A good beginning makes a good ending

Experimental data-tracking of the BMX SX gate start using biomechanical modeling and trajectory optimization

J.D.C. Groenhuis
4484444

DELFT UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL, MARITIME AND MATERIALS ENGINEERING
DEPARTMENT OF BIOMECHANICAL ENGINEERING

Experimental data-tracking of the BMX SX gate start using biomechanical modeling and trajectory optimization

June 15, 2021

Author:
J.D.C. Groenhuis
4484444

Graduation committee:
Dr. Ir. A.L. Schwab (Daily supervisor)
Dr. A. Seth (Daily supervisor)
Dr. I. Janssen (Papendal)
Dr. Ir. J.P. Meijaard (External member)

Image on front cover taken from UCI.org:
https://www.uci.org/bmx-racing/news/2019/bmx-racing-a-look-at-the-uci-world-cup-legends

Preface

In the past months, I have been given the opportunity to research the BMX SX start in cooperation with TU Delft, the Dutch National BMX Team, the National Training Facility Papendal, and the Royal Dutch Cycling Union (KNWU). It has been an amazing experience to be able to work with some true professionals in both scientific research and BMX cycling. I am deeply thankful to Arend Schwab and Ajay Seth, who guided me through the roller coaster that scientific research can be sometimes. They taught me that doing research is essentially the same as the BMX start: sometimes the fastest way to move forward is to first take a step back. Furthermore, I would like to thank Ina Janssen from Papendal and Bernadet van OS from the KNWU for their advice and help during the project. I also have to express my gratitude to Hylke van Grieken and Melle van Dilgt who were so kind to share all their data on the BMX start with me. Hylke collected all kinetic and video data used in this work and could answer all my questions about the BMX start at the beginning of the process. Melle collected all kinematic data used in this study and shared all her knowledge about her experiences with the kinematics of the BMX start. I thank Mathijs Hofmijster and Knoek van Soest from VU Amsterdam for their advice on biomechanical modeling and Jason Moore from TU Delft for sharing his knowledge about optimization. At this point, I have to thank Adam Kewley who helped me a lot with getting access to the CBL workstation which made my optimizations 3 to 4 times faster. Without him, I would probably still waiting for the final results. That being said, I would like to thank the present and former members of the Fietslab and the Computational Biomechanics Lab for providing feedback on my work during the entire process. A big thank you to Mike, Vincent, and Yannick for sharing all their experiences while being on the same boat for the past couple of months. I would also like to thank my housemates Dennis, Kevin, and Ties for their encouraging words and interest during the process. Last, but definitely not least, I am very grateful to my parents, Martin and Sally, and my sister, Esmée, for their endless support during the past years in Delft. Finally, many, many thanks to my girlfriend, Marije, who has been the loveliest support along the way, but also had the courage to ask the right questions at the right moments.

Jan Groenhuis
Delft, June 2021

Abstract

Introduction During Bicycle Motocross races (BMX SX), the start has been proven to be crucial for good overall performance [1]. Riders reaching the bottom of the eight meter high starting ramp in front of the pack, have a favorable position to perform the first jump and can pick the most ideal line through the first corner. The chances of getting involved in collisions with other riders are also highly reduced. Since riders start behind a gate, the anticipation and timing with respect to this gate movement are most important [2]. Most scientific research regarding the BMX SX gate start is focused on defining the performance indicators for a fast start [2] [3] [4] or using in-field experiments to evaluate the effects of minor changes to the bicycle design [5] [6]. However, using the results of these studies to actually improve the start performance would require extensive training using these new conditions. Using predictive simulations, these adaptations can be evaluated without practicing and can thus have a huge contribution to enhancing the gate start technique. However, before these simulation models can have any impact, they must be thoroughly analyzed to prove their validity.

Objective The main goal for this study was to construct a biomechanical model for the BMX SX gate start which could reproduce experimental data. The model must be able to track kinematic data with an accuracy of less than 5% while also match the main kinetic characteristics without tracking those. The kinetic profiles should show the same peak pattern as is commonly seen in cycling and must not differ more than 10% with experimental data. When these goals are reached, this model could serve as a framework for future applications within BMX SX gate start research or other cycling disciplines.

Method A nine degree-of-freedom biomechanical planar model was created within the open-source software package OpenSim [7] [8]. The model consists out of the ground surface, the gate, the BMX SX bicycle, and the rider. The latter two are connected using kinematic constraints on the feet and pedals. The upper body is connected to the frame by a single arm. The model is driven by eight optimal torque actuators located at the hip, knee, ankle, shoulder, and elbow joints. The contact dynamics of the wheels to the ground and the gate are included using the Hunt-Crossley model [9]. Moco [10], a direct collocation package for OpenSim, was used to solve the kinematic tracking optimization problem. The kinematic data was taken from a prior study by Melle van Dilgt [11] who captured three-dimensional kinematics of an elite female BMX SX athlete of the Dutch National team using an Xsens suit (Xsens Technologies, Enschede, The Netherlands). This IMU data was projected on the planar model using OpenSense, a tool within OpenSim that converts experimental IMU data into the model's generalized coordinates. Simulation outcomes were compared to kinetic data collected by Hylke van Grieken [4], who used a fully instrumented bicycle including special cranks (Axis2D, Swift Performance, Brisbane, Australia) to capture the pedal forces executed during in-field experiments with a sample rate of 100 Hz. These experiments used the same elite participant but were taken on a different day using a different bicycle.

Results The optimized tracking simulation showed close agreement with experimental kinematic data, showing an average root mean squared error (RMSE) of 0.337° or 0.52% for the six leg joints. For the tracking of the crank angle and the horizontal displacement of the bicycle similar results were found (RMSEs of 0.079% and 0.6% respectively). Simulated crank torque peak values were off by 4.4%, 7.7%, and 5.1% for the first, second and third torque peak respectively. Overall the crank torque was reproduced with an RMSE of 18%.

Conclusion This work shows the suitability of the designed model for future applications in predictive simulation of the BMX SX gate start. The model can be used to study a wide range of "what-if" scenarios and could lead to the improvement of gate start performance. The way the model is constructed, the main building blocks can be adjusted to more accurate, but also more complex, components if desired.

Contents

List of Figures

List of Tables

1 Introduction

1.1 BMX SX

Although already developed as an alternative form of motor cross racing in the 1960s, BMX (Bicycle Moto-Cross) has only been an Olympic sport since the 2008 Beijing Olympics [12]. Among other things, this might be a reason for the limited scientific interest in this sport [13]. To clearly distinct the BMX freestyle events from the BMX races, the latter will be indicated as BMX SX, where SX stands for Super Cross. During these Super Cross races, the goal is to complete the track, consisting out of a starting ramp followed by 300 to 400 meters long track with hills, jumps and three sharp, banked corners, as fast as possible [14]. The track can differ for each venue, but the start is always the same. An example of a track design can be found in Figure 1.1. Depending on the difficulty of the track, which can be classified using three different levels of difficulty [15], the races last for 35 to 45 seconds [1]. During a race eight or less riders start at the same time from a start ramp which is eight meters high and at least ten meter wide. The initial slope of this start hill usually starts at 18° and suddenly increases to 28° after approximately 3 meters, which can be seen in Figure 1.2. This transition point is often indicated as the 'kink'. The riders gain most of their forward velocity due to this descend [2] and reach velocities at the bottom of the ramp of more than 50 km/h [2] [16].

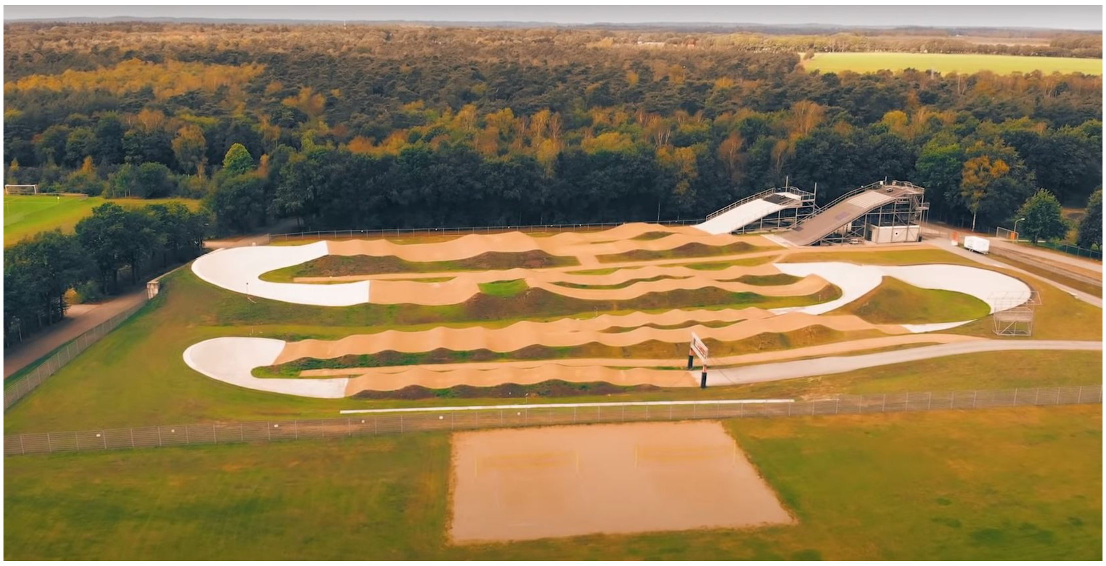


Figure 1.1: BMX SX track at the Dutch National training facility Papendal [17].

Riders compete on special BMX bicycles as seen in Figure 1.3 which have 20 inch wheels and a single geared freewheel drivetrain. The smaller wheels compared to mountainbikes or road bikes allow for larger jumps and makes it easier to go over the obstacles along the track. It is allowed to use multiple gears, but conventional gear shifting systems cannot handle the large forces provided by the athletes, so in practice they are never used. The tires of the BMX bicycle have a comparable width as mountainbike bicycles to increase the traction with the dirt track. BMX riders use shoes which are clipped to the pedals which enables them to push and pull on the pedals. Looking at Figure 1.3, one might wonder why the seat is in an extremely low position. The riders never sit on this seat and complete the entire track standing. Due to safety reasons a seat attached to the bicycle is mandatory.

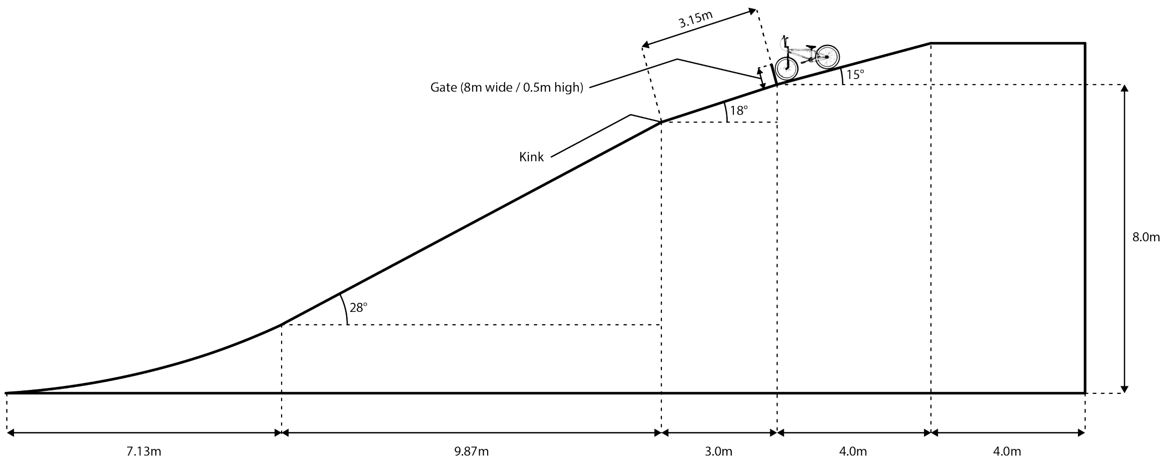


Figure 1.2: Olympic BMX SX start ramp dimensions as defined by the UCI BMX Regulations [18].

Unlike other racing sports, such as athletics or swimming, a BMX SX race starts behind a gate. This gate falls down, as seen in Figure 1.5, releasing the cyclists. Once the gate starts falling the riders can pass the starting line and start their race. The riders are allowed to move before the gate starts dropping, which gives them some preparation time. Besides, the riders are allowed to pass the gate before it has completely dropped. As seen in Figure 1.5, the BMX riders lift their front wheel off the ground to pass the starting line faster. Unlike the most right image, riders usually ride the first couple meters up to the kink on their rear wheel only. Because riders have some preparation time and the possibility to go over the gate before it has touched the ground, massive advantage can be gained from a good start.

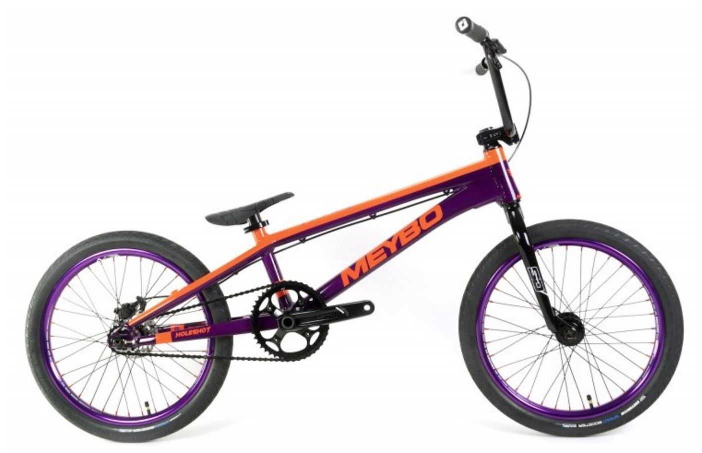
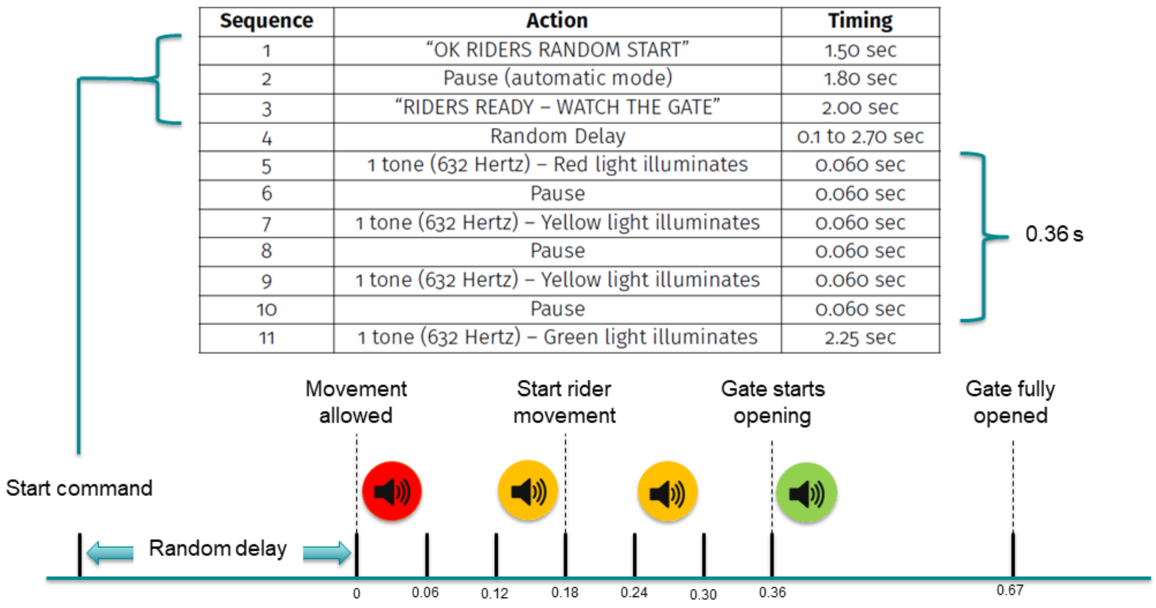


Figure 1.3: A BMX SX bicycle (Holeshot, Meybo, Boxmeer, The Netherlands) [19].

1.2 The importance of the gate start

The start is commonly accepted as the most important part of a BMX SX race. Rylands and Roberts have shown that being within the first three riders at the first corner has a significant correlation with a podium classification at the finish [1]. This suggests that a good start is crucial to get a good result. This has been confirmed by other studies which all concluded that the start and the first straight section of the race are the most important phases of a BMX SX race [20] [21]. These findings make sense considering the fact that overtaking is relatively hard and risky and athletes who lead the race are less likely to make contact with other riders resulting in a crash. Additionally, riders who are in front may have a favorable position when preparing for the first jump and can select the best choice of line along the track [2]. Due to the importance of the start, athletes spend the majority of their training time on improving physical strength and practising the start to enhance the first couple meters of their race [16] [22] [23].

To understand why the start is so important, a closer look must be taken at the gate start action. The riders start with their front wheel positioned against the gate as seen at the left image of Figure 1.5. At the start, the cranks are approximately leveled with the ground to remain balance [24]. The leg closest to the front wheel is called the lead leg and the other leg is called the trail leg. As an example, in Figure 1.5 the right leg is defined as the lead leg. As seen, both legs are slightly bent. The riders use their weight and the contact with the gate to remain their balance, since their feet are clipped to the pedals. When all riders are ready, the standard warning is announced: OK riders, random start, riders ready, watch the gate, as shown in Figure 1.4. After the word gate, there is a random delay between 0.1 and 2.7 seconds, followed by a series of four rapid tones and lights: red, yellow, yellow and green [18]. At the final tone and light the gate starts to drop. However, the riders are allowed to already move after the first signal, which give them a 0.36 seconds period to prepare for the gate drop as indicated by the timeline in Figure 1.4. More details about this and other rules and regulations regarding the BMX SX start can be found in Appendix A.



Sequence	Action	Timing
1	"OK RIDERS RANDOM START"	1.50 sec
2	Pause (automatic mode)	1.80 sec
3	"RIDERS READY – WATCH THE GATE"	2.00 sec
4	Random Delay	0.1 to 2.70 sec
5	1 tone (632 Hertz) – Red light illuminates	0.060 sec
6	Pause	0.060 sec
7	1 tone (632 Hertz) – Yellow light illuminates	0.060 sec
8	Pause	0.060 sec
9	1 tone (632 Hertz) – Yellow light illuminates	0.060 sec
10	Pause	0.060 sec
11	1 tone (632 Hertz) – Green light illuminates	2.25 sec

Figure 1.4: The BMX SX gate start sequence according to the UCI rules & regulations [18].

Usually the riders initiate their motion based on the red light, resulting in movement prior to the actual gate drop [25]. Using a quick forward motion of the hips and shoulders towards the handlebars, the rider pushes the bike backward. By pulling on the handlebars and exerting immense forces on the pedals, the front wheel lifts up slightly from the ground. This wheelie motion is however constrained by the upper body moving forward [22]. This specific motion is often referred to as the 'slingshot manoeuvre' and can be seen in Figure 1.5. This typical action serves the main goal of a fast start, which is to pass the gate with the highest forward velocity without hitting it and to apply maximum force to the pedals [25].

In general, the start can be divided in two different parts: a backward motion of the bike, the so-called recoil-phase, followed by an acceleration phase. However, Kalichová et al. [24] were convinced that in order to get a deeper understanding of the start technique, it is needed to divide the start in more than two phases. As an outcome of their study, they divided the start into five different phases (Figure 1.5).

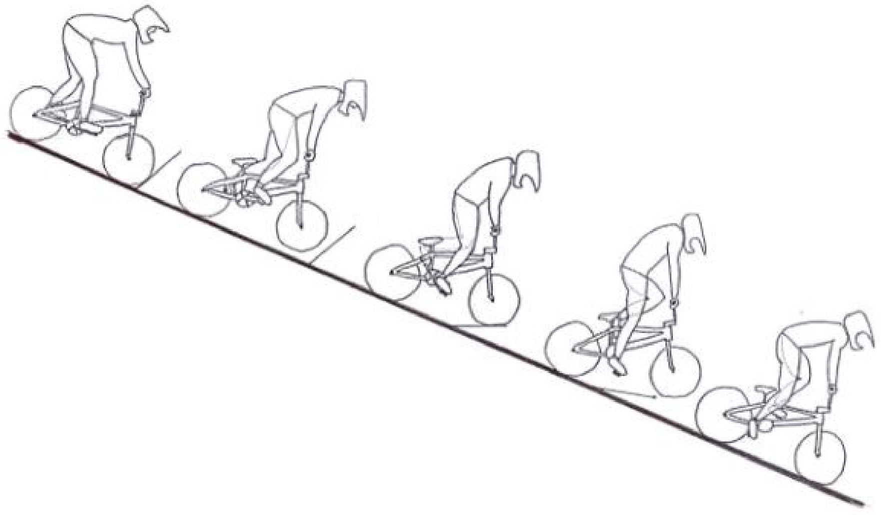


Figure 1.5: The five phases of the BMX SX gate start adopted from Kalichová et al. [24].

Phase 1: Reaction This phase starts with the illumination of the first red light and ends with the initiation of movement. The time taken for this phase is well compared to reaction time.

Phase 2: Preparation movements This phase lasts from the initiation of movement until the first pedal stroke. Prior to this first pedal stroke, the upper body has already been moved forward. Because the center of mass of the rider is shifted forward, the bicycle rolls a little backward due to the conservation of momentum. This is often identified as the recoil motion of the bicycle. To get an idea about the distance the bicycle can roll backward, Appendix B shows a way to calculate the theoretical maximum recoil distance.

Phase 3: First pedal stroke This first pedal stroke ends when the crank is perpendicular to the ground. Since the cranks are initially parallel to the ground, the first pedal stroke covers a range of 90°. During this phase the predominant movement is the lead leg pushing downward.

Phase 4: Dead point pedal passage During one revolution of the cranks, the two cranks are positioned in unfavorable positions twice, called the Top Dead Center (TDC) and Bottom Dead Center (BDC). In these positions the cranks are oriented perpendicular to the ground, which makes it hard to exert force in the tangential direction. This results in a delay between the first and second pedal stroke. This delay is given as the duration of phase 4. During this phase the rider pushes the bike forward with a backward movement of the hips. The phase ends with the forward movement of the trail leg.

Phase 5: Second pedal stroke The second pedal stroke is larger than the first one. The second pedal stroke starts and ends with the cranks perpendicular to the ground, resulting in a total range of 180° being covered during this phase.

Since the gate start action is the most important phase of the race, several studies tried to define the best starting technique. As an example, Gross et al. [2] were interested in the performance indicators for a good start. They used a SRM powermeter to collect crank power and a Vicon 3D motion-capture system to get the kinematics of the start. The idea of their study was that by comparison between the groups of fast and slow starters, the important aspects of the start would become clear. It was found that the fastest athletes reach their most backward position the fastest, which gave them time to "*finely modulate the forward thrust in order to achieve the best balance between forward velocity and gate clearance*" [2].

Gross et al. did also define specific metrics that showed the largest correlation with fast performance. It was suggested that the time to reach the initiation of forward velocity (t_{v_0}) and the distance between the front wheel and the gate when it drops ($|d_{t_0}|$) "*characterize the effectiveness of the preparatory slingshot maneuver prior to the gate drop*". Additionally, the torque and cadence of the first pedal stroke seemed to be the most important for a fast start. It was also found that the forward velocity when the gate dropped ($|v_{t_0}|$) correlated more strongly with a fast gate start than did the mean acceleration after the gate has dropped. As a final conclusion it was therefore stated that BMX SX racers should focus the majority of their training on a fast and well-timed slingshot maneuver and to develop high strength and power at relatively low cadences and high loads.

The importance of the recoil was also one of the key findings of a study by Grigg [3]. In this study the kinematics of the gate start were captured with ordinary cameras and further analysed using Kinovea, a video software package for sport analysis. A correlation was found between the recoil distance and the time needed to reach the kink, i.e. a larger recoil distance correlated with faster kink times. Besides, Grigg defined three different set positions and front hub trajectories for the riders tested. The fastest riders used a set position with a relatively large shoulder angle followed by a trajectory for the front hub which looked like a hairpin. Due to the larger shoulder angle, the hips, and therefore center of mass, were positioned more to the back. This gives the rider the possibility to reach a larger recoil distance.

Besides kinematic aspects of the gate start, also kinetic parameters were investigated. Recently, Hylke van Grieken [4] equipped a BMX bicycle with an extensive set of sensors to capture force, torque and power production as well as a set of kinematic data including crank angle and bicycle position and speed. Based on correlations with starting time, the time needed to reach the bottom of the hill, the most imports factors for a good start were: initial velocity when the gate drops (V_i), peak effective force ($F_{e_{max}}$), peak power during the first two pedal strokes (P_{max}) and the technical indicator $RMPD_{70}$. This latter indicator is the ability of a rider to produce power above 70% of the peak power, so essentially the width of the power peaks. These indicators together were able to predict start performance with an accuracy of 89 to 93%.

BMX SX start research is not just limited to finding performance indicators. Several studies varied bicycle parameters and measured their influence on performance. For instance, Rylands et al. [5] studied the influence of a change in gear ratio. They found that higher torques and powers were generated when using a higher gear ratio and thus advised riders to use this higher gearing. Additionally Campillo et al. [6] changed the length of the cranks and measured its effect on starting performance. It was found that a larger crank length compared to the conventional cranks resulted in a larger forward speeds and smaller forces, indicating that a longer crank could be beneficial. However, none of these recommendations have been adopted yet.

The results of the forementioned studies provide a good overview of how the BMX SX riders currently perform their start. Besides it gives an idea about which factors are the most important and what to focus on. Grigg [3] and Gross [2] found differences between the faster and slower starts, but could not show that improving the important aspects of the start would actually enhance the starting performance. Besides, Ryland et al. [5] and Campillo et al. [6] found that changes to the bicycle design could be beneficial, but riders did not adopt these recommendations. A reasonable explanation is that these adaptations, both for the bicycle and for the rider’s technique, would require extensive training and practice. It would be ideal if all these recommendations could be investigated based on theory without the need for adaptation by the riders.

1.3 Research goals

The gate start is the most crucial aspect of a BMX SX race. Multiple studies have tried to identify the characteristics of a good start. From their results, it is clear that the recoil movement is important to obtain a positive forward velocity when the gate starts to drop. Those studies showed indicators of a fast start but did not show if start performance could be improved by focusing on their key aspects. It would be interesting to know if riders are currently using the optimal technique or that there are different ways to start that might actually be faster. Technique is a difficult word since it is not clear from its definition what it exactly connotes. In this work, technique is defined based on the three main key features of the start indicated by literature. These are the initial posture of the rider on the bicycle, the trajectory of the bicycle in time (the recoil trajectory), and the timing. This latter factor combines two timing aspects indicated by literature, namely reaction time and the initiation of forward velocity. It would be extremely useful to investigate the influence of these factors on overall performance without the need for adaptation by the riders. In-field investigation of these aspects of the start is nearly impossible because this would require extensive training with BMX athletes and measure whether or not this positively changes starting performance. Therefore, an alternative and more interesting approach would be to develop a model that could be used to make changes to these aspects and calculate and predict their influence on performance. For this model to be of any use, it must be proven that it captures the essential dynamics of the BMX SX start. Since such a model does not yet exist, the goal of this work is to develop a BMX SX model that can reproduce experimental BMX starts within a 5% margin. By reproducing experimental data it can be validated that the model is accurate enough to be used in future predictive modeling. To achieve this goal, a BMX SX model must be created and kinematic and kinetic data must be collected. Finally, these two must be combined in such a way that the BMX SX model produces the same motion and forces as seen during experiments. In this work, it was chosen to use an optimization routine that tracks the kinematic data. The outcome of this tracking optimization problem is a set of control efforts resulting in forces on the pedals. These forces are then compared to kinetic data to validate the model.

1.4 Thesis structure

In Chapter 1 this report starts with the background of BMX SX racing including the special gate start and its importance for race performance. This chapter ends with the goals for this thesis work. As will be explained in the first section of Chapter 2, to achieve the goals described in Chapter 1 a predictive model will be constructed. The remainder of Chapter 2 focuses mainly on the background of human modeling in sports and predictive modeling in general. Multiple focus points when modeling human performance will be described even as several optimization strategies. This serves as a foundation for Chapter 3, where the methods used for the construction of the predictive model will be explained. This includes software choices, key model choices and the methods used to gather critical information about the BMX bicycle design such as mass, moment of inertia and dimensions. In Chapter 4 the main findings of this work are presented followed by a comparison with experimental data underlining the validity of the model outcome. Finally, Chapters 5 and 6 discuss the key finding, but also the limitations, of this work. This is followed by a conclusion and recommendations for future work.

2 | Theoretical Background

2.1 Approach

In order to achieve the goals sketched in Chapter 1.3, this work must come up with a way to make predictions about performance in BMX SX racing. This can be done by constructing a biomechanical model of the BMX start. This model then provides a simulation of the start which, in this case, mimics the motion of both human and bicycle. One of the benefits of computational models is the fact that one can execute many different experiments at the same time, whereas these experiments with athletes are in real life simply not possible. Another useful feature of modeling and simulating is the ability of modern computers to execute many trials at once and evaluate which ones lead to better end performance, i.e. optimization. Using these kind of repeated simulations computers can predict the optimal performance strategies for all different kinds of applications. This latter feature is extremely useful for the goals of this thesis. These so-called predictive models can, as the name suggests, make predictions of the outcome of the simulation. For instance, they can predict the final time of a sprint race or the optimal technique to jump as high as possible. A good, end easy to understand, example of a predictive model is a model created for the prediction of performance time in track cycling [26]. This model uses ergometer test data from elite track cyclist and calculates the time needed to complete a 4 km race taking into account all forces acting on the rider-bicycle system. Although this model uses the simplification of modeling the cyclist and the bicycle as one point mass, it still is able to predict performance within a 2% margin. Apart from these point mass model, there are also models that more realistically reconstruct the human. For example, the model created by Anderson et al. [27] modeled the human slightly more accurate to predict the maximum jump height. The model consisted of ten different segments to represent the human body and was actuated by a total of 54 muscles. The model predicted the maximum jump height with an accuracy of 0.1 cm compared to experimental data.

Besides the predictive models, there are also descriptive models. These models are largely data-driven, meaning that they need experimental data to function. With measurements of human walking, a descriptive model can calculate the loads or muscle activation. The goal for this work was to design a model that could track kinematic data, something a descriptive model also does. However, the kinematic data set used, did not information for all states, so descriptive modeling would be difficult. It was chosen to construct a predictive model which could use the kinematic data available. This is slightly different from descriptive modeling in a sense that a descriptive model just calculates the loads causing the observed motion, whereas a tracking predictive model optimizes the activations with the goal of minimizing the differences with kinematic data. A benefit of this latter method is that once solved, the model or optimization goals can be slightly adapted to serve a different research goal. For instance, when the tracking solution is done, this solution could be used as an initial guess for a new optimization which includes minimizing performance time. In this way, the new solution is likely to be close to the original tracking solution, but might have found minor improvements. Therefore, this chapter will primarily focus on the possibilities in predictive human modeling in sports.

2.2 Human modeling in sports

In sports, coaches and athletes try to optimize the technique of the athlete in order to increase performance. This is often done by making small adaptations to their current technique and measure the influence of this adaptation. In the perfect situation, this small adaptation should only effect one parameter at the time. In a carefully controlled laboratory experiment this might be possible, but in sports science this is extremely difficult since changing a certain technical parameters may also change other aspects that influence the performance [28]. Therefore, theoretical modeling is widely used nowadays to help athletes and coaches optimizing the technique. Such models give a simplified representation of the movements executed by the athlete and make it possible to carry out the ideal experiment by making small changes to one specific parameter. The ability to run thousands of simulations on one day without experiencing the fatigue of a human athlete, is one of the main advantages of such models. The technique used can be optimized by characterizing the movements observed using a number of key parameters and then optimizing those parameters to get, depending on the performance score used, a maximum or minimum result [29].

Models can either be used to solve a forward dynamic problem or an inverse dynamic problem [28]. This is also knows as predictive and descriptive modeling as seen before. A forward model can calculate kinematics based on kinetic input, i.e. given the forces and torques exerted on a certain dynamical system, the model provides the motion. An inverse model does the exact opposite, thus calculating the forces and torques that cause an observed movement. The name inverse refers to the flow of calculations which is opposite to the way that motion is actually created in the body. In Figure 2.1 this difference is shown using a visual representation.

As seen from the arrow in red, the inverse dynamic model usually uses kinematic measurements of an athlete as input, which is first digitized by using inverse kinematics. This can be done using position measurement of certain key body markers which are commonly placed on the joint locations. These joint positions can be used to calculate joint angles at each point in time. When a rigid body representation of the human is created, for instance by modeling all human segments by rigid body links connected by ideal revolute joints, the joint angle time history can be used to visualize the movement of the rigid body model. Finally, with the motion and inertia of the model known, the forces or torques responsible for the observed motion can be calculated using optimization criteria [30]. Since each joint in the human body is actuated using several muscles, determining the contribution of each muscle results in a redundancy problem [31]. The problem with muscle redundancy is explained in Figure 2.2. It comes down to an infinite number of possibilities to achieve the same joint torque. This muscle redundancy can make the model complex. Constraints based on the previous states of the muscle and their force relation with respect to length and velocity can be used in the optimization process. Usually the energy expenditure or the muscle stress is minimized in this optimization. Besides, inverse modeling is relatively sensitive to errors made in measurements and modeling. The accelerations determined from kinematic data might have such large errors that often external forces are recorded as well to obtain additional information.

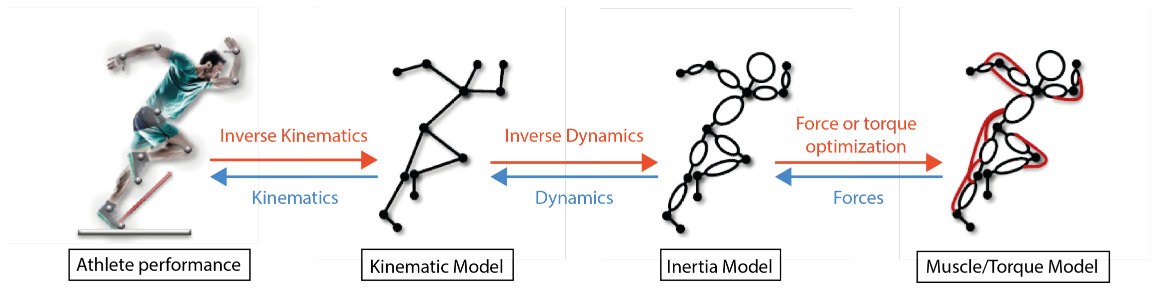


Figure 2.1: Visual comparison between inverse (red) and forward dynamics (blue). Adapted from Geijtenbeek [31].

A forward dynamic model is maybe more commonly used in daily life dynamics. It uses the kinetic input of forces and torques as input to calculate the motion by using straightforward dynamic equations. Using the principles of Newton's Second Law of Motion the accelerations can be calculated from the forces and inertia of the system. Most systems representing the human are however too complex to analyze by hand and require computer analysis [28].

As can be concluded from the definition of inverse and forward dynamics, inverse dynamics can only be used to analyze observed motions. It is therefore also referred to as descriptive modeling, since it describes the motion observed in terms of kinematics and kinetics. In contrast with forward models, inverse dynamics can thus not be used to predict performance. As explained before, this study is focused on predictive modeling in sports. As will be shown next, there are also hybrid approaches which use predictive modeling in combination with experimental data. The next sections will elaborate more on the different approaches when using predictive modeling. Most information gathered for this chapter is based on the book 'Biomechanical Evaluation of Movement in Sport and Exercise' [28], which summarized relevant literature on this topic.

2.2.1 Forward Dynamics

As explained, in forward dynamics the forces are known and the goal is to find the motions resulting from those forces. However, multiple different types of forward dynamics can be distinguished. The first distinction can be made based on the actuation used for the model. There are two main ways as seen at the right hand side of Figure 2.1: muscle forces of joint torques. This results in two different types of forward dynamic modeling: force-driven and torque-driven models. Nonetheless, there is a third way of implementing input to a forward model, namely by prescribing joint angles. These methods will be discussed in the subsections to follow.

Angle Driven

It might sound a little weird: prescribing joint angles, which comes down to prescribing motion, in a forward dynamic model. Usually, as is the case in force and torque-driven models, the kinematics of the system are the output of the model, rather than the input. However, the input of a forward dynamic system can be angle driven. In this case, the output of the model will be both the whole body orientation and the center of mass location of all rigid bodies and the required joint torques [28]. Usually, angle-driven simulation models are used in cases where the movements are not limited by strength, such as movements through air without contact with the fixed world. Examples can be found in high jumping [32] or the dive start in swimming [33]. Since angle-driven models are relatively easy to control, they allow for more complex modeling using more rigid bodies with more degrees of freedom compared to force and torque-driven models. Angle-driven forward modeling can also be a useful alternative for inverse dynamic models, since it is also used to calculate joint torques based on a known kinematic input.

Force Driven

When the human is in contact with its surrounding it is often more convenient to make use of a force driven model. In biomechanics, human forces are the results of muscle action. Force driven models usually contain muscle models to describe how muscles can exert forces and therefore moments on the joints as seen from Figure 2.2. The most widely used muscle model was created by A.V. Hill back in 1938 [34]. This muscle model mainly consists of a contractile component in series with an elastic component [35] and will be explained in slightly more detail in the upcoming sections. Nowadays, advancements and adaptations have been made to the original model, but the essence is still the same after all these years.

In force driven models the muscle forces acting on each joint are the input from which the kinematics can be calculated. This type of human movement modeling follows precisely the procedure for forward dynamic modeling as shown in Figure 2.1. Most force driven models have been used to simulate simple jumping motions, which are relatively easy to model since the movements can be considered in 2D and a certain level of symmetry can be observed [28]. Since the motions modeled with a force driven method are usually relatively simple, it allows for more complex methods to calculate the muscle contribution of each muscle. Muscle force driven models are also used in cycling, since this is a relatively simple movement too. For instance Thelen et al. used 15 muscles per leg to model the bicycle pedaling motion [36].

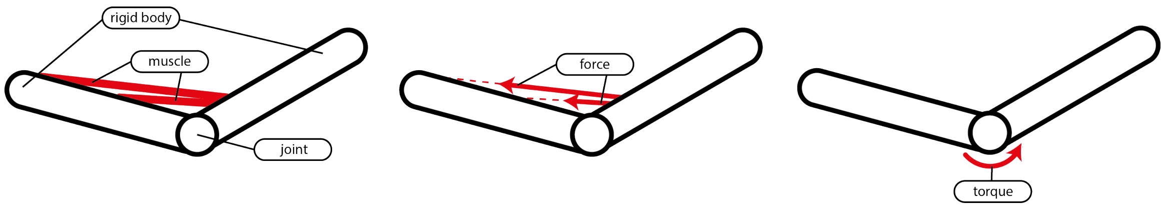


Figure 2.2: Muscle force and joint torque representation. Both muscles contribute to the rotation of the rigid bodies and there are many combinations possible that results in the same net joint torque. This is called a redundancy problem. Adapted from Geijtenbeek [31].

As seen from Figure 2.2 each joint is driven by several muscles. In this simple example, only two muscles are shown, but in reality each joint is powered by a larger number of muscles, depending on the joint. In fact, in Figure 2.2 only the agonists of this joint are drawn, despite the fact that there are also muscle connected at the other side of the joint, the antagonists. Since a muscle can only pull and not push, these so called antagonists allow for torque production in the opposite direction. This can make muscle force driven modeling rather complex since multiple combinations of force contributions can results in the same torque, leading to a redundancy problem. Usually an optimization procedure is followed to calculate the contribution of each single muscle. This optimization can be based on minimizing the stress in each muscle or maximize the torque production of a certain joint. This optimization process could result in long computation times. Assigning each muscle to a specific group could speed up the process. For instance the agonists and antagonists could be combined into two separate muscle groups, which reduces the level of complexity and shortens computation times [37].

To use the muscle force as an input, one must first find an expression for the force that can be produced at each point in time. These relations are usually based on the work of A.V. Hill who, as aforesaid, separated the dynamics of the muscle into a contractile and an elastic element [34]. The contractile part accounts for the effects that length and velocity have on the magnitude of force production. The muscle force-length relationship is usually modeled as being quadratic, meaning that at short and long lengths less force can be produced as compared to a medium stretched muscle. The muscle force-velocity relationship is a little more complicated since it differs for a muscle elongating (eccentric phase) or a muscle shortening (concentric phase) [34]. The force a muscle produces is also depending on the activation level, which ranges from 0 (no activation) to 1 (maximum voluntary activation) [28]. Besides this contractile part of the muscle representation, Hill defined two elastic elements in his muscle model, one in series with the contractile element and one parallel. This latter is often ignored for applications in sport, since these elements do not provide large forces. The elastic elements in series act as a non-linear spring. When a muscle is stretched further than its resting length, the force produced by the elastic elements increases with muscle length.

Torque Driven

The main drawback of using force generated by each single muscle is the number of variables it entails [38]. Taking into account more muscles per joint may improve the quality of the model but slows down the computation of the model. As mentioned previously, assigning muscles to different groups might help. In an extreme way, all muscle acting on a joint can be modeled as one single group. In that case the muscle forces are no longer considered, but only muscle torques are used. This reduces the number of variables in the model making the optimal control problem a lot easier and is often referred to as joint torque modeling. Examples of joint torque driven models can be widely found in different types of sports, like for instance gymnastics [28] and cycling [39]. These torque driven models generally consist of a combination of active and passive torques. The active torque is a product of the muscle acting on the joint and is depending on the joint angle and the angular velocity of the joint since those parameters are correlated with the length and speed of the muscle [38]. The passive part of the joint torque is due to the passive stretching of the muscle tissue. This active and passive torques are basically the equivalents of the contractile and elastic elements in the muscle model defined by Hill [34]. Torque driven modeling is also convenient when it comes down to implementing measured data in the model [28]. Since measurements on individual muscle force production are extremely hard, especially in sports, it is easier to use and measure the torque production. Strength measurements on an individual athlete can be done using an isovelocity dynamometer [29]. With this device, one can determine the maximum torque that can be produced in a single joint as a function of the joint angle and the angular velocity of the joint. Torque driven modeling could therefore be beneficial when one aims to model the difference between athletes, since force driven models must rely on theoretical equations or scaled versions of these equations.

