Gina Magnussen

Power Modeling in Cross-Country Skiing

A Quantitative Approach by Sensitivity Analysis and Monte Carlo Simulation

June 2019

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Senter for toppidrettsforskning

Master's thesis
2019
Gina Magnussen
Master's thesis

NTNU
Norwegian University of Science and Technology

NTNU
Norwegian University of Science and Technology

NTNU
Norwegian University of
Science and Technology

Power Modeling in Cross-Country Skiing

A Quantitative Approach by Sensitivity Analysis and Monte Carlo Simulation

Gina Magnussen

Applied Physics and Mathematics
Submission date: June 2019
Supervisor: Bo Henry Lindqvist (NTNU)
Co-supervisor: Jan Kocbach (Centre for Elite Sports Research (SenTIF), NTNU)

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

A quantitive approach to how variations in input affects the output in calculation of propulsive power in the power balance model in cross-country skiing is provided in the current study. Three sprint races and two long distance races, including both classical and skate technique, was investigated by analysis of data collected by GPS sensors at the Beitosprinten skiing competition in 2017 and 2018. The friction coefficients for these races were estimated in a separate field test, and acceleration and the inclination in the track was calculated by an amended version of the central differences scheme. The results showed that power calculations are the most sensitive to changes in the drag area, and the least sensitive to changes in body mass, when varying model parameters separately in an investigated uphill segment of a race. Varying the body mass m within a range of ± 2 kgs in this segment, the absolute relative difference in propulsive power was at most 3%. Varying the drag area within a range of ± 0.2, the equivalent measure was at most 57.4%. The combined effect of parameter changes in Monte Carlo simulations for each race showed that the maximum relative deviation in absolute value of all races was 11%, also looking at uphill segments. As a result of the analysis in this thesis, it would be desirable with more precise classification of subtechniques to decide correct drag area for use in calculations of propulsive power.

Sammendrag

En kvantitativ tilnærming til hvordan variasjoner i input påvirker output ved beregninger av krefter i kraftbalansemodellen i langrenn er presentert i dette studiet. Tre sprintløp og to langdistanseløp, inkludert både klassisk teknikk og fristil, ble undersøkt ved analyse av data innsamlet med GPS-sensorer under Beitosprinten 2017 og 2018. Friksjonskoeffisienten for disse løpene ble estimert i en separat test i felt, og akselerasjon og stigning i løypen ble regnet ut ved hjelp av en tilpasset versjon av 'central differences'-metoden, en metode for approksimasjon av den deriverte. Resultatene viste at kraftutregninger er mest sensitive for endringer i 'drag'-areal, og minst sensitiv for endringer i kroppsmasse når modellparametre varieres hver for seg i analyse av et segment i en oppoverbakke i et løp. Ved å variere kroppsmassen m innenfor et område på ± 2 kg i dette segmentet, var det absolutte relative avviket i kraftutregningen på det meste 3%. Ved å variere 'drag'-arealet innenfor et område på ± 0.2, ble det tilsvarende resultatet 57.4% på det meste. Den kombinerte effekten av endringer i parametere i Monte Carlo-simuleringene for hvert løp viste at det maksimale relative avviket i absoluttverdi for alle løp var 11%, også her ved analyse av segmenter i oppoverbakke. Som resultat av analysene i dette studiet, er det ønskelig med mer presis klassifisering av delteknikk for å bestemme rett verdi for 'drag'-areal som skal brukes i kraftutregningene.

Preface

This thesis completes my years of studying Applied Physics and Mathematics at The Norwegian University of Science and Technology. The work has been carried out during the spring semester of 2019 and is an extension of my specialization project from the fall of 2018.

The thesis is written in cooperation with Centre for Elite Sports Research (SenTIF) at NTNU. A special thank you to SenTIF for welcoming me and including me in their work and for the opportunity to combine my studies with my personal interest for sports in writing this thesis. I would also like to thank co-supervisor Jan Kocbach for valuable perspectives, useful discussions and honest feedback.

Many thanks to my supervisor Bo Lindqvist for our regular meetings, and for all technical discussions and statistical input on the problems and questions that arose in working with this project.

Lastly, thank you to family, friends and everyone I have gotten to know in these past five years. My time in Trondheim would not have been the same without you.

Gina Magnussen
Trondheim, June 2019

Table of Contents

List of Tables

List of Figures

Abbreviations

DGPS	=	Differential Global Positioning System
		Extension of the GPS system, with higher accuracy
		(From 10m to $10 - 15$ cm)
IMU	=	Inertial measurement unit
P_{prop}	=	locomotive power / propulsive power
SenTIF	=	Senter for toppidrettsforskning
	=	Centre for elite sports research
COM	=	Center of mass
COF	=	Coefficient of friction

Chapter 1

Introduction

In order to improve the performance of cross-country skiers, we need to better understand their use of propulsive power in the cross-country skiing track. One way of improving our understanding is to investigate a model for power balance. Already in 1990, van Ingen Schenau and Cavanagh (1990) proposed a power balance model describing power use and power dissipation in endurance sports. From this model the propulsive power P_{prop} can be calculated directly given sufficient data, and it is used as an alternative measure for estimation of metabolic power of the athletes.

The model has by earlier research been confirmed to be a valid tool for gaining better understanding of the performance of cross-country skiers. For instance, recent research by (Gløersen et al. (2018b)) used the power balance model to estimate and determine the propulsive power of high-level skiers based on data generated from a simulated distance race on roller skis. This study also evaluated the accuracy of the results when the power balance principle is applied to cross-country skiing. (Gløersen et al. (2018a)) also published an article on the accuracy provided by tracking devices used for sports applications. Both articles are presented in Gløersen's newly published doctoral thesis (Gløersen (2019)).

Previous work with simulations using the power balance model has aimed to model power as a function of a single variable or to simulate to estimate finishing times given different parameters. (Swarén and Eriksson (2017)) estimated continuous propulsive power to enable in-depth analyses of power output in cross-country sprint skiing by using real-time

positioning, however only for a limited number of skiers. (Hausken et al. (2014)) was able to quite accurately predict a skier's performance using the power balance model and to estimate the influence of on performance due to changes in various model factors. This was however done by modeling locomotive power as a function of speed. Furthermore, though (Gløersen et al. (2018b)) was one of the first to quantify measurement error in propulsive power using the power balance principle on distance races on roller skis, the case of investigating real-life xc-skiing race data to quantify uncertainty in output propulsive power and assessing the sensitivity given changes in the input parameters, has rarely been adressed.

This project aims to enhance the understanding of how the power balance model can be used as a tool for improving the performance of cross-country skiers by mainly investigating the sensitivity of changes in the input and variety in output given uncertainty in the input. The current study uses GPS data collected at classical and skate, sprint and long distance races, and processes the input to remove noise before calculating propulsive power. A sensitivity analysis for model parameters is conducted and variations in parameters are combinined in a Monte Carlo simulation. The friction coefficient μ was also estimated in a separate field test.

The results in this research gives insight into how variations in input parameters influence the output of propulsive power as well as what the combined effect in calculated power is when several parameters vary simultaneously. The aim of this is to give a quantitive approach to better understand how large the error of calculation of propulsive power can be when there is uncertainty related to our knowledge of the model parameters and input. In other words, how accurate should our input be in order to compute valid results that are useful for coaches and athletes in the work of achieving better performance in the cross-country skiing track.

In this thesis, chapter two presents the model and relevant related theory. Chapter three describes the setting for collecting data, how the data was collected, experiments conducted and what assumptions were made when approaching the model and doing calculations. Chapter four introduces in further detail the analysis and processing of data. Results are presented in chapter five, and discussion and conclusion is given in chapter six and seven, respectively.

Chapter 2

Theory

2.1 Power balance model

The propulsive power of a cross-country skier can be calculated from a power balance model, as stated by van Ingen Schenau and Cavanagh in 1990 (van Ingen Schenau and Cavanagh (1990)):

$$\begin{aligned} \frac{dE_k}{dt} &= mv\frac{dv}{dt} \\ &= P_{prop} - \mu mg\cos(\alpha)v - mg\sin(\alpha)v - 0.5\rho C_d A v^3 \end{aligned} \tag{2.1}$$

On the left hand side of the equation is $\frac{dE_k}{dt}$, which is the rate of change in translational kinetic energy of the skier. Simply put, kinetic energy is the energy of an object with mass in motion. The object has this energy because of motion from one location to another location. Translational kinetic energy is thus energy an object with mass has due to its motion from one position to another. The derivative of this quantity is a measure of how much this energy changes per time unit. Next is P_{prop} the propulsive power of the skier, a measure of how much power the skier produces, or how much energy the skier uses, to move in the direction of motion. Further is m the body mass of the skier, v is the speed along the track and $\frac{dv}{dt} = \dot{v}$ is the acceleration along the same course.

In the next term is μ the friction coefficient (COF), a measure of the friction between skis and snow. The coefficient is influenced by environmental conditions and skiing equipment

as well as the skiing technique. Combined, this affects the gliding and thus the speed of the skier. g is the gravitational acceleration and α is the angle of inclination of the skiing track measured in radians. In the two last terms ρ is the air density and C_d is the drag coefficient. The drag coefficient is a coefficient related to fluid dynamics, measuring the how much drag is induced by an object. Lastly, A is the projected frontal area of the skier. Multiplied with the drag coefficient, this becomes the drag area C_dA. The drag area changes with technique of the skier and if the skier is in the upright or in a tucked position. For instance will the drag area when double poling be larger than the drag area when skiing in the tucked position, and by that change how much the drag affects the skier. All terms described above are gathered in Table 2.1 in order to achieve an overview of the model and its terms. Additionally, Equation (2.2) shows a simplified version of the power balance model and in which direction the terms are acting relative to the direction of motion.

$$\begin{array}{lcccccccc} \frac{dE_k}{dt} & = & mv\frac{dv}{dt} & & & & & & \\ & = & \text{power} & - & \text{friction} & - & \text{gravity} & - & \text{drag} \\ & = & \Longrightarrow & - & \Longleftarrow & - & \Longleftrightarrow & - & \Longleftarrow \end{array} \tag{2.2}$$

The model itself is based on energy balance and is often used as a tool to investigate performance in endurance sports like for instance cross-country skiing. Because it takes power production and power dissipation of the athlete into consideration, it is commonly used to look into how skiers use their energy in a race track. From a physical perspective, the model is simply derived from Newton's second law. This famous physical law states that the sum of the forces in a system is equal to the mass times the acceleration of that same system. For simplicity, the athlete and the equipment is usually modeled as a point mass, and the mechanical energy of the skier is thus equal to translational kinetic energy and potential energy due to gravity. The gravitational potential energy is the stored energy of an object due its vertical position or height and is dependent on the mass of the object and the height. The sum of potential energy and kinetic energy of an object is known as the object's mechanical energy. Considering the skier and its equipment as the system, the propulsive power P_{prop} is then equal to the system's rate of change in mechanical energy and the work done by the environment. In other words, the propulsive power is equal to the sum of the change in kinetic energy of the skier and the forces acting on the skier from the surrounding environment. The latter is mainly due to the air drag force and the frictional forces between skis and the snow, but gravity also plays an important role.

While both air drag and forces of friction act in the direction opposite to the direction of motion, the gravitational force can act both ways depending on the inclination of the track. When skiing uphill the gravity term will decrease the kinetic energy of the skier if all other terms are constant. Thus, in order to maintain the same amount of kinetic energy, a skier will have to use more power to keep up the speed. Opposite can the kinetic energy and the speed increase when the skier is skiing downhill due to the gravitational force, even without using any more power. In practice this means that a skier needs to use more energy when skiing uphill, and that less energy is needed downhill, to maintain the same speed.

Setting up the equation for the forces and multiplying by the velocity v, eq. (2.1) for calculation of propulsive power is obtained.

Table 2.1: Power balance model terms with description

Term		**Explanation/Description**	**Unit**
$\frac{dE_k}{dt}$	=	Rate of change in kinetic energy	$\left[\frac{J}{s}\right]$
m	=	Body mass of the skier	$[kg]$
v	=	Speed along the track	$\left[\frac{m}{s}\right]$
$\frac{dv}{dt}$	=	Acceleration along the track	$\left[\frac{m}{s^2}\right]$
P_{prop}	=	Propulsive power	$\left[\frac{J}{s}\right]$
μ	=	Friction coefficient	$[-]$
g	=	Gravitational acceleration	$\left[\frac{m}{s^2}\right]$
α	=	Angle of inclination of the track measured in radians	$[-]$
ρ	=	Air density	$[\frac{kg}{m^3}]$
C_d	=	Drag coefficient	$[-]$
A	=	Projected frontal area of the skier	$[m^2]$
C_dA	=	Drag area	$[m^2]$

2.2 Friction

Friction is a force that prevents relative motion of systems in contact. This force is dvided into kinetic friction, where systems in contact are moving relative to one another, and static friction, where systems in contact are stationary. Friction is highly complicated depending on for instance speed, surface of materials in contact and temperature. In a cross-country skiing race track, the quality and state of snow, and skis influence the magnitude of the friction present.

The coefficient of friction describes the ratio of the frictional force between two objects and the force pressing them together, usually the normal force. Highly polished surfaces typically have lower coefficients of friction than unpolished surfaces (Colbeck (1994)). Waxing of skis and ski base texturing treatments also influence the coefficient of friction, though temperature and snow quality, e.g., if the now is new or transformed, hardness and texture, has a larger effect on friction (Budde and Himes (2017)). Friction is also shown to increase with speed (Hasler et al. (2016); Braghin (2016)). These relationships are however hard to model in practice.

Calculation of the friction coefficient

For better calculation of power in the power balance model, one can try to find the best estimates possible for the parameters in the model, for instance the friction coefficient. The best friction coefficient is the number that reflects the weather and the environmental conditions of the skiing track on the specific race day. One way of estimating the friction coefficient is based on a classical kinematic equation from physics (Young and Freedman (2012)). This equation states that the velocity v at a given point is equal to the initial velocity v_0 plus the acceleration a times the time difference Δt, assuming that the acceleration is constant. When calculating friction, the acceleration is equal to the friction coefficient μ times the gravitational acceleration g. The equation is easily solved for μ and one obtains Equation (2.3) for the friction coefficient:

$$\mu = \frac{v - v_0}{g\Delta t} \tag{2.3}$$

Under optimal conditions, i.e. meaning no impact from air drag or change in speed due to gravity or use of energy of the skier, the loss of speed is only due to the friction between skis and the snow. The friction coefficient can therefore be calculated by measuring the

speed at two points and measuring the time difference between the same points, given that the conditions are as close to optimal as possible.

There are many ways to calculate the friction coefficient in practice. However, most setups include sensor equipment along a straight line and observing how a test object loses speed as it is passing the sensors under the best conditions possible. The initial velocity v_0, the velocity v when the test object has lost some speed and the corresponding time difference Δt between the measurements is measured to calculate the coefficient of friction. The actual setup used for calculation of the COF in this thesis is described in Chapter 3.

2.3 Drag and drag area

Drag is a frictional force acting on an object opposite to the relative direction of a moving object, but with respect to a fluid surrounding the object. The size of this force generally depends on the shape and area of the object investigated, the velocity, as well as the material of the surface of the object, around which the fluid is flowing. This last dependency is accounted for in the drag coefficient C_d, which is a dimensionless quality. In the cross-country skiing context, this constant varies depending on the shape of the skier and the material of the clothing and equipment of the skier.

Since the drag force depends on area, a larger area means larger drag force and vice versa. So if a skier is skiing upright, the drag force would be larger than if the skier skied in a tucked position at the same speed. If the variations of drag area C_dA of a skier could be implemented into the the power balance model, this would lead to more precise calculations. Drag area will in this thesis be implemented as a function of subtechnique, further described in Chapter 3.

2.4 Data processing: Removing noise in data

All measurements include noise. In practice this means that every measured signal or data point consists of an underlying true signal or observation and some random measurement error. Before doing calculations and drawing conclusions based on collected data, this noise or random error should be removed. If this is not done, the errors can be magnified and contribute to substantial errors in future calculations and conclusions. Removal of noise is important in any field, also in human movement in sports.

2.4.1 Importance of noise removal

Data is often sampled discretely with a given samling frequency. In particular does human movement generally consist of low frequencies whereas the measurement noise and random error usually consist of higher frequencies (Bartlett (2007),
Skaloud and Limpach (2003), Skaloud et al. (2004)). When looking into human movement and analyzing data it is therefore a key interest for the researcher to attempt to remove the high frequency noise. To do this, the limit between the true low-frequency signal and the high frequency noise must be found. This frequency limit is different depending on the sport and is decided based on whether it is a sport with relatively slow movement or a sport with high energy transfers. Either way, it is important to find this limit to keep all important information for further analysis and to avoid working with errors in the data.

Removal of noise in sports biomechanics and human movement is important, but it is not always that easy to remove all of the noise while at the same time keeping all of the true information. This is especially the case when dealing with data where transient signals are present, signals caused by sudden changes in energy over a short period of time. The process of removing noise should be conducted before using the data for further calculations. This is because the calculations usually are highly non-linear and will result in non-linear combinations of random noise. In turn this can affect the noise removal process in a negative way later.

It is also worth noting that even though the noise in the measured data has an amplitude of only 1% of the true signal, this noise can become of intolerable size if further calculations are done based on this noisy data. The noise leads to considerable inaccuracies in the derived data if the noise is not removed. Consider for instance that the position of an athlete is recorded. Then the calculated velocity and acceleration can possibly contain large errors if the noise is not removed from the positioin data before doing calculations.

