	20.62	13.43
Subtalar, left	0.000	0.966	8.48	0.89
Subtalar, right	0.002	0.877	8.88	1.19
Trunk extension	0.000	0.425	11.17	1.02
Trunk bending	0.200	0.978	4.60	0.29
Trunk rotation	0.021	0.989	1.61	0.13
Shoulder flexion, left	0.028	0.993	19.04	0.58
Shoulder adduction, left	0.067	0.997	19.71	0.16
Shoulder internal rotation, left	0.014	0.997	35.95	0.37
Shoulder flexion, right	0.093	0.914	15.47	0.68
Shoulder adduction, right	0.100	0.996	26.47	0.11
Shoulder internal rotation, right	0.000	0.933	46.40	0.54
Elbow flexion, left	0.003	1.000	23.31	0.10
Elbow flexion, right	0.027	0.998	27.10	0.10
Average	0.155	0.802	16.38	5.60

The same treatment has been given to the knee contact forces in table 8. 2.

Table 8. 2. Comparison of the change in R^2 and RMSE of the medial and lateral knee contact forces.

	R^2		RMSE (N)	
	Default	**Improved**	**Default**	**Improved**
Medial	0.956	0.967	91.4	84.7
Lateral	0.405	0.921	130.9	48.4
Average	0.680	0.944	111.2	66.6

As seen in tables 8. 1 and 8. 2, the model has become more accurate in the predictions of both knee contact forces and kinematics. The average RMSE of the position of the DoFs along the cycle has decreased by 10.78° (a 65.8% diminishment); in the prediction of KCFs, the difference is 44.6 N, a 40.1% reduction.

To sum up, from the information extracted from this analysis it is reasonable to state that most indicators have improved. That is not to say that the results could not be better; they most certainly could, but that would most likely involve fine tuning the weight given to the individual terms of the functional, as will be discussed in the conclusion. It is also noteworthy that the total computation time of the final simulation is 7 hours, 18 minutes, and 30 seconds.

9. Pressure visualisation tool

To prove that the techniques discussed up to this point are valid and can produce useful data, a simple program has been written to automatically animate the movement and pressure distribution of the tibial portion of a knee joint prosthesis.

This program takes the .stl files of the surfaces whose pressure distribution is being studied and three .csv files containing all the necessary information to create the motion and pressure map of the piece. The script has been written in python to run on the Blender API.

A modified version of the MATLAB visualiser used in Serrancolí et. al [18] is employed to condition the Results_3D.mat file generated by TrackSim_3D_GC.m. The output of the modified visualiser is the following .csv files:

- **pressure2blender.csv**: This file is an m-by-n matrix containing the numerical value of the pressure applied to each face (n) at every time interval (m).
- **statsblender.csv**: A file containing miscellaneous information, such as the maximum and minimum pressures in pressure2blender.csv.
- **kin2btransposed.csv**: This file is a m-by-6 matrix holding the rotation and position coordinates of the .stl part.

The first step is the retrieval and reading of the csv files. To do so, a function called Csv2list was created. **Csv2list** simply takes a .csv file and converts it into a matrix that can be called and read with python.

Then the matrices created from pressure2blender.csv and statsblender.csv are fed to the function **colormap**, which converts the pressure values in the pressure matrix to a number between 0 and 1. 1 indicating the maximum pressure and 0 indicating no pressure.

When the preceding actions have finished, the script loops over each face and executes 3 functions:

- **CreateMaterial**: Generate a new material and assign it to the current face.
- **CreateNodes**: Prepare the current material to receive the input of the colour map by generating all the necessary nodes.
- **CreateKeyFrames**: Create all the keyframes, the points in time at which the colour of the current face must change. The rest of the time, the colour is interpolated from the two closest keyframes.

Once the script has gone over all faces in the object, the node structure of each material should be as shown in Figure 9. 1.