Hybrid methods

As was made clear in previous parts of this chapter, forward dynamic modeling is needed when one opts to purely predict the performance of an athlete. However, not all forward modeling methods are purely predictive. For instance, when one prescribes joint angles, as with the angle driven forward models, the model has little predictive power. Such models are considered as semi-predictive or data-tracking [30] and are maybe more comparable with inverse dynamic methods [38]. They make use of experimental data and the goal of the model is to minimize the error between measured data and model output. Since the model matches measured performance, the results are often realistic. In contrast, the other group of models is known as fully-predictive or knowledge-based models. These methods do not require measured data and make use of an objective function, such as reaching a target as fast as possible, to make sure that modeled behaviour is comparable to observed motions. In general it can be stated that the data-tracking method provides little predictive power, but shows high accuracy with measured data, while the knowledge-based models can predict performance but are also hard to compute since optimizations of objective functions are required. Yet, there is a belief that prediction of human movement can more realistically be described by a combination of both semi- and fully-predictive modeling, the so called hybrid method [30]. For instance, Pasciuto et al. [40] formulated their model to predict clutch pedal depression as a combination of data-based and knowledge-based input. They changed to contribution of each of the components ranging from fully data-based to fully knowledge-based. It was found that the hybrid solution showed the best results compared to the actually performed motions. A sort of same approach was described by Otten [41], who stated that it is possible to use both inverse (data-driven) and forward dynamics (knowledge-based). When one formulates the equations of motion, it is possible to presume certain movement patterns. For example, when one aims to predict the dynamics of walking one can presume certain arm motions and thus extract the accelerations associated with this motion. When these assumed accelerations are multiplied with the inertia matrix it is possible to remove the rows and columns in the equations of motion dealing with the prescribed motion. Another way of using a mixture of forward and inverse dynamics can be to use forward dynamics to predict the movements and then correct the joint torque when the error between modeled and measured kinematics exceeds a certain limit [30].

Point mass models

The sections above described the most common ways to model human performance. However, there are other, more easier, ways to make predictions about human performance in sport. The foregoing section assumed that in order to calculate human performance, the human itself, although highly simplified and represented by rigid bodies and muscle-like actuators, must be modeled. If the human is even further simplified, it can be reduced to only one single point mass, which can exert forces or power to its surroundings. Using this modeling strategy one does not model the human as an actuator, but just its actions on the environment. Models including a multibody segmented human are biomechanical models, whereas the point mass models are best described as mathematical models. This latter modeling method was used by Lukes et al. [26] for their track cycling model discussed before. In section 2.2.3 more examples of point mass models will be shown. In these models the human and bicycle itself and their internal forces are not modeled, but just their interaction with the environment in terms of propulsive and demanding forces. The propulsive forces are the force delivered by the athlete to propel the bicycle and the demanding forces are a combination of all possible drag forces acting on the bicycle rider system. The benefit of such methods is that the model becomes relatively simple because the number of degrees of freedom and control parameters are highly reduced, leading into faster computation times. On the downside, removing the human aspects makes the model less suited to make predictions about the optimal technique. These models cannot make predictions about the human posture during the start, but can only make predictions about the fastest performance time which is theoretically possible. Another widely seen application of these point mass models is the prediction about pacing strategies. The pacing strategy tells something about the distribution of energy throughout the race and is only applicable to endurance races.

2.2.2 Parameter determination

Since each human is different, a model simulating human movement should allow for individual parameter selection. As seen, it is not always possible to measure such parameters. There are only two ways to select parameters for a model [28]. One can make parameters estimations based on literature or one can use experimental data of the individual athletes. This latter method is often beneficial since parameter selection and model validation can be done using the same participant, but it is not always possible. This section describes methods to gather parametric input for certain aspects of the model.

Inertia parameters Since human body parts are modeled as rigid bodies, accurate information about these body parts is needed in order to realistically model them. These inertia parameter are mass, size, center of mass location, and moment of inertia. Most techniques used to gain information about these rigid segmental inertia parameters is obtained by scaling data from cadavers based on anthropometric measurements, either by a regression equation or a geometric model [28]. Besides, with the advancement of scanning technologies it is nowadays also possible to use 3D body scanners to measure the body composition of an athlete [42].

Strength parameters Determining the contribution of each muscle in the human body is extremely complex. There are two main ways to represent these forces without intensive measurements. The first is to implement all relevant muscles as individual muscle-tendon complexes based on experiments with animals [28]. Not surprisingly, this method does not give a complete representation of the human muscle and is not that accurate, although it can be scaled to match an individual muscle or group. It is easier and more accurate to make use of torque-driven models, because net torque produced at a certain joint can be measured using a dynamometer. This can be done for a range of velocities and angles, to get a complete set of parameters. Nevertheless, data from literature is still needed to determine the parameters of the passive torque produced by the elastic elements of the muscles.

Visco-elastic parameters These are the parameters needed to implement interaction with the environment or to include the modeling of complex joints like the shoulder. For this work, it would entail the contact between the wheels and the ground. These contacts can be modeled as springs with a certain stiffness and damping. The best way to implement these in a model is by directly measuring their parameters. Another alternative method is to optimize them based on simulations and the observed performance.

2.2.3 Examples in Literature

Predictive modeling is widely used tool in sport research, so many examples can be found in literature. Nevertheless, no BMX SX models are present in literature, so the focus is mainly on cycling models, since road and track cycling share most of their characteristics with BMX SX racing.

Point mass models Starting with the simplest models, the point mass models, one can find many examples in cycling literature. Martin et al. [43] developed one of the first mathematical models for the prediction of power output during road cycling. This power was calculated by equating it to all resisting powers: aerodynamic drag, rolling resistance, wheel bearing friction, the rate of change of potential and kinetic energy, and friction in the drive chain, which could all be determined by measurements. Most point mass models function based on this 'first-principle' [44]. Lukes et al. [26] and Underwood and Jermy [45] used the same strategy for their models to predict performance time in track cycling. Key feature of their models was the introduction of a term that accounted for the banked corners of a track cycling velodrome. Finally, Fitton and Symons [46] extended the work of Lukes et al. and Underwood and Jermy by adding an improved relation for the aerodynamic drag for cycling in a velodrome. One major issue with the strategy of equating the power or force required to overcome all resisting factors, is the fact that one must divide by the velocity, which is zero at the start. This will lead to an infinite acceleration at the beginning. Most models solve this issue by assuming an initial non-zero velocity or ignoring the first couple second of the race. Besides, point mass models can make predictions about power output or performance time, but do not capture the technique since human specific dynamics are simply not taken into account. For the purpose of developing a BMX model, it is thus important to have a look at biomechanical models for cyclist instead of purely mathematical models.

Biomechanical models One of the first to construct a cycling model without making use of the point mass simplification were Cangley et al. [47]. They did so because they believed that to reach a higher level of accuracy the rider and the bicycle must be modeled separately. The bicycle model consisted out of a rigid frame connected to a rear wheel and a steering assembly. The contact between the wheels and the ground was included using both holonomic and non-holonomic constraints by making use of an advanced tire model. Because the bicycle had a steering angle, the model needed to be stabilized. It was chosen to include a PID controller which steered into the direction of the fall to keep the bicycle-rider system in an upright position. The human model was constructed out of 14 rigid bodies representing the arms, legs and upper body. The pelvis was fixed to the seat of the bike and the feet to the pedals, resulting in two 5-bar linkages with each two degrees of freedom. The model was driven by applying pedal forces directly on the pedals. These were found by measurements of pedal forces at different steady state power outputs. So while in real life pedaling forces are the result of leg joint torques, in this model the leg pedaling motion is the results of applying pedal forces directly. Only rolling starts were modeled, to ensure that joint torques would not have to be unrealistically high. The model was used to investigate the rider-less self-stability of bicycles and compare that to the golden standard in literature. The model was also used to predict the performance time for a 2.5 mile road race. The predicted time varied only 1.4% with respect to the mean actual time of 14 experienced cyclists. One of the downsides of this model, is the lack of handlebar forces and the simplification of the driving actuation in the model. An upgrade to the model of Cangley et al. would include a muscle or torque driven human body.

Several examples can be found in literature applying just that strategy. These models use a multi-body segmented human body combined with either a force control (muscles) or a torque driven actuation in the joints. However, almost all these studies opted to optimize steady state pedaling, sometimes at a prescribed pedaling rate [48]. Additionally, most models assumed a fixed hip location at the saddle, thus not allowing standing start or standing pedaling [36]. An interesting approach was presented by Raasch et al. [49], who designed a model that optimized maximum startup pedaling, i.e. pedaling starting with zero velocity, by use of a force driven forward dynamical model. The human was only represented by the legs, which consisted of three rigid bodies each (thigh, shank and foot). Each leg was driven by a total of 15 equivalent Hill-type muscles, which were further simplified into nine muscle sets. All muscles in the same set would receive the same activation signal. The hip was modeled to be stationary on the seat and the feet were connected to the pedals. The model made use of an ergometer instead of a real bicycle. The model thus predicts maximum ergometer pedaling, thus ignoring the dynamics of a real bicycle. Based on assumptions previously made by Fregly and Zajac [50] in their pedaling optimization model, the load of the ergometer was modeled using an effective inertial and frictional load. The simulation results showed similarities with measured motion of two elite cyclist within the standard deviation. However, since the cyclist's hip position was fixed to the saddle, this motion does not fully cover the motions observed in the standing track cycling start were the riders rise from their seats to produce even higher powers.

A rather advanced model was constructed by Farahani et al. [51]. In this study a so called inverse-

inverse optimization process was used. As seen in the beginning of this chapter, for inverse dynamics kinematic data is collected from which the muscle activations causing the observed motion can be calculated. Usually in forward dynamic optimization an initial guess about the muscle excitations is made, which is then optimized for a certain criterion. In that case the excitations of the muscles are the independent variable that need to be optimized. In inverse-inverse dynamics, the excitations are no longer independent variable, but are dependent on the kinematics and external forces. Now, the movement has become the independent control variable. Farahni et al. used a full biomechanical human model with Hill-type modeled muscles built in the AnyBody Modeling System (AMS) [52], as depicted in Figure 2.3. They parameterized all kinematic degrees of freedom with time functions that controlled these motions. This leads to the advantage of fewer control parameters and the fact that all time steps can be solved independently, without the need of intermediate integrations. Farahani et al. used an openly available generic model of the human which was then scaled to match the height and mass of their participants. The strength of the muscles was scaled using a length-mass-fat scaling law presented by Rasmussen et al. [53]. This methods equates a linear scaling factor for the muscle force based on the fat percentage of the subject, since fat percentage could be used to calculate the total muscle mass compared to the total mass of the subject. With this approach the authors could predict the optimal cycling technique using different objective functions. As a results they found a set of objective functions resulting in an output motion which showed a good agreement (mean RMSE = 0.104) with experimental cycling data.

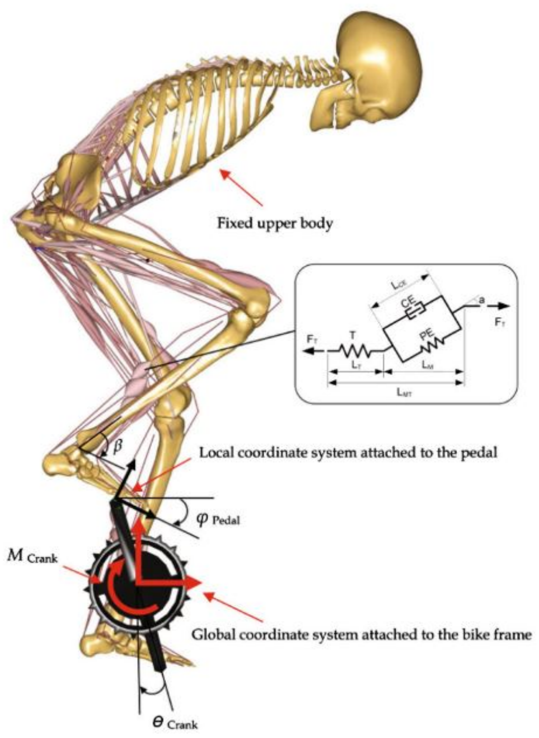


Figure 2.3: Full body biomechanical cycling model developed in AMS containing Hill-type muscles [51].

One of the most recent muscle controlled cycling models was developed by Bobbert et al. [54]. The objective of that study was to optimize the torque production during one full revolution of the cranks. The human model was constructed out of rigid bodies to represent the legs, connected by simple hinge joints and actuated by eight muscles per leg. Since the legs are mechanically decoupled, it was sufficient to model only one of the legs according to their reasoning. Compared to other pedaling optimization this model included more biological details about the activation dynamics. However, it was assumed that muscle could only fully activate or deactivate, resulting in a step-wise increasing activation ranging from 0 to 1. Moreover, the model was optimized for certain prescribed constant pedaling rates. The generic model was not scaled to match the subjects participating in the validation experiments. Eventually, the model was used to investigate the relationship between crank angular velocity and maximum torque production, rather than ideal cycling technique.

By far the most interesting study with respect to the goals of this work, is the model developed by Jansen [38]. In this model, a bicycle model, cyclist model and tire model were combined, just like the model of Cangley et al. [47]. However, this model does not use force input to ensure pedaling motion, but makes use of a cost function to optimize the starting movement of the cyclist. Since the model was only constructed for modeling the standing start, aerodynamic effects were ignored, since they have minor effects on the system at low speeds. Moreover, a steering assembly was left out of the model because during the start the cyclist drive in a straight line. Additionally the model was restricted to remain upright to prevent falling. These simplifications were made to reduce the computation time. Besides the bicycle model, a cyclist model was created, consisting out of ten rigid bodies representing the upper body and limbs. The hands were fixed to the handlebars and the feet to the pedals. Unlike other models, the trunk was not attached to the saddle to allow for an upright standing start techniques. To move the whole system of cyclist and bicycle, joint torque models were used. The model predicts the active joint torque that can be applied considering the current joint angle and velocity, based on the work of van Soest and Bobbert [55]. Besides, the torque limits need to be defined first, since the strength of an athlete is limited. The maximum joint torques found for competitive cyclists, determined in a study by Kordi et al. [56] were taken. Data at the upper end of the ranges were selected, since it was believed that an Olympic cyclist could produce even higher torques than the competitive cyclist measured by Kordi et al. Since every human is different this method seems pretty questionable and comparison with experimental data collected by a different athlete without any form of sensitivity analysis seems rather meaningless. It was further stated that "the maximum isometric joint torques were scaled up by 1.5 to represent the lower limb strength of an Olympic track cyclist." This factor seems to be selected more or less random, without any validation.

Using the model, the optimal joint torque activation for driving the largest distance in a certain time could be obtained, which was the cost function. However, since this model does not make use of rider specific characteristic, such as individual peak power, its ability to predict performance is limited, but it can be used to optimize the general starting dynamics and cyclist setup. Moreover, due to the large computation times, the authors stated that the solutions found are probably not the optimal solutions yet. Despite the high expectations of the optimization process, only small differences between the current starting technique used and the optimized technique were found. The main differences were due to the perfect timing and possibilities of abrupt acceleration and deceleration of body segments of the model, which is not feasible in real life.

2.3 Trajectory optimization

Most biomechanical models described above used some sort of optimization to find the best performance theoretically possible. The best performance can be interpreted in many different ways, for instance as the fastest or the option requiring the least amount of energy. There might exist several roads leading to that specific optimal solution. The goal of this section is to give a short and understandable overview of methods used for predictive optimization problems. These problems are sometime referred to as trajectory optimization problems. This means that it deals with problems that aim to find the best choice of *trajectory* by selecting the best control inputs for the system as a function of time [57]. The trajectory includes both the states (e.g. positions and velocities) and the controls as a function of time. These kind of problems can also be named optimal control problems, since it aims to find the best control possible for a specific goal.

Figure 2.4 shows an example of a trajectory optimization problem. In this example, the task is to hit the crossbar with the ball. The controls of the problem could be the horizontal and vertical velocity at which the ball should be shot. As illustrated in Figure 2.4, there are multiple paths that result in the ball hitting the crossbar. To define which is the best possible solution to a problem the trajectory optimization problem needs a goal. This goal is known as the *objective function*. This function must be maximized or, as more often, minimized. The evaluation of this objective function is used to determine when the solution is optimal. Possible objective functions could be to minimize the time needed to hit the crossbar, or minimize the start velocity of the ball.

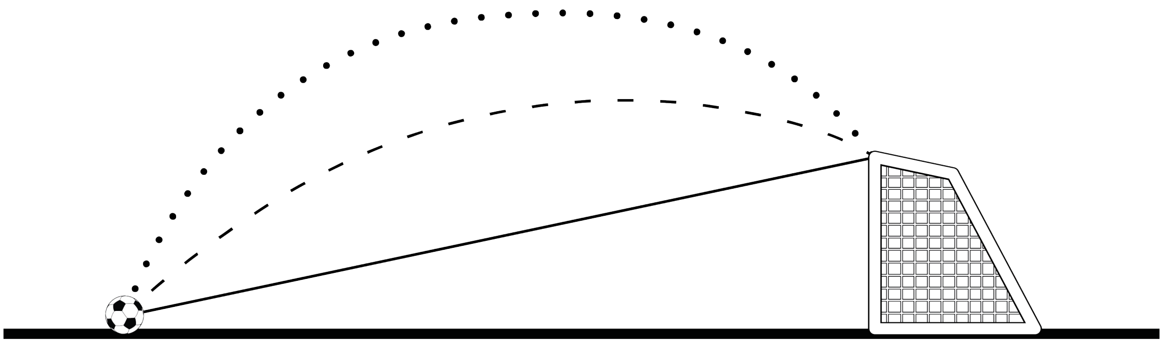


Figure 2.4: An example of a simple optimal control problem. If the task is to hit the crossbar, multiple trajectories exist that fulfill this task. An objective function should determine which trajectory is optimal for the task given. Figure adapted from Kelly [57].

A common objective is to minimize the energy needed to complete a certain task, for instance move from A to B. In that case the trajectory is bounded at the start and end point, using so-called boundary conditions. In these energy saving optimizations, the objective function is usually expressed as an integral function. A common objective function is the control effort squared, which would look as follows:

$$\min \int_{t_0}^{t_{end}} u^2(t)\,dt$$

Objective functions expressed in the integral form showed above are called objectives in Lagrange form. The other type of objective functions is called an objective in Mayer form. Mayer form objective function are used to minimize one of the states at the final or initial time step or the time itself. These functions are therefore also called boundary objective function. An objective in Mayer form is often recognized by a capital J. An objective function could also contain multiple goals either in Lagrange or Mayer form. If an objective contains functions in both Lagrange and Mayer form it is said to be in Bolza form.

Indirect vs Direct Methods Most methods for solving these kind of trajectory optimization can be classified as either direct or indirect methods. Indirect methods analytically determine the conditions for the optimal solution. This is then discretized and numerically solved. This can be best explained using a simple example. Let's assume that we have something that we want to minimize in the form of $y = f(t)$. We know from basic mathematics that y is minimal if $y'(t) = 0$ and $y''(t)$ is positive. Indirect optimization methods use the same principle, but usually the equations to solve are way more complex. One of the downsides of indirect methods is the fact that the equations of motion must be constructed analytically in order to find an accurate solution, which can be very difficult.

On the other hand, there are the direct methods. These methods will minimize the function $y(t)$ using an iterative process in such a way that each subsequent solution is better than the previous one. A distinctive feature of direct methods is that they discretize the trajectory itself. Whereas the original problem is a continuous function of time, the direct methods discretize this trajectory using polynomial splines, a function that is made out of multiple polynomials. Polynomials are really useful for these problems because they require a small number of coefficient. Besides, derivatives and integrals can be calculate relatively easy. Converting the continuous function into a discretized set of splines, is called transcription. Therefore, sometimes direct methods are referred to as direct transcription methods. Summarizing, direct methods first discretize and then optimize, whereas indirect methods first optimize and then discretize [57]. Regarding the complexity of the model that would be needed for the BMX problem of this work, direct methods are the better option.

Direct methods: shooting methods vs collocation Direct methods can be even further classified into a group called shooting methods and a group called collocation methods. The main difference between these two groups is that in shooting methods only the control variables are parameterized and in collocation methods both the control and the states are parameterized. In other words, shooting methods make use of a simulation based on an initial guess and the outcome of the simulation is then used to improve the guess. This is explained in Figure 2.5 with the use of the simple football example. The first iteration shows that the ball is too short to hit the crossbar. The improved guess, iteration 2, comes closer to the crossbar, but is still too short. Iteration 3 is again a little better than the previous guess and hits the crossbar. The process stops when a solution is found that is good enough, i.e. the error between the simulation outcome and the original target is small enough and the objective function is maximized or minimized. In reality more iterations would be needed to complete the goal. This principle works well when the control input is relatively simple and there are just a few path constraints. Because each iteration needs an entire simulation of the system, shooting methods can be relatively slow compared to collocation methods.

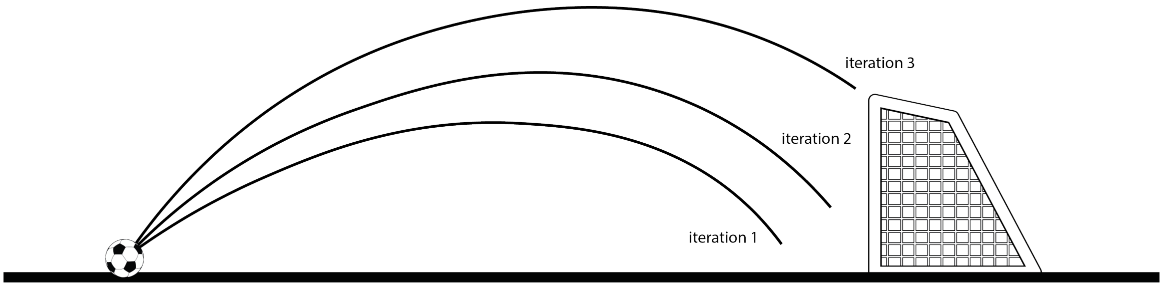


Figure 2.5: An example of how the crossbar task would be solved using single shooting methods. The outcome of each iteration is used to improve the next solution. Figure adapted from Kelly [57].

Collocation methods transcribe the continuous-time trajectory problem into a nonlinear program, often identified as NLP. This is just a fancy name for a constrained parameter optimization problem with the side note that it contains terms in either the objective or the constraints that are nonlinear. These constraints are the dynamics of the system. With collocation methods, the trajectory is split in several parts and the dynamic constraints are only obeyed at specific point along the trajectory, called collocation points. In some special cases the transcription of the optimization problem may not lead to a NLP, but rather just a linear or quadratic program. However, this is only the case if both the objective and the constraints are linear. To obtain such a NLP program, the original continuous-time problem must be discretized. This means that after discretization the continuous function is only represented by specific points in time. These points are called collocation points and this is where the method gets its name from. These collocation points are indicated by the dots in Figure 2.6. For the dynamics of the system this can be a tricky task. To discretize the dynamics it must be converted into a set of constraints. The main idea of these dynamic constraints is that the change in the state between two collocation points must be equal to the integral of the system dynamics. However, integrals are hard to solve analytically, so it is convenient to replace this term with an approximation. It turns out that there are several ways to make such an approximation.

Collocation: trapezoidal vs Hermite-Simpson In this overview two of those methods are discussed: the trapezoidal method and the Hermite-Simpson method. The first method, trapezoidal collocation, is the simplest. The integral is approximated using the mean of two consecutive values and the time step. This is probably more understandable in equation form. Say that $\boldsymbol{x}$ is the state vector containing all positions and velocities and that the vector $\boldsymbol{f}$ contains the dynamics of the system. Then the following equations apply for all collocation points.

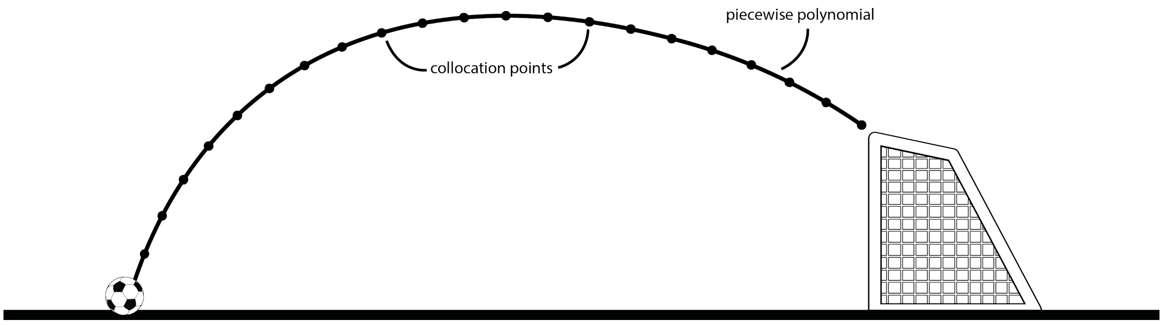


Figure 2.6: An example of how the crossbar task would be solved using collocation methods. The trajectory is split into different sections separated by collocation points. At each collocation point the dynamics must be obeyed. The collocation points are connected using piecewise polynomials. Figure adapted from Kelly [57].

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{f}\\ \int_{t_n}^{t_{n+1}} \dot{\boldsymbol{x}}\,dt &= \int_{t_n}^{t_{n+1}} \boldsymbol{f}\,dt\\ \boldsymbol{x}_{n+1} - \boldsymbol{x}_n &\approx h_n \frac{\boldsymbol{f}_{n+1} + \boldsymbol{f}_n}{2}\end{aligned}$$

The integral for the entire interval between t_0 and t_N is simply a summation of all the segmented approximated integrals. This approach would have been exact if $y(t)$ was a linear function for each segment $(t_{n+1} - t_n) = h_n$. The trapezoidal method thus approximates the system dynamics as being piecewise linear. This is illustrated in Figure 2.7, where the upper figure shows the linear approximation of some random function. Using this simplification it is possible to obtain a set of equations describing the dynamics between each set of consecutive collocation points, called collocation constraints.

The other option to simplify the integral function arising in the system dynamics is called the Hermite-Simpson collocation method. This method reaches higher accuracy because it approximates the dynamics as being piecewise quadratic instead of linear, as seen from Figure 2.7. The function is matched not only at the collocation or knot points, but also at the mid points of each segment. To calculate the integral between two consecutive points in time the Hermite-Simpson method makes use of Simpson's rule for integration as shown in Equation 2.3.

$$\int_{t_n}^{t_{n+1}} \dot{\boldsymbol{x}}\,dt = \boldsymbol{x}_{n+1} - \boldsymbol{x}_n \approx h_n \left(\frac{\boldsymbol{f}_n + 4\boldsymbol{f}_{k+\frac{1}{2}} + \boldsymbol{f}_{k+1}}{6} \right)$$

Summarizing, this overview shows that there are essentially two basic groups of methods to solve an optimization problem. The first group is called shooting methods and is characterized by its simple implementation but respectively long computation times. Its most suited for simple problems like the soccer example shown. The other group consists the direct collocation methods. Since these methods discretize the optimization problem itself into a NLP, they can be hard to implement. However, once setup these methods tend to be faster than the shooting methods. For the purpose of the BMX optimization direct collocation methods are the most interesting to use. Both trapezoidal and Hermite-Simpson collocation should be fine as long as the mesh interval is selected properly. The Hermite-Simpson is more accurate so should be chosen if computation time allows. In Chapter 3 it will be explained how these methods are implemented in the BMX SX model.

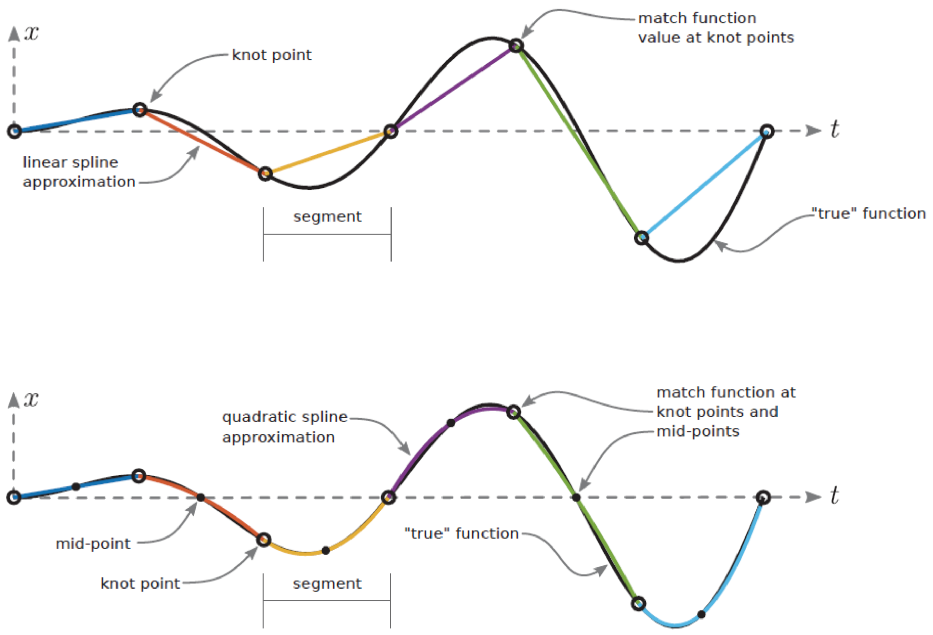

x
knot point
match function value at knot points
t
linear spline approximation
segment
"true" function
x
quadratic spline approximation
match function at knot points and mid-points
t
mid-point
knot point
"true" function
segment

Figure 2.7: The difference between trapezoidal (linear) and Hermite-Simpson (quadratic) collocation. The black line is a random "true" function that is approximated using a trapezoidal method (top figure) and Hermite-Simpson method (bottom figure) at the knot/collocation points. Figure adopted from Kelly [57].

3 Method

The general approach to reach the goal stated in Chapter 1.3 consists of three different parts. First a model of the BMX and rider system must be created. Next, experimental data about the BMX gate start, both kinematic and kinetic, must be collected and processed so it can be used for the model developed in step one. Finally, the model and the data must be combined into a tracking optimization problem. These three steps are described in this chapter.

3.1 Biomechanical Model

According to Chapter 2, authors who constructed simulations for predicting performance in sports may construct their model with different levels of human-like actuation. Some may use point masses to represent the human and their resulting forces on the system, while others accurately model the different human segments and their interactive muscles. The ultimate goal of this work, as stated in Chapter 1.3, is to track the kinematics of the BMX SX gate start. In order to do so, at least the most important limbs of the human athlete must be included in the model. Additionally, the model should be able to track the kinematic data while reproducing kinetic data. This could be done using inverse dynamic techniques. However, not all states are known from the experimental data. Besides, for future application a predictive model could have a way larger impact on improving start performance. The ultimate goal would thus be to construct a model that could track kinematic data and could thereafter be used for semi- or fully predictive purposes. As was stated in Chapter 2 this would be best described as a hybrid model.

Where for point mass models or other simple model approaches, it might be possible to derive the equations of the system by hand, this will be become more and more difficult once the complexity of the model increases. For multi-segmented human body representations, each additional limb will lead to more degrees of freedom and even more added constraints. For instance the revolute joint that connects the upper and lower arm will add five constraint equations to the system. Moreover, the muscle or torque models needed to actuate the model will add even more complexity to the model. Generally speaking, there are two options for constructing the model. The first would be to use more or less conventional multibody dynamics to construct the model using scripting in MATLAB or Python or any other platform that can handle these large sets of equations. This creates a lot of freedom for different applications. On the downside this would imply that all the equations must be manually derived to a certain extend. The other option would be to use a multibody dynamics software program that deals with all dynamic equations. This is often more user friendly, less time consuming and usually less error prone. On the downside, it reduces the possibilities and freedom of the developer. There are several multibody dynamical software packages available which could be used to created a bicycle rider system. Common examples are Adams [58] and MapleSim (Maplesoft, Waterloo Maple Inc., Waterloo, Ontario). However, these package do not include pre-built human models or usable actuation models, such as muscle models. Besides the general multibody dynamical software programs mentioned above there are also several software packages that are especially designed for musculoskeletal models. These packages include build-in muscle models and complex human joints, like the shoulder. Examples of such software packages are OpenSim [7] [8], AnyBody [52] and MSMS [59]. This latter was designed to simulate neural prosthetic systems and to test their performance before received by the patients and might be less suited for the BMX SX model. OpenSim and AnyBody are both very complete software packages including musculoskeletal system of humans which can be used to calculate individual muscle forces, joint contact-forces and moments or be used to perform inverse dynamic analysis. The benefit of OpenSim over AnyBody is the fact it is open-source and freely available. Moreover, OpenSim is known for its user-friendliness and its endless possibilities of implementing new tools into the software. It supports a large community which exchanges models and codes on an active forum. This is accompanied by a large number of tutorials and examples and documentation is published online which enhances learning the OpenSim skills needed to use the software. OpenSim can be used using the desktop application and the so-called Application Programming Interface. However, it is also possible to access OpenSim through C++, Java, MATLAB or Python interfaces, making OpenSim easy to work with for a broad community of users. Based on those arguments, it was chosen to use OpenSim for constructing the BMX model.

3.1.1 OpenSim Examples

OpenSim is nowadays the most used software package for simulating biomechanical models. It the recent years, a large number of studies have been presented making use of OpenSim. It has for instance been used to track motion of football players or identify adaptations to running technique in order to prevent injuries [60]. It has also been used for the purpose of making predictions about the cycling motion. Dudum et al. [61] were interesting in predicting the knee joint moment based on motion measurements of ergometer cycling. They used a full body human model provided by Hamner, Seth and Delp [62], which included the upper and lower extremities with five degrees of freedom each. The lower extremities were actuated by 92 individual musculotendon actuators, whereas the upper extremities were actuated by torque driven actuators. The joint actuation was obtained from measurements using inverse kinematics and inverse dynamics.

Another study that used OpenSim to study cycling was presented by Cardoso de Sousa and colleagues [63]. They studies Functional Electrical Stimulation (FES) cycling, which allows patients suffering from spinal cord injuries to ride a bicycle. The authors combined simplified generic available models for the lower limbs. One adaptation was the locking of the lumbar, pelvis and ankle rotation, leaving only the hip and knee rotations as degrees of freedom. To ensure that the feet stayed connected to the pedals they created a crankset with pedals and foot supports, which locked the feet using contact geometries as shown in Figure 3.1. The feet are connected to the blue foot support which has an empty space. This empty space is filled with the yellow pedals, connecting the pedal to the foot. The reason they constructed this in a rather complex way is the fact that a closed kinematic loop arises when just extra joints are used. The complete model for FES cycling was then used to analyze the effect of different control strategies on pedalling technique.

A final study using OpenSim for cycling purposes investigated the powercadence relationship in endurance cycling [64]. They used a simple two-legged generic model presented by Delp et al. [65]. They further constrained the model by relating the pedal angles to the crank angle using an equation provided by Redfield and Hull [37]. Each leg was actuated by 18 individual muscles per leg. They used the model to predict the mean total muscular power that could be generated as a function of the angular velocity of the cranks. This muscular power can be transformed into the maximum power output. This results in a power-cadence graph that could be compared to experimental measurements.

Finally, a recent work by Haralabidis at al. [66] used OpenSim to track the first meters of a running sprint race. The goal was to construct a model that could reproduce the experimental dynamics, including both kinematics and kinetics. The model created was validated by the experimental data with a RMSD of 1.0° or 2.0 cm. For the contact forces a difference of 8.1% was found. This validation ensures that it captures the essential characteristics of the sprint start, making it a framework for future research regarding optimization simulations in athletics. The model could be used to analyze the effects of adaptations to technical aspects and investigate a variety of "what-if" scenarios. This is more or less the same aim as the goals for this work, so it is interesting to have a closer look to their

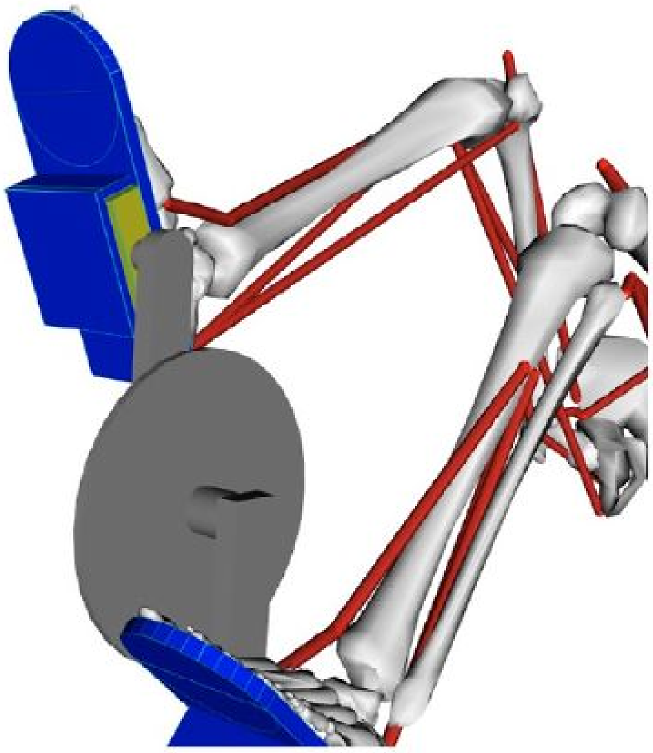


Figure 3.1: Close-up of the pedal-feet connection used by Cardoso de Sousa et al. [63]. The blue foot support is surrounding the yellow pedal which connects the two bodies.

approach. The authors started with the same generic model [62] as was used by Dudum et al. [61]. A number of small adaptations were made resulting in a total of 37 degrees of freedom. These were controlled by 92 Hill-type models for the lower limbs and 14 joint actuators for the upper limbs. To scale the strength of the model to more accurately represent the ability of the athlete, the strength parameters of the generic model were simply doubled. The optimization process was formulated as a set of nonlinear programming problems (NLPs) in MATLAB making use of CasADi and solved using a so-called flipped Legendre-Gauss-Radau direct collocation method.

The examples above show that creating a cycling model with OpenSim is perfectly possible, although some strange model choices were made. Nevertheless, none of these cycling studies used dynamic optimization. Besides the OpenSim cycling models, it showed an example of tracking and reproducing dynamic experimental data using OpenSim and direct collocation optimization. It would be interesting to combine the cycling model with the optimization approach of Haralabidis et al. [66]. The following sections will explain the model choices made for the BMX model and how the model parameters are derived. The model suited for the purpose of this project must consist of three components: the ramp including the gate, the BMX bicycle and the human athlete. The next sections will describe these three parts of the model, starting with the bicycle model.

3.1.2 Bicycle Model

A bicycle is a rather complex construction which contains a high level of instability, non-holonomic constraints, and a set of coupled degrees of freedom. For the purpose of modeling and especially predictive modeling, it is usually best if model complexity can be reduced to the bare essentials. It is always a quest for the optimal balance between complexity and good representation of reality on one side and simple, but thus simplified, representations on the other side. The first will lead to complicated equations which most certain lead to large computation times, while the other edge is too simple to be of any use, since it does not contain the essential complexity to match reality. For the bicycle model, the goal was to leave out anything that has minor influence on the outcome of the model. For instance, since the bicycle is unstable, the riders needs to balance it. However, the riders are very good in balancing the bicycle, so it likely does not influence their pedalling performance. Leaving out the roll angle of the bicycle reduces the complexity massively while it should have minor impact on the models performance. This and other model choices result in the bicycle model design seen in Figure 3.2.

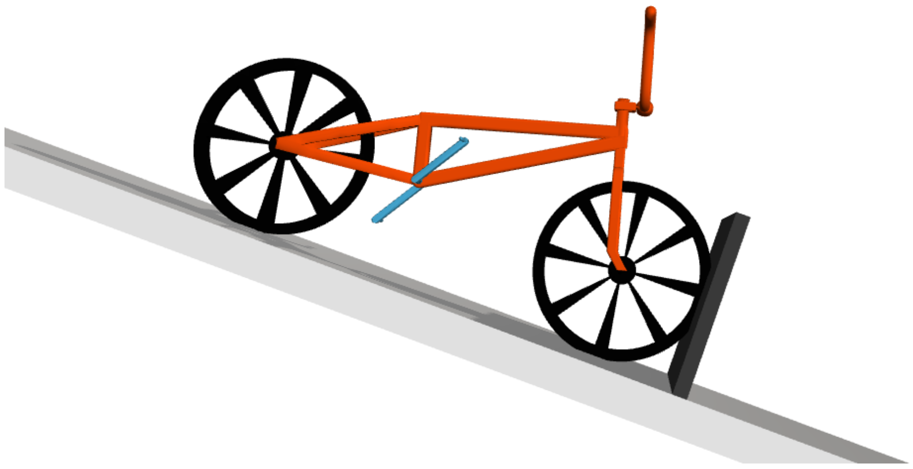


Figure 3.2: The bicycle model created in OpenSim. It consists of four rigid bodies: the frame (orange), the two wheels (black) and the crankshaft-crank body (blue).

The bike consist of essentially four parts: the frame, the crankshaft and cranks, modeled as one, and the two wheels. The frame (displayed in orange) is completely constructed out of one part. This is a simplification of a realistic bicycle, since there is no steering assembly included. The front fork and steer are directly welded to the rest of the frame. The riders travel down the ramp in an almost perfect straight line because usually there are seven other riders beside them, so including a steering angle is not necessary. The crankshaft and cranks (displayed in blue) are made out of one part too. It is connected to the frame using an optimal rotational joint. There are no pedals attached to the ends of the crank because, as will be explained in the following sections, the pedals can be included using a sophisticated way to connect the human feet to the cranks. Finally the wheels are, just like the crankshaft, connected to the frame using perfect pin joints. As seen there is no chain in the model as seen in real bicycles. Modeling a chain and its interaction with the sprockets would be way too complex considering the purpose of the chain, which is a connection between the rotations of the crankshaft and the rear wheel. In OpenSim this connection can be made fairly easy using a so-called CoordinateCouplerConstraint[1]. This constraint, as the name suggests, couples two coordinates with a user defined function. If for this function the transmission factor is chosen, the CoordinateCouplerConstraint functions as a perfect gearing system. A final remarkable detail about the model as seen in Figure 3.2 is the fact that it does not contain a seat. Since BMX riders never sit on their seat during the race, it is not included in the model. The seat is only obligatory due to safety reasons [18], but that is not important for the model.

The entire design of the four parts results in a bicycle with four degrees of freedom: one translation and one rotation for the entire bicycle plus the rotations of the crankshaft-rear wheel system and

[1]https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1CoordinateCouplerConstraint.html

the front wheel. The bicycle in itself is thus a planar model which cannot move laterally, just like a real bicycle, and does not include a roll and yaw degree of freedom. Leaving out these degrees of freedom massively influences the complexity of the model while not taking out essential aspects of the start. To further constrain the model, the frame cannot move vertically. Hereby the rear wheel is forced to stay in contact with the ground which is just as seen in real BMX starts. Lifting the rear wheel from the ground would take out the possibility to generate a propulsive force forward and is thus not beneficial. Moreover, constraining the vertical displacement of the rear wheel massively simplifies the contact dynamics with the ground as will be explained in the upcoming sections. The rotational degree of freedom of the frame corresponds to the possibility of making a wheelie manoeuvre to go over the gate with the front wheel before it has completely dropped.