This also means that this type of error can be even more significant if the random error in the recorded position data is even larger to begin with. To illustrate the importance of noise removal, consider the following example from Bartlett (2007).

Imagine an extremely simplified example where the position of some object is recorded, and in addition the true analytical expression including the noise is known. The position r of an object including noise is

$$r = 2\sin(4\pi t) + 0.02\sin(40\pi t)$$

Here the first term on the right hand side of the equation is the true signal. This part of the recorded signal has a frequency of 4π and an amplitude of 2. The second term is the noise. This term has a ten times higher frequency than the true signal, but has an amplitude that is a hundred times smaller – the noise has an amplitude of only 1% of the true measurement. However, things change when calculating the derivative to find the velocity v:

$$v = 8\pi\cos(4\pi t) + 0.8\pi\cos(40\pi t)$$

Now the amplitude of the noise is ten times larger than in the noise term in the position data and is as high as 10%. When then again differentiating to fint the acceleration a, the effect is very much significant.

$$a = -32\pi^2\sin(4\pi t) - 32\pi^2\sin(40\pi t)$$

The ratio between the frequencies is still the same due to the expression of the original position, but the amplitude is significantly changed. The random error in the acceleration data now has the same amplitude as the true signal, which is an error that is not tolerable.

Table 2.2: Example of functions for position, velocity and acceleration, with and without error, plotted to highlight the importance of filtering.

	Importance of filtering	
	With noise	Without noise
Position	$r = 2\sin(4\pi t) + 0.02\sin(40\pi t)$	$r = 2\sin(4\pi t)$
Velocity	$v = 8\pi\cos(4\pi t) + 0.8\pi\cos(40\pi t)$	$v = 8\pi\cos(4\pi t)$
Acceleration	$a = -32\pi^2\sin(4\pi t) - 32\pi^2\sin(40\pi t)$	$a = -32\pi^2\sin(4\pi t)$

Based on the example above, the noise is clearly magnified by differentiating and disturbs the original true signal. Curves of the true position, velocity and acceleration and their respective errors are shown in Figure 2.1. Table 2.2 shows the expressions in the plot, with and without the noise. The noise is visible in all three subplots of Figure 2.1, but becomes increasingly visible in the velocity and the acceleration. Unless an attempt to remove the noise is made, it can lead to significant inaccuraries and possibly false conclusions can be drawn. This example therefore illustrates the importance of removing random errors from a measurement or signal.

Two commonly used techniques to reduce measurement error and remove high-frequency noise from low-frequency movement data are Butterworth filtering and spline smoothing (Bartlett (2007), p. 134). Only spline smoothing is considered in this thesis.

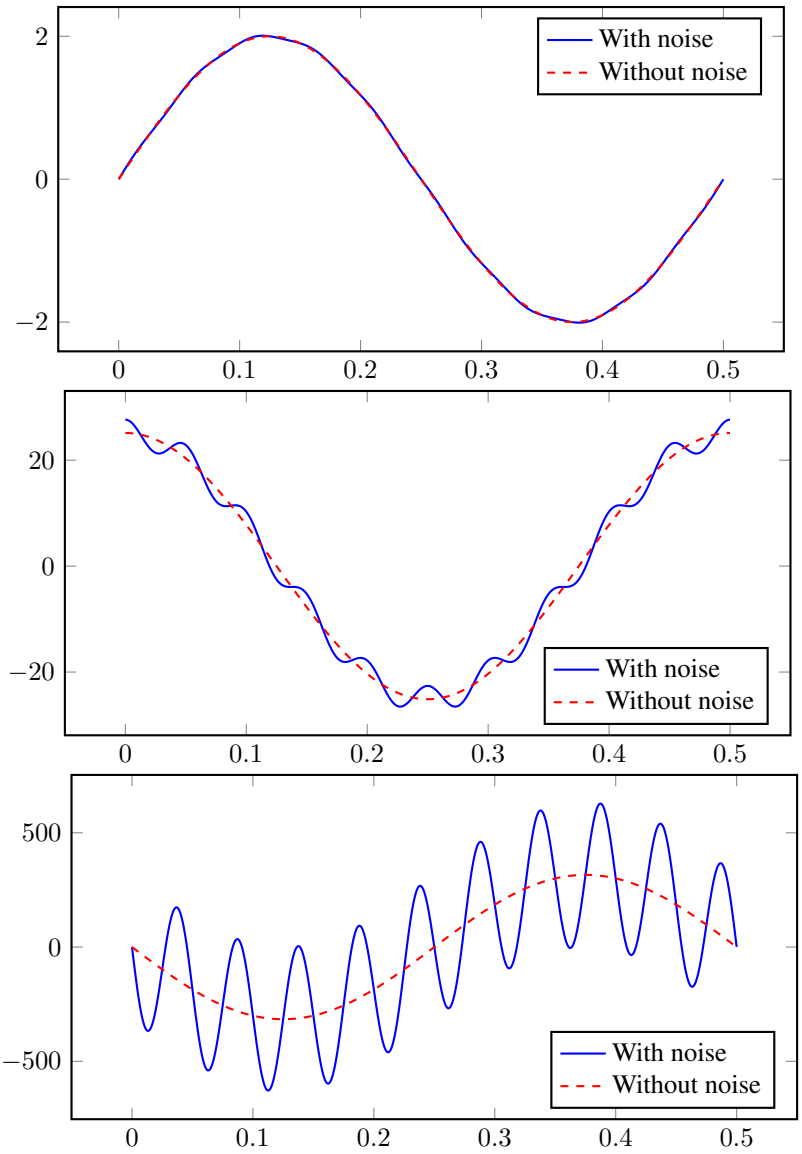

Figure 2.1: Curves (from top to bottom) for position and its first (velocity) and second (acceleration) derivative, with and without noise. Illustration of the imprtance of filtering. Example inspired by Bartlett (2007).

2.4.2 Smoothing splines

The aim of smoothing splines is to fit a curve that fits well to a set of observed data while also being somewhat smooth. As for other regression methods, we want to fit a function $f(x)$ to the data such that the error between the true value at x_i and the estimated function value $f(x_i)$ is as small as possible. The measure of error frequently used is RSS - Residual Sum of Squares - which measures the squared deviation of x_i from $f(x_i)$ for all observations $i = 1, ..., n$. In mathematical terms this means $\sum_{i=n}^{n}(y_i - f(x_i))^2$. With only this restriction it is possible to make RSS zero by interpolating all points. This will however greatly overfit the data. Since we also want a smooth curve, a way to ensure this is to add a term to the RSS ex ssion that controls smoothness of the fitted curve. When this term is added, the expression to be minimized becomes

$$\sum_{i=n}^{n}(y_i - f(x_i))^2 + \lambda \int f''(t)^2 dt \tag{2.4}$$

The term $\sum_{i=n}^{n}(y_i - f(x_i))^2$ is called the *loss function* that makes sure that f fits the data well. If this term is small, the fitted curve is close to the observed data for all data points and the error is small. The term $\lambda \int f''(t)^2 dt$ is called the *penalty* term and penalizes the variability of f. Since the derivative of a quantity is a measure of how much this quantity changes, the first derivative $f'(x_i)$ measures the slope of f at data point x_i. Analogously, the second derivative measures how much the first derivative changes. Broadly speaking, the second derivative of a function is a measure of how rough the function is. If $f(t)$ is wiggly around t, the second derivative is large in absolute value, otherwise it is close to zero. Since the integral notation can be thought of as a summation over the range of t, $\int f''(t)^2 dt$ and is a measure of the total change of $f'(t)$ over its entire range. If $f(t)$ is smooth, $f'(t)$ is close to constant, and $\int f''(t)^2 dt$ will have a small value. Conversely, if $f(t)$ is jumpy and wiggly, then $f'(t)$ will vary a lot, and the sum $\int f''(t)^2 dt$ will take on a much larger value. Hence, since we want to make RSS as small as possible, the penalty term $\lambda \int f''(t)^2 dt$ encourages the function $f(\cdot)$ to be smooth.

In the penalty term, the *smoothing parameter* λ decides how smooth $f(\cdot)$ will be. If $\lambda = 0$, the penalty term has no effect and $f(\cdot)$ will exactly interpolate our data points and possibly be very jumpy. A low lambda will therefore give a flexible fit and mean that the bias of the fitted function is low, but that the variance can be high, and the data is overfitted. The larger λ is, the more weight is put on the smoothing penalty, and the smoother the function $f(\cdot)$

will be. In this case, the variance is lower, but the bias is higher, and one risks underfitting the data. The parameter λ therefore controls the flexibility and the bias-variance trade-off of the smoothing spline. The smoothing parameter λ is connected to the *effective degrees of freedom*, which also controls the flexibility of the smoothed curve. The larger λ is, the more emphasis is put on smoothing, and the effective degrees of freedom is smaller. (James et al. (2017))

For fitting a function to the data, a set of *knots* is defined, dividing the range of data into K regions. For the smoothing splines method, the number of knots is equal to n, giving a knot at each data point $x_i, i = 1, ...n$. The number of knots $K = n$ leads to a more flexible fit.

Between each pair of knots, we want to fit a polynomial. Additionally, we require that the function $f(\cdot)$ is continuous at each knot, and also that its first and second derivative is continuous at each knot. It can be shown that the function that minimizes RSS and that also meets the requirements mentioned above, is a natural cubic spline with knots at each data point (Hastie et al. (2017)). A natural cubic spline fits a cubic polynomial between each pair of knots, but is linear beyond the boundary knots. This leads to more stable predictions for extreme valued data points.

Smoothing splines is chosen as the method to filter the input data of the power balance model as it leaves the data better suited for analysis later, for instance in the derivation of acceleration. Especially is this method suited for trajectory smoothing, as it requires continuous speed (first derivative) and acceleration (second derivative). Furthermore, the interpolating properties of this method provides a kind of spatial filtering which effectively reduces high-frequency noise and gives smooth transitions where jumps in collected GPS data may occur due to satellite constellation. Fitting separate polynomials at different intervals, the method accounts for different behavioural patterns of the data, while at the same time, the smoothness and continuity restrictions will bridge over data outliers (Skaloud and Limpach (2003); Skaloud et al. (2004)).

2.5 Central differences

After the noise is removed, the next step is to find the velocity and the acceleration based on the original position data. For this purpose, the *central difference method* is used. This is a finite difference approximation to the derivative, a commonly used technique in numerical mathematics.

The derivative of a function $f(x)$ with respect to x, $f'(x)$, is defined as

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

where is h is a small number approaching zero. Letting h have a finite value instead, the expression becomes

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{2.5}$$

When h is small, this is an approximation to the derivative. This way of approximating the derivative utilizes function values close to a data point x to find an estimate of the unknown value for the derivative of $f(x)$ in point x. This method is called the central differences method.

Given a function $f(x)$ with n data points $x_i, \;\; i = 1, ..., n$ the approximated derivative becomes

$$f'(x_i) \approx \frac{f(x_{i+h}) - f(x_{i-h})}{x_{i+h} - x_{i-h}} \tag{2.6}$$

2.5.1 Acceleration

The central differences method is used for calculation of the acceleration. With velocity v and time t in the track, the acceleration a in data point i is found by

$$a_i \approx \frac{v_{i+h} - v_{i-h}}{t_{i+h} - t_{i-h}} = \left(\frac{dv}{dt}\right)_i \tag{2.7}$$

2.5.2 Slope angle in the race track

The slope angle α in data point i is found by combination of the central differences method and simple geometry.

$$\alpha = \arctan\left(\frac{dy}{dx}\right) \tag{2.8}$$

where dy is the change in the data values for elevation, $elev$, and dx is the change in the data values for distance traveled, x, in the track. The slope angle is thus

$$\alpha_i = \arctan\left(\frac{elev_{i+h} - elev_{i-h}}{x_{i+h} - x_{i-h}}\right) \tag{2.9}$$

2.6 Sensitivity analysis

Imagine that you have some mathematical model, and that you are interested in knowing the properties of that model. Given some input you feed the model, it gives you an output. But if you change the input slightly, what will happen to the output? Will it change at all, and if it changes, how and how much does the output change? Furthermore, how large is the uncertainty in the resulting output? This is the core of *sensitivity analysis*, a tool for investigating and evaluating how much and in what way variations in input affects the output. (Saltelli et al. (2004)) defines sensitivity analysis as

> 'The study of how the uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input'

The situation above, where one is interested in understanding how a model is influenced by its input, is quite common within many fields. For researchers in this situation, there are many questions of interest to answer. For instance, in which parameters is the model most sensitive to changes? I.e., which parameters gives the largest change or uncertainty in the output given changes in the input?

Continue to think of a real-life example where the model describes a natural phenomenon or a relationship between physical forces. In this case, there might be data collected by some measuring device. Additionally, other parameters may be estimated by an experimental test. Interesting questions can then be: How much does noisy input data influence the resulting output from the model? Should input data be filtered before input, and how much in that case does filtering affect the result? How important is it that the estimated values are estimated with precision down to the third decimal place? Is it important at all to estimate it exactly? Or if it is not possible to estimate it under certain circumstances - is it suffient to estimate the value based on other relevant data? These are all questions which sensitivity analysis seek to answer.

2.6.1 The cross-country skiing case

The questions above are the reason for choosing this method to evaluate how much the output of the power balance model is influenced by changes in its input. If the input speed is filtered, how much does this change the output propulsive power? And is the model more sensitive to changes in the speed, the friction coefficient or the drag area?

How accurately should the friction coefficient be estimated? Or if it in a specific case is not possible to test the skiing conditions and conduct a friction test, will the model provide trustworthy enough results if μ is approximated from other available weather data and previous knowledge from similar races and conditions? What happens if the track profile is not exact? Will collected GPS data give good enough results anyway? If a friction test is not possible to conduct, or if exact body mass of the skiers is not available, how large is the resulting error in P_{prop}? These are a few of many interesting questions to answer when developing the power balance model as a tool for investigating cross-country skiing.

Though these questions have a highly practical application, some mathematical ground should be established before investigating the situation further. Note that the notation used in this context and that the approach of analysis presented here is chosen based on applicability to the cross-country skiing case and the research questions sought to answer.

For notational purposes, assume that a general mathematical model is given by the following equation:

$$Y = g\left(\mathbf{X}, \boldsymbol{\theta}\right) \tag{2.10}$$

where Y is the calculated response from the function $g(\cdot)$, with parameters $\mathbf{X}$ and $\boldsymbol{\theta}$. In this case, $\mathbf{X}$ is a matrix with relevant and collected data and $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_m]$ is a vector of model parameters.

For the case investigated in this thesis, the response Y is the propulsive power P_{prop} and the matrix $\mathbf{X}$ is a collection column vectors where each vector j represents one variable of data $\mathbf{x}^{(j)} = [x_{i=1}^{(j)}, ..., x_{i=n}^{(j)}]$, for instance speed, position, elevation, time from the collected data. Additional data and labels, such as name and body mass of the skier, are added in separate columns.

The parameter vector θ consists of the friction coefficient, the drag area of the skier, the air density and the body mass of the skier, giving the parameter vector $\theta = [\mu, C_dA, \rho, m]$. There are two interesting cases when investigating the the power balance model by a sensitivity analysis. One case is changing a variable one by one, the other is to change variables at the same time. Both cases aim to give the researcher and idea of what happens to the output when the input is varied.

2.6.2 Varying parameters one by one

Case number one is the case where one variable is changed at a time. In this case, each parameter of interest is varied based on a predefined set of values before calculating the response Y. The values are chosen or determined based on relevant literature or previous research, and should cover all reasonable values for the parameter. Additionally is it of interest to choose a lower and upper bound for this interval covering values bordering to the extreme or more unlikely cases, to make sure that all possibilites are covered and tested. What values that are likely and not must be determined in each case. However, the important part in this first case is to vary only one parameter at a time, and keeping all other parameters constant when testing. The reason for this is to shed light on how the model behaves with changes in each variable separate from the others.

2.6.3 The Monte Carlo approach

Whereas case number one considers variations of the parametes isolated from each other, case number two looks deeper into what happens when all variables are varied simultaneously. This method is particularly of interest to discover the range of possible outputs for instance when several input variables contain noise or inaccuracies. However, in contrast to case number one, each variable is not varied based on a set of predefined values, but rather assigned a probability distribution with mean and standard deviation. The chosen distribution for a parameter θ is in our case, for simplicity, the normal distribution $N(\mu_\theta, \sigma^2)$. Further is the mean μ_θ set to be the most likely, or what we believe to be the most likely, value based on literature, experiments or previous experience. The standard deviation σ is chosen accordingly, but based on the corresponding uncertainty connected to the chosen mean value. For each calculation of propulsive power for a skier, a single value is drawn from each of the distributions. Repeating this procedure several times, simulated responses for P_{prop} are produced and will presented in a plot give an idea of how much the the response can vary. If the chosen values for mean and standard deviation are good enough and large enough, respectively, the result should represent the entire range of possible outcomes for the model. One can then judge whether the model still outputs useful information from which it makes sense to draw conclusions, or if the variables must lie within stricter bounds for that to happen. Asigning a distribution to each variable and simulating based on these to model risk and the probability of all outcomes is known as a *Monte Carlo approach*.

Chapter 3

Method

3.1 Setting

The data analyzed in this thesis were collected at the cross-country skiing competition Beitosprinten at Beitostølen, Norway. Beitosprinten is the national opening race of the cross-country skiing season in Norway. Here the top cross-country skiers in Norway compete as well as a few international skiers. There is a range of different races and disciplines possible to take part in at Beitosprinten, such as sprint distances and longer distance races. Out of these, data from 2017 and 2018 for several disciplines have been considered. More specifically are the races in Table 3.1 analyzed further.