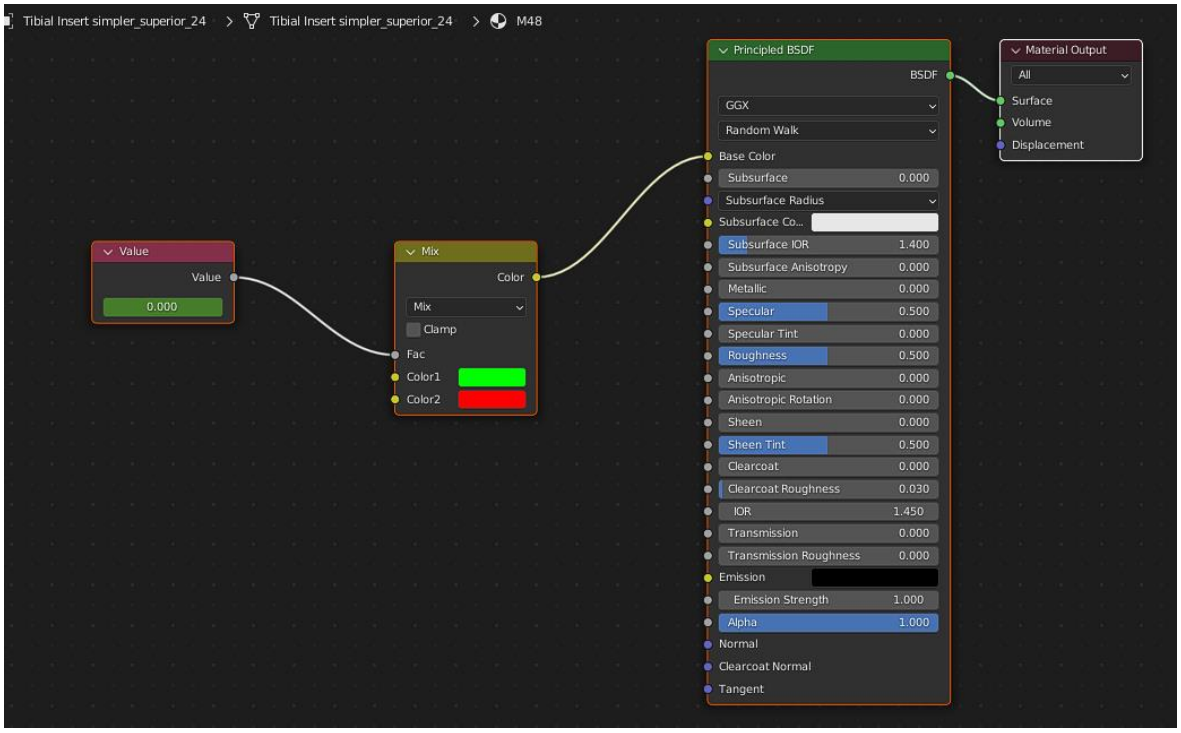

Figure 9. 1. Node structure of a randomly selected material.

Note the “value” node; this is the input that links the material to the colour map.

The process for animating the motion of the object is simpler, the matrix containing the information from kin2btransposed.csv is fed to the function **MoveAnim** and keyframes are assigned to the different positions in the matrix. When finished, Blender should display the keyframes for the motion as shown in Figure 9. 2, where every dot is a keyframe with a defined position.

Figure 9. 2. Keyframes for animating the motion of the object

This is all done by just executing the python script and requires no previous knowledge in Blender or python. The user should something similar to figure 9. 3:

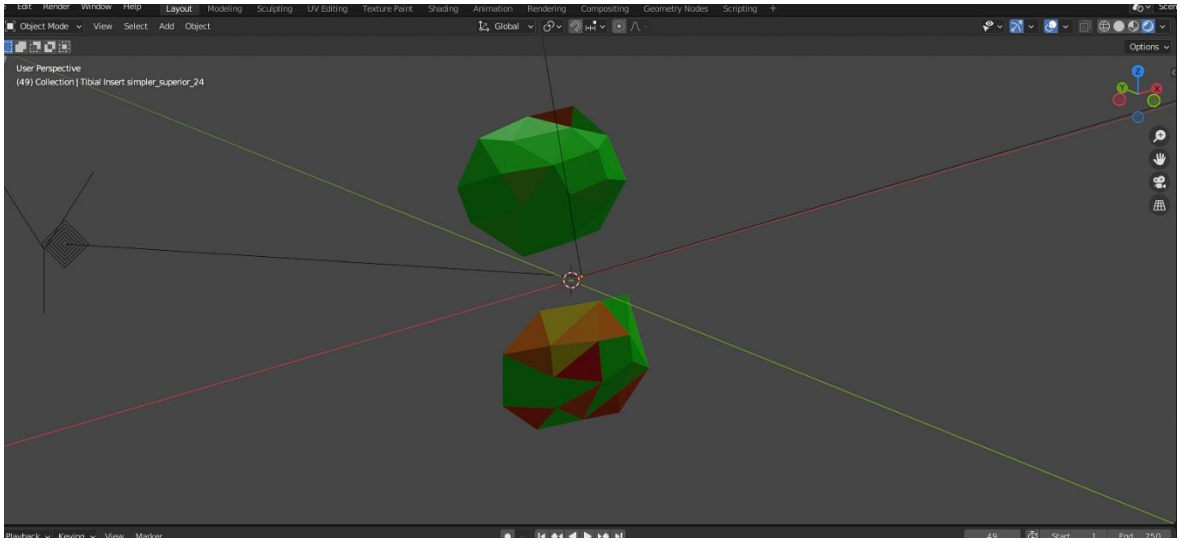

Figure 9. 3. Still frame of the expected result, showing a variety of pressures indicated by different shades of a green-red spectrum, the static femoral component has been removed for clarity.

Environmental impact

Since there is no physical product associated with this project, the greatest potential source of harm for the environment related to this report is the electricity used to compute lengthy simulations. As necessary as they have been, this type of operations require a computer to remain turned on and engaging in resource-intensive tasks for hours at a time.

Considering the fact that the component doing most of the work to solve the problem is the CPU, the energy consumption is estimated as the maximum heat this part can dissipate. For the Intel Xeon W-2123, that is 120 W [45].

Adding up the CPU time of the 50 recorded tests, the total time that the computer at SIMMA Lab has been optimising the problem is 174 hours, 26 minutes, and 40 seconds. This translates to a total energy expenditure of 20.93 kWh. According to data from the Generalitat de Catalunya [46], the average amount of CO_2eq released per kWh in early 2022 was 259g. With a simple conversion of units, the total CO2 equivalent emissions is calculated:

$$20.93\ kWh \cdot \frac{259\ g\ CO_2eq}{1\ kWh} = 5420.87\ g\ CO_2eq$$

That is, approximately 5.42 kg of CO_2 equivalent has been incorporated into the atmosphere as a result of this project.

But that is not all, if the modified program was to be released publicly and its use prevented visits to the doctor or successive surgeries, this technology could contribute to reducing the carbon footprint. The cost of running the final program was 0.877 kWh, which implies a release of 227 g CO_2eq, the same as driving 1790 meters in a 2022 Volkswagen Golf 8 [47].

Conclusions

In order to critically discuss the implications of this project, it is proper to first look back on the goals set at the beginning of this document to appraise the degree to which they have been achieved.

Assessment of the initial goals

1. To review literature, understand the concepts involved in the formulation of the problem, learn to use the tools needed to solve the problem.

As a prerequisite to the rest of tasks of the project, the research involved has been quite extensive because of the number of areas of expertise involved in the problem. On the flip side, most of the tools described in this report have the potential to be used to overcome challenges very different to the ones outlined in this document. Open-source software, such as OpenSim and CasADi, and mathematical implements, such as algorithmic differentiation and optimal control problems, have the potential to solve extremely complex issues in an innumerable quantity of applications.

2. To improve on the original program.

The main goal of the project, to develop the initial program into a more accurate version, has been attained. This statement is supported by the reduction of the RMSE of the knee contact forces by 40.1% to 66.6 N, and the average decrease in the RMSE of the position of the degrees of freedom by 65.8% to 5.6°.

Nevertheless, these gains have come at a cost. Chief among the concessions made is the increase in the computation time, which has grown by 69.5% and reached the mark of 7 hours, 18 and a half minutes. Another compromise that was necessary was the liberalisation of the bounds on knee contact force residuals, which aim to compensate for disparities between the simulated conditions and the real-world circumstances and aid the convergence of the solution.

Ultimately, it will be the user of this technology that will have to decide if what is given up with the changes made is worth the advantages they bring. At least in the case of the extended CPU time requirements, there is a strong case to be made that its effects will not be too bothersome. The time it takes to solve the problem still allows for the program to be run overnight in a common personal computer.