In OpenSim there are multiple parameters needed for each individual body, namely: dimensions, mass, center of mass and inertia. In order to create a model that is close to reality, these values must be determined experimentally or based on valid approximations as explained in Chapter 2.2.2. An official BMX SX bicycle which was used by a rider from the Dutch national BMX team was used for these experiments. The following section will describe the procedures and experiments executed to determine the aforementioned parameters.

Dimensions

The dimensions of the BMX bicycle are key to accurately model the bike. However, some parameters are more critical than others. In essence only a couple dimensions are important: the attachment points for the wheels, hands, and crankshaft must be on the right place. The shape of the frame does not really matter, as long as the locations of these attachments points are correct. A common measure in bicycle design is the distance between the rear and front base, indicated as the wheelbase. Other key dimensions are the chainstay length and the seat tube and head tube angles. These dimensions can either be measured or retrieved based on information provided by the manufacturer. The BMX frame is a Meybo Holeshot model 2018, which comes in different sizes ranging from M to XXL. The dimensions of the frame are different for each of these frame sizes. In this study, it was chosen to measure these dimensions directly because it was not entirely clear what the frame size of the bike was. The frame dimensions needed for the OpenSim model can be found in Figure 3.3. All dimensions are expressed with respect to the rear hub. The rear wheel will be in contact with the ground during the entire start, which makes it the ideal base point. The black dots in Figure 3.3 are the attachment points for the wheels, the crankshaft, and the hands. The top most dot is the location of the center of the two handlebar-ends. The dots and dimensions in grey indicate the location of the center of mass, which will be discussed next.

The dimensions of the other bicycle components are a little easier to measure. It is known that the wheels of BMX bikes are 20 inches, which corresponds to a diameter of 50.8 cm. The length of the cranks is also known as 17.5 cm. These dimensions, provided by the manufacturer, were checked by hand and found to be correct.

Mass

Probably the easiest parameter to determine is the mass of each section of the bicycle. In the most ideal situation the masses of each component would be determined exactly. However, it is most important to have a general idea about the masses of the different components in order to reach forces and torques that are close to reality. A few grams will not have large influences on the final outcome. Moreover, due to the peculiar circumstances of the past year it was preferred to do most of the experiments at home. Therefore, the masses were measured using common-or-garden equipment, in this case an ordinary Soehner bathroom scale with a resolution of 0.1 kg for the frame. This does not provide a very high accuracy but since this is not necessarily required it will do the job. For all other parts an IKEA Ordning scale was used with a resolution of 1 gram and a maximum load of 3000 grams. Each bicycle part was measured five times and the mean values were taken as inputs for the models. An overview of the component's masses can be found in Table 3.1. Eventually the mass of the crankshaft, crank and pedals were combined into one mass, since their modeled as one component. The right crank was slightly heavier because it included the chainring. The mass of the chain was added to the mass of the frame. Note that the rear wheel is heavier than

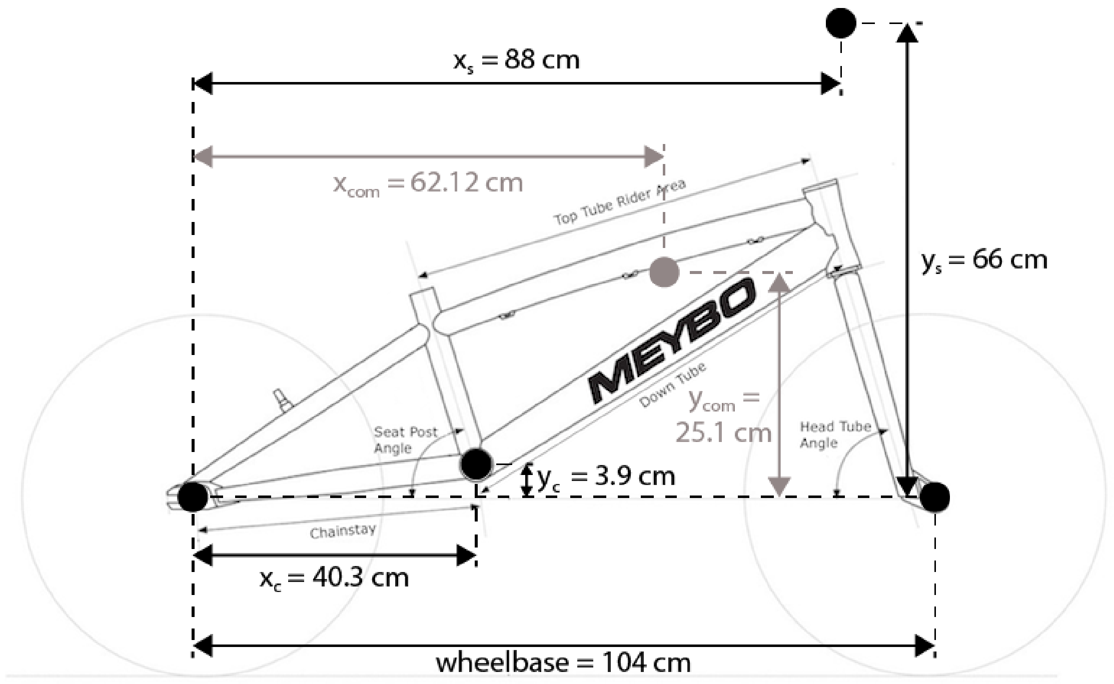


Figure 3.3: The Meybo Holeshot main dimensions needed in OpenSim. The black dots are the locations for the attachment points of the wheels, crankshaft and the hands expressed from the rear wheel location. The grey dot is the center of mass location.

the front wheel since it also contains the rear sprocket and a disc for the disc-brake system as seen in Figure 1.3.

Bicycle component	Mass (kg)
Frame	5.2
Front wheel	1.280
Rear wheel	1.579
Chain	0.319
Crankshaft	0.063
Crank + pedal left	0.745
Crank + pedal right	0.601

Table 3.1: Measured mass for each of the bicycle components.

Center of mass

The center of mass is a little harder to determine experimentally. Moore et al. [67] have clearly explained a way to determine several bicycle parameters including the center of mass and moments of inertia. These methods will be used to determine the center of mass of the BMX components.

Frame To determine the center of mass of the frame it can be hung in three different orientations as explained in the article presented by Moore et al. [67]. This can be seen in Figure 3.4. The orange lashing strap was used to fixate the steering assembly. Once at rest the frame will be positioned with the center of mass exactly located under the point of attachment. A string connected to a mass was connected from the point of attachment vertically down, crossing the center of mass, as seen in Figure 3.4. Once everything was at rest the string was fixed to the frame using tape. Executing this procedure three time, for three different attachment points, will results in three strings stuck to the frame. The intersection of these three strings will give an estimation of the center of mass, shown in Figure 3.5. It would have been more precise to calculate the angle of the headtube with respect to the ground as explained by Moore et al [67]. However, since the experiments were executed at home with the absence of an angle measurement device, this alternative will give a reasonable prediction of the center of mass. The location of the intersection between the three strings was measured with respect to the rear wheel center and visualized in Figure 3.3.

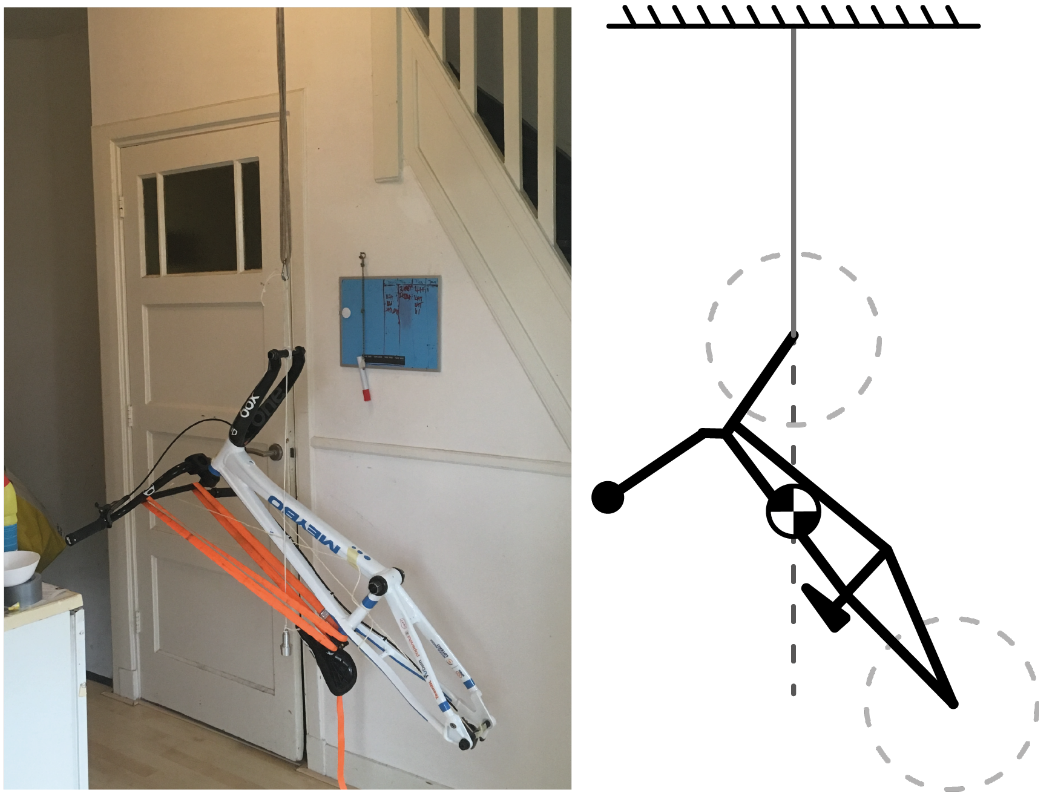


Figure 3.4: Center of mass experiment setup. The bike was hung from the ceiling in three different orientations of which one of them is shown in this figure. A mass on a rope was attached at the hanging-point of the bicycle. As seen from the schematic drawing on the right, the intersection of the ropes from the different orientations will give the center of mass. The orange lashing strap fixates the steer.

Wheels As explained in the work of Moore et al. [67], it is assumed that the center of mass for the wheels coincides with the geometrical center.

Crankshaft and cranks Moore et al. do not explicitly mention the calculation of the center of mass of the crank assembly. Since the crank setup is more or less lateral symmetrical it is assumed that the center of mass is at the geometrical center as well.

Inertia

Frame The moments of inertia can according to Moore at al. [67] be calculated using Equation 3.1. In this equation, T is the period of oscillation when the frame would be hung like a pendulum, m the mass of the frame and L the distance between the point of attachment and the center of mass.

$$I_{zz} = \left(\frac{T}{2\pi}\right) mgL - mL^2 \tag{3.1}$$

Using the experimental setup shown in Figure 3.6 the frame could oscillate like a single pendulum. Again, the orange lashing straps were used to fixate the steering assembly. The rear wheel was fixed to a horizontal beam in such a way that the rear axis could be used as the axis of rotation of

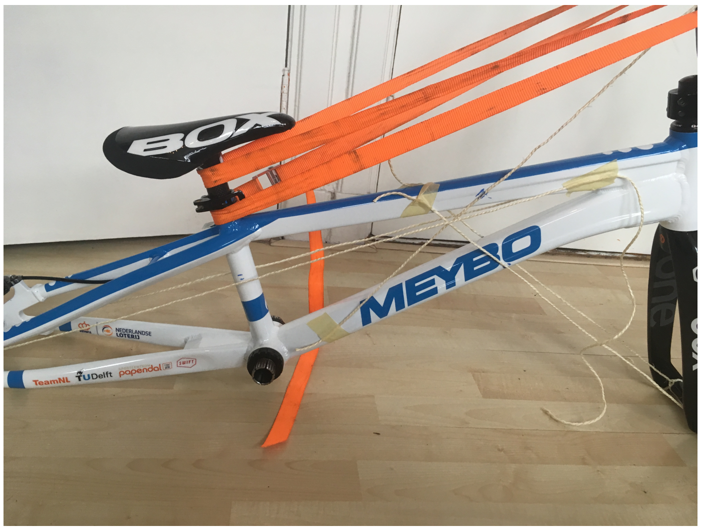


Figure 3.5: The center of mass of the BMX SX frame. The three ropes cross at approximately the same location.

the pendulum. The mass and center of mass are known from the previous experiments, so the only thing to be determined is the period T. Moore et al. used a gyro to capture the dynamics of the swing and calculate the period. A simpler and easier approach used here is to capture the swing with a camera and use a video software program to measure the displacement of one single point on the frame. In this experiment Kinovea (Kinovea version 0.8.15, kinovea.org, France) was used, which is a free software program to analyse videos widely used for sport applications. As seen from Figure 3.7 a point on the handlebar was tracked during the oscillation. This can be done by hand, but Kinovea is also able to track points automatically. The position data from Kinovea was then analyzed by loading the data into MATLAB (MathWorks, Inc., Natick, USA), which resulted in the graphs shown in Figure 3.8.

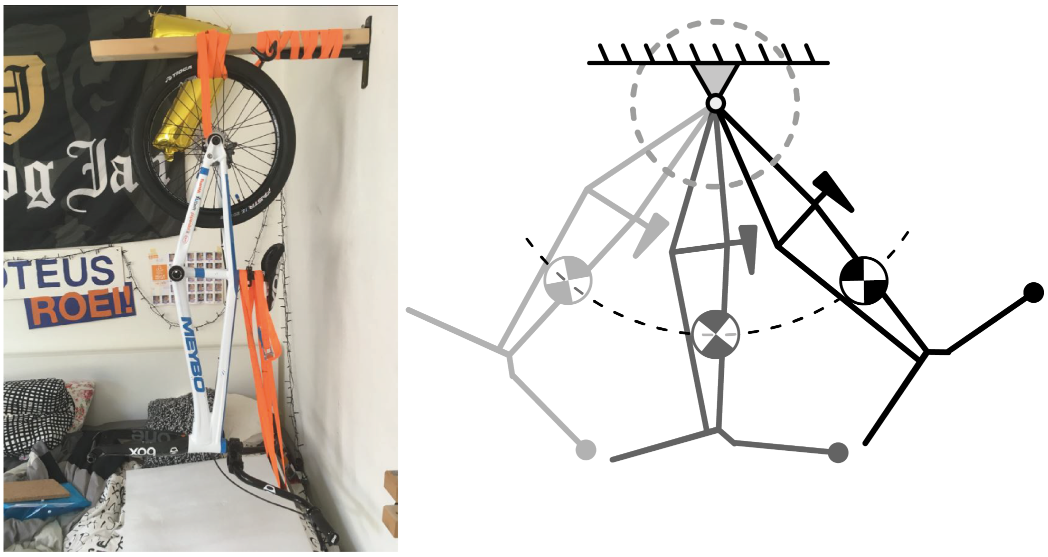


Figure 3.6: The experimental setup to determine the oscillation time of the frame. The rear wheel is used as the rotation point. The orange lashing strap is used to fixate the steering assembly. The right figure shows an exaggerate overview of the pendulum motion of the bicycle. As seen, the center of mass will swing in a perfect circle.

From the graph in Figure 3.8 the period T can be calculated fairly easy by measuring the distance between two peaks, which is approximately 1.9 seconds. Now that the period is known, Equation 3.1 can be used to calculate the inertia of the frame. From the center of mass location in Figure 3.3

the distance L as found in Equation 3.1 can be measured directly. The distance between the rear hub and the center of mass was 67 cm, which results in an inertia of 0.79 kgm^2.

The complete frame was constructed in OpenSim with the dimensions, mass and inertia found in the sections above. In OpenSim a virtual pendulum test was simulated and the results were compared to the experimental data. In Figure 3.8 the yellow line shows the results of this virtual test. It can be found that the virtual period matches the experimental period within an error of 1%.

Wheels The inertia of the wheels has not been determined experimentally but was calculate using basic geometry. It was assumed that the largest part of the weight was distributed along the rim of the wheel. The wheels have an outer radius of 25.4 cm, but the distance to the center of the rim is close to 22 cm. This latter dimension was chosen as the radius of gyration (ρ_{wheel}) of both wheels. The inertia can than be calculated using equation 3.2

$$I_{wheel} = m\rho_{wheel}^2 \tag{3.2}$$

Using this approximation the inertia of the rear and front wheel are 0.0764 kgm^2 and 0.0620 kgm^2 respectively.

Crankshaft, cranks and pedals The inertia of the system of cranks, crankshaft and pedals is a little hard to measure experimentally. Therefore, it was chosen to split it into multiple parts: first the inertia of the cranks, second the inertia of the chainring and finally the inertia of the pedals. The inertia of the crankshaft is neglected since it mass and size are considerably small compared to the other parts. The inertia of each of these individual parts can be calculated fairly easy by assuming it are simple shaped objects of which the inertia can be calculated using the dimensions and mass only. The cranks can be seen as simple circular rods with a length L_{crank}. The inertia can then be calculated using the first part of equation 3.3. The chainring can be seen as a circular disk with radius $r_{chainring}$ with corresponding inertia calculated with the second term in Equation 3.3. Finally the pedals are modeled as point masses at the end of the cranks. The inertia is then only depending on the mass of the pedals and the lengths of the cranks as seen from the final term in Equation 3.3.

$$I_{crankshaft} = 2\left(\frac{1}{3}m_{crank}L_{crank}^2\right) + \frac{1}{2}m_{chainring}r_{chainring}^2 + 2\left(m_{pedal}L_{crank}^2\right) \tag{3.3}$$

With the mass, center of mass and the inertia of the four bicycle parts measured or calculated the information needed for the OpenSim bicycle model is complete. However, an important feature is

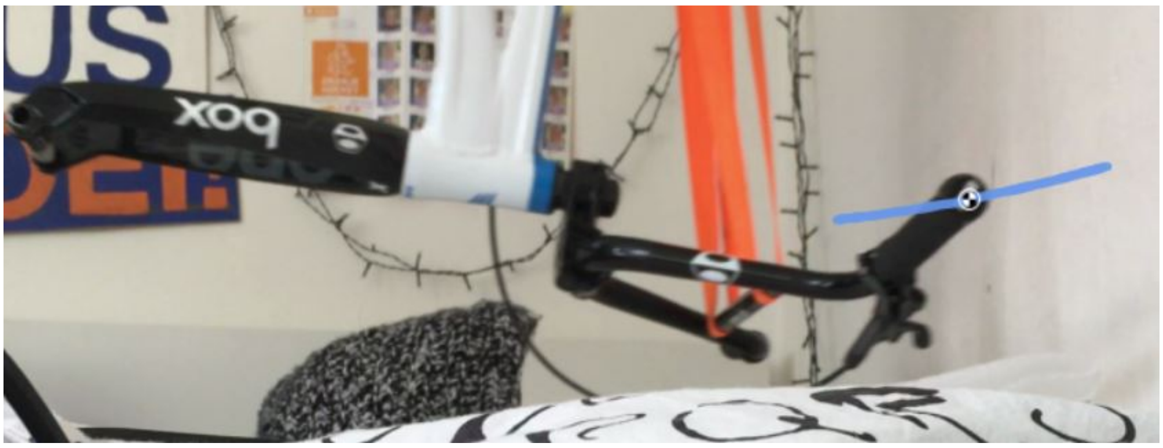


Figure 3.7: The Kinovea tracking of the handlebar tip during the pendulum test. The blue line shows the trajectory of that specific point in time.

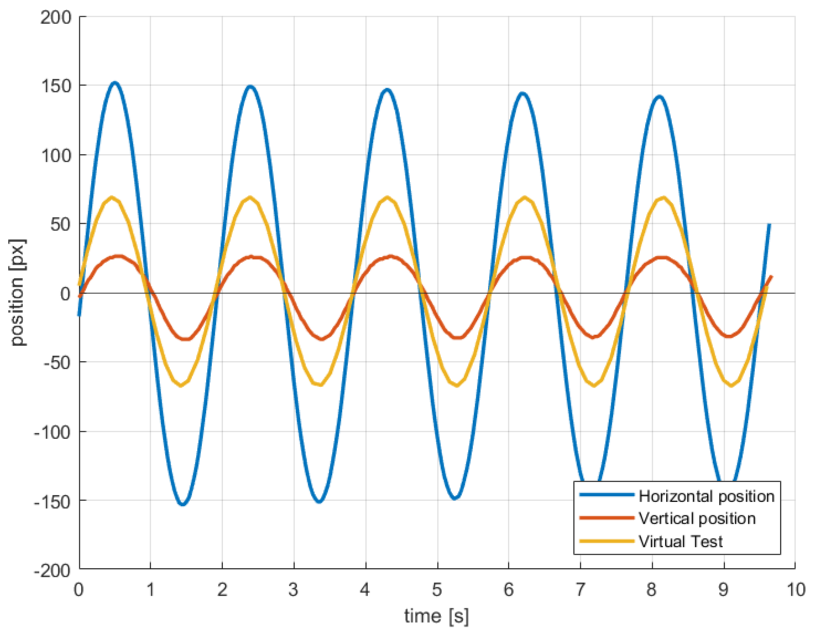


Figure 3.8: The results of the pendulum moment of inertia experiment. In blue the horizontal displacement is shown in pixels and in red the vertical displacement. The yellow line is the result of the of the virtual experiment in OpenSim in arbitrary units.

still missing, namely the interaction with it surroundings. As seen in Chapter 1.1, the BMX SX race starts from a huge ramp behind a gate. The interaction with the ground surface, its slope and the timing of the gate is very important. The next section will discuss these topics in more detail.

3.1.3 Contact modeling

For the bicycle to be able to propel itself, there must be interaction with the ground. Due to the grip between the rubber tires and the ground surface a bicycle can move forward by rotating the rear wheel via the drivetrain. In OpenSim, there are two main ways to model the interaction between two surfaces [68]. The first is using the so-called Hunt-Crossley model [9]. This approach is based on the Hertz contact theory and analytically calculates the interaction forces based on elastic parameters and the deformations. Its main feature is that it uses a sophisticated way to express the coefficient of restitution as a function of speed. Although it is an analytical model, it can compute interaction pretty quick. A disadvantage of this approach is the fact that it is limited to some basic geometrical shapes like planes, spheres and ellipsoids. For modeling the wheel-to-ground interaction two spheres and a plane would be just fine. Since the bicycle itself is a planar model, the wheel could be modeled as spheres instead of two disks. However, a step-wise increasing slope, like the BMX start ramp seen in Figure 1.1 would not be possible, since the contact planes would intersect. The other option is to make use of an Elastic Foundation model [69]. This uses a mesh grid to represent a surfaces, so it can be used to model the contact of every shape of interest. It calculates the interaction forces based on a 'bed-of-springs' elastic model, which comes down to modeling a tiny spring at every mesh center. To smoothly model the interaction between a plane and a rolling wheel, the mesh density must be very high, otherwise the wheel would fall down within every two springs. For exactly this reason, the method can be very expensive in term of computation time.

An entirely different option is to not model the contact at all, but rather use constraints to capture the interaction between the wheel and the ground. OpenSim provides a way to include a rolling on surface constraint consisting out of a set of constraint equations to prevent slip and twist. However, adding constraints is usually not beneficial for the complexity of the model, so contact modeling is usually the better option. For the BMX model it was therefore chosen to use a Hunt-Crossley model to model the interaction between the rear wheel and the ground. Because it is not possible to model the kink or the transition between the section before and after the gate, the ramp was modeled with a constant slope as explained in the next section.

The same approach was used for the contact between the front wheel and the ground. Although there is no actuation transmitted via this contact, the forces are important for the balance of the bicycle and the possibility to lift the front wheel from the ground. A second contact plane was used for the contact with the gate. Because this plane is attached to the gate, it rotates together with the gate. Once the gate is fully opened, the contact planes of the ramp and the gate are aligned. Positioning the rotation point of the gate slightly below the surface of the ramp results in the gate contact plane ending up underneath the contact plane of the ramp. This is important because it basically removes the gate contact plane from the model once the gate is fully opened.

Hunt-Crossley contact model The Hunt-Crossley contact model exist out of two different components. First, there is the normal force component and secondly the friction force component. The normal force is perpendicular with the surface, while the friction force is parallel to the surface. The main equation that is used to calculate the normal contact force between two objects using the Hunt-Crossley approach is given by Equation 3.4.

$$F_n = kx^n \left(1 + \frac{3}{2}cv)\right) \tag{3.4}$$

In this equation, k is the stiffness depending on the material and the geometry, x the penetration depth and v the penetration rate. The exponent n in Equation 3.4 depends on the surface geometry. For an interaction with a sphere and a plane, which will be used for the wheels and the ground surface, n equals $\frac{3}{2}$. The stiffness k can be calculated using $k = \frac{4}{3}\sqrt{R}E$, where R is the relative radius of curvature and E the effective plane-strain modulus. These two parameters, R and E, are dependent on the two surfaces, so on the two radii of curvatures (R_1 and R_2) and the two elastic moduli (E_1 and E_2). R and E can be calculated using Equations 3.5 and 3.6 respectively.

$$R = \frac{R_1 R_2}{R_1 + R_2} \tag{3.5}$$

$$E = \left(\frac{E_1^{\frac{2}{3}} E_2^{\frac{2}{3}}}{E_1^{\frac{2}{3}} + E_2^{\frac{2}{3}}}\right)^{\frac{3}{2}} \tag{3.6}$$

In conclusion, the stiffness k can be calculated if the radii of curvatures and the elastic moduli are known. For the ground surface and the tires of the bicycle, these parameters could be measured or at least be well approximated. The parameters x and v are simply a results of the simulation and are not characteristic parameters of Equation 3.4. The final term in Equation 3.4, c, is perhaps the most complex part of equation. As mentioned before, the key aspects of the Hunt-Crossley method is the dependence of the coefficient of restitution on the velocity. The coefficient of restitution can be calculated using: $e = (1 - cv)$, where c and v are the same as in Equation 3.4. c is a material property that depends on the slope of the coefficient of restitution versus velocity curves at low velocities. This is extremely hard to determine experimentally. Luckily, Equation 3.4 can be simplified, since it was assumed that the bicycle does not translate in the vertical direction. As is illustrated by the schematic drawing in Figure 3.9, because the bicycle has no degree of freedom in the vertical direction, the penetration distance x is fixed. This means that the penetration rate v is zero, meaning that the normal component of the contact force is constant. The equation simplifies to $F_n = kx^n$, which eliminates c from the equation. It is therefore no longer a parameter of interest.

The equations above are used to calculate the normal force component of the contact force. To calculate the friction force, the Hunt-Crossley approach in OpenSim uses a formulation based on a model by Michael Hollars and can be calculated using Equation 3.7.

$$F_f = F_n \left(\min\left(\frac{v}{v_t}, 1\right) \left(\mu_d + \frac{2\left(\mu_s - \mu_d\right)}{1 + \left(\frac{v}{v_t}\right)^2} \right) + \mu_v v \right) \tag{3.7}$$

In Equation 3.7 F_n is the normal force, calculated using Equation 3.4, v_t the transition velocity, and μ_s, μ_d, and μ_v are the coefficients of static, dynamic, and viscous friction respectively. The transition velocity v_t is the speed at which the peak static friction occurs.

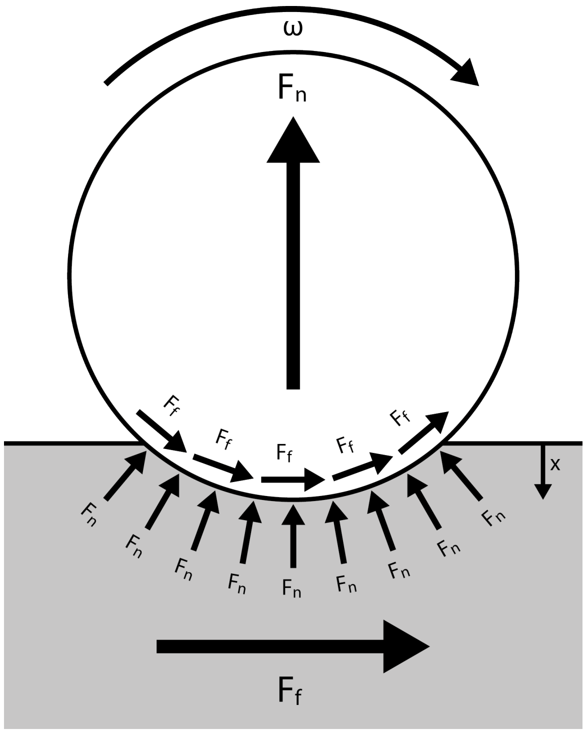


Figure 3.9: Normal and friction force components. The penetration distance x, which depends on the clamping distance of the bicycle, is fixed in the model. This results in a constant normal force.

Combining Equations 3.4 and 3.7, it can be shown that the contact dynamics is dependent on six different parameters: stiffness (k) and the material property c for the normal force and three types of friction (μ_s, μ_d, and μ_v) and the transition velocity (v_t) for the friction force. These parameters must be determined to achieve a realistic contact between the ground surface and the wheels. Determining these parameters experimentally, like done for some of the mass properties, would be a study on its own. In this work it was chosen to make estimations for the contact parameters based on two simple requirements. First, the rear wheel must not slip while applying forces on the cranks, in the range of the forces known to be applied on the pedals by the athletes. Secondly, the torque needed to move the bicycle forward must be in the range experimentally measured in previous studies. Based on initial guesses based on simple experiments, iteratively the contact parameters could be determined as was suggested in Chapter 2.2.2.

It turned out that there were three main contact parameters that influenced the outcome of a simulation: the clamping height (the position of the rear wheel center with respect to the ground), the stiffness and the coefficients of friction. The clamping height is essentially the same as the penetration depth x seen in Figure 3.9, but it also includes the radius of the rear wheel. Guessing these parameters based on literature is challenging because the parameters relate to both the tire and the ground surface characteristics. It was therefore decided to make initial guesses based on two simple experiments: a rolling test and a bouncing test. For the rolling test, the same BMX bicycle as was used in the mass and inertia experimented was placed on a small ramp at a bridge in the center of Delft. The bicycle passively rolled down this small ramp with a person on top of it. The incline of the small ramp was measured using a smartphone and it was assumed that this incline was constant over the entire ramp. The measurements most of the time showed an incline of 2° for the larger part of the hill, but showed an angle of 1° for a few sections. Since the resolution of the smartphone was just 1° it was estimated that the ramp had a constant slope of 1.75°. Starting from a stand-still the bicycle and rider slowly rolled down the ramp. A camera captured the kinematics of the bicycle so the displacement could be retrieved from video analysis using Kinovea. The setup was recreated in OpenSim by modeling the bicycle only with an additional mass attached to the virtual seat to mimic the rider on top of the bike. The bicycle was placed on a virtual hill with the same slope as the ramp of the bridge. Starting from a stand-still the OpenSim forward dynamics tool simulated the bicycle rolling passively down the ramp. The horizontal displacement of the simulation outcome was compared to video data. Based on these results, the contact parameters were adjusted manually and their effect on the distance travelled was evaluated by rerunning the simulation. The final fit shown in Figure 3.10 was created using a stiffness of $1 \cdot 10^8$ N/m, a clamping height of 0.3 m, and friction coefficients which all equaled one. For simplicity it was assumed that all friction coefficients were equal. Because there are multiple contact parameters it was noticed that there existed multiple combinations which led to a good fit with experimental video data. However, since one of the requirements was that the wheel must not slip when large crank torques are applied, this combination of parameters showed to be a valid choice. The evaluation of these parameter choices will be discussed in slightly more detail in Chapter 5.

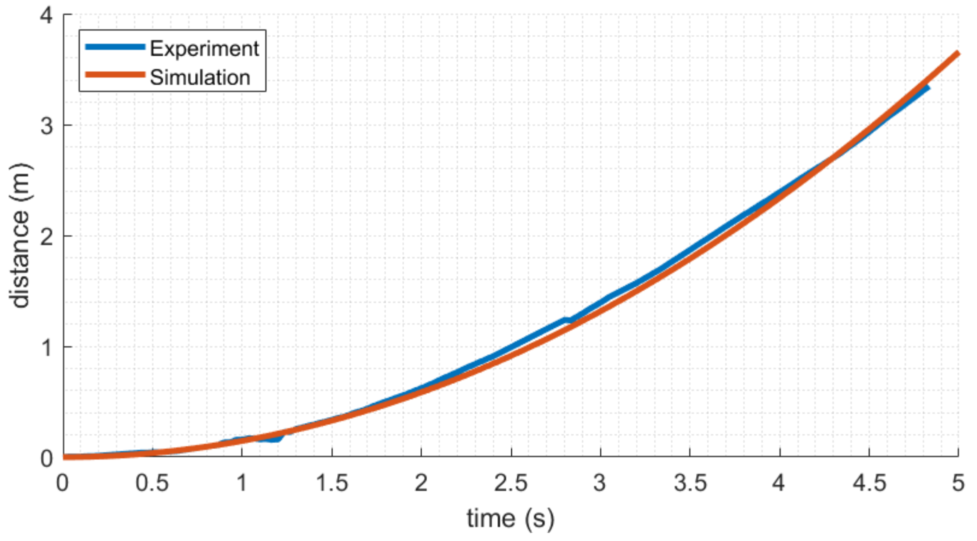


Figure 3.10: Bicycle displacement in time for the rolling experiment and the virtual rolling test. These results were obtained using a stiffness of $1 \cdot 10^8$ N/m, a clamping height of 0.3, and all friction coefficients set to 1.

The rolling test provides a well educated guess for the stiffness, the clamping height, and the friction coefficients. The final parameters, the material property c, in OpenSim also referred to as the dissipation, can be determined using a bouncing test. Because the dissipation tells something about the energy being lost during impact, the parameters c can be guessed based on the rebound height of the wheels. For this experiment one of the wheels was dropped to the ground while its kinematics were captured using video. This video was further analysed using Kinovea. The experiment was copied in OpenSim and the same procedure as for the rolling test was used, i.e. adapting the dissipation manually to get a good agreement between experiment and simulation. Figure 3.11 shows the results when the dissipation was set to 0.08.

Slope of the ramp The slope of the ramp is important because it determines the contribution of gravity to the force pointing in the forward direction. As seen from Figure 1.1, the slope of the official Olympic starting hill starts with an angle of approximately 18° which step-wise increases to 28° during the descend. Due to the impossibility to model this discontinuous surface in OpenSim, the ramp must be modeled as being a straight surface with a constant incline. Luckily, this was no

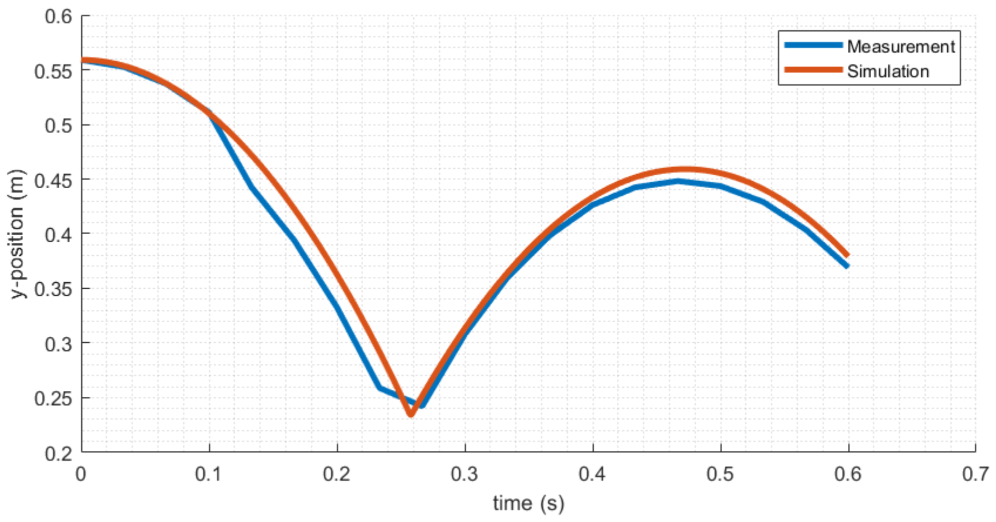


Figure 3.11: Vertical displacement of the rear wheel while dropped to the ground experimentally and virtually. This result was obtained using a dissipation of 0.08.

issue since the kinematic data was collected on the training ramp at the Papendal BMX facility. Van Dilgt [11] determined the slope of the training ramp to be 18°.

Gate timing According to the BMX SX rules and regulations [18] the starting gate must fully open within 0.310 seconds with a deviation of maximum 7%. The gate thus must drop from a vertical position to a horizontal position with respect to the ramp anywhere between 0.289 tot 0.331 seconds. This 7% difference in gate opening speed results in what riders call fast and slow gate. Although the time difference is very small, rider claim that they can determine this difference and adapt their starting technique based on the speed of the gate. For the model the gate timing is thus of large importance. Van Grieken [4] captured video footage of the gate drop, which can be used to determine the opening dynamics using the same approach as used for the video analysis in the frame inertia experiment. Using Kinovea the gate angle can be tracked in time. Kinovea tracks the x and y position of a specific selected point on the gate, as seen in Figure 3.12, from which the gate angle can be calculated. This results in the graph seen in Figure 3.13. The dotted lines show the gate angle as a function of time for four different videos. From this figure, it can be concluded that the gate fully opens within 0.3167 seconds, which makes it a fairly average gate in terms of opening speed.

With the gate kinematics known, the only thing left is to prescribe the gate in the OpenSim model. This can be done in many different ways, for instance by using the PrescribedCoordinate[2] function in OpenSim. This allows the user to define the exact value of a coordinate in time by prescribing it with a custom made function. OpenSim includes a large number of pre-built functions, like a step, spline, constant or linear function. OpenSim then introduces a constraint force or torque that ensures that the coordinates matches the prescribed function. As mentioned before, introducing extra constraints is usually not beneficial for the complexity and thus computation time of a model. This makes it interesting to see if there are other possibilities that do not require introducing extra constraint equations.

A suited solution would be to not prescribe the coordinate itself, but its actuation. Based on the shape of the curve in Figure 3.13 it can be assumed that the opening occurs using linear actuation. The simplified graphs in Figure 3.14 illustrates how this linear actuation results in a gate angle graph that is similar to the experimental data in Figure 3.13. As seen in Figure 3.14 the gate starts opening slowly due to the inertia of the gate. The timing of the gate torque must be altered a little to reach the closest agreement with the experimental data using this symmetrical gate torque. This

[2]https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1Coordinate.html{#}aa4981ae3cb1002e22c20c30698d036cb

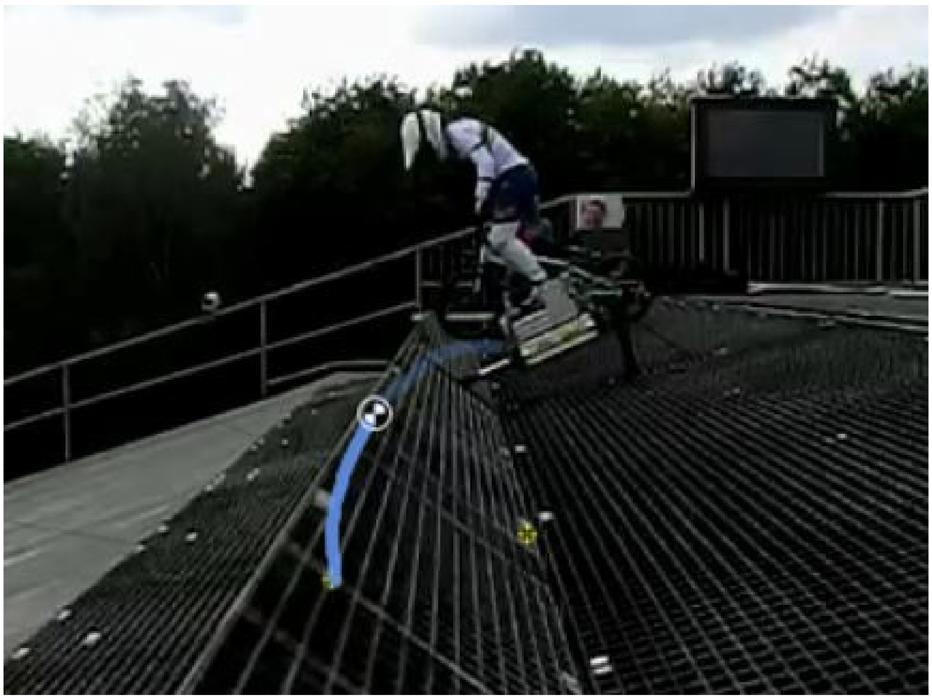


Figure 3.12: The tracking of the gate in Kinovea. The blue line shows the time history of the point being tracked. Due to practical reasons the tracking was done in reverse.

is done manually and the result can be seen in Figure 3.13. The most important part of the gate opening is the beginning. The final part is of less importance because the rider makes a wheelie and goes over the gate anyway. Whether the gate has opened 80° or 85° at the moment the front wheel travels over the gate does not matter for the outcome of the simulation. The torque actuated gate shows a good agreement with the beginning of the gate opening.

Another way to get close agreement with experimental observed gate motion would be to introduce non-symmetrical gate torque. However, gate timing is of large importance in a predictive simulation, but less important in a tracking optimization. Tracking the motion eliminates the freedom of timing from the optimization problem. The simulation cannot go over the gate earlier if it does not match with the measured kinematics. The only purpose of the gate in the tracking problem is to represent the contact force from the gate onto the front wheel. Once the bike recoils this contact is lost and the gate is not important anymore. On the other hand, in a fully predictive model the gate timing is of huge important. In order to be as fast as possible the optimizer likely finds a solution that clears the gate as soon as possible. A mismatch between the simulated and experimental gate kinematics could be fatal for a reliable outcome. Since the goal of this work is to track the kinematics, less effort has been spend on a better agreement between simulated and experimental gate kinematics and the results obtained in Figure 3.13 are sufficient.

3.1.4 Human model

The bicycle must be actuated as realistically as possible. As explained in Chapter 2.2 the best way to do that is with a multi-segmented human. This linked system must be as complicated as necessary. In the past few years researchers have developed sophisticated human model which show huge similarities with real humans. This section describes and explains the model choices made for all issues regarding the human model.