Table 3.1: Data sets analyzed specified with type of race, technique used, gender and year the data were collected.

Race	Technique	Gender	Year
Sprint (prologue)	Classic	Men	2017
Sprint (prologue)	Free/Skate	Men	2018
Sprint (prologue)	Free/Skate	Women	2018
10 km	Classic	Women	2018
15 km	Classic	Men	2018

The data analysis in this thesis will however focus less on the specifics of each atlete and

rather more on the general use of data in the power balance model and the uncertainty related to this.

Race tracks, discipline and skiing technique

The race tracks used during Beitosprinten 2017 and 2018 were two similar tracks for the sprint (slight variation in length from 2017 to 2018) and a 5 kilometer race track used in the long distance races in 2018. In the 10 km race for the women the athletes skied this lap twice, and the men completed this lap three times in their 15 km race. All athletes in each race started separately with 30 seconds between each starting skier. As for the sprint races, the skiers also started separately but with 15 seconds between each athlete. The sprint race is significantly shorter and is divided into rounds starting with the qalification round. The skiers ski a short lap, and a certain number of with skiers with the best finishing times qualify to the next round. The skiers then race against each other in heats, and the top skiers from each heat qualify to the proceeding rounds. This knock-out procedure continues until a winner is crowned after the final round. Sprint data analyzed here only contains data from the qualification round, also called the prologue.

In classic technique races, skiers are restricted to using certain subtechniques, such as diagonal stride and double poling with kick, and skating is not allowed. If the race is a free technique race, the athletes use ski skating techniques, further visualized an described in (Andersson et al. (2010), p. 588). The 10 km and 15 km race, as well as the 2017 sprint race, were classical races, whereas the 2018 sprint was a free technique race where skating was used.

Variations in conditions

Skiing conditions vary of many reasons. For one, the races start at different times of the day. This means that the weather conditions could have changed and that the skiing conditions later in the day are different from the skiing conditions earlier on that same day. Furthermore, the snow conditions, and therefore the skiing conditions, can be different at different locations in the race track. There is also a possible effect depending on which starting number you have, as the skiing conditions and the condition of the snow can change considerably if many skiers have skied before you. This is particularly visible, at least visually, in medium to sharp turns of the track. Here the snow can be a lot "looser" than the snow in a newly prepped race track and can be harder to navigate in. The weather conditions also change over days, since all races analyzed are not completed on the same

day, some parts of the data is also collected from competitions a year apart. These differences will from the power balance model perspective affect the friction coefficient the most, which is of interest for modelling, and which is also experimentally estimated and investigated in this thesis. However, the effect of the differences mentioned is hard to estimate, especially since there are also other variables contributing to variability and the combination of these is hard to quantify.

3.2 Fixing parameter values

Given the setting, there are a few variables in the power balance model that need to be set before calculations can be made. This in particular means the coefficient of friction in the friction term and the air density and the drag area in the drag term. The choice of variables are based mainly on what conditions affect the variables. The coefficient of friction and the air density is affected by and thus determined by environmental conditions whereas drag area is determined by the subtechnique used by the skiers in each race.

Environmental conditions means current weather and the resulting conditions based on weather in the past. To be more clear, it means air temperature, possibly precipitation in the form of rain and or snow, snow temperature and especially the quality of the snow. Snow quality means how hard or soft the snow is, is it floury or crystalized or how old or fresh is it. Despite the fact that the environental conditions can change, the friction coefficient is assumed to be constant for all data from a given race. In contrast is the drag area modelled as a piecewice constant function of subtechnique. A more detailed description of the choice of variables follows.

Friction

The friction coefficient in the power balance model and in the analyses is assumed to be a constant. This is of course a simplification of the truth. The friction coefficient is greatly influenced by the varying conditions mentioned in the previous paragraph, but also by choice of skiing technique and speed. These reasons will however not be considered when finding an appropriate friction coefficient for the races analyzed in this thesis. By choosing to do it this way the model will be unable to express the complex ski-snow interactions and the relationship with varying speed. However, it is still done this way due to simplifying modelling reasons.

In order to use the best friction coefficient possible in the analyses, the friction coefficient

was estimated from an experimental test. The test was conducted in the field close to the race tracks on the particular race day to achieve a measure as accurate as possible for the races held that day. By this test, the weather and snow conditions are taken into account and gives an impression of how much friction will affect the skiing and use of power given the environmental conditions. The exception is the data sets from 2017, where no estimate of the friction coefficient was available at the time this thesis was written.

The friction test was conducted in the following way: On a flat area of snow close to the racing track, with as good and as similar conditions to the race tracks as possible, pairs of photo cells were set up 1m, 19m and 1m apart, respectively, along two parallell straight lines. This adds up to a total of four pairs of photo cells making up the test track, see Figure 3.1. A test skier, using a pair of skis similar to the athletes', was the test object. Tracking units were placed by sensor pair one, pair three and on the test skier in a racing vest, or bib, on the back between the shoulders similar to the competing athletes. The test skier first started a sufficient distance away from sensor pair one, then double poled to gain enough speed, and then tucked into the hockey position before reaching sensor pair one. Holding this position, without moving the skis, passing times were registered between all sensors. From these times, the average speed between sensors one and two and then between sensors three and four was found and used together with the time difference to compute the friction coefficient by Equation (2.3).

The experimental set up for estimation of the friction coefficient is shown in Figure 3.1. Here is the initial speed v_0 from Equation (2.3) set to be the calculated mean speed between sensor pair one and two. Correspondingly is the ending speed v of the test set to be the calculated mean speed between sensor pair three and four. The time difference Δt is found by adding together the time used to travel between photo cell pair two and three, half of the time spent to travel from photo cell pair one to two, and three to four, respectively. If the test is carried out perfectly, it is assumed that the loss of speed is only due to friction between the skis and the snow, i.e., no speed is lost due to air drag, moving skis or gravity reasons. A photograph of the actual setup is shown in Figure 3.2.

Assumptions in the friction test are that the air drag is negligible since the skier is in the tucked position and that the test is conducted on flat ground such that gravity has little or no impact on the friction coefficient. Moreover, it is assumed that the skier is otherwise stable such that the loss of speed of the test skier in this setup is only due to the friction between skis and the snow. Five runs were done from one side, then five test runs from the other side. The reason for this was to even out possible impact from gravity in the test

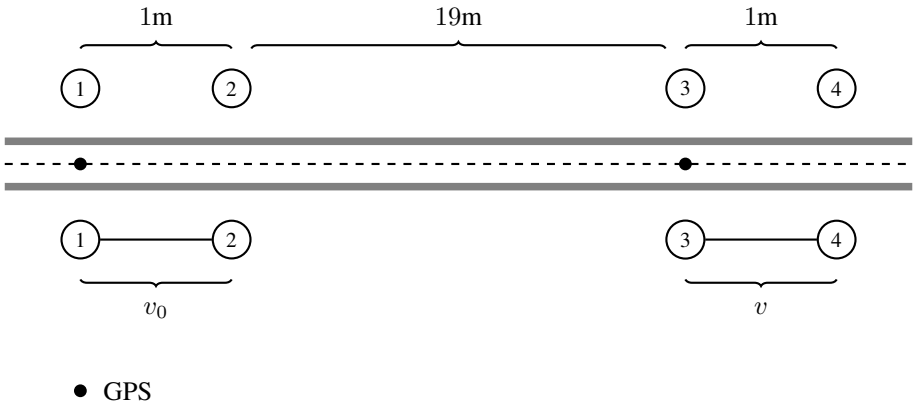

Figure 3.1: Schematic view of field test for estimation of the friction coefficient as seen from above.

and thus also in the computed result. One friction coefficient value was computed from each run, ten in total, five from each side. A mean was calculated for each run, and from these one overall mean was calculated representing the condition on the day of the skiing competition.

The calculated value might not be precisely accurate for the conditions and especially not correct for all times and positions in the track, but this estimate should be closer to the true value than another value taken from the literature. Estimated values are listed in Chapter 5.

Adjustments based on subtechnique

To make model calculations more accurate, there were made certain adjustments based on subtechnique. Certainly, the power used when skiing is different depending on whether the skier uses a skating technique, the diagonal stride tehcnique or is in the tucked position. The drag area C_dA is then different and thus also the drag force in the power balance model. Drag area was therefore implemented as a function of subtechniqe. Different values for drag area were used in calculations depending on whether the race was a classical race or whether the free skating technique was used.

Figure 3.2: Photograph of the setup for estimating the friction coefficient. To the right are the four photo cells in one end of the setup, used to estimate the velocity in one end of the test track.

For the classical races, the subtechnique used at each data point was determined by a classification algorithm. There were four subtechnique categories of the classification:

- Double poling ('DP') (1)
- Diagonal stride ('DIA') (2)
- Double poling with kick ('DPK') (3)
- Other (= tuck and turn) (0)

The classification was performed before the analysis of data in this thesis started, and was added as an additional feature to the data with labels $0, 1, 2, 3$, determining what drag area value to use. For the free skate sprint races no such classification was available at the time this thesis was written. Nevertheless, a simplified classification based on speed was used instead. If the speed was higher than 10 meters per second, the skier was assumed to be in the tucked position, and then a certain value for the drag area was assigned. If the speed was lower than 10 meters per second, the skier was assumed to be using any other more or less upright skate skiing technique. The value assigned at this lower part of the speed scale is higher than the one for high speeds, since the projected frontal area of the skier is larger in the upright position than in the tucked position. All drag area values connected to each subtechnique were found in relevant literature.

Drag area values assigned to each subtechnique classified in the classical races were found in the article by (Ainegren and Jonsson (2018)). This paper describes a study investigating air drag, frontal area and coefficient of drag by letting a highly skilled skier simulating skiing like techniques on a force plate in a wind tunnel. The resulting calculated values for the drag area were used in the implementation of the power balance model.

3.3 Collecting data

Only a limited number of skiers were collected data from. These skiers were mainly elite skiers from the Norwegian national team who all gave their consent to participation in the data collection. All skiers were equipped with an GPS unit (Catapult OptimEye S5 (2019)) placed in an attached pocket in the back of the race bib. When wearing the bib, the GPS unit was then positioned approximately at shoulder height, right between the shoulder blades. The GPS were placed in the pocket right before the starting point in the interval start and collected by the finish line right after the race when data had been collected. Data was then extracted from the units and processed for further analysis later. In particular was three-dimensional data for position projected into the race track, giving data where position is measured as distance traveled in the track. The processed data is the base for the calculations and analyses in this thesis.

3.4 Other assumptions and considerations

The skiing equipment used and grooming of skis is assumed to be of same quality for all the competing athletes, as they all try to choose the best skis and prep for the conditions on the race day. Skis used in the friction test were also of same quality level as the athlete's skis trying to eliminate any possible error due to use of differing equipment. There is assumed to be no difference in the equipment of the competing athletes that will cause differences in power calculations.

The mass m in the model is the mass of skier and equipment combined. Since the mass of additional equipment affects the power usage when skiing, 3 kg is added to the body mass of the skier before further analysis. The power balance model also assumes that the skier and the equipment is a point mass and that one observes the movement from the center of mass (COM) of the object in motion. The COM of the skier is located closer to the hip than the shoulders in the vertical direction if seen in the sagittal plane, the plane that divides

the human body into left and right sections. The exact position of the center of mass, can bee seen in the master thesis of Øyvind Nøstdal Gløersen (Gløersen (2014), p. 40). The data collecting unit is however positioned away from the COM and closer to the neck and is therefore causing slight measurement error. The combination of this systematic error in addition to any other noise registered by the unit, is attempted filtered away by smoothing techniques. These techniques remove the high-frequent noise and keep the underlying trend of the data, which is crucial in order to obtain as true results for power as possible.

Also the distance between the COM and the unit changes as the skiers move. The quality of the GPS is good and can detect typical instantaneous speed differences in cross-country ski racing (Gløersen et al. (2018a)), but the sensitivity opens up for possible unwanted high-frequency noise, especially in the case of speed. Speed changes rapidly as the skiers move. Natural considering the placement of the GPS and for instance skiing using double poling. Detected speed of the GPS increases when using the poles to push back, and decreases after the push until the movement is repeated. This causes a zig-zag pattern when plotting speed as a function of distance traveled in the track. Since the GPS is located on the upper back, the GPS detects mainly the movement and speed of the upper back up the skier rather than the speed and movement of the center of mass. This highlights the importance of noise removal or filtering of the data before calculations, described further in the chapter on data processing. Further preprocessing of the data before the sensitivity analysis, such as calculating the slope angle of the track and calculating the acceleration, will also be described in more detail in the next chapter. The systematic error due to location of the GPS as well as other present high-frequency noise from the measurements motivated use of filtering techniques.

After collecting the data, further processing of the data was needed before calculating propulsive power. This work will be presented in further detail in Chapter 4, though some reasoning for the chosen methods and their values in the analyses will be given here.

3.5 Statistical methods

In the smoothing splines method, the number of degrees of freedom was chosen, by visual inspection, such that the smoothed curves looked reasonably smooth.

For the sensitivity analysis, the following variables were chosen for testing: The friction coefficient μ, drag area C_dA, body mass m and air density ρ. The effect of each variable in the model was tested by calculating propulsive power by varying the variables one by one,

taking values from a vector of predetermined values. The values chosen for each variable covered what is found or known to be the best value for the variable and for the race conditions, found either by estimation or in literature. Additionally, some possible low and high values in the ends of this interval were added, to cover all possible conditions and to investigate what happens when values vary between values that are true and the values that are assumed to be true, even if some slight error is made in the calculation of parameters.

Testing values for μ covered the estimated values from the friction tests. Drag area testing values ranged from the low drag area values representing the tucked position, to the drag area values of a skier in an upright position where the drag area is considerably larger. Experimental values for the drag area for these positions were found in the article by Ainegren and Jonsson (2018). Body masses tested were the measured body mass including equipment and a slightly lower and slightly higher mass ($\pm 2kg$). Air density was set to $1.1^{\mathrm{kg}/\mathrm{m}^3}$ and also here a slightly smaller and a slightly larger value was tested. Lists of values and variations in analysis are found in Chapter 4.

For the power simulation, the same variables were tested, but following a Monte Carlo approach. Each variable was assigned a distribution, as follows

$$\theta_{par} \sim N(\mu_{par}, \sigma_{par}^2)$$

The normal distribution was chosen because it is easy to work with mathematically and in many cases models natural phenomena well. It also a reasonable first choice if it is not clearly known that the distributions follow another distribution.

Means μ_{par} were chosen to be the estimated values from experiments or from the literature, similar to what was done in the analysis for each variable separately. Standard deviations σ_{par} were chosen to cover all reasonable values in the test setting, as well as some values bordering to the more unlikely, so that any uncertainty in the model input is accounted for.

The aim of the sensitivity analysis was to get a clearer view and a better understanding of how each variable and how each term affects the total response $Y = P_{prop}$ of the model. Then, the simulation approach will then give insights into how much variation in the variables will result in varying results for propulsive power curves.

Chapter 4

Data processing and analysis

Data from the data collection were preprocessed in MATLAB and then further processed and analyzed in R. Only processing and analysis in R is included in this thesis. Since a large proportion of the work with this thesis was spent processing the data further and setting up a framework for analysis, this will de described in more detail in the following.

Further processing of input data

Preprocessed data were loaded into R and relevant initial variable vectors were chosen and set up in data frame. Initial variables for each skier were time t, distance travelled in the track x, velocity v, elevation $elev$ and body mass m of skiers. An additional 3 kgs were added to the skiers body mass to account for the mass of equipment, such as skis, boots and poles. Note that exact body mass was not known for every single male skier at the time this thesis was written. In those cases, an average mass of 77 kg was assumed, which is an approximate average of the body mass of the male skiers. In addition to mass, the initial data included a vector of subtechnique classification, however only for the classical races. Additional labels such as gender, year and race discipline were added to the data frames for a better overview. In the cases where DGPS (differential GPS, accuracy down to $10 - 15$ cm) data was available for the track profile, this was used.

Smoothing of velocity and track profile

Since in particular the velocity detected by the GPS units includes noise and measurement error, the velocity is filtered to remove this noise. Plotting the velocity as function of distance traveled in the track confirms that the velocity data includes noise, the curve draws a zig-zag pattern where the data points alternate in being smaller and larger than the previous data point. There is however a general trend in the speed. The smoothing spline filtering aims to keep this trend, but to remove the noise and the corresponding zig-zag pattern. For the cases where DGPS data was not available (all 2018 data), the track profile was smoothed by smoothing splines.

The degrees of freedom were chosen such that the curves looked reasonably smooth, by visual inspection. Calculating acceleration based on a smoothed velocity also gives a much smoother acceleration, as well as a more correct calculation of the propulsive power. For velocity data, only the smoothed velocity is considered in calculations and analyses as this curve best represents the truth which we are interested in investigating.