3. To build a useful environment for visualizing the results in blender.

The last of the main goals has also been achieved, as shown in chapter 9. The most important characteristic of the visualisation tool, its adaptability to new data and problems, should make it suited for both demonstrations and informing health professionals of the anticipated behaviour of the outcome of individual patients if further refined towards a more user-friendly platform.

Avenues for future research

The complexity of the original problem and the time it takes to test different configurations leaves many options unexplored, such as how the regularisation terms can be exploited to achieve faster convergence and mitigate stiffness in the problem at the same time.

A distinct approach that is also relevant to the findings made in the process of enhancing the code is the reduction of residuals. This would contribute to a more accurate picture of the nature of muscle activations and other key parts of musculoskeletal system, but would require more detailed data-gathering strategies and probably more computation time as the problems grow in complexity.

Budget

The resources devoted to completing this project can be divided in three sections:

- Software costs
- Personnel costs
- Energy costs

Hardware costs are not considered since the computer at the SIMMA Lab is used by multiple people for different projects. Table 1 is the summary of the expected cost.

	Cost
Software	1,451.07 €
Personnel	21,000 €
Energy	3.92 €
Total	**22,454.98 €**

Table 1. Summary of the required budget.

Software costs

The software costs (Table 2) comprise licenses and other digital goods required to effectively complete the tasks of the project.

	Price	Amount	Total
MATLAB annual license	800 €	1/3	266.67 €
Microsoft 365 Business Basic	5.10 €	4	20.40 €
AutoCAD monthly subscription	291 €	4	1,164 €
Total			**1,451.07 €**

Table 2. Software costs.

Since most of the software used is open source (CasADi, OpenSim, Blender, etc.), the price is manageable.

Personnel costs

The cost of the labour (Table 3) is slightly higher than average because of the prior experience required to take on this project.

	Cost (€/h)	Hours worked	Total
Junior engineer	35 €	600	**21,000 €**

Table 3. Personnel costs.

Energy costs

The energy costs (Table 4) include only the approximation of the CPU consumption and are calculated according to the average price of electricity in Spain in May 2022 [48].

	Cost (€/MWh)	MWh consumed	Total
Computation	187.10 €	0.02093	**3.92 €**

Table 4. Energy costs.

References

[1] "Isometric Muscle Strength, Adduction (Code C181500),NCI Thesaurus." https://ncithesaurus.nci.nih.gov/ncitbrowser/ConceptReport.jsp?dictionary=NCI_Thesaurus&version=22.05e&ns=ncit&code=C181500&key=n1004018581&b=1&n=null (accessed Jun. 12, 2022).

[2] "Isometric Muscle Strength, Flexion (Code C139216), NCI Thesaurus." https://ncithesaurus.nci.nih.gov/ncitbrowser/ConceptReport.jsp?dictionary=NCI_Thesaurus&version=22.05e&ns=ncit&code=C139216&key=1557342039&b=1&n=null (accessed Jun. 12, 2022).

[3] "Isometric Muscle Strength, Internal Rotation (Code C181498) , NCI Thesaurus." https://ncithesaurus.nci.nih.gov/ncitbrowser/ConceptReport.jsp?dictionary=NCI_Thesaurus&version=22.05e&ns=ncit&code=C181498&key=n440194199&b=1&n=null (accessed Jun. 12, 2022).

[4] H. Long *et al.*, "Prevalence Trends of Site-Specific Osteoarthritis From 1990 to 2019: Findings From the Global Burden of Disease Study 2019," *Arthritis & Rheumatology*, Jun. 2022, doi: 10.1002/art.42089.

[5] H. M. Kremers *et al.*, "Prevalence of total hip and knee replacement in the United States," *Journal of Bone and Joint Surgery - American Volume*, vol. 97, no. 17, pp. 1386–1397, Sep. 2014, doi: 10.2106/JBJS.N.01141.

[6] D. Zhao, S. A. Banks, D. D. D'Lima, C. W. Colwell Jr., and B. J. Fregly, "In vivo medial and lateral tibial loads during dynamic and high flexion activities," *Journal of Orthopaedic Research*, vol. 25, no. 5, pp. 593–602, 2007, doi: https://doi.org/10.1002/jor.20362.