Generic model As seen before, one of the benefits of working with OpenSim is the fact that it comes with an open and co-operative community. Many work done by other researchers is open to

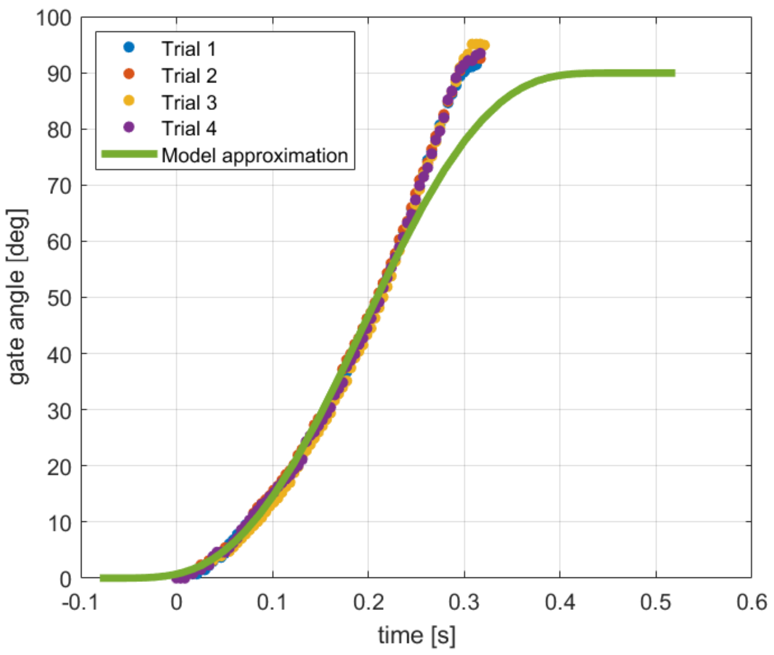


Figure 3.13: The gate angle measured using the video footage and Kinovea. The dotted lines show the four different trials being analyzed. The green line is the model representation.

the public and can be used or altered for your own project. On the OpenSim website[3] a list of all available human models can be found. One could construct a model from scratch, but this would simply be a waste of time, since there is a large variety, with varying difficulties, of models available. The job left is to pick the model which is best suited for the purpose of this project. When observing the pedaling motion of cycling, the flexion degrees of freedom are most important, especially for the lower limbs. Since the bicycle model is also a planar model, it would be best to select a human model that only takes into account the flexion degrees of freedom of the lower limbs. The other rotations of the hip and ankle are far less important and can be omitted. Additionally, the control should be fairly simple. The human body consists out of a large number of muscles with their own characteristics and strength. For the purpose of the BMX SX model, it would be best if only the most important muscles were taken into account, or that the muscles are omitted and replaced by their resulting joint torques. As described in section 1, the choice between a muscle or torque driven human model is mainly a choice about the level of accuracy needed. Muscle simulations are likely to be more biologically correct, but add more complexity to the model compared to torque driven actuation. In terms of coding the model, it is easier to remove the muscles from the model and adding torque actuators instead, then it would be the other way around. Therefore, the main criteria for the generic model is that it should have: 1) a planar representation of the lower limbs and 2) muscle control with only the most important muscles with a maximum of ten muscle per leg. In the list with generic models there is one model that ticks both of these boxes. The model, called *gait10dof18musc*, was originally created to analyze and simulate human gait. It is a highly simplified planar model which is focused on the lower extremity. As the name suggests, it entails ten degrees of freedom and is driven by 18 different muscles (nine per leg). The degrees of freedom are the flexion in the ankles, knees and hips plus an additional rotation between the pelvis and the torso, which adds up to seven. The remaining degrees of freedom are the vertical and horizontal displacement of the pelvis and the xy-rotation. The muscles included are the hamstrings, biceps femoris short head, gluteus maximus, iliopsoas, rectus femoris, vasti, gastrocnemius, soleus, and the tibialis anterior, which are the main muscles in the lower limbs. This is very well comparable to the model created by Bobbert et al. [54], which used eight muscles to actuate the cycling motion.

[3] https://simtk-confluence.stanford.edu/display/OpenSim/Musculoskeletal+Models

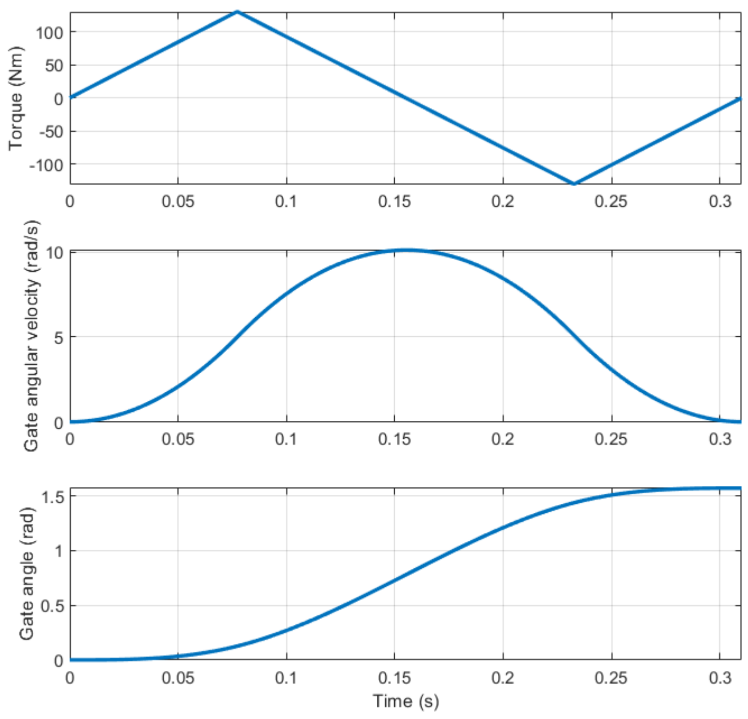


Figure 3.14: Simple approximation of the gate opening torque. A linear, symmetric opening torque results in the gate angular velocity and angle in the bottom two graphs. The angle profile has the same shape as the experimental data seen in Figure 3.13.

Upper extremities One shortcoming of the *gait10dof18musc* model is the fact that it does not include arms. Because it was used to analyze gate, the arms were excluded, since they have a minor function during walking. For the BMX model, a connection between the torso and the handlebars is important because applying forces on the pedals requires a reaction force somewhere else on the bicycle. The beautiful thing about OpenSim is the possibility to not only use a generic model, but also the ability to make changes to it. Users can add elements to the model, replace parts or remove features. It is thus possible to add arms to the gait model. This can be done in multiple different ways. The function of the arms is a force connection between the torso and the handlebars. So the first option would be to just add a force from the torso to the handlebars. This is probably the simplest solution which adds no extra degrees of freedom and just one extra control parameter The question may rise where on the torso this force should be applied. Modeling only one force from the human to the handlebars would allow only one direction of acceleration due to that force. Looking at the motion of human and bicycle during the gate start, the hips will move towards the handlebars and the front wheel will lift from the ground. One line of actuation is therefore probably not enough. To solve this issue there could be simply an extra force, but it would be hard to determine the strength of these forces. Additionally, without modeling arms, it would be difficult to constrain the upper body. The torso cannot move too far from or too close to the handlebars because this would be physically impossible. The only way to solve this issue is to have a joint that connects the torso to the handlebars. This joint must allow both translation and rotation, so have at least two degrees of freedom.

Summarizing, it was concluded that in order to correctly describe the upper body actuation and motion, there must be at least two actuators and two extra degrees of freedom. It was chosen to model the option that matches reality the closest, namely one arm made out of an upper and a lower arm connected by a shoulder joint and an elbow joint. The mass and inertia of both sections of the arm are the summation of the right and left upper and lower arm. This arm is powered by a shoulder torque actuator and an elbow torque actuator. Just like the lower limbs, the arm only contains planar rotation. In reality the shoulder acts like a ball and socket joint, but for simplicity the arm in the BMX model does just contain the planar rotation. The dimensions and mass properties of the arm were taken from a generic model in OpenSim including the upper extremities presented by Rajagopal et al. [70]. Since the mass of the arms was added to the weight of the torso for the generic *gait10dof18musc* model, the total weight of the arms was subtracted from the torso. This solution allows the different direction of actuation and provides the physical boundaries of the torso with respect to the handlebars. How this actuation was implemented in the model will be covered in the following section.

Actuation The *gait10dof18musc* model comes with an advanced muscle actuation system using the so-called Milliard muscle model [71], which is based on the widely used Hill-type muscle model [34]. This muscle model takes into account the muscle activation, tendon dynamics, and the force-length and force-velocity characteristics. All these parameters differ for each muscle and can be changed manually. These models are extremely accurate, but also add complexity to the model. Keeping only the most important lower limb muscles, already helps a lot in terms of complexity, but for the purpose of this thesis it might be better to have an actuation that is even more simplified. For this reason, the muscles are removed from the model and replaced by ordinary torque actuators. These actuators can provide a positive or negative torque around a joint torque. In OpenSim this comes down to adding CoordinateActuators[4].

These actuators act as a force when applied to translational coordinates and as a torque actuator when applied to rotational coordinates. These torque actuators require only one parameter to be determined, namely the optimal torque. This is the maximum generalized force that can be produced by the actuator. An overview of the optimal torques used for the OpenSim model can be found in Table 3.2. These numbers were based on the studies of Jansen [38] and Kordi et al. [56], who used isometric dynamometry techniques. As was explained in Chapter 2.2.2, using an dynamometer the maximum voluntary joint torque could be determined as a function of joint angle and joint velocity. Because the activation of the joint actuators in the model will be the result of tracking the kinematic, the optimal torques must be sufficiently high to be able to produce the same motion. Since these numbers were based on male track cyclists, they should be large enough. A control signal determines by what percentage the torques are used. For example, an actuator with a optimal force of 500 N that is controlled by a signal of 0.1 will produce a net force of 50 N. The minimum and maximum control of the actuators can also be adjusted, which allows for a difference between flexion and extension strength. This reduced the control parameters for the legs from 18 to only 6. A downside is that the actuation will become less realistic. For instance, the torque actuators do not account for activation dynamics, so the torque can change from positive to negative instantaneously. However, the big picture is expected to be similar. For the actuation of the upper and lower arm, two extra ideal joint torque actuators are added resulting in a total of eight control parameters.

To smoothen the actuation and to overcome high frequency oscillations in especially the elbow joint, dampers are introduced for all joints. In OpenSim this can be done using the SpringGeneralizedForce[5] class. This adds a spring to a coordinate of choice with stiffness and viscosity properties. If the stiffness is set to 0 these springs act like pure dampers. These damping coefficients are set to 20 Nms/rad for the elbow and the shoulder joints and 10 Nms/rad for all other joints. Besides that, a CoordinateLimitForce[6] is added to the elbow joint. This particular class provides a force to a coordinate that is about to hit its limit. For the elbow this was necessary to make sure it could remain in a fully stretched position. For this CoordinateLimitForce again stiffness and damping coefficient could be introduced. Since the elbow joint was already damped using the generalized spring force, only the stiffness was used with a value of 4 Nm/° active within a range from 0 to 5°.

Besides the activation torques and contact forces, there is one final force in play, namely the gravity. This is easily implemented in OpenSim by setting a gravitational acceleration, in this case 9.81

[4]https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1CoordinateActuator.html
[5]https://simtk.org/api_docs/opensim/api_docs24/classOpenSim_1_1SpringGeneralizedForce.html
[6]https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1CoordinateLimitForce.html

Joint	Maximum Torque (Nm)
Hip	400
Knee	300
Ankle	250
Shoulder	400
Elbow	250

Table 3.2: Maximum joint torque for each joint. Hip, knee, and ankle joint torques are equal for both legs. The shoulder and elbow joint are the summation of the left and right arm. These values are based on the work of Jansen [38] and Kordi [56] for Olympic male track cyclists.

m/s^2. The aerodynamic forces on the bicycle and rider are in this model ignored due to their minor importance at the low speeds reached at the first section of the start.

Constraints Now that the bicycle and human model are complete, the only thing left is a connection between the two models. Usually connections between different bodies in OpenSim are made using a variety of possible joints. However, this cannot be used for the bicycle rider system since this would result in closed-loop kinematic chains. There are actually two different closed-loop chains: one for the loop between the the legs and cranks and one for the legs connected to the cranks and the upper body connected to the handlebars. One could use a regular joint for one of the attachment points between the bicycle and the human, but not more than that. For these kind of systems, it is generally impossible to derive the equations of motion in terms of the generalized coordinates, resulting in major issues when trying to optimize the system. Luckily, OpenSim has a solution for this problem: kinematic constraints. These kinematic constraints are not real joints but their outcome is the same as if there would be a joint between two bodies. In essence, joints would take out one or multiple degrees of freedom. The kinematic constraint does not take out this degrees of freedom directly but tries to minimize the relative motion between the two bodies, usually with an accuracy in order of 10^{-10}. To connect the human to the bicycle the lower arm is connected to the handlebar using a pin joint which allows for the rotation of the hand around the handlebar. BMX SX riders use pedals with clips that connects the feet to the pedals. This makes it possible to not only push but also pull on the cranks. The easiest way to implement this type of constraint in the OpenSim model would be to use a WeldConstraint[7]. As the name suggest, this simply welds the feet and the pedals together, taking out all six degrees of freedom. The problem with this method it that it over-constrains the model. Due to the pin joint between the hands and the handlebar, there already is no possibility for the feet to laterally translate with respect the bicycle. A WeldConstraint would thus introduce unnecessary constraints. Additionally, when using a WeldConstraint, the lateral position of the pedal and feet must be known exactly in order to not violate the other constraints in the model. In other words, since there are only planar joints in the model, the width between the feet must be exactly the same as the width between the two pedals. Additionally, the orientation of the feet must be exactly the same as the orientation of the pedals. To overcome these issues, the connection between the feet and the cranks are realized using the PointOnLineConstraint[8]. This constrains one point on the feet to be on the same line a single point on the cranks. This line is indicated with the red arrows in Figure 3.15. This type of constraint takes out two translational degrees but no rotational degrees of freedom. A benefit of this method is that since the lateral translation and rotations are still open, the width between the two feet and the orientation of the feet does not matter.

The other kinematic constraint that is implemented in the model is the kinematic coupling between the crankshaft and the rear wheel. As mentioned earlier in this chapter, this is done using a CoordinateCouplerConstraint with a transmission factor equal to the gear ratio of the front and rear sprocket, which equals $\frac{44}{16}$. An overview of the entire model can be seen in Figure 3.16. As shown there is only one arm which is positioned in the middle of the torso and connected to the center between the two handlebar-ends. Since the arms are manually added, their appearance is not as

[7]https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1WeldConstraint.html
[8]https://simtk.org/api_docs/opensim/api_docs33/classOpenSim_1_1PointOnLineConstraint.html

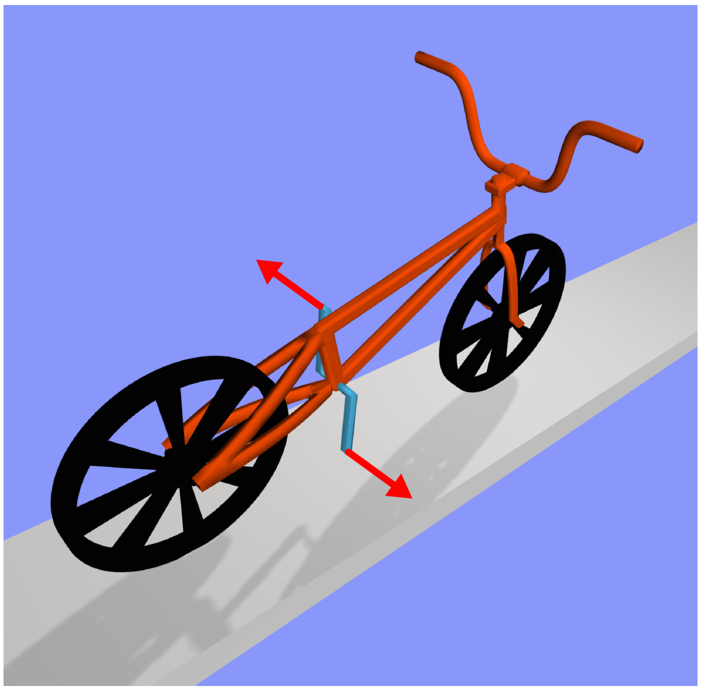


Figure 3.15: Visual interpretation of the OpenSim PointOnLineConstraint used for the connection between the feet and the crankset. The feet are connected to the lines indicated by the red arrows.

detailed as for instance the legs. This does not influence the model performance, but is just for visual support. In total there are nine degrees of freedom. First, a horizontal translation and a rotation for the bicycle frame. Secondly, there is a closed kinematic loop for the lower limbs and the crank system with a total of three degrees of freedom. This is usually thought of as a degree of freedom for the crankshaft and one for each leg. The final degrees of freedom are the shoulder, elbow, and hand rotation and the rotation of the front wheel, making a total of nine degrees of freedom. Since the rear wheel is kinematically constrained to the crankshaft it does not provide an extra degree of freedom.

3.2 Experimental Data

With the model finished, the next step is to collect and manipulate experimental data from the BMX SX gate start. Due to the upcoming Tokyo Olympics and the coronavirus situation, it was not possible to execute experiments with the elite BMX riders. Luckily, two master students from the Technical University Delft and the Vrije Universiteit Amsterdam collected just the data needed. Hylke van Grieken from the TU Delft collected kinetic data from five elite BMX SX athletes [4]. He used a fully instrumented bicycle to capture the pedal forces executed. Melle van Dilgt from the VU Amsterdam capture the kinematics of the BMX SX start to calculate mechanical power output based on kinematics only [11]. These experiments used the same participants, but on different bicycles and on different days. Ideally, these experiments would be redone at the same time, with the same equipment, but unfortunately this was not possible. There exists a dataset of both kinematic and kinetic measurements, but this data is not synchronised nor processed. Since the data available was gathered within a time-span of a couple weeks and used the same participant, the data available could be of great use for the purpose of the tracking solution. The two section below, briefly describe the experiments for the kinetic and kinematic measurements.

3.2.1 Kinetic data

Hylke van Grieken used an instrumented BMX bicycle to determine the key characteristics of the gate start [4]. Its main component was a Swift Axis2D crank (Swift Performance, Brisbane, Aus-

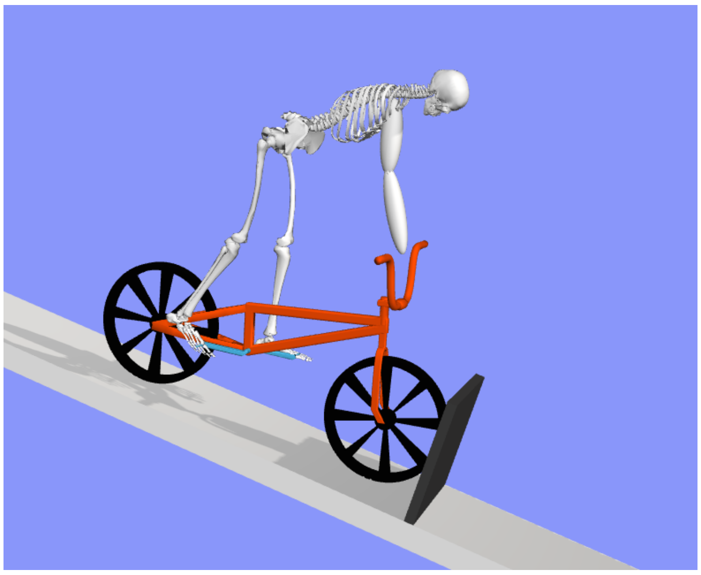


Figure 3.16: The complete BMX SX bicycle-rider system. The lower end of the lower arm is connected to the frame using a pin joint and the feet are connected to the crankset using the PointOnLineConstraint.

tralia) which could measure the forces exerted on the pedals in both the radial and the tangential direction with a frequency of 100 Hz. This was the first setup that could determine pedal forces with such a high frequency during in-field experiments. The Swift Axis2D cranks are also capable of measuring the crank rotation and velocity. Additionally, there were timing chips, gyro's and accelerometers attached to the bicycle as seen in Figure 3.17. From the measured pedal forces, other kinetic parameters could be calculated such as crank torque, work, and power using the crank kinematic data. The participants in the experiments were elite BMX riders from the Dutch national team. The participants were instructed to perform six all-out starts from a standard Olympic ramp. For one particular athlete the kinetic experiment was also collected at the lower training ramp. This latter data will in this work be used to validate the outcome of the tracking solution.

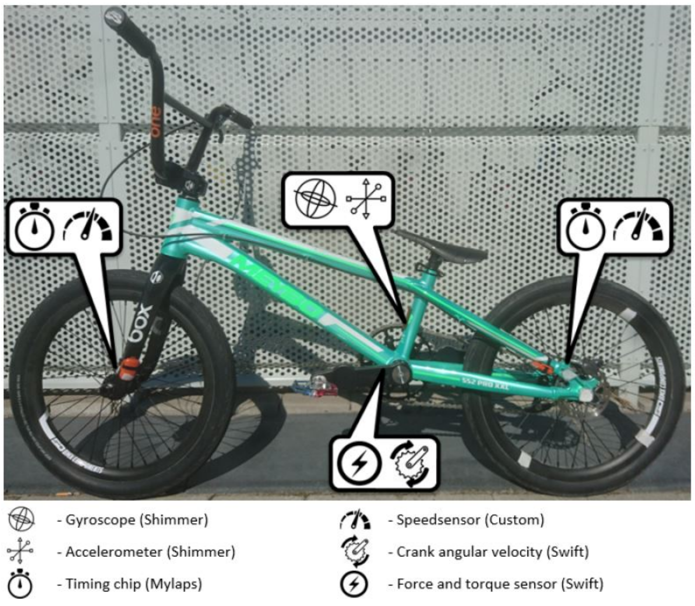

- Gyroscope (Shimmer)
- Accelerometer (Shimmer)
- Timing chip (Mylaps)
- Speedsensor (Custom)
- Crank angular velocity (Swift)
- Force and torque sensor (Swift)

Figure 3.17: The instrumented BMX SX bicycle used by Van Grieken [4] equipped with several sensors including the Swift Axis2D crank.

3.2.2 Kinematic data

Melle van Dilgt collected kinematic data of the BMX gate start with the aim to calculate power from kinematic data only [11]. She used the same participants as in the kinetic experiments, but the starts were performed on the slightly lower training ramp. An Xsens MVN suit (Xsens Technologies, Enschede, The Netherlands) was used to capture the kinematic of the rider and an additional IMU sensor was placed on the rear hub of the bicycle to capture its rotation. An Xsens suit consists of multiple high frequency (240 Hz) IMU sensors that can capture the movement of the human body. Using sensor fusion techniques the data of accelerometers, gyroscopes and magnetometer are combined which result in segment rotations and translations. This is exactly what is needed for the setup of the tracking optimization problem. However, the Xsens system determines the 3D joint angles, whereas the created bicycle rider model only has the possibility of planar rotations. Additionally, the output of the Xsens sensor suit is a so-called .mvnx file. This file contains a time history of all the sensors on the suit containing orientation, position, velocity, acceleration, angular velocity and angular acceleration. To capture only the beginning of the trial, a start and a stop condition were defined. The dataset started at the point where the pelvis showed three consecutive datapoints with an accelerations above a certain threshold, which was defined as 2 m/s^2. The dataset was cut off when the bicycle had covered a total of five meters. The kinematics used for the tracking problem were taken from a female world class BMX athlete (age 26) with an experience of 19 years. Kinetic data for the same athlete was available as well. For the tracking problem the time histories of the generalized coordinates are needed. In the case of the BMX-rider system these are the joint flexion angles of ankles, knees, hips, shoulder, elbow, and hands for the human and crank angle, front wheel angle, bicycle translation, and bicycle orientation for the BMX bicycle. The kinematic data must thus be processed before it can be used in the tracking problem.

The transformation from 3D Xsens data to planar joint angles suited for the optimization problem can be done using OpenSense[9], which is part of OpenSim. OpenSense was created to convert experimental IMU data into the OpenSim generalized coordinates, which is exactly what is needed. A useful side-effect is that OpenSense projects the measured data on the OpenSim model. It uses the relative rotations to find the best solution matching the constraints of the OpenSim model and the experimental data, without the need of scaling the model to the specific participant. The only thing required to use OpenSense is an initial position. The joint angles for this initial position were taken directly from the IMU data. The first data points were used to calculate the Euler angles between all the relevant segments of the human body. Because of the mismatch between the OpenSim model and the anthropometrics of the participant, these joint angles cannot all be satisfied. OpenSim finds the closest match for all these joint angles. In order to be able to use OpenSense the IMU data must be converted into .txt files that are readable for OpenSense. OpenSense requires the orientation of the body segments as a rotation matrix and their linear acceleration in three directions. Since the .mvnx files express the orientation of the segments in so-called quaternions, this must be converted first. The .mvnx files can be loaded into MATLAB (MathWorks, Inc., Natick, USA) to process the data. Once in MATLAB it is easy to convert quaternions into rotation matrices and to export the data in any file you want using the fprintf command. A script was created that could read the IMU data, convert it into the correct form and then print it into an OpenSense readable .txt file. This .txt file could then be forwarded to OpenSense to calculate the corresponding generalized coordinates given the BMX-rider model. These generalized coordinates can be used as a tracking file for the trajectory optimization problem, which will be described next. More details on this can be found in Appendix C.

OpenSense takes the IMU data of the six lower limbs including sensors on the femur (upper leg), tibia (lower leg), and calcaneus (foot). Besides, it takes the sensor data from the pelvis and the rear wheel, adding up to a total of eight IMU sensors. The model has one single arm, which does not correspond with reality. As a matter of fact there is no IMU input for OpenSense for the arm. It computes the joint angles of the missing degrees of freedom based on the best match with the provided input. The joint angles for the shoulder, elbow, and hand are thus a result of the OpenSense optimization routine and not based on IMU sensory input of the upper limbs. Therefore, the joint angles of these degrees of freedom will be tracked by the trajectory optimizer with a lower weight,

[9]https://simtk.org/projects/opensense

as will be explained in further detail in Chapter 3.3.2.

OpenSense does not take into account forces. So the output of OpenSense will be only the nine joint angles, the crank angle, and the rear wheel angle. There will be no bicycle translation or rotation in the OpenSense solution. Additionally, OpenSense provides the rotations only, not the angular speeds and the joint torques. The optimizer must thus be able to track a large section of the generalized coordinates, without knowing the corresponding other generalized coordinates or actuation.

Slip

A major issue with the kinematic data is the fact that it contains a moment of slip in all trials. Apparently, the grip from the tire to the surface cannot handle the large forces applied during the first moments of the start. This is well visualized in Figures 3.18 and 3.19. At approximately 0.17 seconds, the angular velocity of the rear wheel makes a strange jump. This is due to the rear wheel slipping. The forward velocity of the BMX bicycle is calculated based on the rear wheel IMU data. The velocity of the bike, as seen in Figure 3.19, thus makes that same jump.

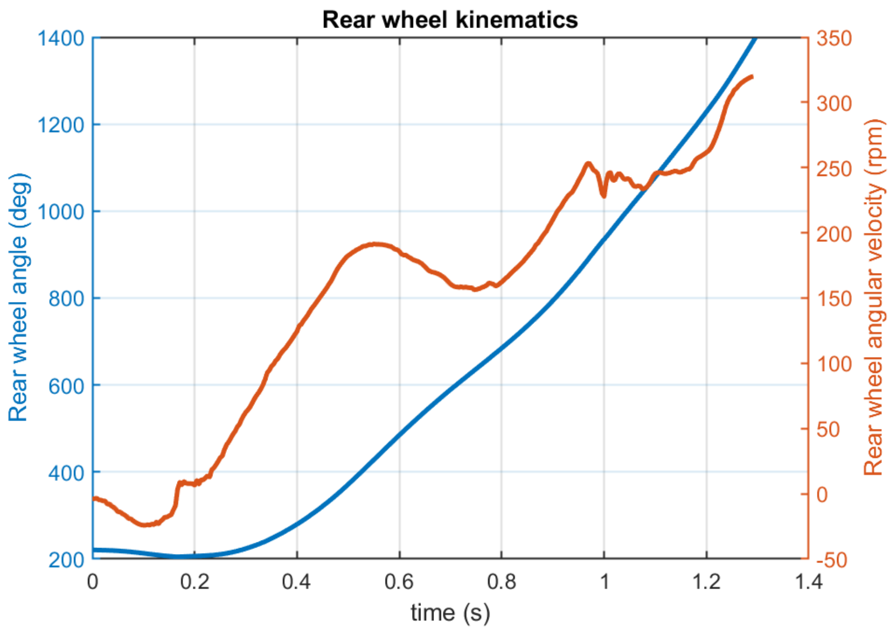


Figure 3.18: Kinematics of the rear wheel as a function of time. The blue line is the rotation of the rear wheel and the orange line shows the angular velocity of the rear wheel expressed in revolutions per minute (rpm). At approximately 0.17 seconds there is a huge jump visible in the angular velocity, indicating that the wheel is slipping. Data taken from Melle van Dilgt [11].

As explained in the previous sections, the rear wheel of the OpenSim model does not slip, which causes trouble when optimizing the model to match the kinematics. Since the wheel cannot slip it applies huge forces in an effort to match the velocity jumps seen in Figures 3.18 and 3.19. Figure 4.3 shows this jump in crank torque when the optimizer tries to track the crank dynamics including the slip. A solution for this issue would be to implement slip in the OpenSim model as well. However, this is pretty complex because this would require extensive knowledge of the bicycle tire and ground surface dynamics. Additionally it makes the model even more complex, which is not desired at this stage. For future work this would be an interesting topic to improve. A simpler solution is to start tracking the IMU data after this moment of slip, which is in this case 0.17 seconds after the first movements observed. Another argument to start the tracking problem after the moment of slip has to do with the CoordinateCouplerConstraint between the crankshaft and the rear wheel. As explained this functions as a direct, bidirectional, coupling between the two rotational degrees of freedom of the crankshaft and the rear wheel. This results in, what is known in cycling

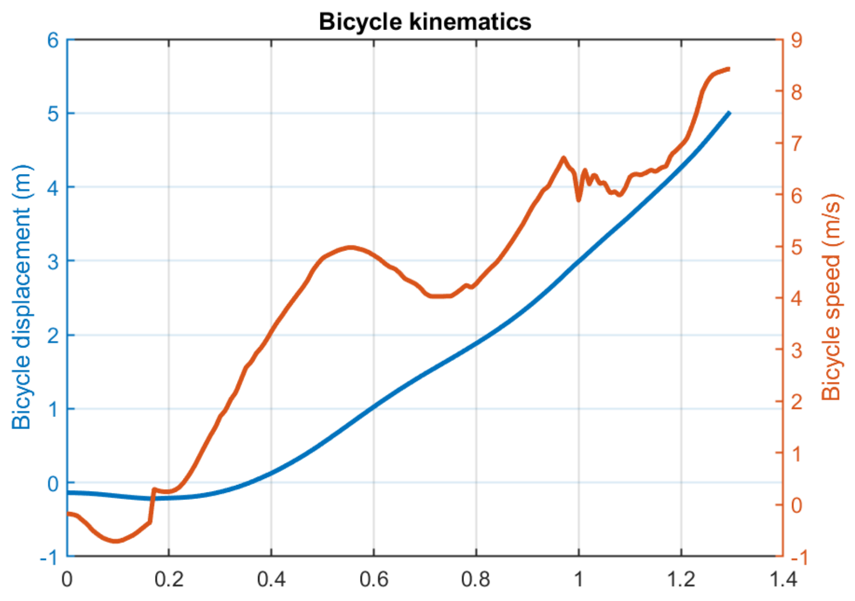


Figure 3.19: Kinematics of the BMX bicycle as a function of time. The blue line is the forward distance travelled and the orange line shows the velocity in that same direction. At the same point in time where the jump in angular velocity was seen, the forward speed makes that same jump because the forward speed was calculated based on the rotation of the rear wheel. Data taken from Melle van Dilgt [11].

as, a fixed gear bicycle. A key aspect of these bicycles is that if you rotate the crankshaft backward, the rear wheel rotates backward as well. In reality the BMX SX bicycle consists of a freewheel hub. This ensures that only when the crankshaft is rotating forward the rear wheel spins in that same positive direction. This freewheel setup is the same as seen for typical road bikes and is necessary in BMX to allow the rear wheel to spin while the cranks are kept level. Riders need this to make the jumps and to smoothly go over the obstacles along the course. The difference between a fixed geared bicycle and a freewheel bicycle is crucial when modeling the start. Using a fixed gear, it is possible to apply a negative crank torque to propel the bicycle backward making the recoil movement. The first optimization solutions showed a negative crank torque at the start, indicating the it adapts to a strategy of actively pedaling backward as shown in Figure 4.3. In reality, with the freewheel hub, the recoil movement is purely the results of shifting the center of mass towards the steer of the bicycle [2]. Modeling the freewheel in OpenSim is possible, but rather cumbersome. The CoordinateCouplerConstraint can only handle user defined function containing the coordinate itself. To make a unidirectional constraint, the function should also be dependent on the coordinates speed, in this case the angular velocity. This angular velocity namely tells something about the direction the crankshaft is spinning in. Attempts have been made to construct a freewheel configuration using the ExpressionBasedCoordinateForce[10]. This force allows for expressions containing both the coordinate itself and its derivative. A function like the one expressed in Equation 3.8 could be used to create a coupling between two bodies only when the rotation is in a positive direction. If $\dot{q}$, which is in this case the crank angular velocity, is positive, Equation 3.8 will be equal to some preset value T_0, but when it is negative it adds up to zero, meaning no coupling torque. Nevertheless, this approach makes the optimization and simulation inevitably slower than just using the CoordinateCouplerConstraint. Leaving out the first section of the start solves the freewheel problem as a side-effect. Because the tracking now starts after the recoil movement, there is no need to apply a negative net torque on the cranks. Therefore, the original CoordinateCouplerConstraint can be used to construct the coupling between the crankshaft and the rear wheel.

[10]https://simtk.org/api_docs/opensim/api_docs/classOpenSim_1_1ExpressionBasedCoordinateForce.html

$$T = \frac{1}{2} T_0 \left(\dot{q} + \sqrt{(\dot{q})^2} \right) \tag{3.8}$$

3.3 Optimization

As described in Chapter 2.3 there are multiple ways to solve an optimization problem. In general, in an optimization problem one tries to minimize or maximize a certain parameters. For instance, in the optimization problem of Jansen [38] the goal was to maximize distance travelled by a track cyclist in a certain time interval. Another example is the frequently used goal to minimize the control effort. There are different methods to find this maximum or minimum. The main categories of solvers are the single shooting methods and the direct collocation methods. The single shooting methods are easier to implement but are mainly suited for simple problems. For more complex problem, where simulation takes more time, the method can be inevitably slow. Direct collocation methods on the other hand are difficult to setup but can be way faster compared to single shooting methods. Since OpenSim is compatible with all types of scripting such as Python or MATLAB, the possibilities for optimization are almost endless. The great thing about OpenSim is the fact that there are two software programs available for optimization problems. The first is Scone, developed by Thomas Geijtenbeek [72], which uses single shooting optimization and the other is Moco, developed by Chris Dembia, Nicholas Bianco and Antoine Falisse [10], making use of direct collocation. Since the BMX model contains closed kinematic loops and is therefore rather complex, the most interesting would be to use direct collocation. However, setting up a direct collocation optimization for this model could become rather complex, since usually a lot of technical expertise is needed. Luckily, Moco takes care of this setup. Since it is especially developed for OpenSim models, the user can simply load their OpenSim model and Moco sets up the optimization problem. In this study it was therefore chosen to use Moco for the tracking optimization problem.

3.3.1 Moco

As mentioned above, Moco was created to optimize the motion and control of models created in OpenSim. It takes care of the implementation of the OpenSim model into dynamic equations that are needed for optimization. Additionally Moco is the first direct collocation software package that can handle kinematic constraints, which is important for the BMX SX model. To be able to make use of kinematic constraints, it is necessary to use the Hermite-Simpson transcription scheme as explained and visualized in Chapter 2.3. Using the Moco software, a number of pre-built optimization goals can be chosen such as motion tracking, parameter optimization or minimizing control effort. A downside of using Moco is that a user can only use the pre-built goals and settings, but with the large number of available goals this is hardly a problem. Besides, since Moco is an open-source optimization software package, users can create their own objective functions. A potential problem with using Moco and thus direct collocation for the BMX SX optimization in specific, is the fact that the front wheel is lifted from the ground. This results in a discontinuous contact force between the front wheel and the ground. Because direct collocation relies on a gradient-based optimization, it converges faster if all functions are continuous. Moco solved this potential problem by adding the possibility to use the SmoothSphereHalfSpaceForce[11] [73] instead of the aforementioned Hunt-Crossley contact force. In essence these two forces are exactly the same with the exception that the SmoothSphereHalfSpaceForce smoothens the transition between contact and no contact. The force does not drop to 0 when contact is lost but remains at a very small, but non-zero, force. In the bicycle model the Hunt-Crossley forces between the front wheel and the ground and gate are therefore replaced by SmoothSphereHalfSpaceForces. The contact between the rear wheel and the ground is kept as a Hunt-Crossley force since the model was created in such a way that it is not possible for the rear wheel to lose contact with the ground surface.

Setting up the problem with Moco takes out a lot of work, but a few parameters must be defined by the user. Despite the fact that Moco takes care of the most complex parts, most time during the work

[11]https://opensim-org.github.io/opensim-moco-site/docs/0.2.0/class_open_sim_1_1_smooth_sphere_half_space_force.html

of this thesis was spent on the optimization part. Setting up the problem with the correct initial guess, boundary constraints, objective functions, and optimization constants can be fairly tricky. The entire Moco xml-script used to setup the optimization problem can be found in Appendix G, but the most important parameters are also discussed here.

The first things that must be set are the boundary conditions for the model. These are the initial and final states for the entire model. The initial states are crucial because the rider must start in the correct posture to accurately track the kinematics. If the initial conditions are not provided to the optimizer it is likely that it finds a solution which matches the initial position pretty well, since the goal is to track the kinematics provided, also for the beginning of the trial. However, specifying the initial conditions ensures that the model starts in the exact same posture as the kinematics showed and helps the optimizer to find solution faster and more accurately. The initial positions and joint angles are directly taken from the kinematic data at 0.2 seconds and are shown in Table 3.3, the final positions were obtained by selecting the final numbers in the OpenSense output file. The states provided in Table 3.3 are the states being tracked. The model is thus not tracking the bicycle displacement or pitch angle. For these states the initial state is based on a previous solution which started at t=0 as seen in Figure 4.1 and 4.2. However, no final value was picked for these states, because small differences between the model and the kinematic data could be fatal for the optimization. For instance, the rear wheel rotation is related to the bicycle displacement and is dependent on the rear wheel circumference. This might differ for the model and reality which can lead to a mismatch in distance travelled. Constraining the final position of the bicycle could thus lead to an infeasible problem setup. The optimizer will find the solution that best tracks the kinematic data provided. If the model is correct, the distance travelled will closely match the experimental distance travelled, without even tracking it.

state	**initial value**	**final value**
time	0.2 s	1.316667 s
right hip	36.84183°	57.38022°
right knee	-31.6591°	-98.8417°
right ankle	-3.13997°	12.72899°
left hip	73.06206°	39.31995°
left knee	-63.0755°	-24.5014°
left ankle	30.31129°	-0.17295°
rear wheel rotation	204.4014°	1386.923°
crank rotation	74.3278°	504.3357°
shoulder	136.954°	171.893°
elbow	15.89449°	53.94788°
hand	164.1472°	157.8397°

Table 3.3: The initial states for the optimization problem setup. These values are the direct outcome of the OpenSense solution at 0.2 seconds and the final point in time. As seen there is no front wheel rotation provided since that degree of freedom was not included in the IMU dataset, nor was it dependent on another degree of freedom that was provided by the IMU dataset. The same reasoning applies to the bicycle translation and rotation. These states are coupled to IMU dependent states via the contact model which OpenSense does not take into account.

The second factor to set are the bounds to the problem. These are the values the optimizer cannot exceed during the entire trajectory. It turned out to be a really important factor to set, since by default the bounds are set to -10 and +10 for all states. Depending on the state the units are m or radians. This caused problems, since the cranks rotate approximately 1.25 full rotations during the trial being tracked. The rear wheel thus rotates $1.25 \cdot 44/16 = 3.4375$ full rotations. This equals $3.4375 \cdot 2\pi = 21.5984$ radians, which exceeds the boundary of 10 radians. The boundaries for the front and rear wheel were thus enlarged to -50 and +50 radians. Boundaries are also important to ensure that the model only finds feasible solutions. Human joints can not fully rotate, but are limited to a certain extend. For instance, the knee joint can flex until the lower leg hits the upper leg and cannot extend further than fully stretched. The limits for each joint in the model are presented in Table 3.4. The human joint limits of the hips, knees and ankles were taken from the generic model. The bounds for the knee were slightly adapted. The lower bound has been changed to -140°, because the kinematic data showed knee angle exceeding the original limit of 120°. The limits for the custom-built arm were selected based on the range of motion for the hip and torso seen from video analysis. The bounds should not obstruct the state tracking, so values for these joint were picked that allowed all possible postures seen from video. To get a better picture of the of the numbers presented in Table 3.4, Figure 3.20 shows the positive and negative directions of the generic model. The blue arrows indicate the positive directions and the red arrow the negative directions. This means that for the hip, ankle and elbow the positive direction is the flexion direction, but for the knee and the shoulder the positive direction is the extension direction. This is important to keep in mind, while analyzing the numbers in Table 3.4 and the results presented in Chapter 4.

state	**lower bound**	**upper bound**
hips	-30°	120°
knees	-140°	15°
ankles	-45°	45°
shoulder	0°	270°
elbow	-10°	150°
hand	90°	225°
front wheel	-50 rad	50 rad
rear wheel	-50 rad	50 rad
crankshaft	-50 rad	50 rad
bicycle translation	-2 m	6 m
bicycle pitch	-5°	60°

Table 3.4: The boundaries for every state in the model. Note that some states are expressed in degrees, other in radians.

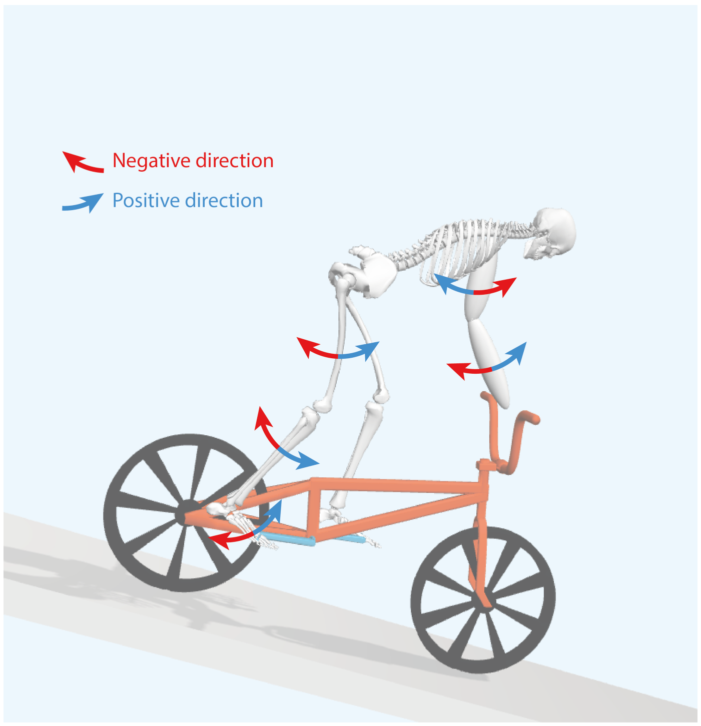


Figure 3.20: The definitions of positive and negative directions for each joint in the model. These settings were taken from the generic model default directions. As seen the positive direction corresponds to joint flexion for the hip, ankle and elbow joints, but to joint extension for the knee and shoulder joints.