Adjusted central differences: Acceleration and slope angle

From the inital loaded data, acceleration a was calculated by central differences. However, since the regular central differences method requires h data points on each side when approximating the derivative for a data point i, adjustments were needed in both vector ends. For the first h data points, the derivative was approximated by

$$f'(x) \approx \frac{f(x_{i+h}) - f(x_1)}{x_{i+h} - x_1}, \quad i = 1, ..., h \tag{4.1}$$

an amended version of central differences. The number of data available data points on the right side of index i will always be h while the number of available data points on the left hand side will increase from 0 to h. The total index difference between data points will therefore range from h to $2h$. A mirrored procedure is used in the end of the vector:

$$f'(x) \approx \frac{f(x_n) - f(x_{i-h})}{x_n - x_{i-h}}, \quad i = n - h + 1, ..., n \tag{4.2}$$

Here, the number of available data points on the left side of index i will always be h, while the number of available data points on the right hand side will decrease from h to 0. In the middle of the vector, the acceleration is calculated by the regular central differences

method as stated in the theory chapter. Combined, these three procedures approximates the derivative of the velocity.

The track slope was calculated by the same central differences procedure as the acceleration. The angle at each point of the race track was calculated by the inverse trigonometry formula described in the theory section.

The step size h was chosen such that a satisfactory smoothness of the data was obtained, while at the same time calculating the derivative using data points not too far apart. The minimum and maximum length between data points in the track was approximately 15 and 30 meters, respectively, for each race, and h was found correspondingly. h was chosen to be race specific and the same value was used for both angle and acceleration calculation within each race.

Subtechnique classification and drag area

Preprocessed data contained classification of subtechnique. From a vector containing values $0, 1, 2$ and 3 for the classical techniques, a script assigned the suitable drag area value (Ainegren and Jonsson (2018)) in a separate vector given as input to the power balance model. Category 0 includes that skiers are both tucking and turning. However, since the data shows that the speed in general is high when the subtechnique is classified into this category, we assume that the skiers are mostly in the tucked position, and hence a low value for the drag area is reasonable. The value is however not set too low, as the this classification category is not precise and skiers also use more upright skiing techniques within this category.

In the skating technique races, drag area values were assigned based on a speed determined classification, above or below $10^{\mathrm{m}}/_{\mathrm{s}}$, for tuck and the upright position, respectively. This classification is even more uncertain than category 0 in the classic races. Since speeds above $10\frac{m}{s}$ generally means skiing in the tucked position, parameter choices are based on this assumption. There is more uncertainty related to the upright cateogory, as this category includes a wider range of techniques with varying drag area, but an intermediate drag area value in the range of drag area values for skating techniques were chosen based on experimental values from Ainegren and Jonsson (2018).

Table 4.1: Example of data frame in R after preprocessing, ready for power calculations.

	skier	time	distance	speed	elevation	subtechnique	technique
1	HSprintP_40	0.00	0.00000000	8.109429	-0.02183887	1	classic
2	HSprintP_40	0.01	0.07708371	8.076857	-0.02176748	1	classic
3	HSprintP_40	0.02	0.15416741	8.044286	-0.02175719	1	classic
4	HSprintP_40	0.03	0.23125112	8.011714	-0.02180455	1	classic
5	HSprintP_40	0.04	0.30833482	7.979143	-0.02190637	1	classic
6	HSprintP_40	0.05	0.38541853	7.946571	-0.02205966	1	classic

...

	year	gender	lap	mass	angle	speedSmoothed	accSpeedSmoothed	mu
	2017	male	1	82	-0.0005728568	7.759799	-0.9422636	0.025
	2017	male	1	82	-0.0009141676	7.749632	-0.9240387	0.025
	2017	male	1	82	-0.0012435023	7.739842	-0.9059670	0.025
	2017	male	1	82	-0.0015614803	7.730424	-0.8880486	0.025
	2017	male	1	82	-0.0018686792	7.721374	-0.9790688	0.025
	2017	male	1	82	-0.0021656378	7.712686	-0.9474127	0.025

Friction

The friction coefficient μ was added in a separate column, where each race had their unique value for μ as estimated in the friction tests in 2018. The exception was 2017, where μ was set to 0.025.

An example of the first six rows of a data frame for the classic sprint in 2017 is presented in Table 4.1.

4.1 Power calculation and sensitivity analysis

After acceleration and slope angle had been calculated, drag area values had been assigned, the friction coefficient for each specific race and the air density had been summoned, all input was ready for power calculation. A model function took the data frame and the parameter vector as input and calculated the propulsive power, while it at the same time calculated the separate terms in the power balance model. Note that where negative propulsive power was calculated, the calculations set P_{prop} to 0 since it in practice does not make sense to use a negative amount of power. Additionally was the calculated power divided by the skiers mass, in order to look into the *relative power* - how much power is used relative to body mass. This is a more universal measure of power, as skiers with higher body mass would need to use more power to obtain the same speed as skiers with lower body mass would.

4.1.1 Each variable separately

Each parameter is tested by varying between the chosen parameter values and calculating the propulsive power. These values were chosen in the range that could reasonably represent changes in the parameter in a race. Three tests on the 2017 sprint data were conducted with different vectors of predetermined values with varying range, and relative power P_{rel} is calculated with varying parameter values, one parameter at a time. Test values for all tests are presented in Table 4.2. Note that only test 3 is presented in the results, the other two are found in the Appendix A.

Table 4.2: Vectors with predetermined values for sensitivity analysis for each variable on an uphill segment from the 2017 sprint data.

Values for sensitivity analysis	
Test 1	
Variable	*Vector of values*
Drag area	$C_dA = [0.20, 0.40, 0.60]$
Friction	$\mu = [0.020, 0.025, 0.030, 0.035]$
Body mass	$m = [m - 2, m, m + 2]$
Air density	$\rho = [0.9, 1.1, 1.3]$
Test 2	
Variable	*Vector of values*
Drag area	$C_dA = [0.30, 0.40, 0.50]$
Friction	$\mu = [0.0225, 0.025, 0.0275, 0.030]$
Body mass	$m = [m - 1, m, m + 1]$
Air density	$\rho = [1.0, 1.1, 1.2]$
Test 3	
Variable	*Vector of values*
Drag area	$C_dA = [0.40, 0.45, 0.50]$
Friction	$\mu = [0.0225, 0.025, 0.0275]$
Body mass	$m = [m - 1/2, m, m + 1/2]$
Air density	$\rho = [1.05, 1.1, 1.15]$

The tests calculated minimum and maximum difference between the calculated curves of calculated relative power for different parameter values, i.e.,

$$\max \Delta P_{rel} \qquad \text{and} \qquad \min \Delta P_{rel}$$

Then the maximum and minimum difference relative to a chosen reference was calculated in percent, as

$$\frac{\max \Delta P_{rel}}{P_{rel,\theta=\theta_{chosen}}} \qquad \text{and} \qquad \frac{\min \Delta P_{rel}}{P_{rel,\theta=\theta_{chosen}}}$$

Maximum and minimum difference in power and maximum and minimum difference in power relative to a reference was then found for anuphill segment, for chosen parameters, see the result chapter.

4.1.2 Monte Carlo simulation approach

The effect of changes in all variables simultaneously was modeled by simulation. A distribution was assigned to each variable with mean and standard deviation, and power calculated by drawing parameter values from these distributions.

Suitable estimates of the standard deviations were chosen based on the empirical rule of statistics, which states that asymptotically 95% of the samples drawn from a normal distribution will lie within two standard deviations of its mean, i.e. the interval $[\mu_\theta - 2\sigma_\theta, \mu_\theta + 2\sigma_\theta]$ covers 95% of samples drawn from $\theta \sim N(\mu_\theta, \sigma_\theta)$. As an example, the standard deviation of the friction coefficient was set to $\sigma_\mu = 0.0025$ as then 95% of drawn samples will be in the interval $[0.020, 0.030]$ if $\mu_{estimated} = 0.025$, which covers most estimated COFs and a reasonable uncertainty in this parameter given the environmental conditions.

Special care was given when setting the standard deviations for the drag area for categroy 0 (classical) and for skate. As previously mentioned, since category 0 is assumed to mostly contain tuck, a low standard deviation would be reasonable as the uncertainty in drag area is low (experimentally tested) given that this is the position (confirmed in Ainegren Johnson). However, since we do not know for sure how much other more upright positions are used in this categroy, this was a reason for increasing this chosen standard deviation.

A similar argument was used for drag area for skate, though this is a more uncertain classification than for the classic races since it is only based on speed. Furthermore, the upright category for skate contains a range of different skating techniques. The standard deviation was therefore chosen such that 95% of sampled values fell within the range of experimental drag area values for skating techniques tested in Ainegren and Jonsson (2018). Chosen means and standard deviations are presented in Table 4.3.

For simulation of propulsive power, N simulations were run for each race and compared to the calculated power vector with 'true' parameters $\mathbf{P}_{rel,true}$. The true propulsive power $\mathbf{P}_{rel,true}$ was subtracted from the power vector $\mathbf{P}_{rel}$ of simulation j, and this difference was divided by the true propulsive power, for each simulation $j = 1, ..., N$ to find the elementwise deviation of simulations relative to what is assumed to be the true calculation.

$$\frac{\mathbf{P}_{rel,j} - \mathbf{P}_{rel,true}}{\mathbf{P}_{rel,true}} \tag{4.3}$$

This calculation was done for one uphill segment in each race, the lengths of the segments varying from 150 to 200 meters. Additionally, 95% confidence intervals were found by sorting all absolute relative differences for all data points for all skiers within each race, and disregarding the lowest 2.5% and highest 2.5% values.

Distributions for Monte Carlo simulation

C_dA **Classical**		
Subtechnique	*Distribution*	95% *intervals*
0: Other (= tuck/turn)	$\sim N(0.23, 0.025^2)$	$[0.18, 0.28]$
1: Double poling	$\sim N(0.44, 0.025^2)$	$[0.34, 0.54]$
2: Diagonal stride	$\sim N(0.54, 0.025^2)$	$[0.44, 0.64]$
3: Double poling with kick	$\sim N(0.46, 0.025^2)$	$[0.36, 0.56]$

(a) Drag area value distributions, classical.

C_dA **Skate**		
Subtechnique	*Distribution*	95% *intervals*
Upright	$\sim N(0.50, 0.075^2)$	$[0.35, 0.65]$
Tuck	$\sim N(0.23, 0.025^2)$	$[0.18, 0.28]$

(b) Drag area value distributions, skate.

Other			
Variable		*Distribution*	95% *intervals*
Mass	m	$\sim N(m_{skier}, 1^2)$	$[80, 84]$ (if $m_{skier} = 82$)
Air density	ρ	$\sim N(1.1, 0.1^2)$	$[0.9, 1.3]$
Friction coefficient	μ	$\sim N(\mu_{estimated}, 0.0025^2)$	$[0.020, 0.030]$ (if $\mu = 0.025$)

(c) Distributions for other parameters.

Table 4.3: Overview of distributions with chosen means and standard deviations for all paramters when sampling in the Monte Carlo simulation.

Chapter 5

Results

In this section, results from power calculations, sensitivity analysis and Monte Carlo simulation will be presented as well as results from friction coefficient estimation. Note that relative propulsive power P_{rel}, propulsive power relative to body mass with unit $^{\mathrm{J}}/_{\mathrm{s}\cdot\mathrm{kg}} = {}^{\mathrm{W}}/_{\mathrm{kg}}$, is used in the subsequent analyses to make the results as general and comparable as possible. Unless otherwise stated, we always mean power relative to body mass when talking about propulsive power in the following.

5.1 Friction coefficient estimation

Example of raw data from the field test for estimation of coefficient of friction.

Table 5.1: Raw data from field test for estimation of the friction coefficient μ.

Test X	**Time (s)**	**Δt**	**Distance** m	**Friction** μ
1st passing time	0.264		1	
Time between points	6.176	6.542	20	
Last passing time	0.467		1	0.0257
Total time	6.902			

Table 5.2: Estimated friction coefficients, results.

Friction coefficient estimation				
Date	16.11.18	16.11.18	17.11.18	18.11.17
Time	09:05	14:49	10:15	No data
Air temperature				
Snow temperature, $^{\circ}C$ (time)	-0.6 (08:35)	-0.6 (14:51)	-3.4 (10:50)	No data
Race	10/15 km classic	10/15 km classic	Free/skate sprint prologue	Sprint prologue classic
μ one way average	0.0242	0.03097	0.0225	No data
Other way average	0.0267	0.0319	0.02743	No data
Total average	0.0255	0.0314	0.02499	Set to 0.025 in calculations (No data)

5.2 Sensitivity analysis

The change in calculated propulsive power given changes given the input parameters depends on the size of each model term. If a parameter in a term is changed in the model with a large absolute value relative to the other terms, this parameter change will increase or decrease P_{prop} more than if a parameter is changed in a model term with smaller absolute value. The impact of change is larger, the larger the absolute value of the term is. The value of each model term for a selected skier from 2017 is presented in Figure 5.1 and gives an idea of how which terms that can lead to the most substantial changes in P_{prop} as parameter values are varied.

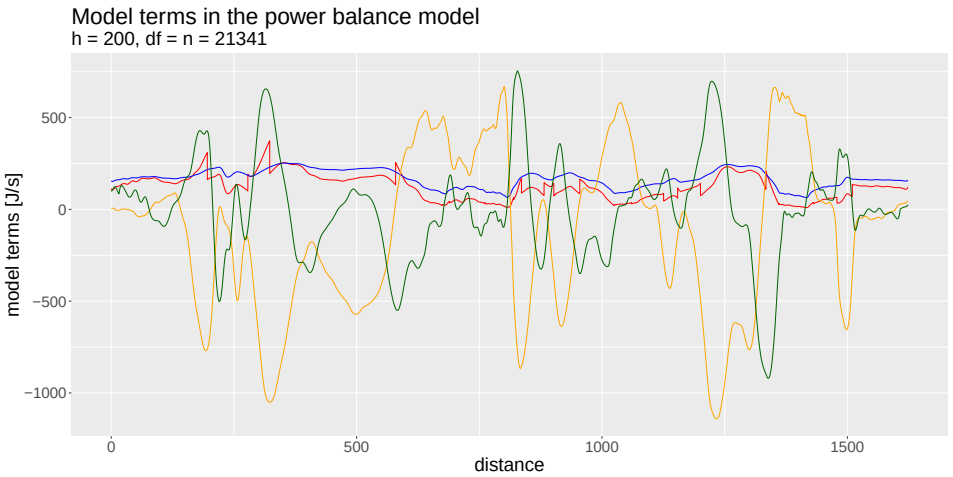

Figure 5.1: Model terms of the power balance model. **Red curve**: Drag, **blue curve**: Friction, **orange curve**: Gravity and **green curve**: Term with acceleration.

Figure 5.1 shows that the two terms with largest values in absolute value is the gravity term and the acceleration term. Note that both these terms include variables that require calculation by the central differences method, the incline angle in the gravity term and the acceleration a in the acceleration term. However, the friction term also requires use of central differences for the incline angle, though the size of the acceleration term in particular is highly dependent on the smoothness and size of the speed.

Investigating an uphill segment

An uphill segment from the 2017 classical sprint was investigated and propulsive power was calculated with varying parameters. Maximum and minimum absolute difference

in relative power P_{rel} (`max.diff` and `min.diff`) and maximum and minimum absolute difference (`rel.max` and `rel.min`, respectively) relative to a chosen reference was computed in three different tests. Only test 3, with the smallest variation in parameters, is shown here. Tests 2 and 3 are found in Appendix A.

Drag area C_dA

Testing relative difference in drag area C_dA with test values $C_dA = [0.40, 0.45, 0.50]$. The reference is $C_dA = 0.45$.

Test 3

Table 5.3: Table with absolute maximum and minimum and absolute relative difference in $\%$ of propulsive power with varying C_dA for an uphill segment. $C_dA = [0.40, 0.45, 0.50]$, reference $C_dA = 0.45$

```
> tab.drag
         skier max.diff min.diff rel.max rel.min
1 HSprintP_40    0.620   0.0618    18.6   0.764
2 HSprintP_47    0.519   0.0460    20.1   0.616
3 HSprintP_46    0.659   0.0239    18.4   0.383
4  HSprintP_4    0.515   0.0342    18.5   0.476
5 HSprintP_30    0.551   0.0379    19.5   0.542
6 HSprintP_23    0.555   0.0375    16.1   0.541
7 HSprintP_20    0.613   0.0408    19.5   0.565
8 HSprintP_19    0.571   0.0344    19.2   0.480
9 HSprintP_10    0.478   0.0428    18.7   0.551
```

The relative difference ranges from 0.4% to 20%, a quite wide range with a high maximum relative difference. This difference is however related to uncertainty in classification of subtechnique. Additionally, variation can be explained by that the difference betweem minimum and maximum tested value for C_dA is 0.10, which is a $22,2\%$ change relative to the reference value. Since also the model is linear in all its terms and parameters except for the cubed velocity in the drag term, potential inaccuracies and changes in velocity has greater impact in this term of the power balance model.

Friction coefficient

Test for relative difference of the friction coefficient with test values $\mu = [0.0225, 0.025, 0.0275]$. The reference is the estimated value, $\mu = 0.025]$.