[7] "Walking Upright," *The Smithsonian Institution's Human Origins Program*, Dec. 31, 2021. https://humanorigins.si.edu/human-characteristics/walking-upright (accessed Jun. 03, 2022).

[8] K. S. Al-Zahrani and M. O. Bakheit, "A historical review of gait analysis," *Neurosciences (Riyadh)*, Apr. 2008.

[9] Z. O. Abu-Faraj, G. F. Harris, P. A. Smith, and S. Hassani, “Human gait and Clinical Movement Analysis,” in *Wiley Encyclopedia of Electrical and Electronics Engineering*, John Wiley & Sons, Inc., 2015, pp. 1–34. doi: 10.1002/047134608x.w6606.pub2.

[10] “Vicon | Award Winning Motion Capture Systems.” https://www.vicon.com/ (accessed Jun. 03, 2022).

[11] “3D Measurement - Codamotion.” https://codamotion.com/3d-measurement/ (accessed Jun. 03, 2022).

[12] F. C. Anderson and M. G. Pandy, “Dynamic optimization of human walking,” *Journal of Biomechanical Engineering*, vol. 123, no. 5, pp. 381–390, 2001, doi: 10.1115/1.1392310.

[13] A. J. van den Bogert, D. Blana, and D. Heinrich, “Implicit methods for efficient musculoskeletal simulation and optimal control,” in *Procedia IUTAM*, 2011, vol. 2, pp. 297–316. doi: 10.1016/j.piutam.2011.04.027.

[14] A. Falisse, G. Serrancolí, C. L. Dembia, J. Gillis, I. Jonkers, and F. 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*, vol. 16, no. 157, Aug. 2019, doi: 10.1098/rsif.2019.0402.

[15] G. Serrancoli *et al.*, “Subject-Exoskeleton Contact Model Calibration Leads to Accurate Interaction Force Predictions,” *IEEE Trans Neural Syst Rehabil Eng*, vol. 27, no. 8, pp. 1597–1605, Aug. 2019, doi: 10.1109/TNSRE.2019.2924536.

[16] A. Falisse, G. Serrancolí, C. L. Dembia, J. Gillis, and F. DeGroote, “Algorithmic differentiation improves the computational efficiency of OpenSim-based trajectory optimization of human movement,” *PLoS ONE*, vol. 14, no. 10, Oct. 2019, doi: 10.1371/journal.pone.0217730.

[17] P. J. Bishop *et al.*, “Computational modelling of muscle fibre operating ranges in the hindlimb of a small ground bird (Eudromia elegans), with implications for modelling locomotion in extinct species,” *PLoS Computational Biology*, vol. 17, no. 4, Apr. 2021, doi: 10.1371/JOURNAL.PCBI.1008843.

[18] G. Serrancolí, J. Torner, S. Perelli, and J. C. Monllau, “On the use of mesh-based joint contact models within simulations using automatic differentiation,” in *Computer Methods, Imaging and Visualization in Biomechanics and Biomedical Engineering II*, 2023.

[19] R. E. Wengert, “A Simple Automatic Derivative Evaluation Program,” *Commun. ACM*, vol. 7, no. 8, pp. 463–464, Aug. 1964, doi: 10.1145/355586.364791.

[20] C. C. Margossian, "A review of automatic differentiation and its efficient implementation," *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 4, Mar. 2019, doi: 10.1002/widm.1305.

[21] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," 2015, doi: 10.48550/ARXIV.1502.05767.

[22] A. Griewank, "Who invented the reverse mode of differentiation," *Documenta Mathematica*, Jun. 2012.

[23] W. Lee, H. Yu, X. Rival, and H. Yang, "On Correctness of Automatic Differentiation for Non-Differentiable Functions." arXiv, 2020. doi: 10.48550/ARXIV.2006.06903.

[24] T. Beck and H. Fischer, "The if-problem in automatic differentiation," *Journal of Computational and Applied Mathematics*, vol. 50, no. 1, pp. 119–131, 1994, doi: https://doi.org/10.1016/0377-0427(94)90294-1.