Thirdly, information about the speed of the model must be provided. Since the tracking starts after 0.2 seconds, the rider and bicycle have already moved and have a non-zero speed. To achieve a close agreement with the kinematic data it is important to have at least a proper idea about the initial speeds of all states. Since the OpenSense output does not provide joint speeds, the speed are taken from a tracking solution using the entire trial, as seen in Figures 4.1 and 4.2. This is the same approach as was used for the initial states that were not traced by OpenSense. The initial speeds for all states are found in Table 3.5. However, since these initial speeds are taken from a previous optimization solution, it is not ascertained that these value are valid. Therefor these values were taken as a guess and the initial speed was said to be -5% or +5% from the value reported in Table 3.5. Because the first optimization solution showed a mismatch between the crankshaft speed at the end of the trial, clearly visible in Figure 4.2, a final speed condition was added for this specific state. The first solution showed a drop in cadence at the end of the trial, so the end constraint used stated that the final crankshaft speed must be higher than 11.5 rad/s.

state	**initial speed**
Bicycle forward speed	0.003617763 $\pm$ 5%
Bicycle pitch speed	0.582009483 $\pm$ 5%
Rear wheel angular velocity	0.769888868 $\pm$ 5%
Front wheel angular velocity	-2.495015296 $\pm$ 5%
Crankshaft angular velocity	0.279959589 $\pm$ 5%
Hand angular velocity	2.203104703 $\pm$ 5%
Elbow angular velocity	0.902891971 $\pm$ 5%
Shoulder angular velocity	4.017798133 $\pm$ 5%
Right hip angular velocity	-3.122837265 $\pm$ 5%
Left hip angular velocity	-2.338567256 $\pm$ 5%
Right knee angular velocity	0.862019925 $\pm$ 5%
Left knee angular velocity	-1.827780104 $\pm$ 5%
Right ankle angular velocity	-2.845731303 $\pm$ 5%
Left ankle angular velocity	0.43460599 $\pm$ 5%

Table 3.5: Initial speeds for the tracking optimization. Values were taken from an optimization of the entire start, starting from t=0. Initial speeds allowed a variance of 5% with respect to the numbers in this table.

The final step in the optimization setup is to define the solver settings. These settings include the mesh interval, which is related to the step size of the simulation, the transcription method, the convergence and constraint tolerances, the initial guess, and finally the finite difference scheme. The BMX tracking problem is solved with a mesh grid size of 135 resulting in a step size of 0.004 seconds. In Chapter 2.3 it was explained that to handle the kinematic constraint the Hermite-Simpson transcription is needed. The constraint and convergence tolerance are both set to 10^{-4}, as was used by Haralabidis et al. [66]. The initial guess is based on previous solution with slightly less bounded conditions. This is usually the way to go with direct collocation problems [10]: start with a simple problem and increase complexity using the previous solution as an initial guess. This process started with an initial guess which only consisted out of the joint angles from OpenSense and random values for all states that were not defined in the OpenSense output file. So all speeds and activations were random constants throughout the whole trial.

As explained in Chapter 2.3, Moco creates a so-called nonlinear program (NLP), which is solved using a direct collocation method. This work uses CasADi [74] to covert the continuous optimal control problem into a finite-dimensional NLP. This NLP is then solved using a gradient-based nonlinear program solver such as SNOPT [75], or, as was used in this work, IPOPT [76]. The finite difference scheme used to solve this NLP was set to 'forward'. CasADi offers two possible schemes, 'forward' and 'central'. The forward scheme is less accurate but it is faster.

A final remark about the optimization setup has to do with the constraint forces needed to obey the kinematic constraints implemented in the model. These are the forces needed to make the rear wheel rotates with a factor of $\frac{44}{16}$ with respect to the crankshaft and to keep the feet attached to the pedals. The maximum default Lagrange multiplier, which is the maximum constraint force that can be applied, had to be adapted to reach the required pedal forces to match the kinematics. The default settings limited the constraint forces to -10^3 and 10^3 Newton, whereas Figure 4.11 show that pedal forces close to 2000 N were reached. In the case of the BMX model it was clear that the constraint forces were somehow limited because the results showed a clear plateau around 1000 N. The downside of Moco is that these default settings are somewhat hidden in the setup files so one should always keep an eye on strange optimization outcomes that could be the results of these hidden default settings.

3.3.2 Objective function

The objective function for the tracking problem is twofold. The first goal is to track the kinematic data containing the joint angles, crank angle and rear wheel angle as close as possible. The corresponding Moco goal is called a StateTrackingGoal[12]. Moco evaluates the squared difference between a state variable value and a reference state variable value. This is then summed over all state variables for which a reference is provided, multiplied by a specific weight for each state, and finally integrated over the phase. The second goal is a term for the control cost. Since not all generalized coordinates are prescribed, there are multiple ways to actuate the model that equally match the tracking data. The solver should select the solution that uses the least control effort, since humans are usually super efficient. Additionally, adding a control goal will smoothen the solution, i.e. prevent high frequency oscillation in the torque activation, which makes the solution more realistic. This control goal was added with a relatively low weight compared to the weight of the tracking goal, since it is of minor importance. The goal ratio between the tracking and control goal was set to 1:100. Furthermore, OpenSense does not only provide the states for the lower limbs, the crankshaft and rear wheel rotation, but it also provides information about the arm states. Because the single arm is completely artificial, there is no IMU data for it states. One could try to fuse the IMU data from the left and right arm into a planar single arm model, but this is easier said than done. The human shoulder has not one, but three, degrees of freedom which allows a wider range of motion compared to the single arm used in the OpenSim model. However, it is not necessarily needed to provide arm IMU kinematics to OpenSense. Since the hand is connected to the handlebar, the OpenSense solution should obey this constraint. While converging the IMU data to the OpenSim model, it also provides the state of the arm. These states were used in the tracking optimization, but they were assigned a lower weight. Compared to the weight of the other states being the tracked, the weight for the shoulder, elbow and hand angle was 100 times lower, equaling the weight of the control goal.

An overview of the entire XML-coding to setup the Moco optimization problem, including all settings and the objective function described above, can be found in Appendix G.

[12]https://opensim-org.github.io/opensim-moco-site/docs/0.4.0/class_open_sim_1_1_moco_state_tracking_goal.html

4 Results

4.1 Full start tracking

As was explained in Chapter 3.2.2, the slip in the data caused problem when tracking the kinematic data. Just from looking at the main tracking result, which is the tracking of the joint angles of the leg, this issue is not clearly visible yet. Figure 4.1 shows the time history of the six joint angles of the legs. In blue the results of the tracking optimization are shown, whereas the dotted orange line shows the kinematic data derived from IMU and OpenSense. The mean RMSE between the simulated and experimental joint angles for the legs is 1.73°, which is pretty accurate compared to other tracking literature [66] [77]. Both knee angles start close to 0 which indicates that the legs are almost stretched at the initial posture of the gate start. The right leg is slightly more bent indicating that this is the trail leg. The ankle angles are both close to zero too, meaning that they are perpendicular to the shanks. This is just as expected based on the balancing task during the initial position. As seen there is a small deviation between the curves at the beginning, especially for the right ankle. In these graphs, this is the only sign pointing in the direction of the slipping issue indicated in Chapter 3.2.2.

The slipping can be much better seen in Figure 4.2. The two top graphs show the bicycle's distance travelled and the crank rotation. These graphs are essentially the same since the crank rotation is directly coupled to the distance travelled. The bottom graphs show the bicycle's speed and the cadence, which is the crank angular velocity. Again, these graphs are essentially the same as seen from their very similar shape. The blue curve in all four figures is the simulated result, the orange line is the kinematic data from the VU Amsterdam determined using IMUs, and the yellow lines are the measurements from the TU Delft. Recall that this latter experimental dataset was gathered on a different trial while using a different bike. However, the same participant was used and it is interesting to compare the simulation results with that data as well.

As seen from the top left graph, the bicycle displacement starts at zero and becomes negative at first, indicating that the bicycle rolls backward. This is the recoil movement of the start. After this recoil the displacement graph gradually becomes steeper, showing a growing velocity in time. This is seen from the bottom left graph as well, showing the forward speed. It shows a negative velocity at the beginning, indicating the recoil movement, which more or less piecewise grows in time. This shows the different pedal strokes very nicely. Each pedal stroke contains a top and bottom dead center, where it is hard to execute a net effective torque on the cranks. From the right figures, showing the crank angle and cadence, it can be seen that the velocity drops are approximately located at 180 and 360 degrees respectively. In these orientations the crank arms are directed in a vertical direction, making it hard to apply an effective force on the pedals. The movement starts with a crank angle of approximately 90° which is a horizontal orientation with respect to the ground surface. This position is needed to remain balance during the initial position.

The different lines for the simulation and experimental data at the top plots are more or less the same. The RMSEs with the VU and TU data are 0.037 and 0.048 m for the distance travelled and

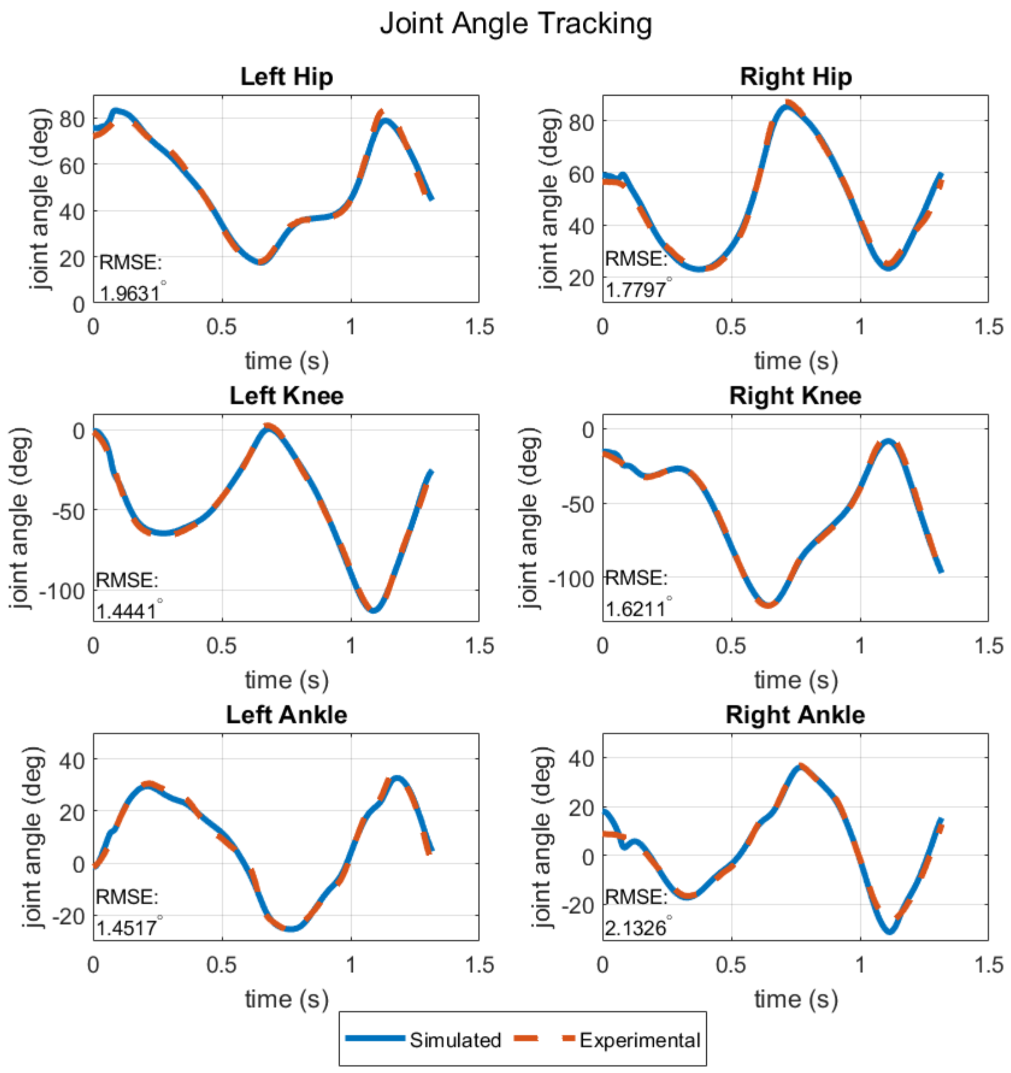


Figure 4.1: The tracked joint angles for the lower limbs of a complete start. In orange are the experimental state trajectories from OpenSense and in blue the optimized and tracked states using Moco.

1.5° and 8.35° for the crank angle respectively. The VU data is the dataset being tracked, so it makes sense that is shows a lower RMSE compared to the TU data which is not tracked. The bottom graphs show a little more deviation between the three lines. The orange VU Amsterdam data contains the slip, occurring at 0.2 seconds, which is in this case not as clear as was shown in Figures 3.18 and 3.19. The simulation tries to follow this slip jump which can be seen from the blue curves. Surprisingly, his jump does not take place at the same point in time. This can be explained from the initial guess for this specific solution, which was taken from an optimization with a different data set, where the slip happened a little earlier. The RMSEs are 0.35 m/s for the VU data and 0.45 m/s for the TU data. For the tracking of the cadence RMMEs of 5.6 rpm and 7.8 rpm were achieved respectively. A remarkable final conclusion from these graphs is the fact that at the end of the trial the blue velocity start dropping, whereas the experimental curves are growing. This could be due to the control goal, which searches for the solution with the least amount of effort, while still tracking the kinematic data well enough. As was stated in Chapter 3, to overcome this problem the final cadence was bounded to be at least larger than 11.5 rad/s or 110 rpm approximately.

Figure 4.3 shows the resulting torque on the crank during a tracking optimization of the full start, containing the slip. The rear wheel of the model cannot slip and thus produces large forces to be

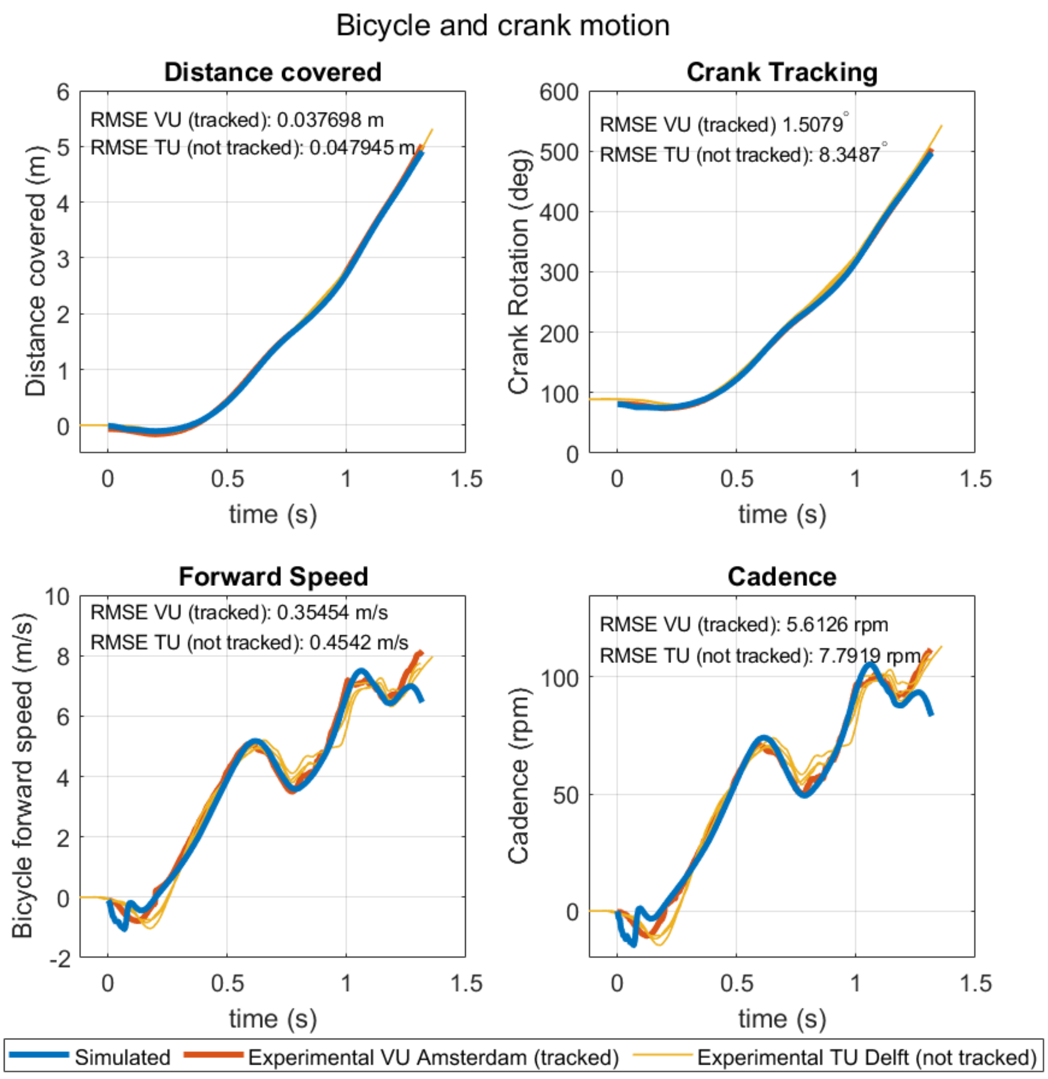


Figure 4.2: Bicycle motion, crank angle, and cadence of a complete start. The blue curve is the optimization outcome, the orange curve is the tracked kinematics from the VU Amsterdam and the yellow curves are the experimental data from the TU Delft taken from a different trial.

able to track the crank rotation. At the same point in time where the velocity jump occurred, the simulated total crank torque in blue shows a huge and narrow peak. After this peak, the torque profile looks similar to the experimental crank torque, displayed in orange. This gives hope that the model is valid, but that the slip makes the kinematic data hard to track. Additionally, the second torque peak seems to be higher than the experimental graphs. This is somehow compensated by the third torque peak which is considerably lower. Two other remarkable conclusion from this figure, are the negative torque at the start and end of the trial. As seen from Figure 4.2, the velocity starts at zero and becomes negative at first. Because the model does not have a freewheel, the virtual rider can actively propel the bicycle backward by applying in a negative. As seen from the orange curves, in reality the athletes apply a positive torque around 50 to 60 Nm. These issues were solved by starting the optimization after the moment of slip, as explained before.

Because the slipping behaviour of the rear wheel in the experimental datasets was causing invalid results for the simulation, it was chosen to start tracking the data after the slip. This entails that the simulation, for this specific trial, starts at 0.2 seconds just after the slip occurs. All results shown in the remainder of this chapter are generated using tracking optimizations starting after the slip.

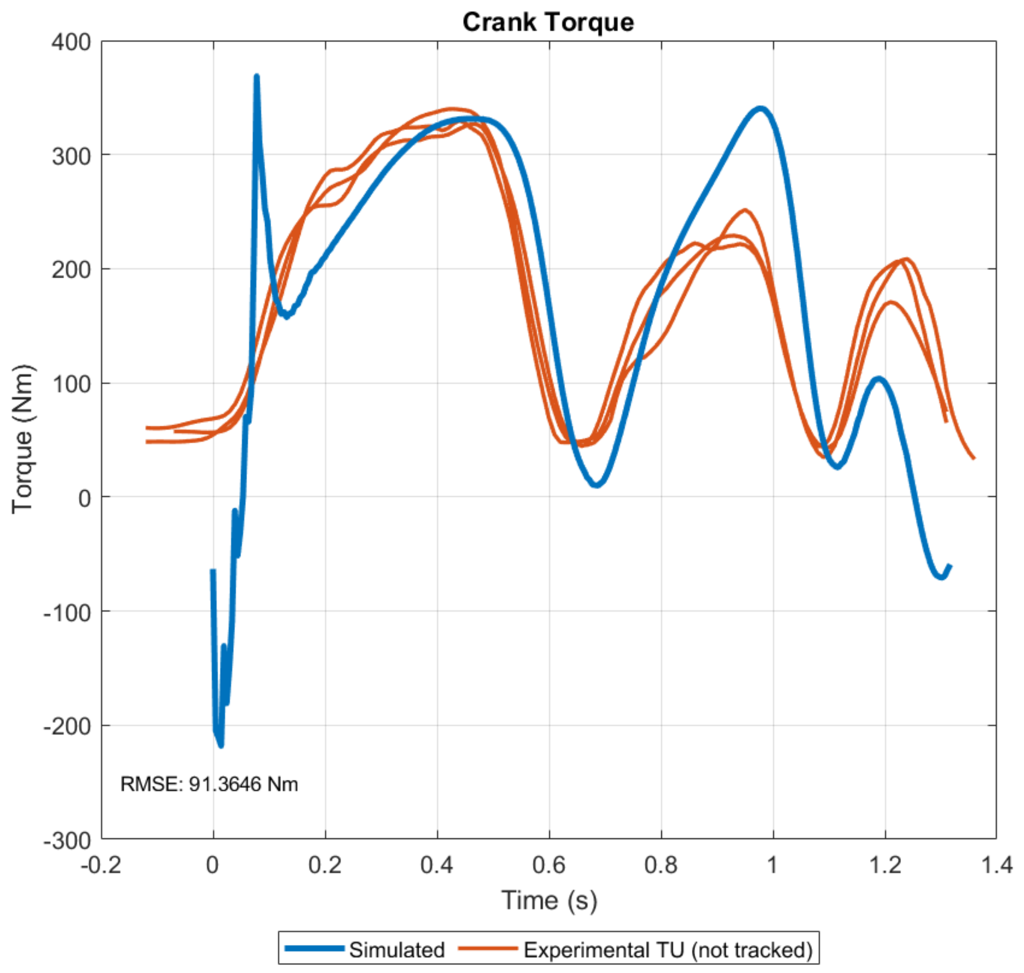


Figure 4.3: Resultant crank torque of a full gate start optimization. In blue the simulated results and in orange the experimental data obtained by Hylke van Grieken [4]. As seen there is a huge peak at 0.08 seconds, the same point in time where the velocity jump happens in Figure 4.2.

The results of the tracking optimization will be divided into several sections: the joint angles being tracked, the trajectory of the bicycle, crank angle and cadence, pedal force, crank torque and finally work and power. The kinematic parameters will be compared to the experimental IMU state trajectories being tracked. The kinetic parameters will be compared to the experimental data from three different trials with the same participant but gathered on a different day using a different bike to have an idea about the general force patterns.

4.2 Joint angles

The results of tracking the joint angles, starting after 0.2 seconds, are visualized in Figure 4.4. Again the blue curves show the optimization results and the orange dotted lines represent the kinematic data obtained from the Xsens IMU data and the OpenSense configuration. The results do not massively differ from the results for the complete start seen in Figure 4.1. However, the tracking is slightly more accurate reaching a mean RMSE error of 0.377°. Expressed as a percentage of the total range of motion of each individual joint, the average relative error is 0.52%, with a maximum of 0.8% for the left ankle. These results are just a good as or even better than the results presented in previous tracking studies using OpenSim and direct collocation optimization methods presented by Haralabidis et al. [66] and Lin and Pandy [77].

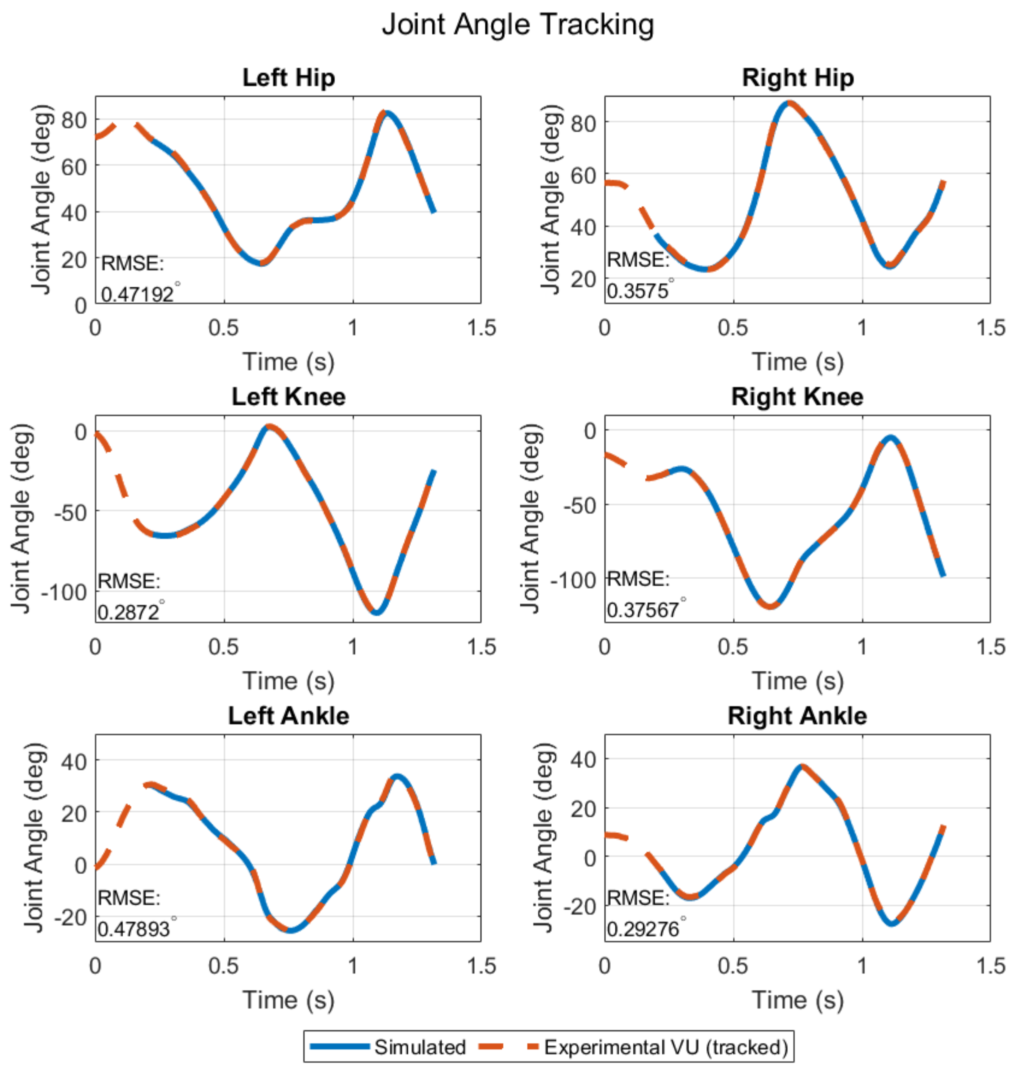


Figure 4.4: The tracked joint angles for the lower limbs of the delayed start. In orange are the experimental state trajectories from OpenSense and in blue the optimized and tracked states from Moco. The simulation starts after 0.2s to overcome the slipping of the rear wheel in the kinematic dataset.

4.3 Bicycle motion, crank angle and cadence

The graphs in Figure 4.5 have stayed more or less the same with the introduction of the delayed start. Again the RMSEs are a little smaller compared to the tracking of the entire start. The only noticeable difference is the fact that the velocity drop at the end of the two bottom graphs has disappeared due to the end velocity constraint. Additionally, since the optimization started after the slip, the speed curves show a better agreement with the experimental speeds, indicated by the speed RMSEs which have also dropped.

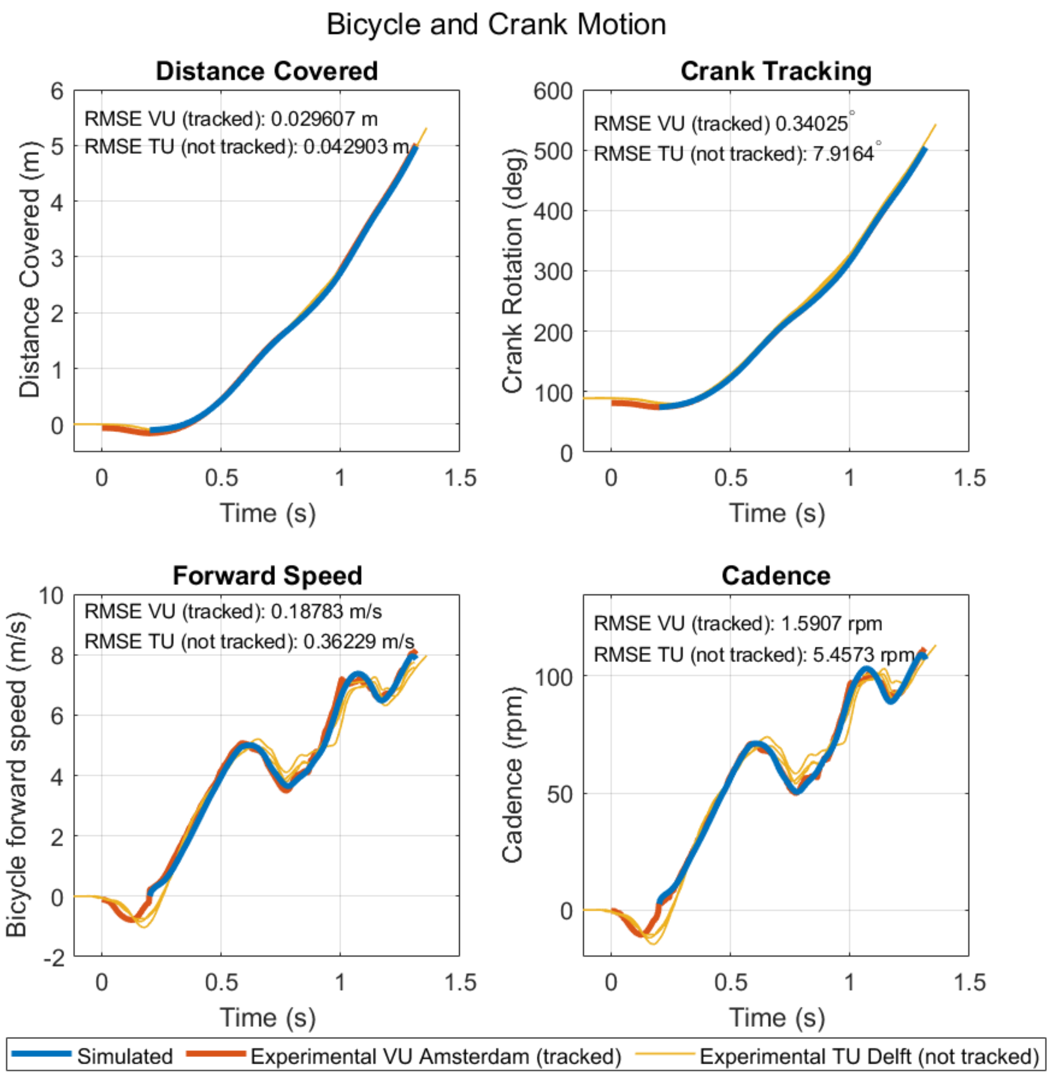


Figure 4.5: Bicycle motion, crank angle and cadence for the delayed start. The blue curve is the optimization outcome, the orange curve is the tracked kinematics from the VU Amsterdam and the yellow curves are the experimental data from the TU Delft taken from a different trial. The simulation starts after 0.2 seconds to overcome the slipping of the rear wheel in the kinematic dataset.

To visually compare the simulation results with real gate starts, Figure 4.8 shows a series of snapshots of the gate start for both the athlete in reality and the simulation. Unfortunately the video was shot from the other side of the track resulting in the gate opening mechanism in the line of sight, making it hard to clearly observe the motion. The first snapshot is taken at the beginning of the simulation at approximately 0.2 seconds after the first moment of hip motion. As seen for both images the left knee is slightly bent and the head is above the front wheel. Due to the poor video resolution and the dark colored pants for the right leg, it is hard to accurately see the joint motion of the right leg. The second image shows the center of the first pedal stroke. The left leg is now almost fully stretched and the front wheel is lifted even further from the ground to go over the gate. The third and final image shows the initiation of the second pedal stroke. As seen the cranks are aligned with the chainstay. From these images, there are two clearly distinctive differences between the video and simulation snapshots. First of all, there is a difference in the height of the front wheel. The video shows that the rider is struggling to keep the front wheel lifted from the ground, which requires a lot of torque. On the right side the simulation has lifted the front wheel even further from the ground compared to the second snapshot. The other difference worth to be mentioned is the position of the shoulders. The video shows a slightly curved back, whereas the simulation shows a perfectly straight back. This difference is caused by the lumbar extension joint. Originally this joint was in the generic model, but it was removed to keep the model simple and the reduce the number of control inputs. The model is thus not able to rotate the torso with respect to the abdomen and cannot curve its back. Because the upper body was not tracked, this does not results in troubles with the optimization.

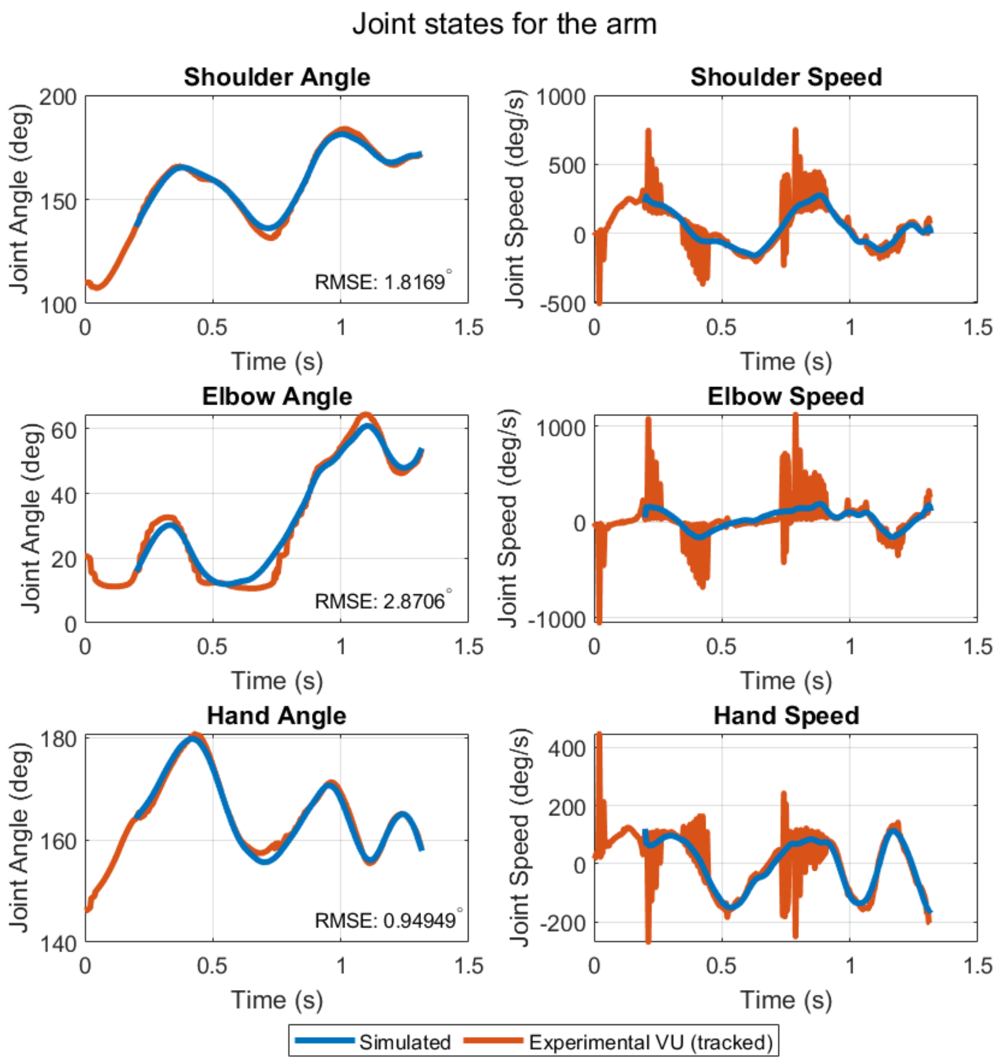


Figure 4.6: The states for the joints of the artificial arm. On the left the angles for the shoulder, elbow and hand, on the right the speeds of these joints. In blue the optimization results and in orange the speeds obtained by differentiating the joint angles from IMU data and OpenSense.

The legs and the crank dynamics were the main states to track during the optimization, but the states of the arm were also part of the kinematic dataset being tracked. As stated in Chapter 3, the joints of the arm were tracked with a lower goal weight with respect to the aforementioned states. However, the tracking of these states is still reasonably accurate as seen from Figure 4.6. The average RMSE for the three joint angles is 1.88°, which is as expected considerably higher than seen for the joint of the lower extremities. Looking at the hand angle, the three different pedal strokes can be perfectly seen by the peaks in the graph. The elbow plot shows that the arm starts in an almost stretched position, flexes slightly during the first pedal stroke and than stretches again. This is because the rider moves its hips towards the handlebar during the first pedal stroke and than shifts them back to make space for the second pedal stroke.

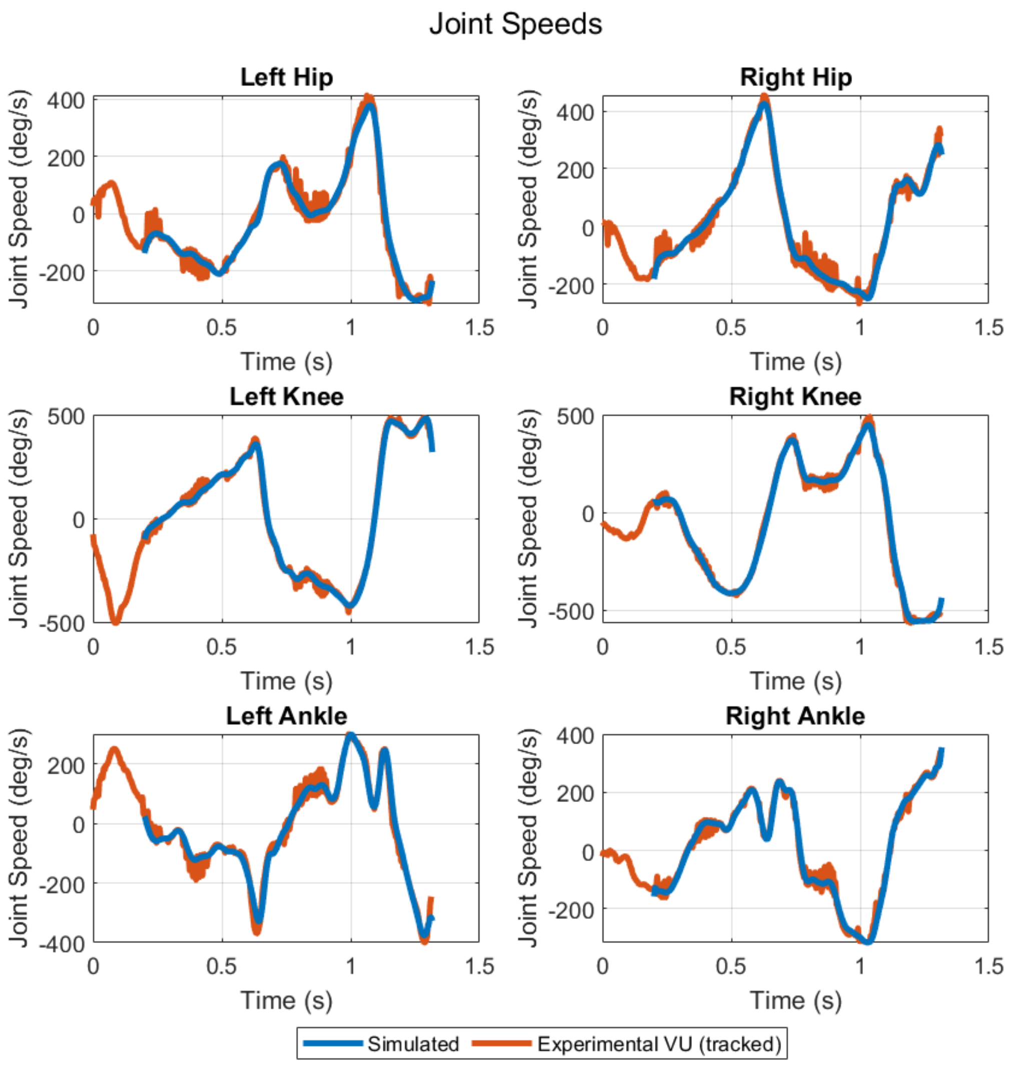


Figure 4.7: The joint speeds for the joints of the left and right leg. In blue the optimization results and in orange the speeds obtained by differentiating the joint angles from IMU data and OpenSense.

The graphs on the right showing the joint speed are a little messy. OpenSense does not provide the joint speeds, but rather the joint angles. To compute the joint speeds from the joint angles, the rate of joint angle change was calculate. This can be done by dividing the difference in joint angle between two consecutive time points by the time-step between those two points. Due to the measurement errors, these differentiated results can be become very noisy, as seen in Figure 4.6. The same can be observed in Figure 4.7, which shows the joint speeds of the two legs. Again the orange lines show some noisy behaviour, although less intense as for the arm. Because these joint angles were accurately tracked by the optimization, the joint speeds are also very similar.