Test 3

Table 5.4: Table with absolute maximum and minimum and absolute relative difference in $\%$ of propulsive power with varying μ for an uphill segment. $\mu = [0.0225, 0.025, 0.0275]$, reference $\mu = 0.025]$.

```
> tab.mu
        skier max.diff min.diff rel.max rel.min
1 HSprintP_40    0.417    0.189    12.7    2.33
2 HSprintP_47    0.398    0.174    15.6    2.32
3 HSprintP_46    0.424    0.137    12.0    2.20
4  HSprintP_4    0.397    0.157    14.4    2.19
5 HSprintP_30    0.396    0.159    14.2    2.26
6 HSprintP_23    0.394    0.157    11.5    2.26
7 HSprintP_20    0.409    0.162    13.2    2.23
8 HSprintP_19    0.412    0.158    14.1    2.20
9 HSprintP_10    0.384    0.168    15.2    2.16
```

The relative difference ranges from 2.2% to 16%, also a quite wide range. In this test the difference between minimum and maximum tested value for μ is as low as 0.0050, though this is a change of 20% relative to the reference value $\mu = 0.025$. Additionally, the power calculations can be affected by the calculation of the incline angle in the same term, even though the absolute size of this model term is not large compared to the others (See Figure 5.1) and changes in parameters in the friction term therefore do not have as high impact on the propulsive power calculations.

Body mass

Test for relative difference of body mass m with test values $[m - 1/2, m, m + 1/2]$. The reference is the measured body mass m.

Test 3

Table 5.5: Table with absolute maximum and minimum and absolute relative difference in $\%$ of propulsive power with varying body mass of the skiers, m (± 0.5 kgs), for an uphill segment of the 2017 sprint race. Test values: $[m - 1/2, m, m + 1/2]$, reference is each skier's body mass m.

```
> tab.mass
        skier max.diff min.diff rel.max rel.min
1 HSprintP_40   0.0222  0.00271   0.673  0.0334
2 HSprintP_47   0.0179  0.00195   0.703  0.0260
3 HSprintP_46   0.0239  0.00106   0.674  0.0170
4  HSprintP_4   0.0178  0.00145   0.645  0.0201
5 HSprintP_30   0.0205  0.00173   0.735  0.0246
6 HSprintP_23   0.0212  0.00175   0.620  0.0252
7 HSprintP_20   0.0231  0.00189   0.744  0.0260
8 HSprintP_19   0.0195  0.00144   0.665  0.0200
9 HSprintP_10   0.0169  0.00186   0.670  0.0238
```

The relative difference ranges from 0.02% to 0.74%, a quite small range. The difference between minimum and maximum tested value for m is only 1 kg, only being about 1.2% of the reference mass of 82 kgs. This means that if the measured mass of a skier and the equipment is within an uncertainty range of $\pm 1/2$ kg, the maximum absolute relative difference is less than 1%. Even for the test where the body mass was varied within an uncertainty range of ± 2 kgs, the maximum absolute relative difference was about 3% at most.

Air density

Test for relative difference of ρ with test values $\rho = [1.05, 1.1, 1.15]$. The reference is $\rho = 1.1$.

Test 3

Table 5.6: Table with absolute maximum and minimum and absolute relative difference in $\%$ of propulsive power with varying ρ for an uphill segment for the 2017 sprint race. Test values $\rho = [1.05, 1.1, 1.15]$, reference $\rho = 1.1$.

```
> tab.air
        skier max.diff min.diff rel.max rel.min
1 HSprintP_40    0.165  0.02022    6.56   0.259
2 HSprintP_47    0.138  0.01504    5.49   0.193
3 HSprintP_46    0.176  0.00782    6.98   0.100
4  HSprintP_4    0.137  0.01118    5.45   0.143
5 HSprintP_30    0.147  0.01240    5.83   0.159
6 HSprintP_23    0.148  0.01227    5.88   0.157
7 HSprintP_20    0.163  0.01337    6.49   0.171
8 HSprintP_19    0.152  0.01127    6.05   0.145
9 HSprintP_10    0.127  0.01400    5.05   0.180
```

The relative difference of ranges from 0.1% to 7%. The difference between minimum and maximum tested value for ρ is 0.1, about 9% change relative to the reference.

5.3 Monte Carlo approach

Calculated power with estimated and set values is considered the 'true' power curve. The difference between simulated power and true power relative to true power, in percent, for one uphill segment in each race, as well as 95% confidence intervals (CIs) for these deviations are presented in Table 5.7.

Table 5.7: 95% confidence intervals for absolute relative deviation from P_{prop} with true parameters, from N Monte Carlo simulations of a selected uphill segment in each race.

Race	*CI: Deviation in* $\%$	*Simulations*
Sprint (M), 2017	$[0.0675, 7.723]$	$N = 100$
Sprint (W), 2018	$[0.0821, 10.984]$	$N = 200$
Sprint (M), 2018	$[0.0825, 6.863]$	$N = 200$
10 km (W), 2018	$[0.0572, 10.495]$	$N = 50$
15 km (M), 2018	$[0.0382, 3.973]$	$N = 10$

The Monte Carlo simulations show that after having sorted the absolute values of the relative deviations in an uphill segment in each race (in $\%$), and removed the largest 2.5% values, simulated races varies at most 11% from the race with 'true' parameters. Deviations for all data points and all skiers are considered together, giving a total of $n \cdot$ No. of laps $\cdot N$ deviations in the results for each race. (Number of laps for sprint races is equal to one.)

Figure 5.2 illustrates the relative deviations from the propulsive power calculation with 'true' parameters for an uphill segment in the 10 km classic race for women in 2018. The deviations from P_{prop} with 'true' parameters within a 95% confidence interval are not larger than 8% in absolute value and are centered around zero.

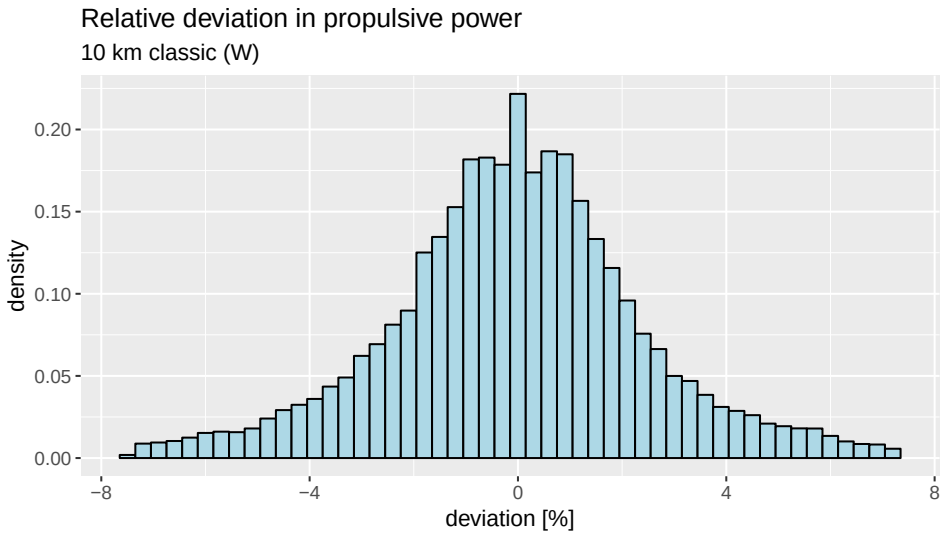

Figure 5.2: Relative difference (in %) of propulsive power within a 95% confidence interval for the simulations of an uphill segment from the 10 km classic race (W) 2018.

Chapter 6

Discussion

The accuracy in measurements, choice of methods and parameters as well as their corresponding errors are important factors when investigating the power balance model for better understanding of cross-country skiing in a race setting. There are numerous sources of uncertainty already starting in the preprocessing of the data. The measurement units used for collecting data in this thesis is highly accurate (Gløersen et al. (2018a)), though also sensitive and register the movements of the upper back where the GPS is located rather than the movement of the COM of the skier. Though improvements with regards to positioning of this GPS, so that it can measure the movement of the COM, is tractable, it is not done for the data analyzed at this point. As therefore filtering techniques must be used to provide data to the model that more accurately represents the truth, a new issue presents itself when having to choose both method and the method parameters that follow.

Filtering method

Though the choice of smoothing splines as a method for removing high-frequency noise is justified (see theory sectiob er add references again), the smoothing parameter must still be chosen such that noise is removed yet does not remove any of the underlying truth in the data. The effective degrees of freedom - a function of the flexibility of the fitted curve - was here chosen mainly by visual inspection of the resulting curve for the smoothed speed, and what curve was deemed to reasonably represent the speed of the COM of the skier in motion. However, a more quantitative approach would be to use cross-validation to find

the best smoothing parameter minimizing Equation (2.4).

Choice of number of knots and their positioning is not an issue in smoothing splines, as the number of knots is equal to n, positioned at the n data points. However, other similar smoothing and regression techniques allow for this, and can be evaluated for instance in sections of the data where a higher (or lower for that matter) accuracy is wanted, for instance in an uphill segment, a downhill segment of a segment over a hill top. An example is regression splines, upon which the smoothing splines technique is based. A higher number of knots and small spacing will give a more flexible fit, whereas fewer knots with larger spacing will result in a smoother curve. The number of knots for a specific data set can also be found by cross-validation.

Note also that the definition of smoothing splines in this thesis is slightly different from the one used by Gløersen (2019) and that the weights w_i in the loss function were all set to one instead of using weighting for adjusting for measurement accuracy.

Central differences and choice of h

In the central differences method used in this thesis for calculation of the derivative of speed, a key issue is finding a suitable step size h. One wants to find an h that leaves the acceleration sufficiently smooth while at the same time representing the actual acceleration. Choosing h too small will include the sensitivity of the velocity data in the calculated acceleration, i.e. if the function for speed is jumpy, so can the acceleration be with a small choice of h. A small step size, means that the difference in value between two data points is small, especially if the data are collected with a high sampling frequency. Consequently, the denominator in the central differences scheme will be small, and dividing by a number much smaller than the numerator can give a unaccurately high acceleration and vice versa, resulting in a acceleration that also has a zig-zag pattern. This again highlights the importance of sufficient smoothing of the velocity, though a thought is also to consider smoothing the acceleration. The same arguments above are applicable for calculating the incline angle in the track by the same method.

An example of step size choice is $h = 200$, which for the 2017 classic sprint race meant using points for calculation that were 30 meters apart (from 15 to 30 meters apart in the beginning and the end of the race). The chosen step size was $1 - 2\%$ of the total race track length. By visual inspection, this step size seemed to give a descent result for the acceleration and the angle. A too large step size would mean comparing data points that are too far apart from each other in the race track to make sense.

Classification of subtechnique

Implementing the drag area as a function of subtechnique is definitely a step in the direction of more accurate calculations of propulsive power. The classification of subtechniques by accelerometer data and gyroscope data (collected by IMUs) in the classical races is quite accurate, whereas the the skating classification is only based on a speed assumption and category 0 in the classical races includes both tucking ang turning.

Firstly, by only using the speed assummption for skate, one is not able to use the true drag area for a data point since the exact subtechnique is not known. Furthermore, one is not sure if the skier is only tucking for speeds larger than $10^{\mathrm{m}}/_{\mathrm{s}}$ and using upright skiing techniques for speeds lower than that, and if so, which ones. An improvement for skate would therefore be to have classification data available.

As for the classical races, the uncertainty mostly lies within category 0. Since the speed is high within this category as a whole, it makes sense to choose a low drag area value, but this could be investigated in more detail by possibly filtering over the speed for each data point, choosing one drag area value for the highest speeds (where the skiers are the most likely to be in the tucked position) and another value for C_dA for the lower speeds. The main uncertainty is not knowing exactly what subtechniques are used by the skiers in which category, thus complicating the choice of values for C_dA, both the mean and the standard deviation for distributions in the Monte Carlo simulation.

Size of model terms

The model terms, as shown in Figure 5.1, vary greatly in size and value. Expecially the acceleration term and the gravity term which require calculation by central differences, which again is highly influenced by the choice of h. It is however interesting to note that also the friction term requires central differences for calculation of angle, but that the variation in this term is low compared to the gravity and acceleretaion term. The reason for this can be that the friction term is multiplied by μ, a factor of size $\sim 10^{-2}$, which makes this term about fourty times smaller than the very similar gravity term in the model (the only change between them is the friction coefficient). The effect of central differences is thus downsized.

As one is not certain to have removed all noise from the speed, this could also be a reason for change in the model terms. However, as all model terms include the speed, one can argue that this is of less importance. The exception is the drag term, where the speed is

cubed. Thus, local variations can be magnified and contribute to even more to changes.

Relative differences in sensitivity analysis

The relative difference results in the analysis quantifies how much a change in a single parameter changes the calculated propulsive power. By the results, one is able to see which parameter contributes to the least and the most change, in this case the body mass m and the drag area C_dA, respecitvely. It is however worth mentioning that this change in propulsive power is dependent on the size of the change of the parameter relative to the size of the parameter itself. As stated in the Chapter 5, a change of ± 2 kgs in mass from the body mass gives maximum a relative change of 3% in propulsive power, but note that this relative parameter variation is low, since 4 kgs out of 82 kgs (body mass of skier in the tests presented) is only 4.9%. In comparison, a change of ± 0.2 in drag area value gives a maximum relative change of 57.4%. Note however in this case that this relative parameter variation is a lot higher, as 0.2 out of 0.4 is 50%, ten times higher than for the body mass.

Monte Carlo approach

The idea behind the analysis of the Monte Carlo simulations was to attempt quantify how much error one can possibly make when there is uncertainty in the estimation or choice of model parameters. The lower the absolute difference between between the 'true' and the simulated power curve relative to the power curve is for each skier, the more precise and trustworthy is the calculated propulsive power. If the calculations for power are precise, it means that the data we have to calculate propulsive power by the power balance model are precise enough to investigate differences between athletes and differences between races. To be more sure of the of the variations in the simulated races, a larger number of simulations should be considered.

Viewing the simulated values differently, one could have found the difference of all skiers relative to a reference skier, in each race. If this difference does not vary much in the simulations, this suggests that the parameters we have used, and within the chosen uncertainty, are precise enough to detect differences between athletes.

Additional comments

Only uphill segments have been investigated in this thesis, but it would also be interesting to look at other track segments such as downhill segments, segments with little or no incline angle or segments covering hill tops. Examples of this as well as a scientific view

on cross-country skiing can be seen in the article and the TV episode about Didrik Tønseth in the series 'Vitenskapen bak medaljen' (NRK (2019)) on NRK.

Additionally would it be interesting to investigate the differences of calculated propulsive power with and without DPGS data. Also, a completely different interesting approach would be to analyze the data as a time series.

Chapter 7

Conclusion and further work

Propulsive power has been calculated, a sensitivity analysis has been performed and Monte Carlo simulations produced in an uphill segment of all five races investigated in this thesis. The sensitivity analysis showed that the body mass was the parameter that percentwise gave the smallest change in propulsive power when varying the parameters separately within reasonable ranges for which each parameter can vary in race conditions. The drag area gave the largest percentwise change in propulsive power. In total, and in decreasing order; Drag area, the friction coefficient, air density, the body mass are the most sensitive for changes in calculation of power using the power balance model. The sensitivity is highly dependent on how much the parameter varies relative to its estimated value.

The Monte Carlo simulations showed that the maximum relative deviation in absolute value of all races was 11% in the investigated uphill segments. More simulations should be run in order to quantify this deviation more accurately and support the findings in this thesis.

Since the sensitivity analysis showed that the power balance model is most sensitive for changes in the drag area, it is important for future research to improve classification of subtechniques in order to use the correct drag area value for each data point. This is especially important for skate, where the techniques are divided into only two categories, which are divided by an assumption of speed rather. It would be desirable to also classify the skating subtechniques by gyroscope and accelerometer data as is done for the classical races investigated in this thesis.

Bibliography

Ainegren, M., Jonsson, P., 2018. Drag area, frontal area and drag coefficient in cross-country skiing techniques .

Andersson, E., Supej, M., Sandbakk, Ø., Sperlich, B., Stöggl, T., Holmberg, H.C., 2010. Analysis of sprint cross-country skiing using a differential global navigation satellite system. European Journal of Applied Physiology 110, 585–595. doi:10.1007/s00421-010-1535-2.

Bartlett, R., 2007. Introduction to sports biomechanics: Analysing human movement patterns.

Braghin, F., 2016. The engineering approach to winter sports. Springer.

Budde, R., Himes, A., 2017. High-resolution friction measurements of cross-country ski bases on snow. Sports Engineering 20, 299–311. doi:10.1007/s12283-017-0230-5.

Catapult OptimEye S5, c., 2019. Optimeye s5 specifications. URL: https://images.catapultsports.com/wp-content/uploads/2017/07/OptimEye-S5.pdf.

Colbeck, S., 1994. A review of the friction of snow skis. Journal of Sports Sciences 12, 285–295. URL: https://doi.org/10.1080/02640419408732174, doi:10.1080/02640419408732174, arXiv:https://doi.org/10.1080/02640419408732174. pMID: 8064975.

Gløersen, Ø., Kocbach, J., Gilgien, M., 2018a. Tracking performance in endurance racing sports: Evaluation of the accuracy offered by three commercial gnss receivers aimed at the sports market. Frontiers in Physiology 9. doi:10.3389/fphys.2018.01425.

Gløersen, Ø., Losnegard, T., Malthe-Sørenssen, A., Dysthe, D.K., Gilgien, M., 2018b. Propulsive power in cross-country skiing: Application and limitations of a novel wearable sensor-based method during roller skiing. Frontiers in Physiology 9, 1631. URL: https://www.frontiersin.org/article/10.3389/fphys.2018.01631, doi:10.3389/fphys.2018.01631.

Gløersen, Ø.N., 2014. Quantitative technique analysis in XC-skiing. Master's thesis. University of Oslo. COM location on p. 40.

Gløersen, Ø.N., 2019. On the bioenergetics of cross-country skiing.