[25] R. Kearfott, "Chapter 1 Automatic Differentiation of Conditional Branches in an Operator Overloading Context," Jun. 1996.

[26] J.-F. M. Barthelemy and L. E. Hall, "Automatic differentiation as a tool in engineering design," *Structural optimization*, vol. 9, no. 2, pp. 76–82, 1995, doi: 10.1007/BF01758823.

[27] G. B. Dantzig, "Reminiscences about the origins of linear programming," *Operations Research Letters*, vol. 1, no. 2, pp. 43–48, 1982, doi: https://doi.org/10.1016/0167-6377(82)90043-8.

[28] M. Diehl, "Lecture Notes on Numerical Optimization (Preliminary Draft)," 2016.

[29] M. Kelly, "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *SIAM Rev.*, vol. 59, no. 4, pp. 849–904, Jan. 2017, doi: 10.1137/16M1062569.

[30] G. Serrancolí, A. L. Kinney, and B. J. Fregly, "Influence of musculoskeletal model parameter values on prediction of accurate knee contact forces during walking," *Medical Engineering & Physics*, vol. 85, pp. 35–47, 2020, doi: https://doi.org/10.1016/j.medengphy.2020.09.004.

[31] B. J. Fregly *et al.*, "Grand challenge competition to predict in vivo knee loads," *J Orthop Res*, vol. 30, no. 4, pp. 503–513, Apr. 2012, doi: 10.1002/jor.22023.

[32] F. de Groote and A. 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, vol. 288, no. 1946. Royal Society Publishing, Mar. 10, 2021. doi: 10.1098/rspb.2020.2432.

[33] "MATLAB - MathWorks - MATLAB & Simulink." https://www.mathworks.com/products/matlab.html (accessed Jun. 02, 2022).

[34] "Create and Share Toolboxes - MATLAB & Simulink." https://www.mathworks.com/help/matlab/matlab_prog/create-and-share-custom-matlab-toolboxes.html (accessed Jun. 02, 2022).

[35] "CasADi." https://web.casadi.org/ (accessed Jun. 02, 2022).

[36] J. Andersson, J. Åkesson, and M. Diehl, "CasADi: A symbolic package for automatic differentiation and optimal control," in *Lecture Notes in Computational Science and Engineering*, 2012, vol. 87 LNCSE, pp. 297–307. doi: 10.1007/978-3-642-30023-3_27.

[37] "SimTK: OpenSim: Project page." https://simtk.org/projects/opensim (accessed Jun. 02, 2022).

[38] S. L. Delp *et al.*, "OpenSim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, Nov. 2007, doi: 10.1109/TBME.2007.901024.

[39] "NCSRR - About NCSRR." https://opensim.stanford.edu/about/index.html (accessed Jun. 03, 2022).

[40] "National Center for Medical Rehabilitation Research (NCMRR) | NICHD - Eunice Kennedy Shriver National Institute of Child Health and Human Development." https://www.nichd.nih.gov/about/org/ncmrr (accessed Jun. 02, 2022).

[41] "Features — blender.org." https://www.blender.org/features/ (accessed Jun. 03, 2022).

[42] "Add-ons — Blender Manual." https://docs.blender.org/manual/en/latest/editors/preferences/addons.html (accessed Jun. 03, 2022).

[43] "Features | GitHub." https://github.com/features (accessed Jun. 04, 2022).

[44] "Git Guide." https://github.com/git-guides (accessed Jun. 04, 2022).

[45] "Second Generation Intel® Xeon® Scalable Processors Datasheet, Volume One: Electrical," 2019. [Online]. Available: www.intel.com/design/literature.htm.

[46] "Factor d'emissió de l'energia elèctrica: el mix elèctric. Canvi climàtic," 2022. https://canviclimatic.gencat.cat/ca/actua/factors_demissio_associats_a_lenergia/ (accessed Jun. 11, 2022).

[47] "Golf 8, activando generaciones | Volkswagen España." https://www.volkswagen.es/es/modelos/golf-8.html (accessed Jun. 15, 2022).

[48] "Precio de la electricidad | OCU." https://www.ocu.org/vivienda-y-energia/gas-luz/informe/precio-luz (accessed Jun. 15, 2022).