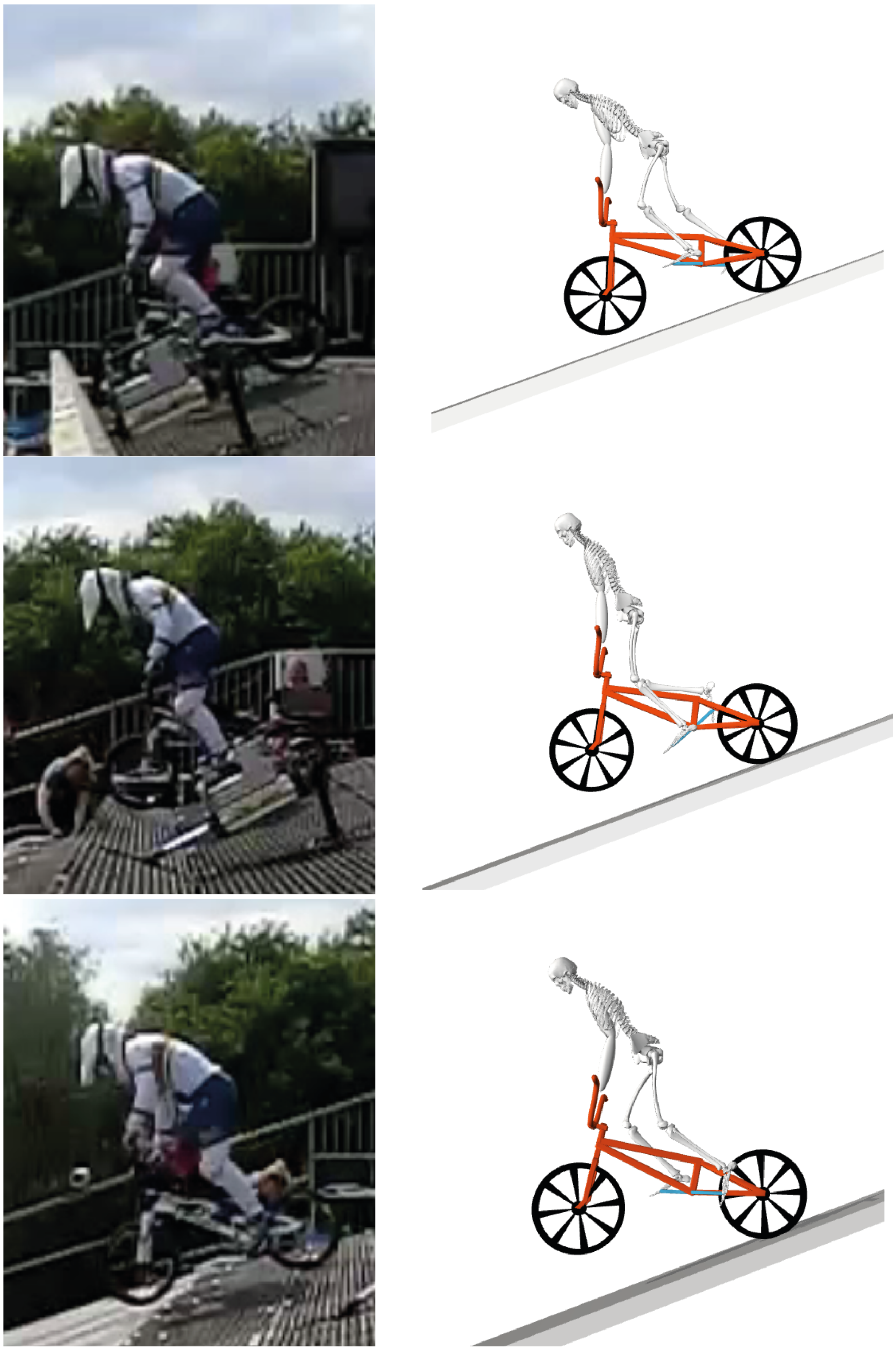


Figure 4.8: Snapshots of the start from experimental trials and simulation. The first image is taken at the begin of the simulation, just after the moment of slip. The second image is the moment when maximum torque is applied, i.e. the first pedal stroke with the lead leg. The third image is taken when the left crank is aligned with the chainstay. For the left leg the kinematics can be well compared, but due to the low resolution of the video and the dark colored pants for the right leg it is harder to compare other kinematics.

4.4 Joint Torques

The model is driven by eight optimal torque actuators located at the eight human joints. Figure 4.9 shows the joint torque generated by each of these joint actuators. As seen none of the actuators reaches the maximum torque anywhere in the simulation, indicating that the model is strong enough to track the kinematics. As was explained in Chapter 3, this was expected since the strength of the generic model was based on elite male track cyclists. Since the right leg is the trail leg, it starts with a flexion torque up till the top dead center where it switches to a extension torque. After the bottom dead center it becomes a flexion torque again. For the left leg this is exactly opposite. The definition of positive and negative torque follows the definitions of positive and negative rotation as visualized in Figure 3.20.

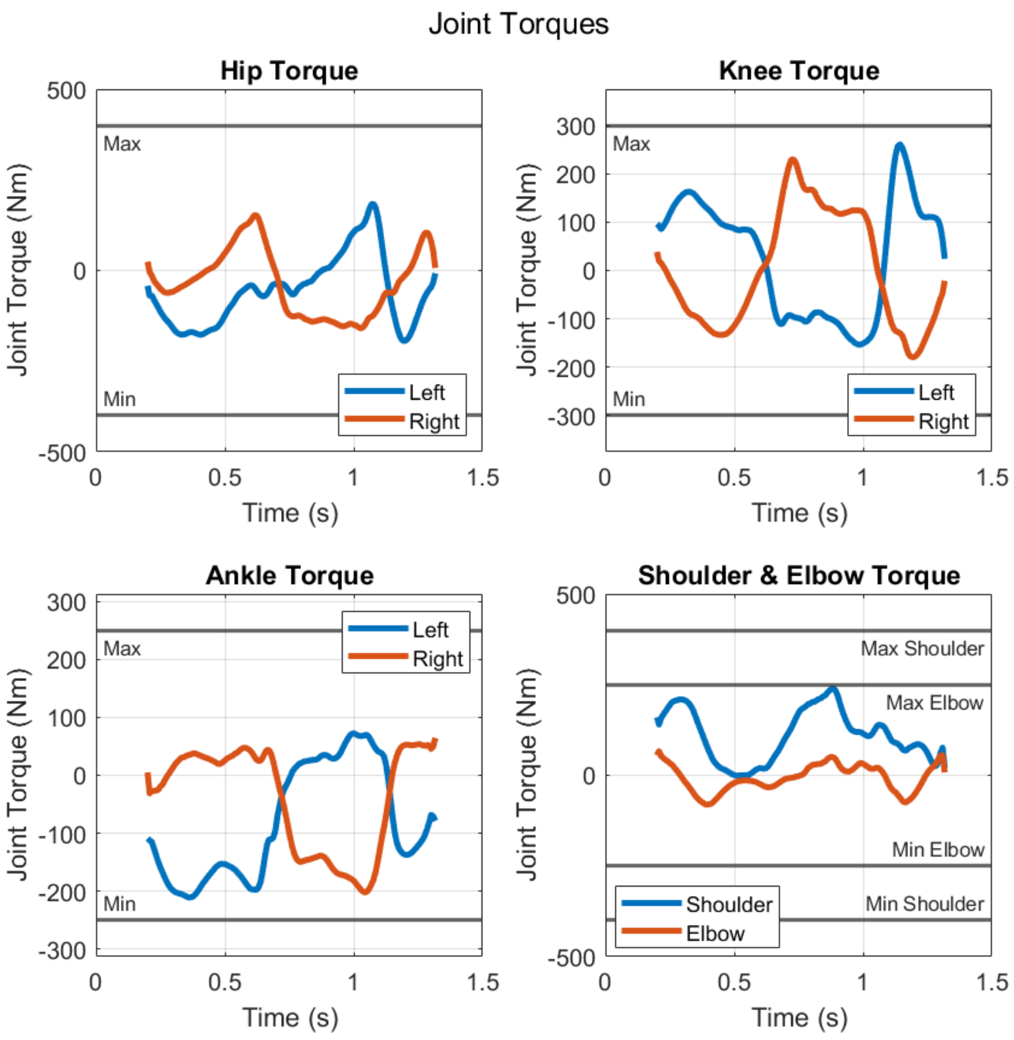


Figure 4.9: The joint torques produced by the eight actuators. The horizontal lines represent the theoretical minimum and maximum joint torques for the model. Hip: (+) = flexion, (-) = extension. Knee: (+) = extension, (-) = flexion. Ankle: (+) = flexion, (-) = extension. Shoulder: (+) = extension, (-) = flexion. Elbow: (+) = flexion, (-) = extension.

Looking at the graph for the hips, it is seen that the left leg, the lead leg, starts with a negative or extension torque to push the pedal downward. After a half rotation of the crankshaft, this torque changes to a flexion torque to pull the pedal up. For the right leg, the trail leg, this is exactly opposite. The left ankle also starts with a negative torque to keep the foot perpendicular to the shank and to transmit the force generated by the hip and knee joint effectively to the cranks. Once the left leg starts pulling, this ankle torque becomes positive, which is the flexion direction. Again for the right leg this is exactly opposite. From these joint torques the three individual pedal strokes can be well retrieved. Looking closely to the joint torque for the right ankle, it is visible that at the start and very end of the graph there are small but steep spikes in the graph. This might be the result of the initial or final speed or orientation which might be slightly off.

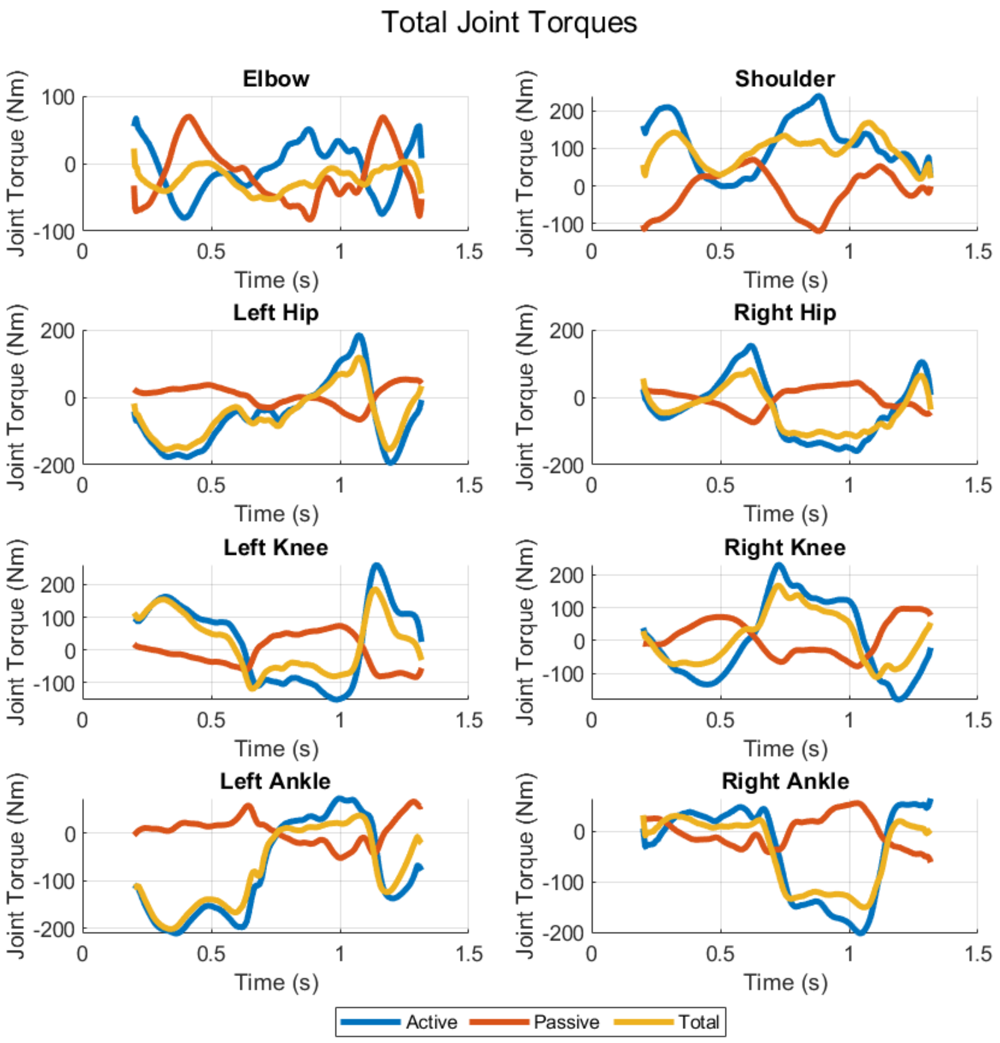


Figure 4.10: The total joint torque for each joint. In blue the active torque provided by the joint actuators. This is the same torque as presented in Figure 4.9. In orange the passive joint torque, produced by the damping torque. The yellow curve shows the total combined joint torque. Hip: (+) = flexion, (-) = extension. Knee: (+) = extension, (-) = flexion. Ankle: (+) = flexion, (-) = extension. Shoulder: (+) = extension, (-) = flexion. Elbow: (+) = flexion, (-) = extension.

Figure 4.9 shows the active joint torques for each joint. However, as was explained in Chapter 3 a damping torque was added to each joint. Figure 4.10 shows the total joint torque for each joint. The blue curves are the same as presented in Figure 4.9 and represent the torque produced by the joint torque actuators. This torque is referred to as the active torque of the joint. The orange lines show the damping torque for each joint. This is dependent on the damping coefficient and the angular velocity of the joint. This torque is referred to as the passive joint torque. As seen for the elbow and shoulder in the upper two graphs of Figure 4.10, the active and passive joint torques seem to counteract each other. The resultant torque for the elbow, shown as the yellow line, is close to zero during the entire start. This does not mean that there is no torque coming from the elbow. This phenomena is in human movement science referred to as co-contraction. Co-contraction happens when an agonist and an antagonist actuator are activated simultaneously to provide joint stability. This is exactly what happens in the elbow. While the athlete is pulling on the handlebars, the elbow must remain stable in an almost flexed position. This is one of the reasons the damping had to be included in the model to achieve realistic results. For the joint of the legs, this co-contraction is less visible and the damping forces provide less torque, but just enough to stabilize the joints and to remove high oscillations from the activation. The coordinate limit force introduced for the elbow showed to produce zero torque during the entire trial. Apparently the damping was sufficient to stabilize the elbow joint in an almost stretched position.

4.5 Pedal force

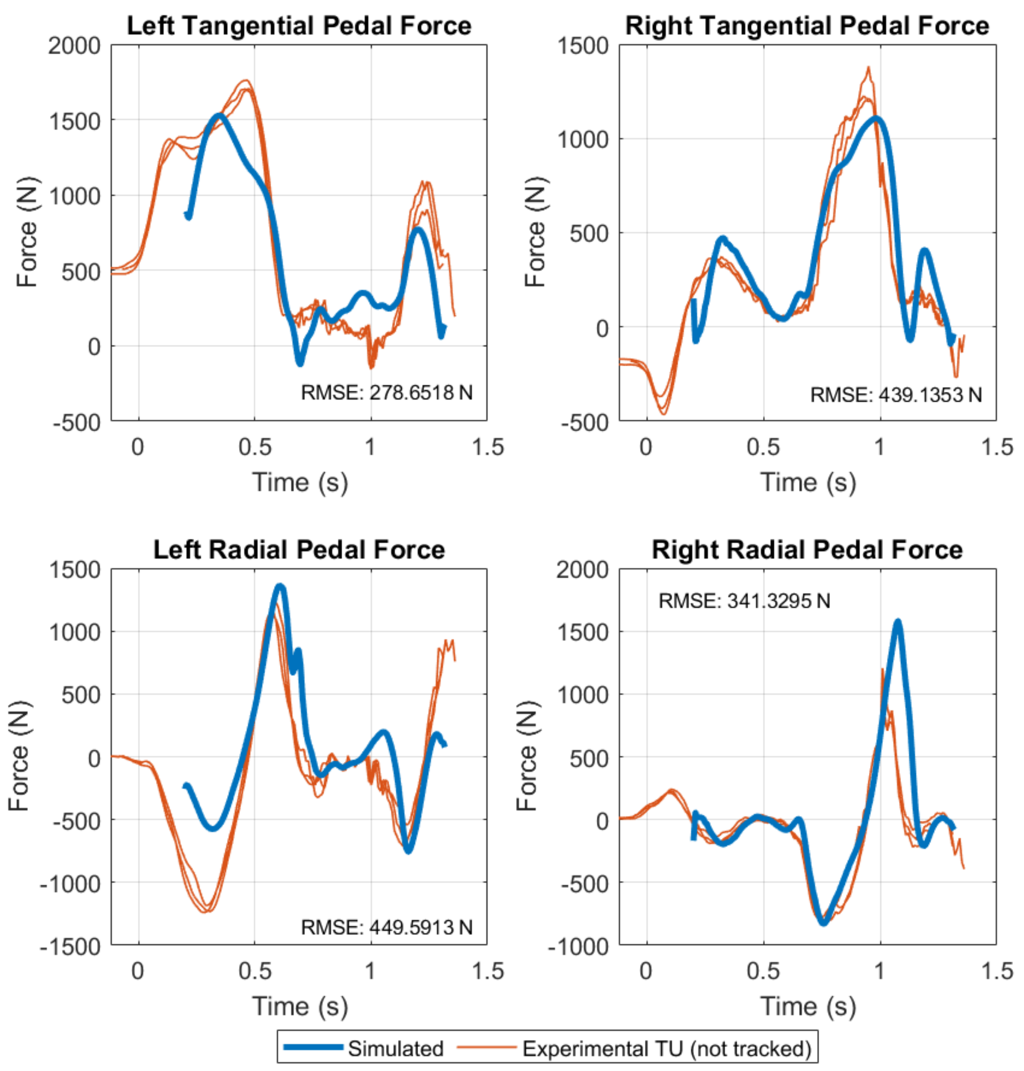


Figure 4.11: Forces applied on the pedals in tangential (effective) and radial (ineffective) directions. In blue the simulated results and in orange the experimental data obtained by Hylke van Grieken [4].

Figure 4.11 shows the forces exerted on the pedals in the tangential and radial direction for both legs. The pedal forces are the constraint forces introduced in the model to keep the feet attached to the pedals. In blue the pedal forces from the simulation are shown and the orange lines correspond to the experimental data determined by Van Grieken [4]. This data was not tracked, but the outcome can give a good idea about the validation of the model. As seen the general shapes are very similar. The left tangential force peaks at around 0.4 seconds, which corresponds to a crank angle around 150°. Shortly after this tangential peak, the radial peak occurs, indicating one of the dead centers. The same pattern can be observed for the right pedal forces. RMSEs between 220 and 450 N are reached. Noteworthy is the third, smaller, peak for the right tangential pedal force around 1.2 seconds, which does not seem to correspond to the experimental data. It looks like the simulation switches to a different way of applying force for the final part of the trial compared to the experimental results. The second peak for the left tangential force is also smaller for the simulation than for the experimental data, which supports the change of technique argument for the final section of the start. In general, the force peaks for the tangential force are slightly lower for the simulation compared to the experimental peaks. However, for the simulation there is a tiny peak for the left tangential force around 1 second. The model seems to be able to apply a pulling force more effectively than the experimental data shows.

A nice way to visualize pedal forces is by plotting the effective force against the crank angle in a polar plot. This clearly shows at what sections of the pedal strokes the pedal forces reaches their peak values. Figure 4.12 shows such a representation of the pedal forces. From this graph it is clear that the effective force peak for the left leg is lower than the experimental data, something that was also visible in Figure 4.11. This is compensated by a larger effective force in the remainder of the pedal cycle. As seen the force applied by the left leg during the top left quarter, the pulling force, is larger compared to experimental force. For the right leg these differences are less clear since the simulated and experimental data almost coincide. The largest force is applied in the bottom right quarter, during the pushing phase of the pedal stroke. A smaller peak can be found around 270° which is caused by pulling on the pedals.

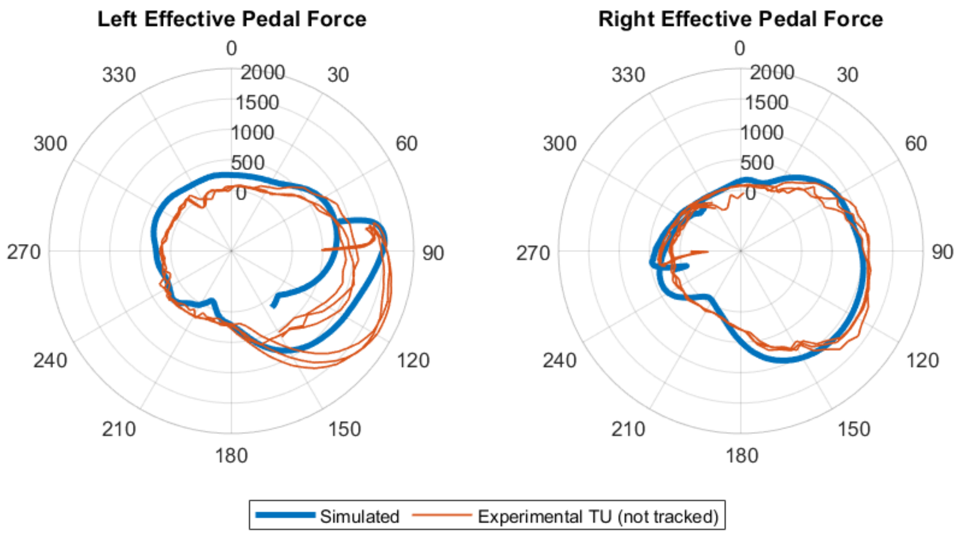


Figure 4.12: Polar plot of the effective pedal force for the left and right leg. In blue the simulated force and in orange the experimental force measured by Van Grieken [4]. As seen the largest force is applied during the bottom right quarter.

4.6 Crank torque

Figure 4.13 shows the resulting torque on the crankshaft for the simulated and experimental outcome. Again, the simulated curve starts at 0.2 seconds, so after the slip. The crank torque is simply the summation of the effective pedal forces multiplied by the crank arm which is 175 mm in length. The crank torque peaks for the simulation are very similar to the experimentally obtained crank torque peaks. The smaller effective force peaks seen in Figure 4.11 are compensated by the stronger pulling forces, resulting in similar torque peaks. The first torque peak is approximately equal in height at around 325 Nm. The drop after the first torque peak seems to happen slightly earlier for the simulation. This results in a smaller area under the graph compared to the orange experimental curves. The second peak however is a little larger for the simulation compared to the average experimental data. Besides, it seems to be a little wider. Moreover, the drop between the first and second pedal stroke and after the third pedal stroke is a little lower than the experimental data. Also at the start of the trial there is a strange difference between the blue and orange graphs, which could be explained by the initial speeds which were based on previous optimization solutions. It was tried to resolve this issue by loosening the initial constraints, but this resulted in other errors. In general the similarity between the simulation and measured torque is surprisingly high, considering that none of the force data was being tracked and the kinetic data was taken from a completely different trial. In total there was a RMSE of 56 Nm, which corresponds to a relative error of approximately 18%.

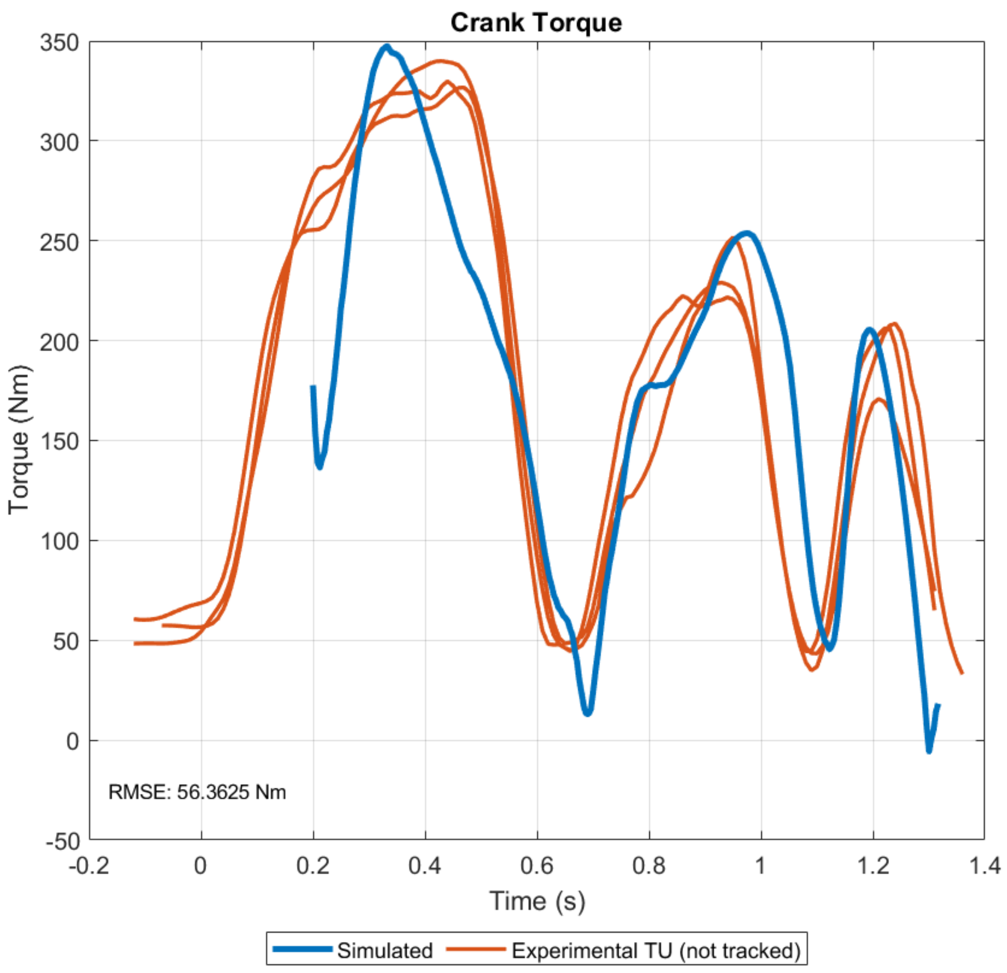


Figure 4.13: Resultant crank torque. In blue the simulated results and in orange the experimental data obtained by Hylke van Grieken [4].

4.7 Work and power

As a final comparison between simulated and experimental outcome the kinetic parameters work and power can be set side by side. Figure 4.14 shows the power profile of the gate start. The power can be calculated by multiplying the crank torque with the cadence for each point in time.

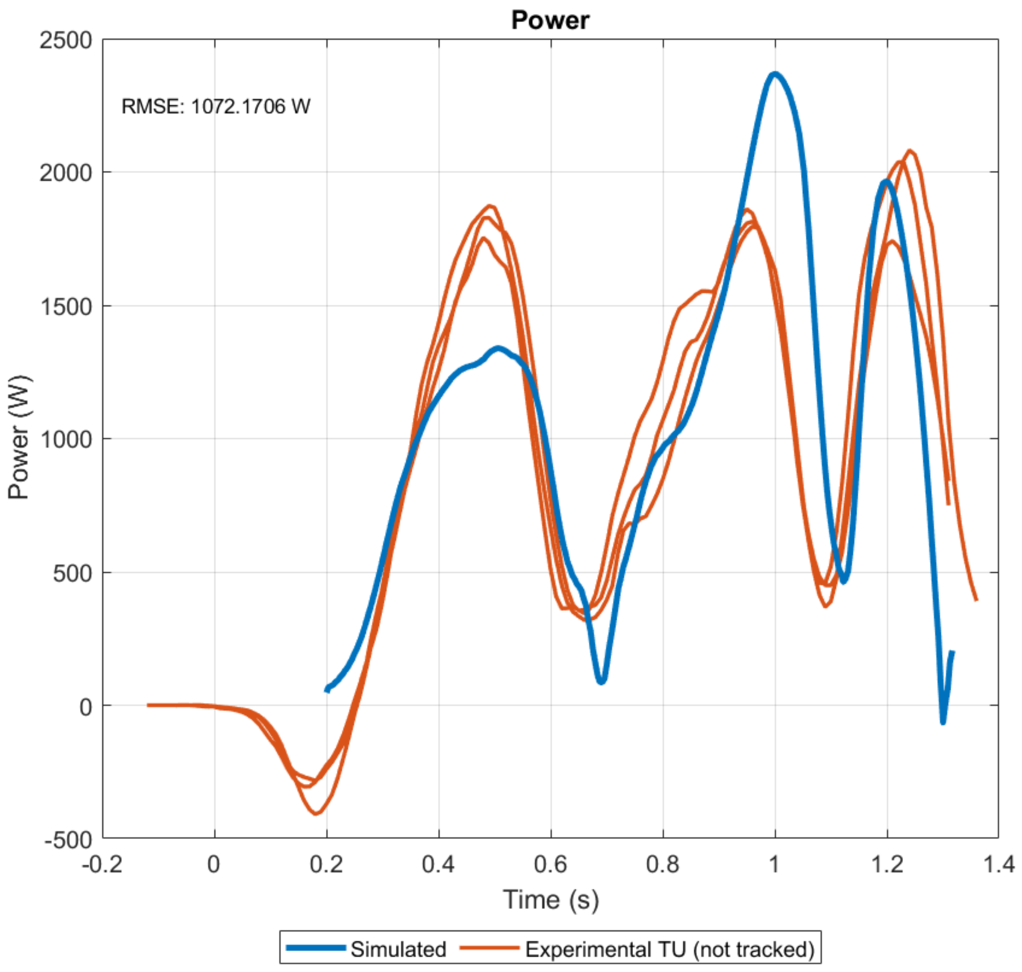


Figure 4.14: Power profile of the BMX SX gate start. In blue the simulated results and in orange the experimental data obtained by Hylke van Grieken [4].

The first noticeable difference is the dissimilarity in power at 0.2 seconds. The experimental data from the TU Delft shows a negative power because the cadence is negative, whereas the experimental from the VU Amsterdam being tracked is positive at that point in time. This can be well observed in the cadence plot in Figure 4.5. The tracked solution shows thus also a positive cadence resulting in a positive work and power. Secondly, the first power peak is lower than the experimental power peak, whereas the second power peak is larger for the simulation. This can be partly explained by the crank torques in Figure 4.13. The first torque peak is approximately just as high as the experimental torque peak, but the crank velocity is lower, resulting in a lower power peak. For the second peak, this works the other way around. The torque peak is slightly higher, resulting in a higher power peak. Additionally, the crank speed shows to be slightly higher for the second peak as seen from Figure 4.5, which enlarges this effect. The third power peak seems to be equally high for both the simulated and measured outcome. The first power peak is 478.9 W lower, the second power peak 544.8 W higher, and the final power peak is just 11.8 W higher than the average of the experimentally determined peak powers. This results in relative errors of -36%, 23%, and 0.6% for the three peak power respectively. A significant difference is the fact that for the experimental data the overall power peak occurs at the third pedal stroke, whereas for the simulation the second pedal stroke shows the higher power output. Due to the various differences in both torque and cadence, the total RMSE for the complete power profile is 1072 W or close to 43%.

Figure 4.15 shows the cumulative work done. The work can be calculated by multiplying the crank torque with the crank angle difference for each time-step. Adding the work done between each two consecutive time-points, results in the cumulative work. Because it is cumulative the relative errors will also add up. However, the figure shows that the simulated result is very similar to the experimental work done. As seen the work increased also piecewise and especially for the first step, there is a large similarity between the blue and orange curves. At the second step, between 1 and 1.2 seconds, there is larger deviation, due to the crank torque difference seen in Figure 4.13. Nevertheless, the simulated work shows a RMSE of only 43 J with respect to the experimental work, which corresponds to a relative error of just 3.6%.

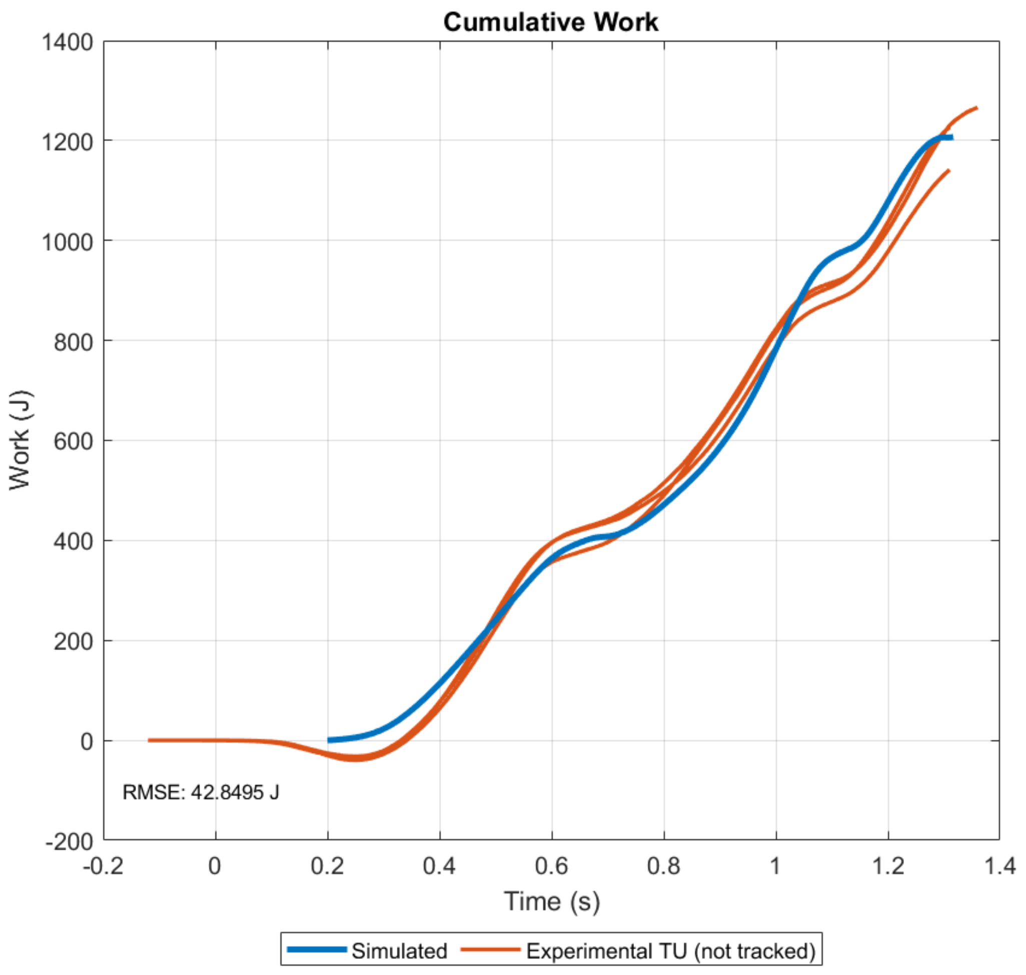


Figure 4.15: Cumulative work done during the BMX SX gate start. In blue the simulated results and in orange the experimental data obtained by Hylke van Grieken [4].

5 | Discussion

5.1 Study objectives and key findings

The major aim of this study was to develop a model that can reproduce experimental BMX SX start kinematics within a 5% margin. To achieve this goal a BMX SX model had to be built from scratch in OpenSim containing the most important aspects and characteristics of the gate start. This entails the possibility of the recoil motion and the wheelie during the first section of the start. The combination of bicycle and rider was optimized using an optimization package primarily made for models in OpenSim called Moco. It uses a direct collocation optimization routine to solve for the model's actuation that matches the experimental kinematic data as close as possible. The results showed an average tracking error of 0.377° for the joints of the lower limbs. This equals an average relative tracking error of only 0.52%, which is ten times more accurate than originally desired. Compared to other tracking studies, such as the tracking of the running sprint start presented by Haralabidis et al. [66] and Lin and Pandy [77] reaching an accuracy close to 1%, these tracking results are almost twice as accurate. Also for the tracking of the crank angle and distance travelled, extremely accurate results were found with a relative error of 0.079% for the crank angle and 0.6% for the distance travelled. The difference between those two has to do with the exact numbers used to convert the crank angle into distance travelled. These numbers are dependent on the crank length, gear ratio, and the circumference of the rear wheel. Due to the elastic behaviour of the tires and the way this is modeled in OpenSim, there exists a small difference for these numbers between the experimental and simulation approach. Nevertheless, based on these kinematic results, it can be concluded that the major goal for this study was achieved.

Looking at the kinetics of the simulation outcome, the optimization solution follows the same profile in terms of crank torque, power, and cumulative work. The characteristic torque peak pattern, as seen during BMX SX but also during other types of cycling, is clearly the same for the experimental and simulation results as shown in Figure 4.13. Peak values for the torque peak differ 4.4%, 7.7%, and 5.1% for the first, second, and third torque peak respectively, with an absolute difference of only 15 Nm. This data was not tracked and was not even taken from the same trial, so differences between simulation and the experimental outcome were expected. The relative errors indicate that the second simulated torque peak shows the weakest agreement with experimental data. The torque profile clearly shows some strange behaviour at the start and end of the trial. This is probably due to the initial and final constraints on the states. Figure 4.7 shows some rapid changes in joint speed at the start and finish of the trials. The mismatch between kinematic and simulated speed is best shown in the plot for the right hip, where the end velocity of the simulation tends to drop quickly near the end. Constraining the end velocities of the states could solve this issue, as seen for the crank velocity. However, final speeds with the accuracy needed are hard to obtain from joint angles only. The same is valid for the initial velocity. The approach used in this work allows for a variability of 10% for the initial velocity based on a previous solution. Future work could focus on a strategy to retrieve the initial and final velocities of the joint angles from the Xsens IMU data or search for a way to track joint velocities as well.

As seen from the individual pedal forces displayed in Figure 4.11 and 4.12, the simulation tends to perform the third torque peak with both the right leg and the left leg, whereas the experimental data shows a major contribution of the left leg. The tracking problem ends at the end of the kinematic data trial (at 5 meters), which makes the solution prone to these end-effect errors. In reality, the athletes have to continue their descent down the ramp, whereas the model just stops after 5 meters and does not take into account how the descent continues. According to these kinetic results, the model has managed to produce the same kinetic characteristics (pedal forces, crank torque, power, and work) without even tracking them.

In general, these results give confidence that the model could function as a framework for further research regarding simulations of the BMX SX gate start. Although many simplifications were made, e.g. the omittance of a steering assembly, having only one arm, the highly simplified actuation, and the rather simplistic way of modeling the wheel to ground contact, the model is perfectly able to track kinematic data with an extremely low RMSE, while also following the kinetic characteristics of the BMX SX start. This model framework could be used in the future to study the influence of specific training aspects and investigate a large range of "what-if" scenarios as will be further described in Chapter 6.

5.2 Comparison with literature

Joint torques

Joint torques found for tracking the kinematic data roughly range between -190 and 175 Nm for the hip joints, -210 and 65 for the ankle joint, -175 and 250 Nm for the knee joints, between 0 and 250 Nm for the combined shoulder joint, and finally between -70 and 60 Nm for the combined elbow joint. These ranges are well in line with joint torque published in literature. Kordi et al. [56] determined the joint torques during cycling experimentally and reported knee joint torques around 200 Nm for cycling at a power output of 1500 W. Given the fact that these experiments were done with professional male cyclists, the joint torque actuation for the simulation model might be a little high considering the fact that the kinematic tracking data was taken from a female elite BMX SX athlete. However, as seen from the experimentally determined power output presented in Figure 4.14, this female athlete is able to produce peak powers of 2000 W, which is considerably higher than the 1500 W used in the experiments of Kordi et al. For this same power output of 1500 W, Kordi et al. [56] found an average of 400 Nm for the joint actuation of the hip. This is considerably higher compared to the model outcome. For the ankle joint an average joint torque of 150 Nm was found, which is well in line with the model outcome. Watier et al. [78] found max joint torque 110, 225, and 100 Nm for the hip, knee, and ankle joint respectively by measuring professional triathletes. As seen there is a large difference in the joint torques for the hip between the measured data of Kordi et al. [56] and Watier et al. [78], which could be due to due position of the athlete. Kordi et al. measured the athletes while seated, whereas riders were allowed to stand during the experiments of Watier et al. Comparing the simulation joint torque with the maximum torques found by Watier et al., it can be concluded that the knee torque shows a good agreement, but for the ankle and hip torque, there is a larger difference. However, Watier et al. did not exactly specify the level of the triathlete participating in their study, so it is hard to draw conclusions from these results. Unfortunately, this work was the first to estimate the joint torques during the BMX SX gate start, so no comparisons can be made with other BMX literature.

Crank torque, pedal forces, and power profile

The crank torque and power profile outcomes of the simulation are already validated by comparison with experimental data from Van Grieken [4]. It was found that the second torque peak is a little higher compared to in-field measurements. This causes the second power peak to be also higher than experimentally determined. Other reported torque peaks in literature show numbers in the same range as determined by Van Grieken. Gross et al. [2] found a torque peak for the first pedal stroke of 231 $\pm$ 24 Nm in a study with junior and elite BMX cyclists, which is slightly lower compared to the experimental results of Van Grieken who found an average torque peak of 297 $\pm$

23 Nm for the female BMX athletes. Regarding individual effective pedal forces, Janssen and Cornelissen [79] reported 1060 ± 188 N for the first pedal stroke of the lead leg and 934 ± 116 N for the trail leg for female junior BMX and track cycling athletes. As seen from Figure 4.11, pedal forces reported by Van Grieken [4] were almost 1.5 times as large. This could be explained by the fact that the female athlete participating in the experiments of Van Grieken was a world-class BMX athlete, which is hard to compare with junior athletes.

In terms of power, the simulated peak power exceeds the peak value experimentally determined for the second pedal stroke. Van Grieken [4] reports an average peak power for the female athletes of 2414 ± 199 W occurring at the third pedal stroke. This value is however taken from experiments on the larger Olympic starting hill, whereas the experiments and the experimental data taken in Figure 4.14 are from the smaller training ramp. For the training ramp, an average peak power of 1953 W was found over the three trials being analyzed. The simulated power peak is, as said, slightly higher at 2367 W. Surprisingly the peak happens at the seconds pedal stroke rather than the third. Comparing the peak power with other literature leads to the same conclusion as comparing it with experimental data from Van Grieken. Gross reports a peak power during the start of 2000 ± 260 W for the faster group of junior and elite BMX athletes, which is close to the experimental data shown in Figure 4.14. In a recent study Daneshfar et al. [80] reported peak power for sub-elite male BMX riders of 1288.7 ± 62.6 W, which is almost half the experimental peak power observed in Figure 4.14. This could be due to the relatively low sampling frequency (1Hz) of the power meter used compared to the sample rate of the Swift 2DAxis, which has a sampling frequency of 100 Hz. Finally, Bertucci and Hourde [81] reported an average peak power output of 1968 ± 210 W for elite male athletes on a non-Olympic starting ramp with a constant incline of 17.5%, which is comparable to the non-standard training ramp at Papendal where the kinematic and kinetic data used in this study were taken from. On average, the power profile shows good agreement with experimental data. The simulated peak power lies within the range of values reported in literature, although the moment of peak power is not in line with literature.

A final kinetic measure to compare with literature is the force exerted on the handlebar. There is however only one study that presented these forces [82] and only the peak value was reported, so comparison has little meaning. The results of this comparison can be found in Appendix F.

5.3 Points of improvement

For the construction of the BMX SX model, a lot of simplifications were made. These could be adjusted in future work to reach a better agreement with experimental data. Due to the structure of the model, each of the components can be adjusted individually if higher accuracy regarding that specific aspect of the model is desired. The sections below will describe the main building blocks which could be improved in future work.