Hasler, M., Schindelwig, K., Mayr, B., Knoflach, C., Rohm, S., Putten, J.V., Nachbauer, W., 2016. A novel ski–snow tribometer and its precision. Tribology Letters 63. doi:10.1007/s11249-016-0719-2.

Hastie, T., Tibshirani, R., Friedman, J.H., 2017. The elements of statistical learning: Data mining, inference and prediction. Springer.

Hausken, K., Moxnes, J.F., Sandbakk, Ø., 2014. Using the power balance model to simulate cross-country skiing on varying terrain. Open Access Journal of Sports Medicine , 89–98doi:10.2147/oajsm.s53503.

van Ingen Schenau, G.J., Cavanagh, P.R., 1990. Power equations in endurance sports. Journal of Biomechanics 23, 865–881. doi:10.1016/0021-9290(90)90352-4.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2017. An introduction to statistical learning: With applications in R. Springer.

NRK, 2019. Vitenskapen bad medaljen. URL: https://www.nrk.no/vitenskapen-bak-medaljen-_-didrik-tonseth-1.14405976. (Didrik Tønseth).

Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M., 2004. Sensitivity analysis in practice.

Skaloud, J., Gontran, H., Merminod, B., 2004. Gsm-distributed rtk for precise analysis of speed skiing .

Skaloud, J., Limpach, P., 2003. Synergy of cd-dgps, accelerometry and magnetic sensors for precise trajectography in ski racing .

Swarén, M., Eriksson, A., 2017. Power and pacing calculations based on real-time locating data from a cross-country skiing sprint race. Sports Biomechanics , 1–12doi:10.1080/14763141.2017.1391323.

Young, H.D., Freedman, R.A., 2012. University physics. Pearson Education.

Appendices

Appendix A provides an overview of scripts made and used in this thesis as well as the actual code. Appendix B includes tables for calculations not presented in the result chapter.

Appendix A

Sensitivity analysis: Additional tables

Men's sprint 2017 (classic)

Drag area: Test 1

Table A.1: Minimum and maximum absolute difference and maximum and minimum absolute relative difference in percent of propulsive power with varying C_dA for an uphill segment of the 2017 sprint. Test values $C_dA = [0.2, 0.4, 0.6]$, reference $C_dA = 0.4$.

```
> tab.drag
          skier max.diff min.diff rel.max rel.min
1 HSprintP_40     1.65   0.1647    52.8    2.04
2 HSprintP_47     1.38   0.1226    57.4    1.65
3 HSprintP_46     1.76   0.0637    52.2    1.02
4  HSprintP_4     1.37   0.0911    52.5    1.27
5 HSprintP_30     1.47   0.1010    55.7    1.45
6 HSprintP_23     1.48   0.1000    45.4    1.45
7 HSprintP_20     1.63   0.1089    55.7    1.51
8 HSprintP_19     1.52   0.0919    54.8    1.28
9 HSprintP_10     1.27   0.1141    53.2    1.47
```

Drag area: Test 2

Table A.2: Minimum and maximum absolute difference and maximum and minimum absolute relative difference in percent of propulsive power with varying C_dA for an uphill segment of the 2017 sprint. Test values $C_dA = [0.3, 0.4, 0.5]$, reference $C_dA = 0.4$.

```
> tab.drag
       skier max.diff min.diff rel.max rel.min
1 HSprintP_40    0.827   0.0824    26.4   1.021
2 HSprintP_47    0.692   0.0613    28.7   0.823
3 HSprintP_46    0.879   0.0318    26.1   0.511
4  HSprintP_4    0.687   0.0455    26.2   0.636
5 HSprintP_30    0.735   0.0505    27.9   0.724
6 HSprintP_23    0.741   0.0500    22.7   0.723
7 HSprintP_20    0.817   0.0545    27.8   0.754
8 HSprintP_19    0.762   0.0459    27.4   0.640
9 HSprintP_10    0.637   0.0570    26.6   0.735
```

Friction: Test 1

Table A.3: Table with absolute maximum and minimum difference and minimum and maximum absolute relative difference in % of propulsive power with varying μ for an uphill segment of the 2017 sprint. $\mu = [0.020, 0.025, 0.030, 0.035]$, reference $\mu = 0.025]$.

```
> tab.mu
         skier max.diff min.diff rel.max rel.min
1 HSprintP_40     1.25    0.568    38.0    6.99
2 HSprintP_47     1.19    0.521    46.8    6.95
3 HSprintP_46     1.27    0.412    35.9    6.59
4  HSprintP_4     1.19    0.471    43.2    6.56
5 HSprintP_30     1.19    0.476    42.7    6.79
6 HSprintP_23     1.18    0.471    34.6    6.77
7 HSprintP_20     1.23    0.486    39.5    6.70
8 HSprintP_19     1.24    0.475    42.2    6.59
9 HSprintP_10     1.15    0.504    45.7    6.47
```

Friction: Test 2

Table A.4: Table with maximum and minimum absolute and relative difference in % of propulsive power with varying μ for an uphill segment of the 2017 sprint. $\mu = [0.0225, 0.025, 0.0275, 0.030]$, reference $\mu = 0.025]$

```
> tab.mu
         skier max.diff min.diff rel.max rel.min
1 HSprintP_40    0.625    0.284    19.0    3.49
2 HSprintP_47    0.596    0.260    23.4    3.48
3 HSprintP_46    0.636    0.206    18.0    3.29
4  HSprintP_4    0.595    0.236    21.6    3.28
5 HSprintP_30    0.594    0.238    21.3    3.39
6 HSprintP_23    0.590    0.235    17.3    3.39
7 HSprintP_20    0.613    0.243    19.8    3.35
8 HSprintP_19    0.618    0.237    21.1    3.29
9 HSprintP_10    0.576    0.252    22.8    3.23
```

Mass: Test 1

Table A.5: Table with maximum and minimum absolute and absolute relative difference in % of propulsive power with varying body mass m ($\pm$2 kgs) for an uphill segment of the 2017 sprint.

```
> tab.mass
        skier max.diff min.diff rel.max rel.min
1 HSprintP_40   0.0888  0.01086    2.69  0.1336
2 HSprintP_47   0.0716  0.00779    2.81  0.1041
3 HSprintP_46   0.0956  0.00425    2.70  0.0680
4  HSprintP_4   0.0711  0.00579    2.58  0.0805
5 HSprintP_30   0.0819  0.00691    2.94  0.0985
6 HSprintP_23   0.0847  0.00702    2.48  0.1010
7 HSprintP_20   0.0923  0.00755    2.98  0.1040
8 HSprintP_19   0.0780  0.00577    2.66  0.0801
9 HSprintP_10   0.0676  0.00743    2.68  0.0953
```

Mass: Test 2

Table A.6: Table with maximum and minimum absolute and relative difference in % of propulsive power with varying body mass m ($\pm$1 kgs) for an uphill segment of the 2017 sprint.

```
> tab.mass
        skier max.diff min.diff rel.max rel.min
1 HSprintP_40   0.0444  0.00543    1.35  0.0668
2 HSprintP_47   0.0358  0.00389    1.41  0.0520
3 HSprintP_46   0.0478  0.00212    1.35  0.0340
4  HSprintP_4   0.0355  0.00289    1.29  0.0402
5 HSprintP_30   0.0409  0.00345    1.47  0.0492
6 HSprintP_23   0.0423  0.00351    1.24  0.0505
7 HSprintP_20   0.0461  0.00377    1.49  0.0520
8 HSprintP_19   0.0390  0.00288    1.33  0.0400
9 HSprintP_10   0.0338  0.00371    1.34  0.0476
```

Air density: Test 1

Table A.7: Table with maximum and minimum absolute and absolute relative difference in % of propulsive power with varying ρ for an uphill segment of the 2017 sprint. $\rho = [0.9, 1.1, 1.3]$, reference $\rho = 1.1$

```
> tab.air
         skier max.diff min.diff rel.max rel.min
1 HSprintP_40    0.661   0.0809    20.1   0.995
2 HSprintP_47    0.553   0.0602    21.7   0.804
3 HSprintP_46    0.703   0.0313    19.9   0.500
4  HSprintP_4    0.549   0.0447    19.9   0.622
5 HSprintP_30    0.588   0.0496    21.1   0.707
6 HSprintP_23    0.592   0.0491    17.4   0.707
7 HSprintP_20    0.654   0.0535    21.1   0.737
8 HSprintP_19    0.609   0.0451    20.8   0.626
9 HSprintP_10    0.509   0.0560    20.2   0.718
```

Air density: Test 2

Table A.8: Table with maximum and minimum absolute and absolute relative difference in % of propulsive power with varying ρ for an uphill segment of the 2017 sprint. $\rho = [1.0, 1.1, 1.2]$, reference $\rho = 1.1$

```
> tab.air
         skier max.diff min.diff rel.max rel.min
1 HSprintP_40    0.331   0.0404    13.1   0.519
2 HSprintP_47    0.277   0.0301    11.0   0.386
3 HSprintP_46    0.352   0.0156    14.0   0.201
4  HSprintP_4    0.275   0.0224    10.9   0.287
5 HSprintP_30    0.294   0.0248    11.7   0.318
6 HSprintP_23    0.296   0.0245    11.8   0.315
7 HSprintP_20    0.327   0.0267    13.0   0.343
8 HSprintP_19    0.305   0.0225    12.1   0.289
9 HSprintP_10    0.255   0.0280    10.1   0.359
```

Appendix B

Code and short description

The following section includes all R code written to produce the results in this project and an order of how the scripts should be run if tested.

Order of scripts

- data_import.R (Loads processed MATLAB data and puts it in an R data frame)
  - SortClassicalData.R (Sorts long distance classical race data into R data frames)
  - SortSprintData.R (Sorts sprint race data into R data frames)
  - SetMassGenderYear.R (Race and skier specific features are set for the data.)
- calculate_acc_and_slope_angle.R (Acceleration and slope angle is calculated, smoothing is done.)
  - smoothing_splines.R (SmoothingSplines()-function)
  - angle_and_slope_calc_functions.R (AngleCalculation() calculates slope angle)
  - central_difference.R (General central differences function CentralDifferences())
- model_calc.R (Calculates power and other terms in the power balance model)
- ParameterSensitivity.R (Script for testing sensitivity of parameters)
- MonteCarloSimulation.R (Monte Carlo simulation)

- drag_area_values.R (Functions setting drag area values based on technique and subtechnique)
  - GetDragAreaValueClassical() (Setting drag area based on subtechnique for classical races)
  - GetDragAreaValueSkate() (Setting drag area based on speed in the skate races)
- 2017_classic_sprint_men.R (Testing script for the classic sprint for men, 2017)
- 2018_skate_sprint.R (Testing script for the skate sprint 2018, men and women)
- 2018_classic_long_distance.R (Testing script for the long distance classical races 2018, 10 km women, 15 km men)

Framework

./../data/data_import.R

```
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

# THIS SCRIPT LOADS MATLAB DATA INTO R DATA FRAMES AND PRODUCES:
    # data_c2018m: Men 2018 classic 15 km (68951 observations per variable)
    # data_c2018f: Women 2018 classic 10 km (76031 obs pr var)
    # data_2017m: Men 2017 sprint (classic) (21341 obs pr var)
    # data_2018m: Men 2018 sprint (skate) (1000 obs pr var)
    # data_2018f: Women 2018 sprint (skate) (1000 obs pr var)

# This script is only needed once, in the beginning of the project to get data on the
    right form
# Slope and acceleration is added later and saved, see other files.

rm(list = ls())
library("R.matlab")
source("./utils/SortClassicalData.R")
source("./utils/SortSprintData.R")
source("./utils/set_mass_gender_year.R")



### CLASSICAL SKIING DATA ------------------------------------------------
# Read MATLAB files (Run once)
# data_c2018m_unprocessed <- readMat("./SenTIF_material/beito2018_classic_men_
    classification.mat") # Men
# data_c2018f_unprocessed <- readMat("./SenTIF_material/beito2018_classic_women_
    classification.mat") # Women
# save(data_c2018m_unprocessed, file = "./data/data_c2018m_unprocessed.RData")
# save(data_c2018f_unprocessed, file = "./data/data_c2018f_unprocessed.RData")
```

```
# rm(data_c2018m_unprocessed, data_c2018f_unprocessed)

# Sort into a data frame
load("./data/data_c2018m_unprocessed.RData")
load("./data/data_c2018f_unprocessed.RData")
data_c2018m <- SortClassicalData(data_c2018m_unprocessed) # 21 skiers
data_c2018f <- SortClassicalData(data_c2018f_unprocessed) # 7 skiers

data_c2018f <- SetMassGenderYear("data_c2018f")
data_c2018m <- SetMassGenderYear("data_c2018m")



# DGPS data
# dgps_c2018m <- readMat("./SenTIF_material/DPGS_elevation_beito_distance2018_men.mat")
# dgps_c2018f <- readMat("./SenTIF_material/DPGS_elevation_beito_distance2018_women.mat")
# dgps_raw <- readMat("./SenTIF_material/DGPS_raw.mat")
# dgps_nofix <- readMat("./SenTIF_material/DPGS_elevation_beito_distance2018_zero_where_nofix.mat")
#
# dgps_men <- data.frame(distance = dgps_c2018m$DGPS.elevation[[4]], elevation = dgps_c2018m$DGPS.elevation[[1]])
# dgps_women <- data.frame(distance = dgps_c2018f$DGPS.elevation[[4]], elevation = dgps_c2018f$DGPS.elevation[[1]])
# dgps_raww <- data.frame(distance = t(dgps_raw$DGPS.raw[[1]]), elevation = dgps_raw$DGPS.raw[[2]])
# dgps_no_fix <- data.frame(distance = dgps_nofix$DGPS.elevation[[4]], elevation = dgps_nofix$DGPS.elevation[[1]])
#
# # Check by plots
# ggplot(dgps_men) + geom_line(aes(distance, elevation))
# ggplot(dgps_women) + geom_line(aes(distance, elevation))
# ggplot(dgps_raww) + geom_line(aes(distance, elevation))
# ggplot(dgps_no_fix) + geom_line(aes(distance, elevation))
#
# ggplot(data_c2018m) + geom_line(aes(distance, elevation, color = skier)) + facet_wrap(~skier)


### SPRINT DATA ------------------------------------------------------------

###  BEITO 2018  ###
## MEN
data_mat <- readMat("./SenTIF_material/beito2018_sprint_men_version1.mat") # Beito 2018 sprint men
data_2018m <- SortSprintData(data_mat) # 13 skiers
data_2018m <- SetMassGenderYear("data_2018m")


## WOMEN
```

```
data_mat <- readMat("./SenTIF_material/beito2018_sprint_women_version1.mat") # Beito 2018 women
data_2018f <- SortSprintData(data_mat) # 11 skiers
data_2018f <- SetMassGenderYear("data_2018f")



###  BEITO 2017 ###
#Data
#data_mat <- readMat("./SenTIF_material/beito_data_sprint_processed.mat") # Beito 2017 (men only, 1000 data points)
data_mat <- readMat("./SenTIF_material/beito2017_sprintprologue_men_classification.mat") # Beito 2017 men with classification
dgps_data <- readMat("./SenTIF_material/DPGS_elevation_beito_sprint2017.mat") # Elevation, men 2017
#dgps <- data.frame(elevation = dgps_data$DGPS.elevation[[1]], distance = dgps_data$DGPS.elevation[[4]])
data_2017m <- SortSprintData(data_mat) # 9 skiers

# Insert DGPS data
data_2017m$elevation <- rep(dgps_data$DGPS.elevation[[1]], length(unique(data_2017m$skier)))
data_2017m$distance <- rep(dgps_data$DGPS.elevation[[4]], length(unique(data_2017m$skier)))

data_2017m <- SetMassGenderYear("data_2017m")


### OVERALL ------------------------------
rm(data_mat,data_c2018m_unprocessed, data_c2018f_unprocessed)
#data <- rbind(data_c2018m, data_c2018f, data_2018m, data_2018f, data_2017m)


## SAVE DATA FRAMES
#save(data, file = "./data/data.RData")
save(data_c2018m, file = "./data/data_c2018m.RData")
save(data_c2018f, file = "./data/data_c2018f.RData")
save(data_2017m, file = "./data/data_2017m.RData")
save(data_2018m, file = "./data/data_2018m.RData")
save(data_2018f, file = "./data/data_2018f.RData")
```

./../utils/calculate_acc_and_slope_angle.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019


## ACCELERATION, SLOPE, ANGLE
# Calculate acceleration and slope angle in the track


# Libraries
library(ggplot2)
library(magrittr)

# Sources
source("./utils/angle_and_slope_calc_functions.R")
source("./utils/central_difference.R")
source("./utils/smoothing_splines.R")



# Initialize
window <- 2


## 2018 CLASSIC MEN ------------------------------------------------------
# No. of observations: 68951
data_c2018m <- CentralDifferences("time", "speed", data_c2018m, window, col.name = "
    acc") # Acceleration
data_c2018m <- AngleCalculation(data_c2018m, window) # Slope angle
data_c2018m <- SmoothingSplines("distance", "speed", data_c2018m, 68951) # Smooth
    speed
data_c2018m <- CentralDifferences("time", "speedSmoothed", data_c2018m, window, col.
    name = "accSpeedSmoothed") # Acceleration smoothed speed
# data_c2018m['CdA'] <- GetDragAreaValueClassical(data_c2018m$subtechnique)

data_c2018m['window'] = window
save(data_c2018m, file = "./data/data_c2018m.RData")


## 2018 CLASSIC WOMEN ----------------------------------------------------------
# No. of observations: 76031
data_c2018f <- CentralDifferences("time", "speed", data_c2018f, window, col.name = "
    acc") # Acceleration
data_c2018f <- AngleCalculation(data_c2018f, window) # Slope angle
data_c2018f <- SmoothingSplines("distance", "speed", data_c2018f, 76031) # Smooth
    speed
data_c2018f <- CentralDifferences("time", "speedSmoothed", data_c2018f, window, col.
    name = "accSpeedSmoothed") # Acceleration smoothed speed

data_c2018f['window'] = window
save(data_c2018f, file = "./data/data_c2018f.RData")
```

```
46
47
48
49 ## 2017, MEN CLASSIC
       ------------------------------------------------------------------------
50 # No. of observations: 21341
51 window <- 150
52 data_2017m <- CentralDifferences("time", "speed", data_2017m, window, col.name = "acc
       ") # Acceleration
53 data_2017m <- AngleCalculation(data_2017m, window) # Slope angle
54 data_2017m <- SmoothingSplines("distance", "speed", data_2017m, 21341) # Smooth speed
55 data_2017m <- CentralDifferences("time", "speedSmoothed", data_2017m, window, col.
       name = "accSpeedSmoothed") # Acceleration smoothed speed
56
57
58 data_2017m['window'] = window
59 save(data_2017m, file = "./data/data_2017m.RData")
60
61
62
63 ## 2018, MEN SKATE ----------------------------------------------------------------
64 # No. of observations: 1000
65 data_2018m <- CentralDifferences("time", "speed", data_2018m, window, col.name = "acc
       ") # Acceleration
66 data_2018m <- AngleCalculation(data_2018m, window) # Slope angle
67 data_2018m <- SmoothingSplines("distance", "speed", data_2018m, 1000) # Smooth speed
68 data_2018m <- CentralDifferences("time", "speedSmoothed", data_2018m, window, col.
       name = "accSpeedSmoothed") # Acceleration smoothed speed
69
70 data_2018m['window'] = window
71 save(data_2018m, file = "./data/data_2018m.RData")
72
73
74
75 ## 2018, WOMEN SKATE --------------------------------------------------------
76 # No. of observations: 1000
77 data_2018f <- CentralDifferences("time", "speed", data_2018f, window, col.name = "acc
       ") # Acceleration
78 data_2018f <- AngleCalculation(data_2018f, window) # Slope angle
79 data_2018f <- SmoothingSplines("distance", "speed", data_2018f, 1000) # Smooth speed
80 data_2018f <- CentralDifferences("time", "speedSmoothed", data_2018f, window, col.
       name = "accSpeedSmoothed") # Acceleration smoothed speed
81
82
83 data_2018f['window'] = window
84 save(data_2018f, file = "./data/data_2018f.RData")
```

./../testing/2017_classic_sprint_men.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

## TESTING Classic sprint races men 2017 (No. of observations: 21341)
#    - Sensitivity analysis of parameters in an uphill segment
#    - Monte Carlo simulation in the same uphill segment


library(ggplot2)
library(ggpubr)
library(ggsci)
library(viridis)

source("./testing/MonteCarloSimulation.R")
source("./testing/ParameterSensitivity.R")
source("./utils/model_calc.R")
source("./utils/drag_area_values.R")

load("./data/data_2017m.RData")

## Set global parameters for data_2017m
CdA <- GetDragAreaValueClassical(data_2017m$subtechnique, testing = FALSE)
par <- list(data_2017m$mu[1], 9.81, 0.50, 0.23, 0.45, 1.1, CdA) # data_2017m$mu[1] = 0.025


### ----- Regular model calculation -----
window <- 200

# High h -----
data_2017m <- CentralDifferences("time", "speed", data_2017m, window, col.name = "acc") # Acceleration
data_2017m <- AngleCalculation(data_2017m, window) # Slope angle
data_2017m <- SmoothingSplines("distance", "speed", data_2017m, 21341) # Smooth speed
data_2017m <- CentralDifferences("time", "speedSmoothed", data_2017m, window, col.name = "accSpeedSmoothed") # Acceleration smoothed speed

# Power calc
df <- model(data_2017m, par, speed.type = "speedSmoothed", acc.type = "accSpeedSmoothed", classical.or.skate = "classical", testing = FALSE)
#skier <- df[df$skier == "HSprintP_40", ] # One skier only

# relative.df <- data.frame()
# for(i in unique(df$skier)){
#   relative.df <- rbind(relative.df, data.frame(skier = i, diff = (df[df$skier==i, ]$rel.power - df[df$skier=="HSprintP_40", ]$rel.power),
#                                                distance = df[df$skier==i, ]$distance))
# }
#
```

```
# ggplot(relative.df) + geom_line(aes(distance, diff, color = skier))
# ggplot(df) + geom_line(aes(distance, rel.power, color = skier))
# ggplot(df) + geom_line(aes(distance, speedSmoothed, color = skier)) + facet_wrap(~
    skier)
#
# #[relative.df$distance>600 & relative.df$distance<800, ]
#
# ## Plots
# high.h.total <- ggplot(skier, size = 2) + geom_line(aes(distance, drag), color = "
    red") +
#   geom_line(aes(distance, friction), color = "blue") +
#   geom_line(aes(distance, gravity), color = "orange") +
#   geom_line(aes(distance, kin.energy), color = "darkgreen") +
#   theme(text = element_text(size = 20)) +
#   labs(title = "Model terms in the power balance model", subtitle = "h = 200, df =
    n = 21341", x ="distance [m]", y = "model terms [J/s]")
# high.h.total


### --------- Sensitivity analysis ----------

## Friction
tab.mu <- data.frame()

for(i in unique(data_2017m$skier)){
  # Set 'data' and parameter vector
  data <- data_2017m[data_2017m$skier==i, ]
  CdA <- GetDragAreaValueClassical(data$subtechnique, testing = FALSE)
  par <- list(data_2017m$mu[1], 9.81, 0.50,0.23,0.45,1.1, CdA)

  # Calculate power with varying parameters for friction
  friction <- ParameterSensitivity("friction", data, par, "classical", speed.type = "
      speedSmoothed", acc.type = "accSpeedSmoothed")
  # Look at uphill segment
  up.friction <- friction[which(friction$distance>600 & friction$distance<800),]
  diff.mu <- abs(up.friction[up.friction$mu==min(up.friction$mu), ]$rel.power - up.
      friction[up.friction$mu==max(up.friction$mu), ]$rel.power)
  mu.ref <- up.friction[up.friction$mu==data$mu[1], ]$rel.power

  rel.max <- max(diff.mu)/mu.ref[which.max(diff.mu)]
  rel.min <- min(diff.mu)/mu.ref[which.min(diff.mu)]
  max(diff.mu)
  min(diff.mu)

  tab.mu <- rbind(tab.mu, data.frame(skier = i, max.diff = max(diff.mu), min.diff =
      min(diff.mu), rel.max = rel.max*100, rel.min = rel.min*100))

}

# up1 <- ggplot(up.friction) + geom_line(aes(distance, rel.power, color = factor(mu))
    ) + labs(title="Propulsive power with varying mu") +
#   theme(text=element_text(size=20), legend.position = "bottom")
```

```
# up2 <- ggplot(up.friction) + geom_line(aes(distance, elevation)) + theme(text=
    element_text(size=20))
# ggarrange(up1,up2, nrow=2, legend = "bottom")




### ----- Drag area -------

tab.drag <- data.frame()

for (i in unique(data_2017m$skier)){
  data <- data_2017m[data_2017m$skier==i, ]
  CdA <- GetDragAreaValueClassical(data$subtechnique, testing = FALSE)
  par <- list(data_2017m$mu[1], 9.81, 0.50,0.23,0.45,1.1, CdA)

  drag.area <- ParameterSensitivity("drag.area", data, par, "classical", speed.type =
       "speedSmoothed", acc.type = "accSpeedSmoothed")
  up.drag <- drag.area[which(drag.area$distance>600 & drag.area$distance<800), ]
  diff.drag <- abs(up.drag[up.drag$drag.area==min(up.drag$drag.area), ]$rel.power -
      up.drag[up.drag$drag.area==max(up.drag$drag.area),]$rel.power) # Absolute
      difference in relative power
  drag.ref <- up.drag[up.drag$drag.area==0.45, ]$rel.power # Relative power for
      lowfriction

  rel.max <- max(diff.drag)/drag.ref[which.max(diff.drag)] # Max difference relative
      to rel.power for low friction
  rel.min <- min(diff.drag)/drag.ref[which.min(diff.drag)] # Min difference relative
      to rel.power for high friction
  max(diff.drag)
  min(diff.drag)

  tab.drag <- rbind(tab.drag, data.frame(skier = i, max.diff = max(diff.drag), min.
      diff = min(diff.drag), rel.max = rel.max*100, rel.min = rel.min*100))

}

pl1 <- ggplot(up.drag) + geom_line(aes(distance, rel.power, color = factor(drag.area)
    ), size = 1) +
  labs(title = "Power with varying drag area for uphill segment") + theme(text=
      element_text(size = 20), legend.position = "bottom")
pl2 <- ggplot(up.drag) + geom_line(aes(distance, elevation), size = 1) + labs(
    subtitle = "Track profile") +
  theme(text=element_text(size = 20))
pl3 <- ggplot(up.drag) + geom_line(aes(distance, speedSmoothed), size = 1) + labs(
    subtitle = "Speed") +
  theme(text=element_text(size = 20))
ggarrange(pl1,pl2,pl3, nrow=3,legend="bottom")



### Mass
```

```
131 tab.mass <- data.frame()
132
133 for (i in unique(data_2017m$skier)){
134   data <- data_2017m[data_2017m$skier==i, ]
135   CdA <- GetDragAreaValueClassical(data$subtechnique, testing = FALSE)
136   par <- list(data_2017m$mu[1], 9.81, 0.50,0.23,0.45,1.1, CdA)
137
138   mass <- ParameterSensitivity("mass", data, par, "classical", speed.type = "
          speedSmoothed", acc.type = "accSpeedSmoothed")
139   up.mass <- mass[which(mass$distance>600 & mass$distance<800),]
140   diff.mass <- abs(up.mass[up.mass$mass == max(mass$mass),]$rel.power - up.mass[up.
          mass$mass == min(mass$mass),]$rel.power) # Absolute diff in relative power
141   mass.ref <- up.mass[up.mass$mass==data$mass[1], ]$rel.power # Relative power for
          ref mass
142
143   rel.max <- max(diff.mass)/mass.ref[which.max(diff.mass)] # Max difference relative
          to rel.power for ref mass
144   rel.min <- min(diff.mass)/mass.ref[which.min(diff.mass)] # Min difference relative
          to rel.power for ref mass
145   max(diff.drag)
146   min(diff.drag)
147
148   tab.mass <- rbind(tab.mass, data.frame(skier = i, max.diff = max(diff.mass), min.
          diff = min(diff.mass), rel.max = rel.max*100, rel.min = rel.min*100))
149
150 }
151
152 # ggplot(mass, aes(distance, rel.power)) + geom_line(aes(color=factor(mass))) +
153 #   labs(title = "Power with varying skier mass") + theme(text=element_text(size =
      20), legend.position = "bottom")
154
155 mass1 <- ggplot(up.mass) + geom_line(aes(distance, rel.power, color = factor(mass)))
156 mass2 <- ggplot(up.mass) + geom_line(aes(distance, elevation))
157 ggarrange(mass1,mass2,nrow=2,legend="bottom")
158
159
160
161 ## Air density
162 #
163 tab.air <- data.frame()
164
165 for (i in unique(data_2017m$skier)){
166   data <- data_2017m[data_2017m$skier==i, ]
167   CdA <- GetDragAreaValueClassical(data$subtechnique, testing = FALSE)
168   par <- list(data_2017m$mu[1], 9.81, 0.50,0.23,0.45,1.1, CdA)
169
170   air <- ParameterSensitivity("air.density", data, par, "classical", speed.type = "
          speedSmoothed", acc.type = "accSpeedSmoothed")
171   up.air <- air[which(air$distance>600 & mass$distance<800),]
172   diff.air <- abs(up.air[up.air$rho == max(air$rho), ]$rel.power - up.air[up.air$rho
          == min(air$rho), ]$rel.power) # Absolute diff in relative power
173   air.ref <- up.air[up.air$rho==1.1, ]$rel.power # Relative power for ref rho
```

```
174
175   rel.max <- max(diff.air)/air.ref[which.max(diff.air)] # Max difference relative to
          rel.power for ref rho
176   rel.min <- min(diff.air)/air.ref[which.min(diff.air)] # Min difference relative to
          rel.power for ref rho
177   max(diff.air)
178   min(diff.air)
179
180   tab.air <- rbind(tab.air, data.frame(skier = i, max.diff = max(diff.air), min.diff
          = min(diff.air), rel.max = rel.max*100, rel.min = rel.min*100))
181
182 }
183
184
185
186 ### ----- Monte Carlo Simulation -----
187 set.seed(1234)
188 mc <- data.frame()
189 #data_2017m <- data_2017m[data_2017m$distance>600 & data_2017m$distance<800, ]
190 for(skier in unique(df$skier)){
191   data <- MonteCarloSimulation(1, data_2017m[data_2017m$skier==skier, ], "classical",
          "speedSmoothed", "accSpeedSmoothed")
192   ref <- df[df$skier==skier, ]
193   for (run in unique(data$run)){
194     mc <- rbind(mc, data.frame(skier = data[data$run==run, ]$skier,
195                                distance = data[data$run==run, ]$distance,
196                                run = run,
197                                diff = (data[data$run==run,]$rel.power - ref$rel.power
                                       ),
198                                rel = ((data[data$run == run, ]$rel.power - ref$rel.
                                       power)/ref$rel.power)*100))
199   } # end runs
200   #mc[which(is.nan(mc$rel)), ]$rel <- 0
201   # print(skier)
202   # val <- max(abs(mc[mc$skier==skier, ]$rel))
203   # print(val)
204 }
205
206 #mc[which(is.nan(mc$rel)), ]$rel <- 0
207 mc.up <- mc[which(mc$distance>600 & mc$distance<800), ]
208
209 ggplot(mc.up) + geom_histogram(aes(rel,..density..), bins = 30, na.rm = TRUE) + facet
      _wrap(~skier)
210 ggplot(mc.up) + geom_histogram(aes(diff,..density..), bins = 30, na.rm = TRUE) +
      facet_wrap(~skier)
211
212 vec <- sort(abs(mc.up$rel))
213 ci.2017m <- c(vec[round(0.025*length(vec))], vec[round(0.975*length(vec))])
214 ci.2017m
215
216 # hist.2017 <- sort(mc.up$rel)
217 # hist2017 <- data.frame(deviation = hist.2017[ round(0.025*length(hist.2017)):round
```

```
    (0.975*length(hist.2017))])
# ggplot(hist2017) + geom_histogram(aes(deviation,..density..), color = "black", fill
    = "white", bins = 50)

mean(abs(mc.up$rel))
min(abs(mc.up$rel))
max(abs(mc.up$rel))
```