Slip and freewheel

As explained in Chapter 3.2.2 the original kinematic data contained a clear slip of the rear wheel. This is best seen in Figures 3.18 and 3.19. As shown in this latter figure, the slip happens approximately at the moment of furthest recoil. The solution to overcome this slipping was to start the tracking after this moment of slip. Consequently, the recoil moment, which was indicated by literature to be of huge importance [20], is cut out of the trajectory. The recoil motion might be the most interesting part of the entire gate start, so leaving that out of the tracking is not ideal. Having the recoil motion in the tracking trajectory will also require a way to model the freewheel instead of a fixed gear coupling between the crankshaft and the rear wheel. A possible solution has been presented in Chapter 3.2.2 using the ExpressionBasedCoordinateForce. Future work should focus on how to implement this in the current model or develop a new coupling to accurately model the freewheel. Overcoming the slip issue is a little more complex. A first approach would be to take out the slip in the kinematic data. This would imply a rear wheel outer tire with more grip so that slip does not happen anymore, which might be advantageous for overall performance as well. Remarkably, the data set of Van Grieken [4] does not contain the slip, as seen from Figure 4.5. This could be due to the different bicycles used during the kinematic and kinetic experiments. Redoing the kinematic experiments with the same bicycle with improved grip might solve the slipping issue. Another approach would be to have a more sophisticated tire model to represent the contact between the wheels and the ground. In the current model, the rear wheel height is fixed, clamping the rear wheel to the ground. This is the fastest and most simple way to get the interaction needed to propel the bicycle. However, a more realistic way would be to include the vertical degree of freedom for the bicycle and rear wheel. This requires a more complex tire and ground contact model.

Contact modeling

When constructing the contact model for the rear wheel and the ground, the only demands were a no-slip condition and that the torque needed to propel the bicycle was close to experimentally determined values. As seen from the graphs in Figure 4.13 these goals were reached, especially for the first and final torque peak. However, the second peak is a little larger than the experimental data while the crank angle and bicycle displacement are accurately tracked. This could be due to the simplified contact model that transmits the force exerted on the pedals to the ground. A more complex contact model that accounts for the elastic behaviour of the tires, such as the approach presented by Cangley et al. [47] or Jansen [38], might be able to get a closer agreement with experimental data. Moreover, it might allow slip in the model, which makes it possible to track the entire start including the recoil movement. Furthermore, the contact geometries of the wheels were modeled using spheres. In OpenSim, a couple of default shapes could be selected for the Hunt-Crossley contact force. Unfortunately, disks were not available, so spheres were selected since the model has no roll angle degree of freedom. The contact surface between the ground and the sphere is larger than for a disk. Future models could take this into account when improving the contact modeling. Nevertheless, the current approach shows an agreement for the peak torques with an average error of only 5.7%, which is within the error margins expected due to the different trials the kinematic and kinetic data were taken from.

To investigate the influence of the clamping height, the stiffness, and the friction coefficients on the simulation, Figure 5.1 shows the outcome of a variation of these parameters on the distance travelled and the total rotation of the crankshaft both starting at 0. The idea of these tests was to construct a model of just the bicycle with a weight added to the virtual seat to mimic the weight of the rider. This is the same approach as was used during the rolling test presented in Chapter 3. Next, a constant crank torque of 250 Nm was applied to the bicycle, which is approximately the average peak torque during the gate start. A simulation with a duration of 0.5 seconds was run and the outcome of these simulations was the distance travelled and the crankshaft rotation. For each virtual experiment, just one of the parameters was changed while all others were held constant. The crankshaft rotation and the bicycle displacement can tell whether or not the rear wheel had slipped or show other effects of these contact parameters. For simplicity, the three friction coefficients were said to be equal.

The graphs in Figure 5.1 show that the parameters have a huge impact once a threshold is passed. For the vertical distance of the bicycle or the clamping height, this threshold seems to be 0.304 m. This number is a combination of the thickness of the ground surface as seen in Figure 3.16 and the radius of the front wheel, which is approximately 0.25 meters. At this particular clamping distance, the crank rotation makes a large jump, indicating that the rear wheel is slipping. This is also seen from the distance travelled which drops rapidly, indicating that the crank torque is not effectively transmitted to the ground surface. This makes sense because once the bicycle is placed too high above the ground surface there is not enough contact surface to create the grip needed to propel the bicycle forward. As seen, the simulation result is barely influenced between a vertical distance of 0.295 and 0.303 meters. The relative difference between the crank rotations for these two clamping heights is approximately 2%. What does differ is the time needed to run the simulation. The smaller the vertical distance, the longer it takes to run the simulation. Therefore, it was chosen to run the optimizations with a height of 0.3 meters, which shows to be a balanced combination of grip and simulation time.

The middle graph shows the influence of the stiffness of the ground surface. The same reasoning as for the vertical distance of the bicycle is valid. Below a stiffness of 0.5×10^8 N/m the crank rotation

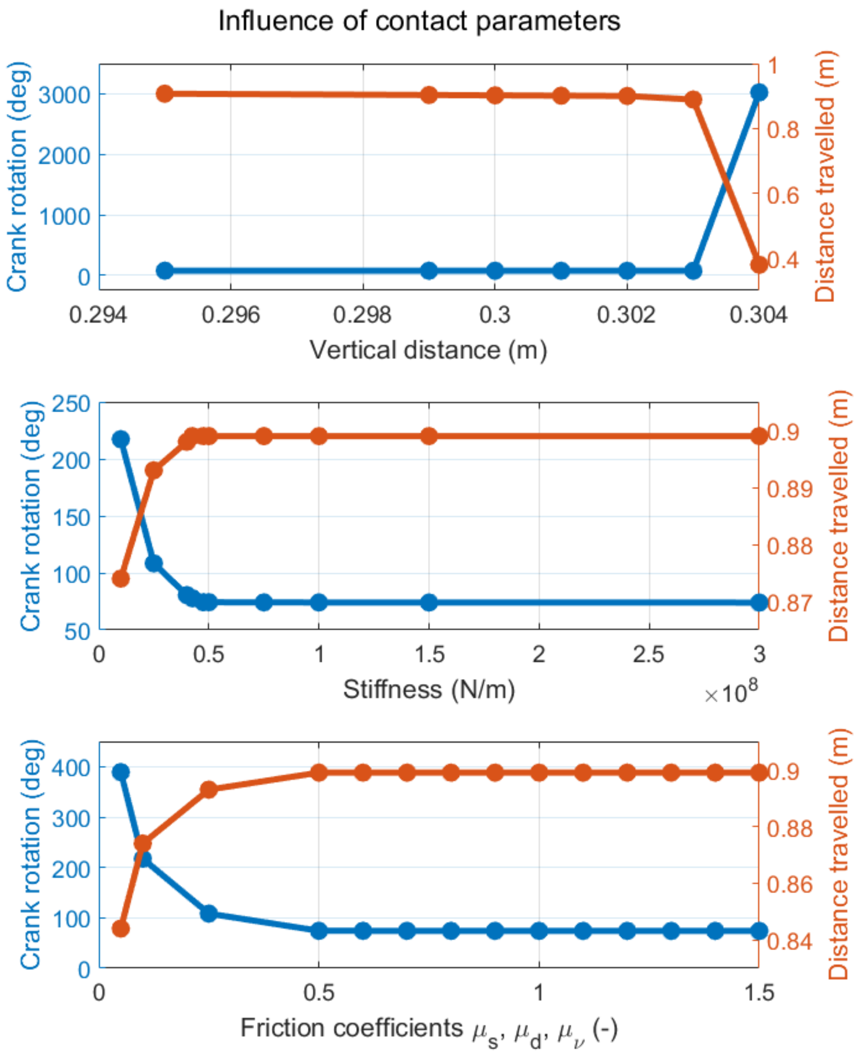


Figure 5.1: The influence of ground contact parameters on bicycle displacement and crankshaft rotation. The blue axes show the crankshaft rotation as a function of the contact parameter. The orange axes show the displacement of the bicycle as a function of that same parameter.

starts to increase while the distance travelled drops. Apparently, below this threshold, the stiffness provided not enough force to avoid the rear wheel from slipping. Again for higher stiffness, the simulation does not change significantly, but the simulation time increases massively. Therefore, a value of 1×10^8 N/m was chosen.

The bottom graph shows almost the same pattern as the middle graph. Below a friction coefficient of 0.5, the rear wheel starts slipping. For the optimization, a value of 1 was chosen for the contact parameters to prevent slipping. This shows to be a great pick to ascertain valid results, while also keeping a reasonable simulation time.

Another approach to improve the ground contact could be to track external forces as well. If kinetic and kinematic data is taken from one and the same trial the crank torque could be tracked to improve the reliability of the solution. This approach was used in the work of Haralabidis et al. [66]. There exists a dataset that combined the Xsens kinematic measurements of Van Dilgt [11] with the Swift Axis2D crank for measuring pedal forces by Van Grieken [4], but these two individual datasets are not synchronized nor processed yet. Performing the data analysis necessary to use these datasets at the same time was out of the scope of this work, but could be an interesting path to follow for future work.

Actuation

The model in its current form is driven by eight optimal torque actuators in each joint. These actuators are optimal in the sense that they have an efficiency of 100%, no delays are included, and they are not constraint by the joint's orientation or velocity. This is far from realistic since normal human actuation is the result of multiple muscles per joint. The forces these muscles can produce are dependent on the length of the muscles, i.e. the orientation of the joints, and the contraction velocity of the muscle fibers, i.e. the joint velocities. Additionally, human muscles suffer from activation dynamics caused by the re-uptake of Ca^{2+} ions [36], which is not captured by the optimal torque actuators as well. Future work should create a model with a higher level of accuracy regarding the actuation of the model. In OpenSim, muscle models exist that are specially designed to function in combination with Moco or other forms of direct collocation optimization. An alternative could be the ActivationCoordinateActuator[1] which takes the activation dynamics into account but is still optimal in the sense that it does not account for length and velocity dependency. A final option could be to develop a custom designed joint torque actuator that accounts for the orientation and velocity of the joint. Examples of such models have been presented in literature [83] [84], but it does not yet exist within OpenSim. The effectiveness of this method has been proven by the work of Jansen and McPhee [85].

Pitch angle

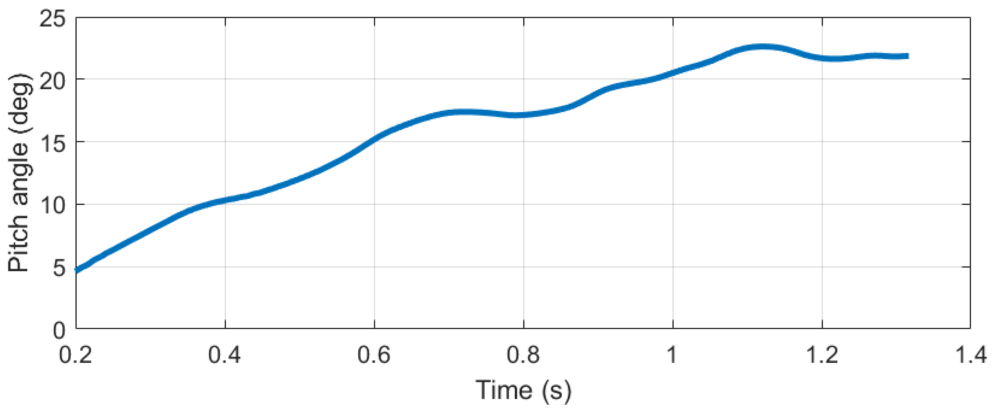


Figure 5.2: The simulated pitch angle in time. The pitch angle is generally growing in time, whereas it was expected to gradually drop to zero.

A big downside of the kinematic dataset is the lack of information about the pitch angle of the bicycle, i.e. the wheelie angle. Comparing the video footage with the simulation in OpenSim, it is clear that the pitch angle is incorrect. The pitch angle is displayed in Figure E.2. As seen, the pitch angle is growing in time starting from 5° up to a maximum of approximately 23°. The model can distribute its weight and torque production perfectly and is be able to keep the front wheel from the ground. In reality, this is extremely hard for athletes, and a lot of torque is required to keep the front wheel in the air. Among other things, this could be the reason why the second torque peak shown in Figure 4.13 is slightly higher compared to experimental results. From video analysis, it is seen that male athletes ride on their rear wheel only up to the kink, but the female athletes cannot produce enough torque to do so. Usually, they place their front wheel back to the ground during the second pedal stroke. The accuracy of the model tracking could have been improved if the pitch angle was also tracked during the optimization. The pitch angle was not directly measured by Van Dilgt [11] as it was claimed that the rotational energy of the bicycle was negligible with

[1]https://opensim-org.github.io/opensim-moco-site/docs/0.4.0/class_open_sim_1_1_activation_coordinate_actuator.html

respect to the total rotational energy of the rider and thus was not needed in the calculation of power output. Because no pitch angle information was provided to the OpenSense data conversion, OpenSense does not provide a pitch angle that can be tracked by the optimization. Its solution shows a stationary bicycle with a moving human on top of it. However, the pitch angle could be calculated based on the Xsens IMU data only. Since the feet are clamped to the pedals and the hands to the handlebar, the orientation of the bicycle could be reconstructed after some data conversion steps. However, it is probably easier to directly measure the pitch angle in future experiments. Being able to track the pitch angle would be a huge benefit for the simulation validity. Attempts have been made to provide a valid guess for the pitch angle, but this is extremely difficult because the pitch angle should match other experimental data as well. Optimization solution tracking the guess pitch angle as well showed high frequency oscillations to match the pitch angle. More details about this guessed pitch angle can be retrieved in Appendix E

6 Conclusion

6.1 Conclusion

The goal of this work was to create a framework model for the BMX SX gate start which was able to track kinematic data of the human limbs with an accuracy of less than 5% of the range of motion for each joint. This tracking should then lead to the same force and crank torque profile as experimentally measured. The OpenSim model was able to track the kinematic data with an average RMSE error of 0.377°, which comes down to an average relative error of just 0.52% for the six joint angles being tracked. The crank angle could be tracked with an even lower RMSE of 0.34° (relative error 0.079%). Pedal force and torque profiles look very similar, especially for the final pedal stroke. The second pedal stroke shows a larger torque compared to experimentally determined crank torque. On average a relative error of 18% was obtained for reproducing the crank torque of the first three pedal strokes. Peak values for the torque peak differ 4.4%, 7.7%, and 5.1% for the first, second, and third torque peak respectively, with an average absolute difference of only 15 Nm. This work has shown that this approach could be used in future applications to investigate the influence of specific parameters and to study an infinite number of possible "what-if" scenarios.

6.2 Practical application and future work

Besides the recommendations for model improvements made in Section 5.3, future work should focus on using the model to investigate possibilities for improvements to the BMX SX gate start technique.

The ultimate goal would be to use the model to discover training focus points for professional BMX SX riders to help them improve their starting technique. This can be done by solving the trajectory optimization problem for a different objective function such as minimizing the time to reach the kink or maximizing distance traveled after 1 second. Using the tracking solution as a first guess can help the optimizer to find solutions close to their current technique which are likely feasible. The optimization problem could also be solved using a bilateral goal by combining the tracking goal with a performance goal.

Another path to follow is to make small changes to the model and discover their influence on performance time. Obvious examples would be to make changes to the BMX bicycle such as the crank length, gear ratio, frame mass, or frame size. Attempts to study the influence of the gear ratio [5] and crank length [6] have already been presented, but their conclusions are questionable since making such drastic changes require extensive training before any conclusions can be drawn. Using simulations this training is not needed, making it way more powerful than these in-field studies. Another adaptation to the bicycle design could be adding a gear shifting system to the model. Currently, gear shifters are not used because they cannot handle the huge forces applied. The model could prove that gear shifting is beneficial and thus there needs to be more focus on designing an automated gear shifting system that can handle the large forces applied.

A final idea is to investigate which leg should be the lead leg. BMX athletes encounter a lot of injuries during their career, which could result in a strength difference between the left and right leg. Among coaches, there is no agreement on which leg should be the lead leg since both the first and second pedal stroke seem important. A predictive simulation could easily determine the most beneficial starting posture and answer this question.

Bibliography

[1] Lee Rylands and Simon J. Roberts. Relationship between starting and finishing position in World Cup BMX racing. *International Journal of Performance Analysis in Sport*, 14(1):14–23, 2014.

[2] Micah Ambrose D Gross, Florian Schellenberg, Gabriel Lüthi, Matthew Baker, and Silvio Lorenzetti. Performance determinants and leg kinematics in the BMX supercross start. *Journal Of Science & Cycling*, 6(2):3–12, 2017.

[3] Josephine Grigg. *A biomechanical analysis of the BMX SX gate start*. PhD thesis, Bond University, 2019.

[4] H.S.J. van Grieken. *Pedalling performance in the BMX supercross gate start*. PhD thesis, Delft University of Technology, 2019.

[5] Lee P. Rylands, Simon J. Roberts, and Howard T. Hurst. Effect of gear ratio on peak power and time to peak power in BMX cyclists. *European Journal of Sport Science*, 17(2):127–131, 2017.

[6] Philippe Campillo, Timothee Doremus, and Jean-michel Hespel. Pedaling Analysis in BMX By Telemetric Collection of mechanic variables. *Brazilian Journal of Biomotricity*, 1(2):15–27, 2007.

[7] Scott L. Delp, Frank C. Anderson, Allison S. Arnold, Peter Loan, Ayman Habib, Chand T. John, Eran Guendelman, and Darryl G. Thelen. OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950, 2007.

[8] Ajay Seth, Jennifer L. Hicks, Thomas K. Uchida, Ayman Habib, Christopher L. Dembia, James J. Dunne, Carmichael F. Ong, Matthew S. DeMers, Apoorva Rajagopal, Matthew Millard, Samuel R. Hamner, Edith M. Arnold, Jennifer R. Yong, Shrinidhi K. Lakshmikanth, Michael A. Sherman, Joy P. Ku, Scott L. Delp, and 1. OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLOS Computational Biology*, 2018.

[9] Kenneth Hunt and Erskine Crossley. Coefficient of restitution interpreted as damping in vibroimpact. *Journal of Applied Mechanics, American Society of Mechanical Engineers*, 42:440–445, 1975.

[10] Christopher L Dembia, Nicholas A Bianco, Antoine Falisse, Jennifer L Hicks, and Scott L Delp. OpenSim Moco: Musculoskeletal optimal control. *bioRxiv*, page 839381, 2019.

[11] Melle van Dilgt. *A new method to determine mechanical power output during the BMX gate start using inertial sensors*. PhD thesis, Vrije Universiteit Amsterdam, 2019.

[12] J.E. Nash. Expensive dirt: bicycle motorcross and everyday life. *Journal of Popular Culture*, 20(2):97–122, 1986.

[13] Lee P. Rylands and Simon J. Roberts. Performance Characteristics in BMX Racing: A Scoping Review. *Journal Of Science & Cycling*, 8(1):3–10, 2019.

[14] J. Louis, F. Billaut, T. Bernad, F. Vettoretti, C. Hausswirth, and J. Brisswalter. Physiological demands of a simulated BMX competition. *International Journal of Sports Medicine*, 34(6):491–496, 2013.

[15] Manuel Mateo-March, Cristina Blasco-Lafarga, Dominic Doran, Rubén C. Romero-Rodríguez, and Mikel Zabala. Notational analysis of European, World, and Olympic BMX cycling races. *Journal of Sports Science and Medicine*, 11(3):502–509, 2012.

[16] John F. Cowell, Michael R. McGuigan, and John B. Cronin. Movement and Skill Analysis of Supercross Bicycle Motocross. *Journal of Strength and Conditioning Research*, 26(6):1688–1694, 2012.

[17] Niek Kimmann. I moved to switzerland. https://www.youtube.com/watch?v={&}ab_channel=NiekKimmann. Accessed: 16-03-2020.

[18] Union Cycliste Internationale. Bmx sx constitutions and regulation. https://www.uci.org/docs/default-source/rules-and-regulations/part-vi--bmx.pdf?sfvrsn=c7be9239_10. Accessed: 15-01-2020.

[19] Meybo Bikes. Bikes. https://www.meybobikes.com/. Accessed: 15-01-2020.

[20] Mikel Zabala, Cristóbal Sánchez-Muñoz, and Manuel Mateo. Effects of the administration of feedback on performance of the BMX cycling gate start. *Journal of Sports Science and Medicine*, 8(3):393–400, 2009.

[21] W. Bertucci, R. Taiar, and F. Grappe. Differences between sprint tests under laboratory and actual cycling conditions. *Journal of Sports Medicine and Physical Fitness*, 45(3):277–283, 2005.

[22] H.S.J. van Grieken. Literature Survey: Pedal to the metal, 2018.

[23] Josephine Grigg, Eric Haakonssen, Rob Orr, and JW Keogh. Literature Review: Kinematics of the BMX SX Gate Start. *Journal Of Science & Cycling*, 6(1):3–10, 2017.

[24] Miriam Kalichová, Sylva Hebíčková, Romana Labounková, Petr Hedbávný, and Gustav Bago. Biomechanics Analysis of Bicross Start. *World Academy of Science, Engineering and Technology - International Journal of Medical Science and Engineering*, 7(10):614–622, 2013.

[25] Josephine Grigg, Eric Haakonssen, Robin (Rob) Orr, Wade Bootes, and Justin Keogh. Kinematics of the Bmx Sx Gate Start Action. *ISBS Proceedings Archive*, 36(1):10–13, 2018.

[26] Richard Lukes, John Hart, and Steve Haake. An analytical model for track cycling. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 226(2):143–151, 2012.

[27] Frank C. Anderson and Marcus G. Pandy. A dynamic optimization solution for vertical jumping in three dimensions. *Computer Methods in Biomechanics and Biomedical Engineering*, 2(3):201–231, 1999.

[28] Maurice R. Yeadon and Mark A. King. Computer Simulation Modelling in Sport. In *Biomechanical Evaluation of Movement in Sport and Exercise*, pages 176 – 206. The British Association of Sport and Exercise Sciences Guide (2nd ed.). Routledge., 2008.

[29] Mark King. Computer Simulation Modelling in Sports Biomechanics. *Biomechanics in Sports Portuguese Journal of Sport Sciences*, 29(11):19–22, 2011.

[30] Mahdokht Ezati, Borna Ghannadi, and John McPhee. A review of simulation methods for human movement dynamics with emphasis on gait. *Multibody System Dynamics*, 2019.

[31] Thomas Geijtenbeek. Lecture: Inverse and forward simulation of human motion, March 2019. Course: Neuromechanics & motor control, Delft University of Technology.

[32] Jesús Dapena. Simulation of modified human airborne movements. *Journal of Biomechanics*, 14(2):81–89, 1981.

[33] Hirokazu Kiuchi, Motomu Nakashima, Kuangyou B. Cheng, and Mont Hubbard. Modeling fluid forces in the dive start of competitive swimming. *Journal of Biomechanical Science and Engineering*, 5(4):314–328, 2010.

[34] A.V. Hill. The Heat of Shortening and the Dynamic Constants of Muscle. *Proceedings of the Royal Society of London B - Biological Sciences*, 126(843):136–195, 1938.

[35] Bertram Müller, Sebastian I. Wolf, Gert-Peter Brueggemann, Zhigang Deng, Andrew McIntosh, Freeman Miller, and William Scott Selbie. *Hill-Based Muscle Modeling*, pages 1–22. Springer International Publishing, Cham, 2018.

[36] Darryl G. Thelen. Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *Journal of Biomechanical Engineering*, 125(1):70–77, 2003.

[37] Rob Redfield and M. L. Hull. Prediction of pedal forces in bicycling optimization methods. *Journal of Bio*, 19(7):523–540, 1986.

[38] Conor Jansen. Predictive Dynamic Simulation of Seated Start-Up Cycling Using Olympic Cyclist and Bicycle Models. Master's thesis, University of Waterloo, 2018.

[39] Andrea Zignoli, Francesco Biral, Barbara Pellegrini, Azim Jinha, Walter Herzog, and Federico Schena. An optimal control solution to the predictive dynamics of cycling. *Sport Sciences for Health*, 13:381–393, 2017.

[40] Ilaria Pasciuto, Sergio Ausejo, Juan Tomás Celigüeta, Ángel Suescun, and Aitor Cazón. A comparison between optimization-based human motion prediction methods: Data-based, knowledge-based and hybrid approaches. *Structural and Multidisciplinary Optimization*, 49(1):169–183, 2014.

[41] E. Otten. Inverse and forward dynamics: Models of multi-body systems. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1437):1493–1500, 2003.

[42] Toshiyuki Abe, Toshiharu Yokozawa, Junji Takamatsu, and Yasushi Enomoto. Determination of Body Segment Inertia Parameters Using a 3D Human Body Scanner and 3D CAD Software. *Symposium on Biomechanics in Sports*, 2010.

[43] J C Martin, D L Milliken, J E Cobb, K L McFadden, and A R Coggan. Validation of a mathematical model for road cycling power. *Journal of Applied Biomechanics*, 14(3):276–291, 1998.

[44] T. Olds. Modelling human locomotion: Applications to cycling. *Sports Medicine*, 31(7):497–509, 2001.

[45] Lindsey Underwood and Mark Jermy. Mathematical model of track cycling: The individual pursuit. *Procedia Engineering*, 2(2):3217–3222, 2010.

[46] Billy Fitton and Digby Symons. A mathematical model for simulating cycling: applied to track cycling. *Sports Engineering*, 21(4):409–418, 2018.

[47] P. Cangley, L. Passfield, H. Carter, and M. Bailey. A model for performance enhancement in competitive cycling. *Movement and Sports Sciences - Science et Motricite*, 75(1):59–71, 2012.

[48] Matthew L. Kaplan and Jean H. Heegaard. Predictive algorithms for neuromuscular control of human locomotion. *Journal of Biomechanics*, 34(8):1077 – 1083, 2001.

[49] Christine C Raasch, Felix E Zajac, Baoming Ma, and William S. Levine. Muscle coordination of movement of maximum-speed pedaling. *Journal of Biomechanics*, 30(6):595 – 602, 1997.

[50] Benjamin J. Fregly and Felix E. Zajac. A state-space analysis of mechanical energy generation, absorption, and transfer during pedaling. *Journal of Biomechanics*, 29(1):81 – 90, 1996.

[51] Saeed Davoudabadi Farahani, William Bertucci, Michael Skipper Andersen, Mark de Zee, and John Rasmussen. Prediction of crank torque and pedal angle profiles during pedaling movements by biomechanical optimization. *Structural and Multidisciplinary Optimization*, 51(1):251–266, 2015.

[52] Michael Damsgaard, John Rasmussen, Søren Tørholm, Egidijus Surma, and Mark de Zee. Analysis of musculoskeletal systems in the anybody modeling system. *Simulation Modelling Practice and Theory*, 14:1100–1111, 11 2006.

[53] John Rasmussen, Mark de Zee, Michael Damsgaard, Søren Tørholm Christensen, Clemens Marek, and Kar Siebertz. A General Method for Scaling Musculo-Skeletal Models. In *2005 International Symposium on Computer Simulation in Biomechanics*, 2005.

[54] Maarten Frank Bobbert, L. J.Richard Casius, and Arthur J. Van Soest. The relationship between pedal force and crank angular velocity in sprint cycling. *Medicine and Science in Sports and Exercise*, 48(5):869–878, 2016.

[55] Arthur J. van Soest and Maarten F. Bobbert. The contribution of muscle properties in the control of explosive movements. *Biological Cybernetics*, 69(3):195–204, 1993.

[56] Mehdi Kordi, Stuart Goodall, Paul Barratt, Nicola Rowley, Jonathan Leeder, and Glyn Howatson. Relation between Peak Power Output in Sprint Cycling and Maximum Voluntary Isometric Torque Production. *Journal of Electromyography and Kinesiology*, 35:95–99, 2017.

[57] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[58] R. R. Ryan. *ADAMS — Multibody System Analysis Software*, pages 361–402. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.

[59] Davoodi R. and Loeb G.E. Msms software for vr simulations of neural prostheses and patient training and rehabilitation. *Studies in Health Technology and Informatics*, 163:156–62, 2006.

[60] Jeffrey A. Reinbolt, Ajay Seth, and Scott L. Delp. Simulation of human movement: Applications using OpenSim. *Procedia IUTAM*, 2:186–198, 2011.

[61] Karim C. Dudum, Jake E. Deschamps, Juan D. Gutierrez-Franco, Luke I. Kraemer, Alejandro M. Gonzalez-Smith, Eshan M. Dandekar, Scott J. Hazelwood, and Stephen M. Klisch. Using OpenSim to Predict Knee Joint Moments During Cycling. In *Summer Biomechanics, Bioengineering and Biotransport Conference*, 2015.

[62] Samuel R. Hamner, Ajay Seth, and Scott L. Delp. Muscle contributions to propulsion and support during running. *Journal of Biomechanics*, 43(14):2709–2716, 2010.

[63] Ana Carolina Cardoso de Sousa, Felipe Moreira Ramos, Marien Cristina Narvaez Dorado, Lucas Oliveira da Fonseca, and Antônio Padilha Lanari Bó. A Comparative Study on Control Strategies for FES Cycling Using a Detailed Musculoskeletal Model. *IFAC-PapersOnLine*, 49(32):204–209, 2016.

[64] Umberto Emanuele and Jachen Denoth. Power-cadence relationship in endurance cycling. *European Journal of Applied Physiology*, 112(1):365–375, 2012.

[65] Scott L. Delp, J. Peter Loan, Melissa G. Hoy, Felix E. Zajac, Eric L. Topp, and Joseph M. Rosen. An Interactive Graphics-Based Model of the Lower Extremity to Study Orthopaedic Surgical Procedures. *IEEE Transactions on Biomedical Engineering*, 37(8):757–767, 1990.

[66] Nicos Haralabidis, Gil Serrancolí, Steffi Colyer, Ian Bezodis, Aki Salo, and Dario Cazzola. Three-dimensional data-tracking simulations of sprinting using a direct collocation optimal control approach. *PeerJ*, 9:1–33, 2021.

[67] Jason K. Moore, Mont Hubbard, A.L. Schwab, and J.D.G. Kooijman. Accurate Measurement of Bicycle Parameters. In *Proceedings, Bicycle and Motorcycle Dynamics 2010. Symposium on the Dynamics and Control of Single Track Vehicles*, 2010.

[68] Ajay Seth, Michael Sherman, Jeffrey A. Reinbolt, and Scott L. Delp. OpenSim: A musculoskeletal modeling and simulation framework for in silico investigations and exchange. *Procedia IUTAM*, 2:212–232, 2011.

[69] Antonio Pérez-González, Carlos Fenollosa-Esteve, Joaquín L. Sancho-Bru, Francisco T. Sánchez-Marín, Margarita Vergara, and Pablo J. Rodríguez-Cervantes. A modified elastic foundation contact model for application in 3D models of the prosthetic knee. *Medical Engineering and Physics*, 30(3):387–398, 2008.

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

[71] Matthew Millard, Thomas Uchida, Ajay Seth, and Scott L Delp. Flexing Computational Muscle : Modeling and Simulation of Musculotendon Dynamics. *Journal of Biomechanical Engineering*, 135(February):1–11, 2013.

[72] Thomas Geijtenbeek. SCONE: Open Source Software for Predictive Simulation of Biological Motion. *Journal of Open Source Software*, 4(38):1421, 2019.

[73] G. Serrancolí, A. Falisse, C. Dembia, J. Vantilt, K. Tanghe, D. Lefeber, I. Jonkers, J. De Schutter, and F. De Groote. Subject-exoskeleton contact model calibration leads to accurate interaction force predictions. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(8):1597–1605, 2019.

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

[75] Philip E. Gill, Walter Murray, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[76] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[77] Yi Chung Lin and Marcus G. Pandy. Three-dimensional data-tracking dynamic optimization simulations of human locomotion generated by direct collocation. *Journal of Biomechanics*, 59:1–8, 2017.

[78] Bruno Watier, Antony Costes, and Pierre Moretto. An inverse dynamic study suggests that cyclists marginally use hip joint torque at maximal power. In *XXIV Congress of the International Society of Biomechanics*, 2013.

[79] Ina Janssen and Jesper Cornelissen. Pedal Forces During the BMX and Track Sprint Cycling Start. In *35th Conference of the International Society of Biomechanics in Sports*, pages 793–796, 2017.

[80] Amin Daneshfar, Carl Petersen, Daniel Gahreman, and Beat Knechtle. Power Analysis of Field-Based Bicycle Motor Cross (BMX). *Open Access Journal of Sports Medicine*, 11:113–121, 2020.

[81] William M. Bertucci and Christophe Hourde. Laboratory testing and field performance in BMX riders. *Journal of Sports Science and Medicine*, 10(2):417–419, 2011.

[82] D. Princelle, T. Monnet, E. Brunet, J. Sastre, and M. Domalain. Dynamic analysis of the BMX start: interactions between riders and their bike. *Computer Methods in Biomechanics and Biomedical Engineering*, 22(sup1):S311–S313, 2019.

[83] Blake M. Ashby and Scott L. Delp. Optimal control simulations reveal mechanisms by which arm movement improves standing long jump performance. *Journal of Biomechanics*, 39(9):1726–1734, 2006.

[84] Dennis E. Anderson, Michael L. Madigan, and Maury A. Nussbaum. Maximum voluntary joint torque as a function of joint angle and angular velocity: Model development and application to the lower limb. *Journal of Biomechanics*, 40(14):3105–3113, 2007.

[85] Conor Jansen and John McPhee. Predictive Dynamic Simulation of Seated Start-Up Cycling Using Olympic Cyclist and Bicycle Models. In *12th Conference of International Sports Engineering Association*, Brisbane, Queensland, Australia, 2018.

A UCI BMX Regulations

A.1 BMX track

The track must be of a compact, closed looped design, forming a circuit where length measured along its centre line is not less than 300 metres nor greater than 400 metres. The track must be a minimum of 10 metres wide at its start and may not taper to a width of less than 5 metres at any point along its length.

A.2 Starting hill

The starting hill must accommodate a track width of at least 10 metres and be at an elevation at least 1.5 metres, preferably 2.5 metres above the grade of the first straight. The initial incline extending from the starting gate to level grade must be at least 12 metres in length. It is recommended that starting hills used for the challenge categories during international BMX events should not exceed 6m in height (as measured between the flat area on which the starting gate is mounted to the bottom of the starting hill ramp). Also, it is recommended that the ramp should not exceed an incline of 20 degrees. Comment: It is strongly recommended that tracks intended for beginner and intermediate riders should have starting hills less tall and less steep than the maximum limits described above. (text modified on 01.01.19).

A.3 Starting gate

The starting gate shall be a minimum of 7,3 metres in width for BMX events on the UCI BMX calendar. The gate shall have a height of at least 50 cm, with an angle no greater than 90 degrees with the slope of the ramp which supports the bicycles' wheels when they are in their starting position. Starting positions 1 through 8 must be clearly marked on the gate. Position 1 should be on the side of the gate closest to the inside of the first turn. The electronically controlled gate, to be used at all BMX events on the UCI BMX calendar, must be outfitted with a system of appropriately coloured starting lights located so as to be clearly visible from all starting lanes without disadvantage to any rider who is in the "riders ready" position. In case of a failure of the gate release system, the gate shall fall to the dropped position. A voice box system is mandatory at all UCI sanctioned events described in annex 3. Whenever a timing scoring system is utilised, the timing system must be activated, whereupon the time starts running, at the moment the gate-start mechanism is activated causing the gate to drop.

A.4 Initial straight

The initial straight shall be a minimum of 40 metres in length. It is recommended that the bottom of the front side of the first obstacle in the initial straight shall be located not less than 35 metres from the starting gate nor less than 20 metres from the point of curvature of the first turn. However, on tracks especially designed for highly skilled riders, the distance between the starting gate and the front side of the first obstacle may be shorter.

A.5 First turn

The first turn may go in either direction and shall be banked to a degree which allows safe entry and exit for riders of all ages at race speeds. At the first turn, the track shall be a minimum of 6 metres wide measured along a straight line extending from its surface at the inner radius to the top of the berm at its outer radius.

A.6 Turns and obstacles

The track shall have a minimum of 3 turns. The track shall be a minimum of 5 metres wide throughout each turn. All obstacles on the track must be constructed with the safety of all riders, regardless of age, in mind. Consideration must be given to the abilities of the youngest riders in competition when designing obstacles intended to present special challenges to older competitors. On the first straight the minimum distance between two obstacles shall be 10 metres. An obstacle is defined by its front and back slope and can be a single obstacle, double, triple or multi-jump as well as a 4-pack, 5-pack or multi-pack. Tracks may be designed to include alternate sections to be traversed only by Championship categories. These sections may offer obstacles which are inherently more challenging than those found on the track's main circuit.

A.7 Random Start Gate Timing Cadence

Starting cadence The UCI starting cadence can be used either automatically with built-in delays, or manually, requiring the operator to press the start button for the second half of the cadence. In general the cadence consists of the phrase Ok riders, Random start. Riders ready? Watch the gate. This is followed by 4 tones that coincide with the display of a light tree and the gate begins to fall on the last tone and light. After the word gate there is a time delay between .1 seconds and 2.7 seconds for the lights, the tones generated by the controls and the gate cycle. This time delay must be totally random, produced by the controls and not predictable by the riders or the starter. Further, the starter should have no control or input as to time interval. Additional items that are an integral part of the cadence are warning tones advising the rider that the gate is about to be raised by the operator and warning tones that advise the riders to stand down if the cadence is interrupted by the operator. In specific the cadence consists of the following;

(a) "OK RIDERS RANDOM START" as spoken words within 1.5 seconds. In automatic mode, there is a 1.8 second pause, before the second set of words.

(b) "RIDERS READY - WATCH THE GATE" are spoken within 2.0 seconds.

(c) A delay randomly between 0.1 seconds and 2.7 seconds will occur after the second set of words concludes before the LED lights and pulse tones are activated. Note that the random delay and all pulse tones are generated by the controller chip, and therefore they are not included in the mp3 files.

(d) Three pulses of a 632 Hertz tone are played, followed by the fourth long tone of 2.25 seconds. The short tone pulses are 60 milliseconds long with 60 milliseconds of silence between them. The four LED lights (red, yellow, yellow, and green) are synchronized exactly with the start of each tone burst.

    (i) The red light illuminates with the first pulse.

    (ii) The first yellow light is added with the second tone pulse.

    (iii) The second yellow light is added with the third tone pulse.

    (iv) The green light is added with the forth, long tone pulse.

(e) When the green light comes on, the gate start drop signal is activated. All lights remain illuminated for the duration of the final tone burst, then all lights extinguish.

(f) At the conclusion of the tone sequence, an LED on the control box flashes to alert the operator to press the stop button to raise the gate for the next start.

(g) Upon pressing the stop button, five pulses of 1150 Hz, each 0.25 second long with a 0.25 second period between the pulses will sound before the up solenoid is triggered to raise the gate.

(h) For safety, the stop button can be pressed at any time (up to the end of the second set of words) after the start button was pressed, to abort the sequence. A Stand Down tone as follows may be played:

- A tone of 740 Hz for 0.22 seconds followed immediately by 680 Hz for .44 seconds will sound when the gate is aborted.

Alternatively, it is acceptable that no tone is played when the stop button is pressed.

Sequence	**Action**	**Timing**
1	OK RIDERS RANDOM START	1.50 sec
2	Pause (automatic mode)	1.80 sec
3	RIDERS READY WATCH THE GATE	2.00 sec
4	Random Delay	0.1 to 2.70 sec
5	1 tone (632 Hertz) Red light illuminates	0.060 sec
6	Pause	0.060 sec
7	1 tone (632 Hertz) Yellow light illuminates	0.060 sec
8	Pause	0.060 sec
9	1 tone (632 Hertz) Yellow light illuminates	0.060 sec
10	Pause	0.060 sec
11	1 tone (632 Hertz) Green light illuminates	2.25 sec

Table A.1: Starting sequence for BMX SX races.

A.8 Gate drop

The gate shall drop at an average speed of approximately 0.310 seconds from upright position to down position (90° angle). A variable of $\pm 7\%$ is allowed, giving the gate the maximum variation from 0.289 to 0.331 seconds.

B | Recoil distance

To have an idea about the theoretical possibilities for the recoil during the gate start, it is often a good idea to do a simple calculation first. What is theoretically the maximum recoil distance possible? To answer this question let's assume that the net force applied via the cranks and the pedals to the bicycle-rider system in the horizontal direction is constant and can jump between negative to positive. Furthermore, let's assume that the time to do this recoil movement is limited by the time between the first red light and the final green light equalling 0.36 seconds, as seen in Figure 3.12. In this period of time the bicycle must start at x=0, travel backward and return at x=0 after t=0.36 seconds. Because the force is said to be constant, the acceleration will also be constant. The distance travelled backward is thus equated using $s = v_0 t + \frac{1}{2}at^2$. This results in Equation B.1.

$$s = s_1 + s_2 + s_3 = 0 \tag{B.1}$$

$$s = \left(\frac{1}{2}(-a)\tau^2\right) + \left(-a\tau^2 + \frac{1}{2}(+a)\tau^2\right) + \left(\frac{1}{2}(+a)t_3^2\right) = -a\tau^2 + \frac{1}{2}at_3^2 = 0 \tag{B.2}$$

From Figure B.1 it can be seen that $t_1 = t_2$, which will be called τ from now on. Now Equation B.1 can be transformed into Equation B.2. If this equation is solved, it is found that $t_3 = \sqrt{2}\tau$. Since it is known that the total time must equal 0.36 seconds, τ can be calculated as $(2+\sqrt{2})\tau = 0.36$. This shows that τ is equal to 0.1054 seconds. Now it is fairly easy to compute the theoretical maximum recoil distance as $s_{max} = a\tau^2 = \frac{F}{m}\tau^2$. Depending on the maximum force that can be applied, the recoil distance can now be calculated. If a guess is made that the maximum acceleration is close to the gravitational acceleration g and the mass of the rider bicycle system is approximately 80 kg, the force must be close to 800 N. Figure B.1 shows for three different maximum forces (600, 800 and 1000 N) the theoretical maximum recoil distance, which show to be 0.083, 0.111, and 0.139 m respectively. These values are close to experimentally determined recoil distances by Grigg [3] which were measured to be 116.7 ± 48.6 mm.

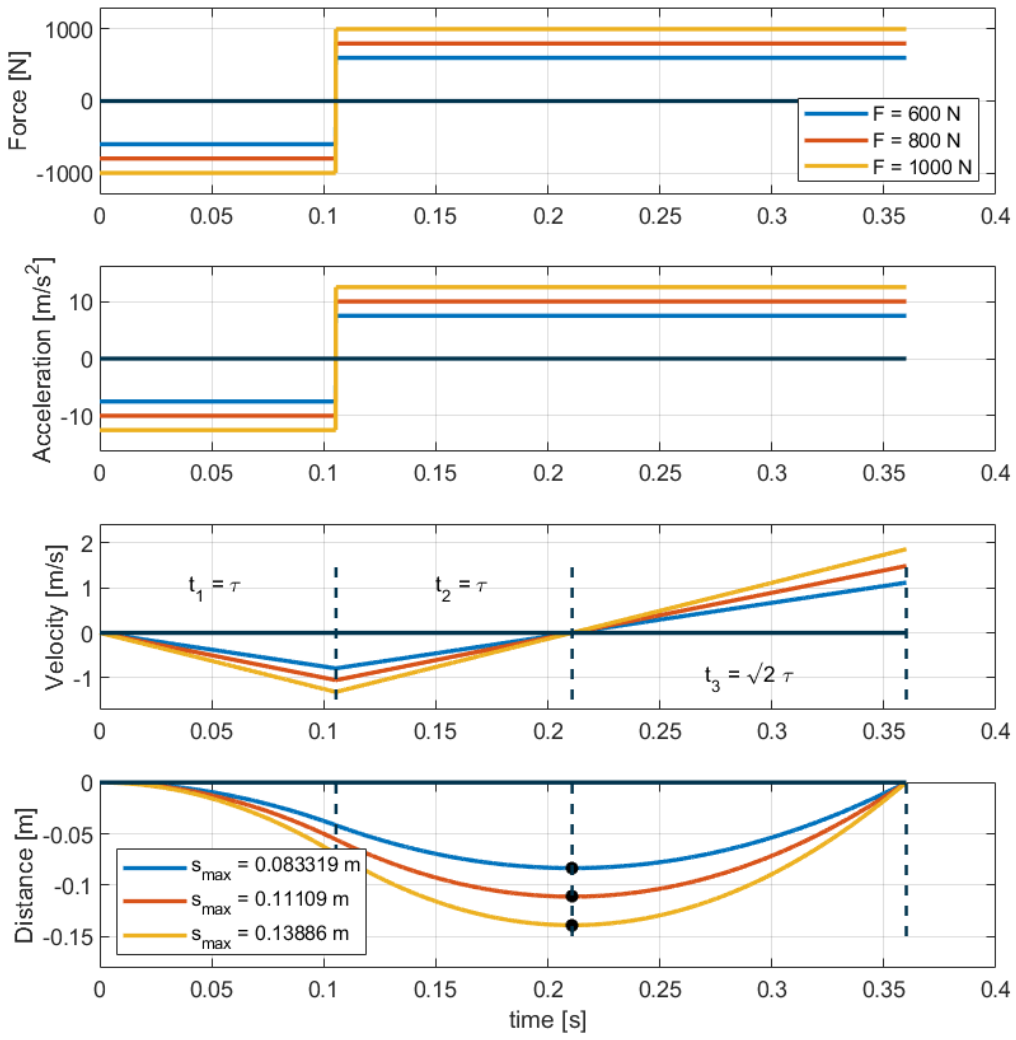


Figure B.1: Back-of-the-Envelope calculation for the theoretical maximum recoil distance. Shown are the constant horizontal force exerted on the bicycle-rider system, the constant acceleration, the linear velocity profile and the quadratic distance curve. The maximum recoil distances depending on the maximum force are 0.083, 0.111, and 0.139 m respectively.

C | Data conversion

To get from the Xsens output to the joint angles of the OpenSim model, the data must be converted. The Xsens data is stored as a .mnvx file which has a data file for each point in time containing all the information. Transforming this file into a file containing the joint angles of the OpenSim model in time, which is usually a .sto file, is done by taking the following steps.

1. The first step is to load the .mvnx file into MATLAB using the by Xsens provided script called load_mvnx.m. This piece of MATLAB code opens the .mvnx file and transforms it into a struct with several fields such as orientation and acceleration. In total there were 23 IMU sensors in the Xsens suit, which are all logged into the fields of the struct. Since each sensor collects data in three dimensions the total length of the fields is $23 \cdot 3 = 69$.

2. Secondly, the data of the IMU sensor attached to the hub of the rear wheel is added as the 24th segment. The angular velocity of this sensor is used to calculate the bicycle's speed and distance travelled.

3. Next, the data is manipulated to cut the dataset at the start and finish of the trial. These start and stop criteria were defined by Van Dilgt [11]. The start condition was defined as the moment when the pelvis showed an acceleration larger than 2 m/s^2 for three consecutive frames. The stop condition was defined as the moment the bicycle reached a distance of 5 meters.

4. The IMU will be projected on the OpenSim model using OpenSense. This requires the data to be exported as a .txt file with the columns being the entries of the rotation matrix in time containing the orientation of the body segment and the three components of linear acceleration. This conversion is done using a custom made MATLAB code called mvnx2txt.m. It simply takes the orientations and accelerations from the struct of step 1 and puts in the right format. The orientation from the struct is written as a quaternion, so the MATLAB built-in command quat2rotm was used to get the rotation matrices. The converted data was then exported using a combination of fprintf, writecell and dlmwrite commands.

5. The quaternions from the struct can also be used to compute the Euler angles for each body segment. This is needed to be able to define the initial posture that OpenSense needs to calibrate the model. Using the MATLAB built-in commands quat2eul or rotm2eul, the Euler angles can be easily computed.

6. The Euler angles of the body segments are not the same as the joint angles from the OpenSim model. To transform the Euler angles into the joint angles a custom made MATLAB script was used called eul2ja, where ja stands for joint angles. It takes the Euler angles of two connected bodies and uses the cross product and basic geometry to calculate the joint angle in the x-y plane, corresponding to the joint angles in the OpenSim model. The initial value for these joint angles are used in the construction of the OpenSim model as the initial joint angles. However, since the anthropometrics of the OpenSim model are not scaled to match the BMX athlete, there is an off-set for these initial angles in order to obey the constraints in the model.

OpenSim searches for way that matches the initial angles provided as close as possible.

7. The .txt files created are loaded into a .xml code provided within the OpenSim download. It is basically the setup file for the OpenSense tracking. It contains the names of the .txt files where the IMU data is stored and the names of the corresponding body segments in the model. This file is loaded using the IMUDataConversion.m MATLAB code written by James Dunne. This file is part of the OpenSim download. The script loads the .xml setup file and transforms the orientation and linear acceleration data from the .txt data into a .sto file, which is the common motion file in OpenSim.

8. The next step is to calibrate the model and the data using the OpenSense_CalibrateModel.m script also written by James Dunne and available within the OpenSim download. The script calibrates the data with the model based on the rotation of the IMU data with respect to the OpenSim world frame and some other settings. The output of the script is a calibrated version of the OpenSim model.

9. The final step is to combine the .sto IMU data from step 7 with the calibrated model from step 8. This can be done using the OpenSense_OrientationTracking.m script once more written by James Dunne. This is essentially where the magic happens. The script uses the OpenSense IMU inverse kinematics tool to convert the IMU data into the model's joint angles. For it to function, it must know the start and end time of the trial as well. To output of the script is a .sto file with the joint angles in time.

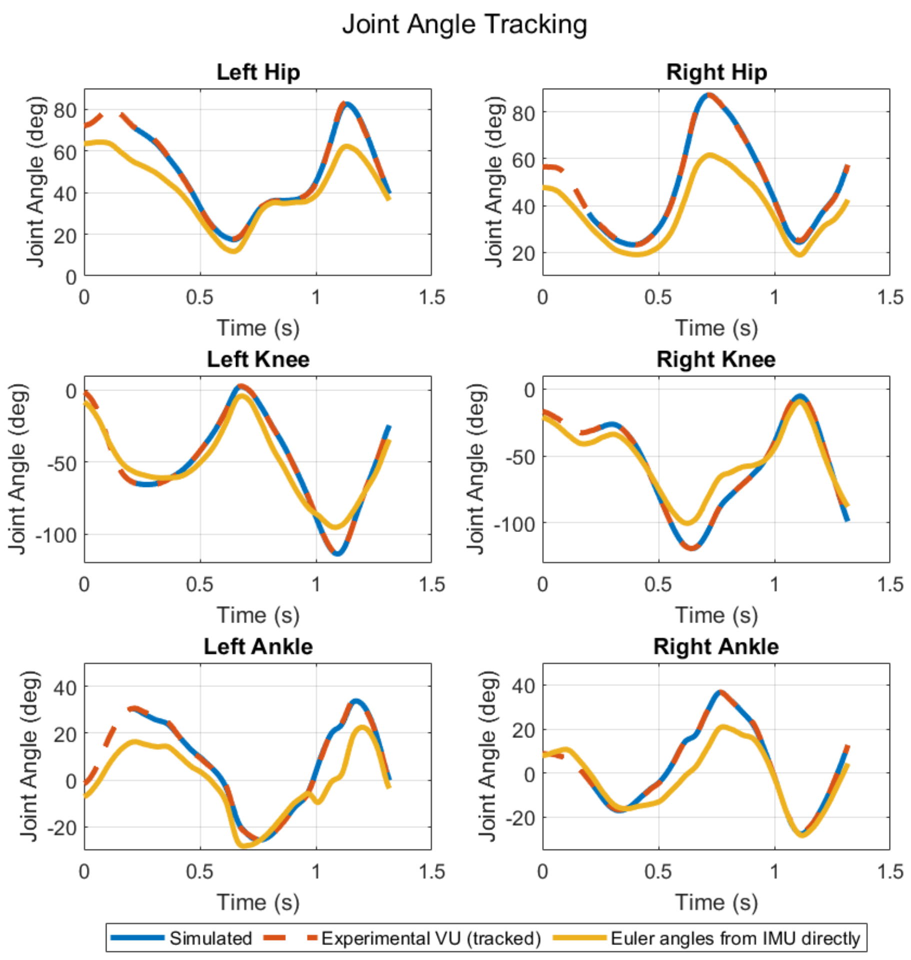


Figure C.1: The difference between the joint angles computed using OpenSense and the Euler angles directly determined from the IMU data. As seen especially for the hips these results differ quite a bit.

As was explained in the steps above, the IMU data taken from the Xsens suit is processed using OpenSense, which projects the joint angles onto the generic model. For sure, the BMX athlete and the generic model do have different anthropometrics. This leads to a difference between the original joint angles during the in-field experiments and the joint angles which come out of the OpenSense configuration and which are used for the tracking problem. Figure C.1 shows the difference between the OpenSense joint angles in orange which are being tracked and the joint angles calculated from the Euler angles directly in yellow. As seen there is quite a deviation between those two approaches, especially for the hip joints. Besides the mismatch in anthropometrics, the initial posture could also be of huge importance. As explained in step 6, the initial position of the OpenSim model used to calibrate the model was based on the Euler angles from the orientation matrices. These initial angles could not be matched by the model as seen from the initial joint angles in Figure C.1. Again, especially for the hips this deviation is substantial. Scaling the model could solve this problem. Another issue that could explain the deviation between the orange and yellow curves, is the lack of a degree of freedom between the abdomen and the torso. The lumbar extension, as this degree of freedom is often called, was removed from the generic OpenSim model for simplicity reasons. Reintroducing this lumbar extension with corresponding torque actuators could improve the similarity between the joint angles.

D | Body meshes

The realistically looking human as seen for instance in the generic model displayed in Figure 3.16 are the result of recreating the human bones in some sort of 3D sketching software. If OpenSim users would like to create such detailed body appearance for their own model, it is possible to import a so-called mesh file into the model. This appearance does not have any influence on the model outcome, since that is only determined by the mass properties and inertia. However, the model would be more understandable if the bike would actually look like a bicycle instead of just a cube. To construct the mesh object needed for the visualization of the bicycle frame, the BMX SX bicycle as seen in Figure 3.17 was reconstructed using Fusion 360 (Autodesk, San Rafael, CA, USA). Because the steering assembly and the frame were modeled as one part, this could also be constructed as one in the Fusion model. Figure D.1 shows the complete model of the bicycle frame. The most important parts of this design are the dimensions and the locations of the attachment point, i.e. the location of the rear and front wheel and the location of the handlebars.

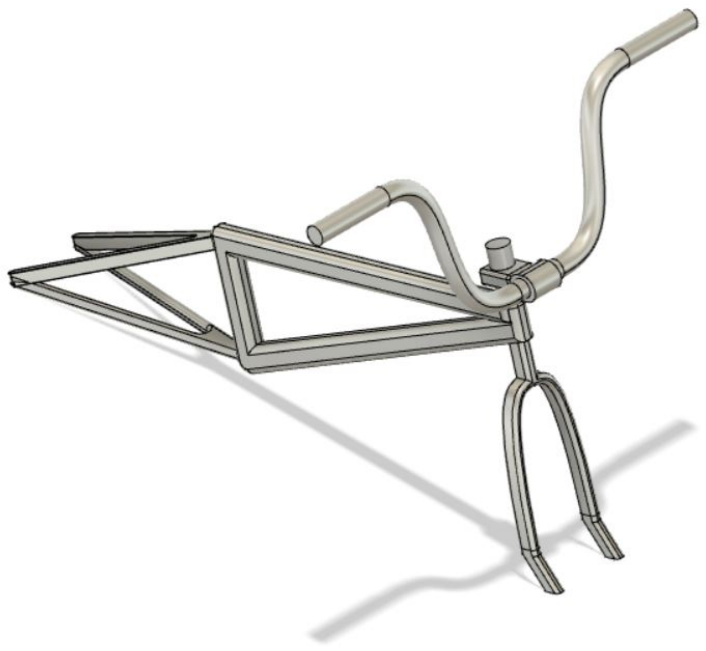


Figure D.1: CAD model of the BMX bicycle frame created using Autodesk Fusion 360.

The model design in Fusion can be exported as a mesh-framed body which can then be used in the OpenSim model. Figure D.2 shows the mesh grid applied to the frame model. If a close look is taken at the front fork, one can discover tiny little triangles which form the mesh grid.

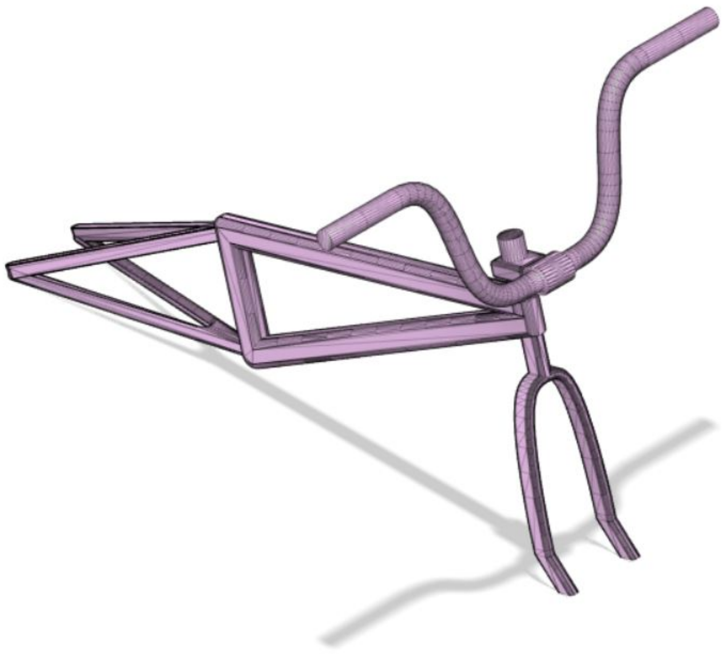


Figure D.2: Mesh grid of the bicycle frame CAD model which can be loaded into OpenSim.

This computer aided design (CAD) model can also be used to derive the inertia and center of mass location of the design as was used by [38]. However, in order to do so, the materials used and the wall thicknesses of all the tubes must be known exactly. This makes this approach error prone, which is why it was in this work chosen to experimentally determined these properties. To compare both approaches, the center of mass location using both techniques are shown in Table D.1. As seen there is a difference of 9% for the x-position and 13.7% for the y-position. For the moment of inertia this method becomes even more error prone because this is also influenced by the material of the bicycle.

	Experiments	Fusion
x	61.96 cm	67.583 cm
y	25.5 cm	28.993 cm
z	0 cm	0 cm

Table D.1: The locations of the center of mass determined using two different techniques.

The same approach as for the bicycle frame was used, was also applied to the wheels and the crankshaft. The wheels could also been constructed in OpenSim using disks or spheres, but then the rotation of the wheels could not be observed during simulations. Therefore, it was chosen to construct wheels in Fusion 360 including a few spokes, as seen in Figure D.3, to visualize the rotation when playing the results of a simulation. For the front and rear wheel, the same CAD model was used.

The construction of the crankshaft was mainly focused on getting the correct locations of the pedals. For simplicity the pedals itself were omitted from the sketch even as the chain ring. As shown in Figure D.4, the crankshaft and cranks were made out of one single part to keep the number of bodies in the model as low as possible.

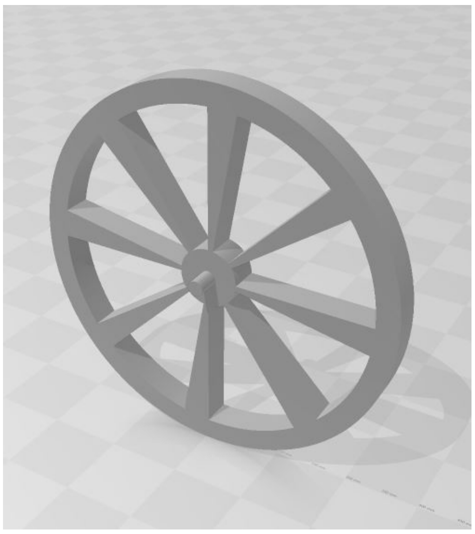


Figure D.3: CAD model of the wheels with eight simplified spokes to visualize the rotation when simulating.

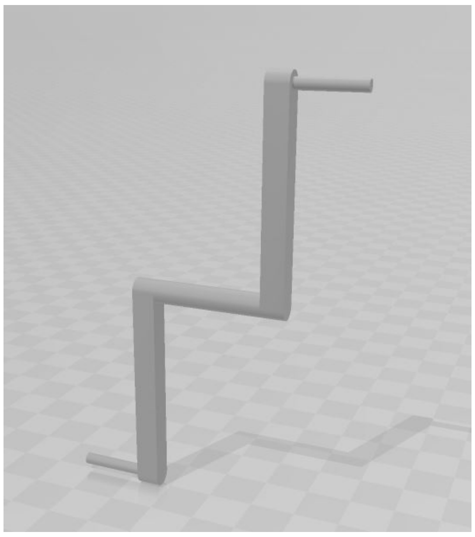


Figure D.4: CAD model of the crankshaft. The crankshaft and cranks were made out of one part, for simplicity reasons.

E | Guessed Pitch Angle

A potential way to overcome the lack of pitch angle data stated in Chapter 5, is to make a guess about the pitch angle (ϕ) based on video footage. From these videos it can be seen that the total wheelie lasts for approximately 0.6 seconds. This can be split into two parts: the part where the bicycle is moving backward (the recoil) and the part where the bicycle is moving forward. If it is assumed that the largest pitch angle is reached at the far most backward position, the time to reach the maximum pitch angle is the same as the recoil time. From video it was seen that this is approximately 0.18 seconds. Finally, it was assumed that the transition between $\phi = 0$, $\phi = \max$, $\phi = 0$, was quadratic and the maximum pitch angle was 15°. The guessed pitch angle in time is then visualized in Figure E.1.

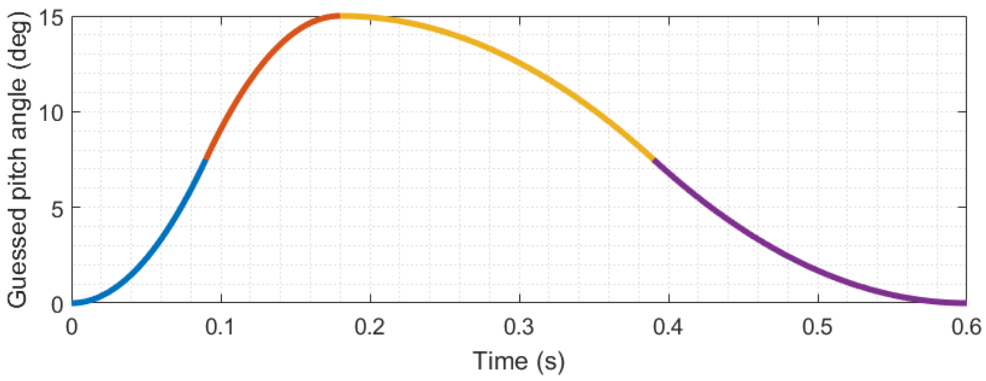


Figure E.1: A guessed pitch angle. It was assumed to be quadratic in four different parts, indicated by the four colors. The timing was guessed based on videos of the start.

If this guessed pitch angle is added to the states to be tracked, the simulation pitch angle will look like the yellow line displayed in Figure E.2. As seen it tries to match the guessed pitch but does not do a good job. Since the pitch angle is guessed, it does not match other provided kinematics. The initial guess for this new optimization with pitch tracking was the previous solution without pitch tracking. This explains why the solution starts at approximately the same pitch angle as the blue curve, which was the previous solution. The optimizer has a hard time bringing the pitch angle back to zero while also matching the other kinematics. As a matter of fact, the tracking similarities between these other states such as hip, knee, and ankle angles showed approximately the same RMSEs. The optimizer thus can track the joint angles while also tracking the pitch angle.

When looking at the crank torque plot of the pitch angle tracking solution in Figure E.3, it is seen that there are high frequent oscillations in the actuation at the beginning. These are the results of the incorrect tracking of the pitch angle. The optimizer tries to minimize this error by introducing these oscillations in the actuation. Additionally, the torque becomes negative after the second torque peak. As seen from Figure E.2, this is the moment when the pitch angle drops to zero. To maintain

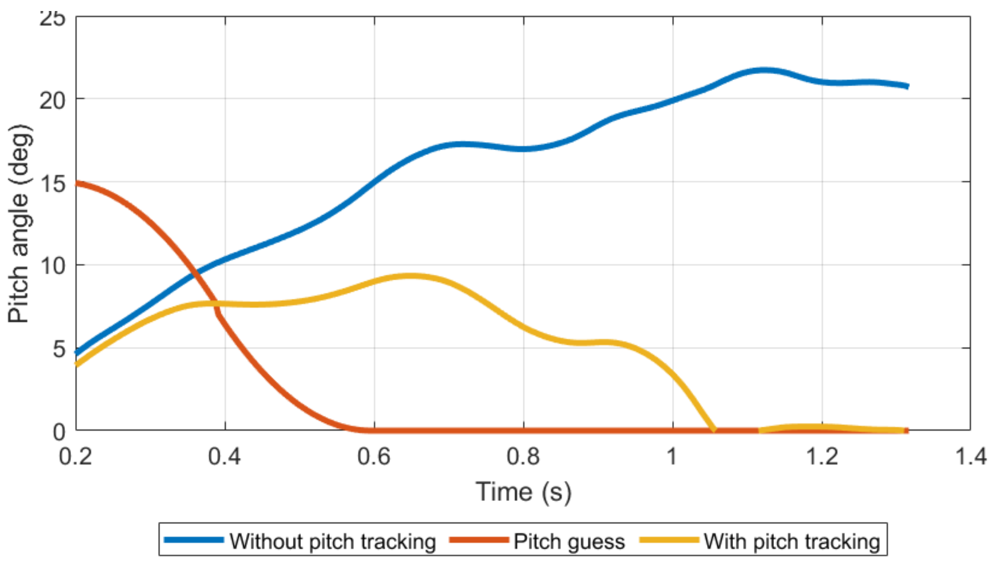


Figure E.2: A comparison between the pitch angles while being tracked and not being tracked. The blue line is the solution without the tracking of the pitch angle, the yellow is the pitch angle when the optimizer tried to track the guessed pitch angle, displayed in orange.

this pitch angle of 0° the produced torque oscillated around 0 Nm. This could be explained by the poor pitch angle guess and the initial guess used. When a dataset could be collected with the pitch angle included, these combined results will probably improve.

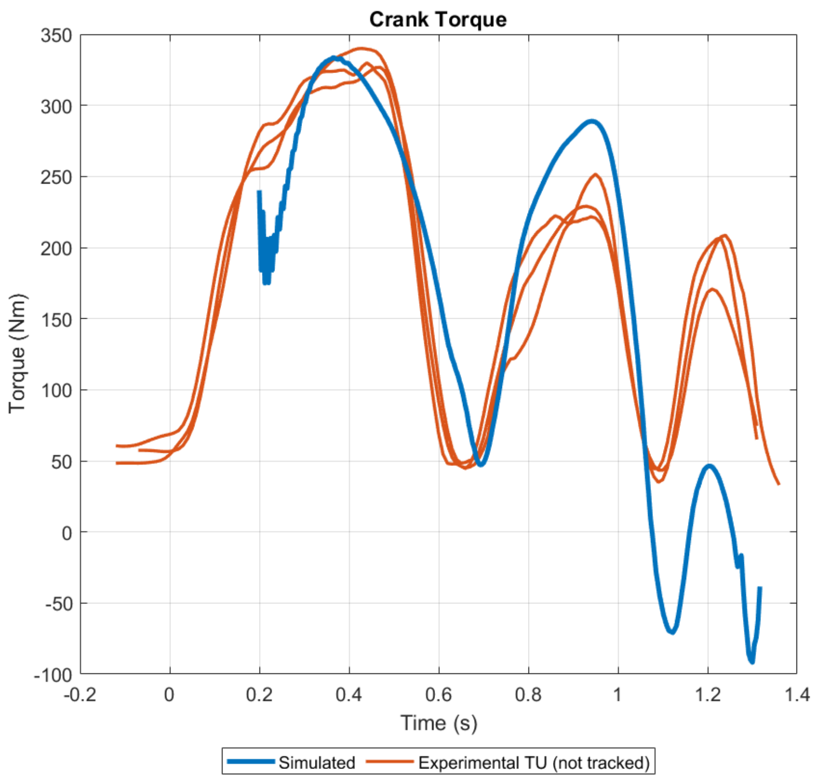


Figure E.3: The crank torque applied when the optimizer tried to track the pitch angle guessed. There are oscillations introduced at the beginning of the trial and the third torque peak has been shifted with approximately 150 Nm.

F Handlebar force

An extra, but interesting, kinetic factor to compare with literature is the force exerted on the handlebars. This force was not experimentally determined by Van Grieken [4], but there exists one study in literature that experimentally determined these forces during the BMX gate start. Princelle et al. [82] estimated the handlebar forces "*based on the fundamental principle of the dynamics using the forces measured on each pedal and the riders centre of mass acceleration*". The position of the center of mass was tracked based on a scaled generic human model and 3D motion tracking using 19 cameras. This resulted in an estimated handlebar force of approximately 2000 N (21.3 N/kg).

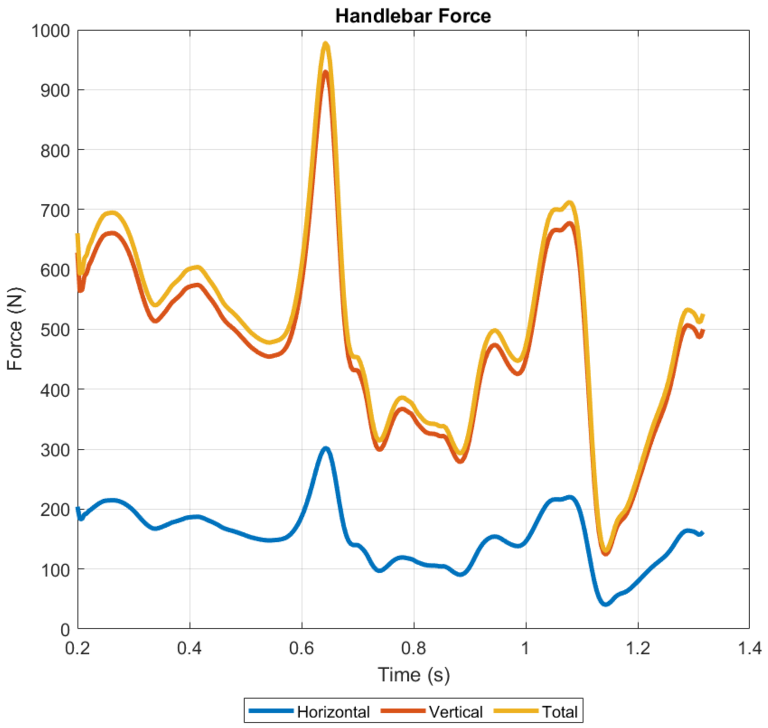


Figure F.1: The forces exerted on the handlebar. In blue the horizontal force and in orange the vertical handlebar force. The combined total handlebar force in shown in yellow.

The handlebar forces from the simulation are shown in Figure F.1. The blue and red curve show the horizontal and vertical handlebar force respectively. As seen the vertical handlebar force is almost three times larger than the horizontal force. Note that the positive directions are forward, so in the direction of travelling, and up. As seen both forces are positive meaning the horizontal force is directed forward and the vertical force is pointing up. The forces are expressed in the ground reference frame which is neither connected to the slope of the hill nor the bicycle. As seen there is a large force peak just after 0.6 seconds for both the vertical and horizontal force. This could be due to the initiation of the second pedal stroke. As seen from figure 4.11 the second pedal stroke, which is performed with the right leg mainly, starts just after 0.6 seconds. Just before the second pedal stroke, the hips are moved backwards to create space to effectively perform the second pedal stroke and to overcome the top dead center of the right leg. During the second pedal stroke the hips are moved towards the handlebar again. The huge force peak in the vertical direction could explain why the front wheel can be kept in the air. As seen from Figure 4.13, at 0.6 seconds there is barely an effective crank torque. The cranks are at their dead centers, making it hard to propel the bicycle. It seems like the model compensates this by an immense force on the handlebar. When comparing the simulated handlebar forces with the experimental results of Princelle et al. [82] it is concluded that simulated peak force is twice as low as the experimentally estimated handlebar force. However, since the result of Princelle et al. was not directly measured and the results have not been validated, it is hard to draw conclusions about the accuracy of these numbers.

G Moco Setup

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<MocoStudy name="
    Predict_BMX_Gate_r18_afterslip_initialspeeds_cadence2_02">
      <!--The optimal control problem to solve.-->
      <MocoProblem name="problem">
         <!--List of 1 or more MocoPhases.-->
         <MocoPhase name="phases">
            <!--OpenSim Model to provide dynamics.-->
            <ModelProcessor name="model">
               <!--Base model to process.-->
               <model>
                  <Model name="BMX_gate_start_tracking_model">

                     <!--Here is were usually the OpenSim Model is
                         written.-->

                  </Model>
               </model>
            </ModelProcessor>
            <!--Bounds on initial value.-->
            <MocoInitialBounds name="time_initial_bounds">
               <!--1 value: required value. 2 values: lower, upper
                   bounds on value.-->
               <bounds>0.2</bounds>
            </MocoInitialBounds>
            <!--Bounds on final value.-->
            <MocoFinalBounds name="time_final_bounds">
               <!--1 value: required value. 2 values: lower, upper
                   bounds on value.-->
               <bounds>1.3167</bounds>
            </MocoFinalBounds>
            <!--The state variables' bounds.-->
            <state_infos>
               <MocoVariableInfo name="/jointset/GateToPlatform/
                   GateAngle/value">
                  <!--1 value: required value over all time. 2 values
                      : lower, upper bounds on value over all time.-->
                  <bounds>-2 2</bounds>
                  <!--1 value: required initial value. 2 values:
                      lower, upper bounds on initial value.-->
                  <initial_bounds></initial_bounds>
```

```
                    <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                    <final_bounds></final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/GateToPlatform/GateAngle/speed">
                    <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                    <bounds>-100 100</bounds>
                    <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                    <initial_bounds></initial_bounds>
                    <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                    <final_bounds></final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/FrameToPlatform/Frame_tx/value">
                    <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                    <bounds>-2 6</bounds>
                    <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                    <initial_bounds>-0.104911752</initial_bounds>
                    <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                    <final_bounds>4.5 6</final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/FrameToPlatform2/Frame_rz/value">
                    <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                    <bounds>-0.1 1</bounds>
                    <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                    <initial_bounds>0.080582739</initial_bounds>
                    <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                    <final_bounds></final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/RearWheelToFrame/RearWheel/value">
                    <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                    <bounds>-50 50</bounds>
                    <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                    <initial_bounds>3.567478</initial_bounds>
                    <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                    <final_bounds>24.20637</final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/FrontWheelToFrame/FrontWheel/value">
                    <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds>-50 50</bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>1.293695824</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds></final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/BracketToFrame/Bracket/value">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds>-50 50</bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>1.297265</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds>8.802318</final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/LowerArmToHandlebar/Hands/value">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>2.864908</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds>2.754822</final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/LowerArmToUpperArm/Elbow/value">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>0.277411</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds>0.941568</final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/UpperArmToTorso/Shoulder/value">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>2.390298</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds>3.000099</final_bounds>
            </MocoVariableInfo>
```

```
<MocoVariableInfo name="/jointset/hip_r/hip_flexion_r/
    value">
    <!--1 value: required value over all time. 2 values
        : lower, upper bounds on value over all time.-->
    <bounds></bounds>
    <!--1 value: required initial value. 2 values:
        lower, upper bounds on initial value.-->
    <initial_bounds>0.643011</initial_bounds>
    <!--1 value: required final value. 2 values: lower,
         upper bounds on final value.-->
    <final_bounds>1.001474</final_bounds>
</MocoVariableInfo>
<MocoVariableInfo name="/jointset/hip_l/hip_flexion_l/
    value">
    <!--1 value: required value over all time. 2 values
        : lower, upper bounds on value over all time.-->
    <bounds></bounds>
    <!--1 value: required initial value. 2 values:
        lower, upper bounds on initial value.-->
    <initial_bounds>1.275173</initial_bounds>
    <!--1 value: required final value. 2 values: lower,
         upper bounds on final value.-->
    <final_bounds>0.686263</final_bounds>
</MocoVariableInfo>
<MocoVariableInfo name="/jointset/knee_r/knee_angle_r/
    value">
    <!--1 value: required value over all time. 2 values
        : lower, upper bounds on value over all time.-->
    <bounds>-2.4435 0.2618</bounds>
    <!--1 value: required initial value. 2 values:
        lower, upper bounds on initial value.-->
    <initial_bounds>-0.55255</initial_bounds>
    <!--1 value: required final value. 2 values: lower,
         upper bounds on final value.-->
    <final_bounds>-1.72511</final_bounds>
</MocoVariableInfo>
<MocoVariableInfo name="/jointset/knee_l/knee_angle_l/
    value">
    <!--1 value: required value over all time. 2 values
        : lower, upper bounds on value over all time.-->
    <bounds>-2.4435 0.2618</bounds>
    <!--1 value: required initial value. 2 values:
        lower, upper bounds on initial value.-->
    <initial_bounds>-1.10087</initial_bounds>
    <!--1 value: required final value. 2 values: lower,
         upper bounds on final value.-->
    <final_bounds>-0.42763</final_bounds>
</MocoVariableInfo>
<MocoVariableInfo name="/jointset/ankle_r/
    ankle_angle_r/value">
    <!--1 value: required value over all time. 2 values
        : lower, upper bounds on value over all time.-->
    <bounds></bounds>
    <!--1 value: required initial value. 2 values:
        lower, upper bounds on initial value.-->
    <initial_bounds>-0.0548</initial_bounds>
```

```
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds>0.222163</final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/ankle_l/ankle_angle_l/value">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>0.529032</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds>-0.00302</final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/FrameToPlatform/Frame_tx/speed">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>0.003617763</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds></final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/FrameToPlatform2/Frame_rz/speed">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>0.582009483</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds></final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/RearWheelToFrame/RearWheel/speed">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>0.731394 0.808383</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds></final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/FrontWheelToFrame/FrontWheel/speed">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
```

```xml
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>-2.495015296</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds></final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/BracketToFrame/Bracket/speed">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds>-11.7139 13</bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>0.265962 0.293958</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds>11 13</final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/LowerArmToHandlebar/Hands/speed">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>2.092949 2.21326</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds></final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/LowerArmToUpperArm/Elbow/speed">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>0.857747 0.948037</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds></final_bounds>
            </MocoVariableInfo>
            <MocoVariableInfo name="/jointset/UpperArmToTorso/Shoulder/speed">
                <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                <bounds></bounds>
                <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                <initial_bounds>3.816908 4.218688</initial_bounds>
                <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                <final_bounds></final_bounds>
            </MocoVariableInfo>
```

```
                <MocoVariableInfo name="/jointset/hip_r/hip_flexion_r/speed">
                   <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                   <bounds></bounds>
                   <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                   <initial_bounds>-3.27898 -2.9667</initial_bounds>
                   <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                   <final_bounds></final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/hip_l/hip_flexion_l/speed">
                   <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                   <bounds></bounds>
                   <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                   <initial_bounds>-2.455 -2.22164</initial_bounds>
                   <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                   <final_bounds></final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/knee_r/knee_angle_r/speed">
                   <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                   <bounds></bounds>
                   <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                   <initial_bounds>0.818919 0.905121</initial_bounds>
                   <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                   <final_bounds></final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/knee_l/knee_angle_l/speed">
                   <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                   <bounds></bounds>
                   <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                   <initial_bounds>-1.91917 -1.73639</initial_bounds>
                   <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                   <final_bounds></final_bounds>
                </MocoVariableInfo>
                <MocoVariableInfo name="/jointset/ankle_r/ankle_angle_r/speed">
                   <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                   <bounds></bounds>
                   <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                   <initial_bounds>-2.98802 -2.70344</initial_bounds>
```

```xml
                  <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                  <final_bounds></final_bounds>
               </MocoVariableInfo>
               <MocoVariableInfo name="/jointset/ankle_l/ankle_angle_l/speed">
                  <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                  <bounds></bounds>
                  <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                  <initial_bounds>0.412876 0.456336</initial_bounds>
                  <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                  <final_bounds></final_bounds>
               </MocoVariableInfo>

            </state_infos>
            <!--Set state variable bounds for all states matching a regular expression.-->
            <state_infos_pattern>
               <MocoVariableInfo name="/jointset/.*/speed">
                  <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                  <bounds>-50 50</bounds>
                  <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                  <initial_bounds></initial_bounds>
                  <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                  <final_bounds></final_bounds>
               </MocoVariableInfo>
            </state_infos_pattern>
            <control_infos>
               <MocoVariableInfo name="/forceset/ShoulderTorque">
                  <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                  <bounds>-0.65 0.65</bounds>
                  <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                  <initial_bounds>0.38 0.4</initial_bounds>
                  <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                  <final_bounds></final_bounds>
               </MocoVariableInfo>
               <MocoVariableInfo name="/forceset/ElbowTorque">
                  <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
                  <bounds>-0.5 0.5</bounds>
                  <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
                  <initial_bounds>0.18 0.22</initial_bounds>
                  <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
                  <final_bounds></final_bounds>
               </MocoVariableInfo>
```

```
         <MocoVariableInfo name="/tau_ankle_angle_r">
            <!--1 value: required value over all time. 2 values: lower, upper bounds on value over all time.-->
            <bounds>-1 1</bounds>
            <!--1 value: required initial value. 2 values: lower, upper bounds on initial value.-->
            <initial_bounds></initial_bounds>
            <!--1 value: required final value. 2 values: lower, upper bounds on final value.-->
            <final_bounds></final_bounds>
         </MocoVariableInfo>
      </control_infos>

      <!--The control variables' bounds.-->
      <!--Integral/endpoint quantities to minimize or constrain.-->
      <goals>
         <MocoControlGoal name="control">
            <!--In cost mode, the goal is multiplied by this weight (default: 1).-->
            <weight>1</weight>
         </MocoControlGoal>
         <MocoStateTrackingGoal name="mytracking">
            <!--Trajectories of states (coordinates, speeds, activation, etc.) to track. Column labels should be state variable paths, e.g., 'knee/flexion/value'-->
            <TableProcessor name="reference">
               <!--File path to a TimeSeriesTable.-->
               <filepath>OpenSenseWithBike_02_afterslip.sto</filepath>
               <!--Operators to apply to the source table of this processor.-->
               <operators>
                  <TabOpLowPassFilter>
                     <!--Low-pass cutoff frequency (Hz) (default is -1, which means no filtering).-->
                     <cutoff_frequency>-1</cutoff_frequency>
                  </TabOpLowPassFilter>
               </operators>
            </TableProcessor>
            <MocoWeightSet name="state_weights">
               <objects>
                  <MocoWeight name="/jointset/LowerArmToUpperArm/Elbow/value">
                     <!--Weight (default: 1)-->
                     <weight>0.01</weight>
                  </MocoWeight>
                  <MocoWeight name="/jointset/LowerArmToHandlebar/Hands/value">
                     <!--Weight (default: 1)-->
                     <weight>0.01</weight>
                  </MocoWeight>
                  <MocoWeight name="/jointset/UpperArmToTorso/Shoulder/value">
```

```
                                <!--Weight (default: 1)-->
                                <weight>0.01</weight>
                            </MocoWeight>
                            <MocoWeight name="/jointset/RearWheelToFrame/RearWheel/value">
                                <!--Weight (default: 1)-->
                                <weight>1</weight>
                            </MocoWeight>
                            <MocoWeight name="/jointset/BracketToFrame/Bracket/value">
                                <!--Weight (default: 1)-->
                                <weight>1</weight>
                            </MocoWeight>
                        </objects>
                    </MocoWeightSet>
                    <!--Flag to determine whether or not references contained in the reference_file are allowed to be ignored by the cost.-->
                    <allow_unused_references>false</allow_unused_references>
                    <weight>100</weight>
                </MocoStateTrackingGoal>
            </goals>
            <!--Variable info to apply to all Lagrange multipliers in the problem. The default bounds are [-1000 1000].-->
            <MocoBounds name="multiplier_bounds">
                <!--1 value: required value. 2 values: lower, upper bounds on value.-->
                <bounds>-3000 3000</bounds>
            </MocoBounds>
        </MocoPhase>
    </MocoProblem>
    <!--The optimal control algorithm for solving the problem.-->
    <MocoCasADiSolver name="solver">
        <!--The number of uniformly-sized mesh intervals for the problem (default: 100). If a non-uniform mesh exists, the non-uniform mesh is used instead.-->
        <num_mesh_intervals>135</num_mesh_intervals>
        <!--'trapezoidal' for trapezoidal transcription, or 'hermite-simpson' (default) for separated Hermite-Simpson transcription.-->
        <transcription_scheme>hermite-simpson</transcription_scheme>
        <!--Maximum number of iterations in the optimization solver (-1 for solver's default).-->
        <optim_max_iterations>100000</optim_max_iterations>
        <!--Tolerance used to determine if the objective is minimized (-1 for solver's default)-->
        <optim_convergence_tolerance>0.001</optim_convergence_tolerance>
        <!--Tolerance used to determine if the constraints are satisfied (-1 for solver's default)-->
        <optim_constraint_tolerance>0.001</optim_constraint_tolerance>
        <!--A MocoTrajectory file storing an initial guess.-->
        <guess_file>Predict_BMX_Gate_r18_afterslip_initialspeeds_02_solution.
```

```
            sto</guess_file>
        <!--The finite difference scheme CasADi will use to
            calculate problem derivatives (default: 'central').-->
        <optim_finite_difference_scheme>forward</
            optim_finite_difference_scheme>
    </MocoCasADiSolver>
</MocoStudy>
```