./../testing/2018_skate_sprint.R

```
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

# TESTING Skate sprint 2018

## TESTING Skate sprint races 2018
#    - Sensitivity analysis of parameters in an uphill segment
#    - Monte Carlo simulation in the same uphill segment

library(ggplot2)
library(ggpubr)
library(ggsci)
library(viridis)

source("./testing/MonteCarloSimulation.R")
source("./testing/ParameterSensitivity.R")
source("./utils/model_calc.R")


### MEN ----------------------------------------------------
load("./data/data_2018m.RData")
### ----- Regular model calculation -----
window <- 20

# High h -----
data_2018m <- CentralDifferences("time", "speed", data_2018m, window, col.name = "acc
    ") # Acceleration
data_2018m <- SmoothingSplines("distance", "speed", data_2018m, 1000) # Smooth speed
data_2018m <- SmoothingSplines("distance", "elevation", data_2018m, 1000) # Smooth
    elevation
data_2018m <- CentralDifferences("time", "speedSmoothed", data_2018m, window, col.
    name = "accSpeedSmoothed") # Acceleration smoothed speed
data_2018m$elevation <- data_2018m$elevationSmoothed
data_2018m <- AngleCalculation(data_2018m, window) # Slope angle

ggplot(data_2018m) + geom_line(aes(distance, elevation)) + facet_wrap(~skier)

# Set global parameters for data_2018m
CdA <- GetDragAreaValueSkate(data_2018m$speedSmoothed, 0.50, 0.23, testing = FALSE)
par <- list(data_2018m$mu[1], 9.81, 0.50, 0.23, 0.45, 1.1, CdA)

# Power calc
df2018m <- model(data_2018m, par, speed.type = "speedSmoothed", acc.type = "
    accSpeedSmoothed", classical.or.skate = "skate", testing = FALSE)

# Monte Carlo simulation
set.seed(1234)
mc2018m <- data.frame()
tic()
for(skier in unique(df2018m$skier)){
```

```
  #data <- sim.all[sim.all$skier==skier, ]
  data <- MonteCarloSimulation(200, data_2018m[data_2018m$skier==skier, ], "skate", "
      speedSmoothed", "accSpeedSmoothed")
  ref <- df2018m[df2018m$skier==skier, ]
  for (run in unique(data$run)){
    mc2018m <- rbind(mc2018m, data.frame(skier = data[data$run==run, ]$skier,
                               distance = data[data$run==run, ]$distance,
                               run = run,
                               diff = (data[data$run==run,]$rel.power - ref$rel.power
                                   ),
                               rel = ((data[data$run == run, ]$rel.power - ref$rel.
                                   power)/ref$rel.power)*100))
  } # end runs
} # skier
toc()
# Find suitable uphill range
xrange <- c(700,875)

# Check elevation is strictly uphill and rel.power > 0
ggplot(df2018m[which(df2018m$distance>xrange[1] & df2018m$distance<xrange[2]), ]) +
     geom_line(aes(distance, elevation)) + facet_wrap(~skier)
ggplot(df2018m[which(df2018m$distance>xrange[1] & df2018m$distance<xrange[2]), ]) +
     geom_line(aes(distance, rel.power)) + facet_wrap(~skier)

mc2018m.up <- mc2018m[which(mc2018m$distance>xrange[1] & mc2018m$distance<xrange[2]),
     ]
sum(is.nan(mc2018m.up$rel))/length(mc2018m.up$rel) # Check no NaNs

vec.2018m <- sort(abs(mc2018m.up$rel))
ci.2018m <- c(vec.2018m[round(0.025*length(vec.2018m))], vec.2018m[round(0.975*length
     (vec.2018m))])
ci.2018m

mean(abs(mc2018m.up$rel))
min(abs(mc2018m.up$rel))
max(abs(mc2018m.up$rel))

ggplot(mc2018m.up) + geom_histogram(aes(rel,..density..), bins = 30, na.rm = TRUE) +
     facet_wrap(~skier)




### WOMEN ------------------------------------------------
load("./data/data_2018f.RData")

### ----- Regular model calculation -----
window <- 20

data_2018f <- CentralDifferences("time", "speed", data_2018f, window, col.name = "acc
     ") # Acceleration
data_2018f <- SmoothingSplines("distance", "speed", data_2018f, 1000) # Smooth speed
```

```
data_2018f <- SmoothingSplines("distance", "elevation", data_2018f, 1000) # Smooth
    elevation
data_2018f <- CentralDifferences("time", "speedSmoothed", data_2018f, window, col.
    name = "accSpeedSmoothed") # Acceleration smoothed speed
data_2018f$elevation <- data_2018f$elevationSmoothed
data_2018f <- AngleCalculation(data_2018f, window) # Slope angle

# Set global parameters for data_2018m
CdA <- GetDragAreaValueSkate(data_2018f$speedSmoothed, 0.50, 0.23, testing = FALSE)
par <- list(data_2018f$mu[1], 9.81, 0.50, 0.23, 0.45, 1.1, CdA)

# Power calc
df2018f <- model(data_2018f, par, speed.type = "speedSmoothed", acc.type = "
    accSpeedSmoothed", classical.or.skate = "skate", testing = FALSE)

# Monte Carlo simulation
set.seed(1234)
mc2018f <- data.frame()
for(skier in unique(df2018f$skier)){
  #data <- sim.all[sim.all$skier==skier, ]
  data <- MonteCarloSimulation(200, data_2018f[data_2018f$skier==skier, ], "skate", "
      speedSmoothed", "accSpeedSmoothed")
  ref <- df2018f[df2018f$skier==skier, ]
  for (run in unique(data$run)){
    mc2018f <- rbind(mc2018f, data.frame(skier = data[data$run==run, ]$skier,
                                         distance = data[data$run==run, ]$distance,
                                         run = run,
                                         diff = (data[data$run==run,]$rel.power - ref
                                             $rel.power),
                                         rel = ((data[data$run == run, ]$rel.power -
                                             ref$rel.power)/ref$rel.power)*100))
  } # end runs
}


mc2018f.up <- mc2018f[which(mc2018f$distance>550 & mc2018f$distance<700), ]

ggplot(df2018f[which(df2018f$distance>550 & df2018f$distance<700), ]) + geom_line(aes
    (distance, rel.power)) + facet_wrap(~skier)
ggplot(df2018f[which(df2018f$distance>550 & df2018f$distance<700), ]) + geom_line(aes
    (distance, elevation)) + facet_wrap(~skier)

sum(is.nan(mc2018f.up$rel))/length(mc2018f.up$rel) # fraction of NaNs


vec.2018f <- sort(abs(mc2018f.up$rel))
ci.2018f <- c(vec.2018f[round(0.025*length(vec.2018f))], vec.2018f[round(0.975*length
    (vec.2018f))])
ci.2018f

ggplot(mc2018f.up) + geom_histogram(aes(rel,..density..), bins = 30, na.rm = TRUE) +
    facet_wrap(~skier)
```

```
133
134 mean(abs(mc2018f.up$rel))
135 min(abs(mc2018f.up$rel))
136 max(abs(mc2018f.up$rel))
```

./../testing/2018_classic_long_distance.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

## TESTING Classical 10/15 km 2018
#    - Sensitivity analysis of parameters in an uphill segment
#    - Monte Carlo simulation in the same uphill segment

library(ggplot2)
library(ggpubr)
library(ggsci)
library(viridis)

source("./testing/MonteCarloSimulation.R")
source("./testing/ParameterSensitivity.R")
source("./utils/model_calc.R")
source("./utils/angle_and_slope_calc_functions.R")
source("./utils/central_difference.R")
source("./utils/smoothing_splines.R")
source("./utils/drag_area_values.R")

### WOMEN ----------------------------------------------------

load("./data/data_c2018f.RData")

test <- data_c2018f[data_c2018f$skier=="D10CL_46", ]
ggplot(test) + geom_line(aes(distance, elevation)) + facet_wrap(~lap)
ggplot(test) + geom_line(aes(distance, elevation)) + facet_wrap(~lap)
#(250/(length(test$time)/2))*(test$distance[152062])

### ----- Regular model calculation -----
window <- 250

#data_c2018f <- CentralDifferences("time", "speed", data_c2018f, window, col.name = "
    acc") # acc from unfiltered/unsmoothed speed
data_c2018f <- SmoothingSplines("distance", "speed", data_c2018f, 1000) # Smooth
    speed
data_c2018f <- CentralDifferences("time", "speedSmoothed", data_c2018f, window, col.
    name = "accSpeedSmoothed") # acc based on smoothed speed
data_c2018f <- SmoothingSplines("distance", "elevation", data_c2018f, 1000) # Smooth
    elevation
data_c2018f$elevation <- data_c2018f$elevationSmoothed
data_c2018f <- AngleCalculation(data_c2018f, window)

# Set global parameters for data_2018m
CdA <- GetDragAreaValueClassical(data_c2018f$subtechnique, testing = FALSE)
par <- list(data_c2018f$mu[1], 9.81, 0.50, 0.23, 0.45, 1.1, CdA)

# Power calc
df_c2018f <- model(data_c2018f, par, speed.type = "speedSmoothed", acc.type = "
    accSpeedSmoothed", classical.or.skate = "classical", testing = FALSE)
```

```
47
48 # Monte Carlo simulation
49 set.seed(1234)
50 mc_c2018f <- data.frame()
51 for(skier in unique(df_c2018f$skier)){
52   for(lap in unique(df_c2018f$lap)){
53     print(skier)
54     print(lap)
55     data <- MonteCarloSimulation(50, data_c2018f[(data_c2018f$skier==skier & data_
           c2018f$lap==lap), ], "classical", "speedSmoothed", "accSpeedSmoothed")
56     ref <- df_c2018f[(df_c2018f$skier==skier & df_c2018f$lap == lap), ]
57     for (run in unique(data$run)){
58       print(run)
59       mc_c2018f <- rbind(mc_c2018f, data.frame(skier = data[data$run==run, ]$skier,
60                                                distance = data[data$run==run, ]$distance,
61                                                run = run,
62                                                lap = data[data$run==run, ]$lap,
63                                                diff = (data[data$run==run,]$rel.power -
                                                       ref$rel.power),
64                                                rel = ((data[data$run == run, ]$rel.power
                                                       - ref$rel.power)/ref$rel.power)*100))
65     } # runs
66
67   } # lap
68 } # skier
69
70 # Find suitable uphill range
71 xrange <- c(835,980)
72 # Check elevation is strictly uphill and rel.power > 0
73 ggplot(df_c2018f[which(df_c2018f$distance>xrange[1] & df_c2018f$distance<xrange[2]),
       ]) + geom_line(aes(distance, elevationSmoothed)) + facet_wrap(~skier)
74 ggplot(df_c2018f[which(df_c2018f$distance>xrange[1] & df_c2018f$distance<xrange[2]),
       ]) + geom_line(aes(distance, rel.power)) + facet_wrap(~skier)
75 mc_c2018f.up <- mc_c2018f[which(mc_c2018f$distance>xrange[1] & mc_c2018f$distance<
       xrange[2]), ]
76 sum(is.nan(mc_c2018f.up$rel))/length(mc_c2018f.up$rel) # Check no NaNs
77
78 ggplot(mc_c2018f.up) + geom_histogram(aes(rel,..density..), bins = 30, na.rm = TRUE)
       + facet_wrap(~skier)
79
80 vec.c2018f <- sort(abs(mc_c2018f.up$rel))
81 ci.c2018f <- c(vec.c2018f[round(0.025*length(vec.c2018f))], vec.c2018f[round(0.975*
       length(vec.c2018f))])
82 ci.c2018f
83
84 hist.vec <- sort(mc_c2018f.up$rel)
85 hist.frame <- data.frame(deviation = hist.vec[round(0.025*length(hist.vec)):round
       (0.975*length(hist.vec))])
86
87 ggplot(hist.frame) + geom_histogram(aes(deviation,..density..), color = "black", fill
        = "lightblue", bins = 50) +
88   theme(text = element_text(size = 12)) + labs(title="Relative deviation in
```

```
      propulsive power",
89                                                  subtitle = "10 km classic (W)", x = "
                                                         deviation [%]")
90
91  mean(abs(mc_c2018f.up$rel))
92  min(abs(mc_c2018f.up$rel))
93  max(abs(mc_c2018f.up$rel))
94
95
96  ### MEN ----------------------------------------------------
97
98  load("~/Masteroppgave/data/data_c2018m.RData")
99  ### ----- Regular model calculation -----
100 window <- 250
101
102 data_c2018m <- SmoothingSplines("distance", "speed", data_c2018m, 100) # Smooth speed
103 data_c2018m <- CentralDifferences("time", "speedSmoothed", data_c2018m, window, col.
        name = "accSpeedSmoothed") # acc based on smoothed speed
104 data_c2018m <- SmoothingSplines("distance", "elevation", data_c2018m, 1000) # Smooth
        elevation
105 data_c2018m$elevation <- data_c2018m$elevationSmoothed
106 data_c2018m <- AngleCalculation(data_c2018m, window)
107
108 ggplot(data_c2018m) + geom_line(aes(distance, speedSmoothed, color = factor(lap))) +
        facet_wrap(~skier)
109 ggplot(data_c2018m) + geom_line(aes(distance, elevationSmoothed, color = factor(lap))
        ) + facet_wrap(~skier)
110
111
112 # Set global parameters for data_2018m
113 CdA <- GetDragAreaValueClassical(data_c2018m$subtechnique, testing = FALSE)
114 par <- list(data_c2018m$mu[1], 9.81, 0.50, 0.23, 0.45, 1.1, CdA)
115
116 # Power calc
117 df_c2018m <- model(data_c2018m, par, speed.type = "speedSmoothed", acc.type = "
        accSpeedSmoothed", classical.or.skate = "classical", testing = FALSE)
118
119 # Monte Carlo simulation
120 set.seed(1234)
121 mc_c2018m <- data.frame()
122 for(skier in unique(df_c2018m$skier)){
123   for(lap in unique(df_c2018m$lap)){
124     print(skier)
125     print(lap)
126     data <- MonteCarloSimulation(10, data_c2018m[(data_c2018m$skier==skier & data_
            c2018m$lap==lap), ], "classical", "speedSmoothed", "accSpeedSmoothed")
127     ref <- df_c2018m[(df_c2018m$skier==skier & df_c2018m$lap == lap), ]
128     for (run in unique(data$run)){
129       print(run)
130       mc_c2018m <- rbind(mc_c2018m, data.frame(skier = data[data$run==run, ]$skier,
131                                                distance = data[data$run==run, ]$
                                                       distance,
```

```
                                          run = run,
                                          lap = data[data$run==run, ]$lap,
                                          diff = (data[data$run==run,]$rel.power
                                                - ref$rel.power),
                                          rel = ((data[data$run == run, ]$rel.
                                               power - ref$rel.power)/ref$rel.
                                               power)*100))
    } # runs

  } # lap
} # skier


# Find suitable uphill range
xrange <- c(2400,2600)

# Check elevation is strictly uphill and rel.power > 0
ggplot(df_c2018m[which(df_c2018m$distance>xrange[1] & df_c2018m$distance<xrange[2]),
     ]) + geom_line(aes(distance, elevationSmoothed)) + facet_wrap(~skier)
ggplot(df_c2018m[which(df_c2018m$distance>xrange[1] & df_c2018m$distance<xrange[2]),
     ]) + geom_line(aes(distance, rel.power)) + facet_wrap(~skier)
mc_c2018m.up <- mc_c2018m[which(mc_c2018m$distance>xrange[1] & mc_c2018m$distance<
     xrange[2]), ]
sum(is.nan(mc_c2018m.up$rel))/length(mc_c2018m.up$rel) # Check no NaNs

vec.c2018m <- sort(abs(mc_c2018m.up$rel))
ci.c2018m <- c(vec.c2018m[round(0.025*length(vec.c2018m))], vec.c2018m[round(0.975*
     length(vec.c2018m))])
ci.c2018m

mean(abs(mc_c2018m.up$rel))
min(abs(mc_c2018m.up$rel))
max(abs(mc_c2018m.up$rel))

ggplot(mc_c2018m.up) + geom_histogram(aes(rel,..density..), bins = 30, na.rm = TRUE)
     + facet_wrap(~skier)
```

Functions

./../utils/SortSprintData.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

SortSprintData <- function(data){
  # Sort loaded MATALB data frame into organized R data frame
  #
  #   Args:
  #     data: Loaded MATLAB data to sort
  #
  #   Returns: Sorted R data frame, ready for further data analysis
  #
  # Note: Loads differently based on sprint style (classic 2017 or skate 2018)

  df <- data.frame()
  dimname <- dimnames(data$data[[1]][[1]])[[1]]


  if (sum(dimname =="technique")){
    # For classic sprint 2017
    for(skier in 1:length(data$data)){
      df <- rbind(df, data.frame(
        skier = data$data[[skier]][[1]][[which(dimname =="name")]],
        time = data$data[[skier]][[1]][[which(dimname =="time")]],
        distance = data$data[[skier]][[1]][[which(dimname =="distance")]],
        speed = data$data[[skier]][[1]][[which(dimname =="speed")]]/3.6,
        elevation = data$data[[skier]][[1]][[which(dimname =="elevation")]],
        subtechnique = t(data$data[[skier]][[1]][[which(dimname =="technique")]]),
        technique = "classic"
      ))
    }
  } else {
    # For skate sprint 2018
    for(skier in 1:length(data$data)){
    df <- rbind(df, data.frame(
      skier = data$data[[skier]][[1]][[which(dimname =="name")]],
      time = data$data[[skier]][[1]][[which(dimname =="time")]],
      distance = t(data$data[[skier]][[1]][[which(dimname =="distance")]]),
      speed = data$data[[skier]][[1]][[which(dimname =="speed")]],
      elevation = data$data[[skier]][[1]][[which(dimname =="elevation")]],
      technique = "skate"
    ))
  } # end for
  }# end if



  return(df)

```

```
50
51
52 }
```

./../utils/SortClassicalData.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019


SortClassicalData <- function(data){
  # Sort data structure into organized R data frame
  #
  # Arguments:
  #   data: Data frame loaded from MATLAB to sort
  #
  # Returns:
  #   Data frame 'frame.name' with sorted data, ready for analysis and further calculations

  # Note: Indices in lists are found based on investigation of the lists when loaded in R.


  # Allocate
  frame.name <- data.frame()
  names <- data.frame()
  # Extract names of skiers
  for (i in 1:length(data$data)){
    names <- rbind(names, data$data[[i]][[1]][[1]])
  }
  # Put into data frame
  num.skiers <- dim(unique(names))[1] # Number of unique skiers
  num.laps <- dim(names)[1]/num.skiers # Number of laps

  # Go through every skier, for every lap/round
  for(skier in 1:num.skiers){
    append.lap <- data.frame()
    for(lap in 1:num.laps){
      lap.data <- data$data[[skier + (lap-1)*num.skiers]][[1]] # Possibly change this line (avoid raw data)
      append.lap <- rbind(append.lap, data.frame(
        skier = lap.data[[1]],
        time = lap.data[[9]],
        distance = lap.data[[3]],
        speed = lap.data[[4]]/3.6, # speed is given in km/h, want m/s
        elevation = lap.data[[8]],
        subtechnique = t(lap.data[[10]]),
        technique = "classic",
        lap = lap))
    }
    # Put in data frame
    frame.name <- rbind(frame.name, append.lap)
  }

  # Save and return
```

```
49   return(frame.name)
50
51
52 } # end function
```

./../utils/set_mass_gender_year.R

```r
SetMassGenderYear <- function(data.frame){
  # Assign mass and gender of skier and year of race as columns in the data frame
  #
  #   Args:
  #     data.frame: Race data frame name to assign column variables to
  #
  #   Returns:
  #     Data frame with additional columns "mass", "gender" and "year"
  #
  # Note: Function assumes data.frame s in if sentences already exist

  # Ensure correct data.frame name
  frame.names <- c("data_c2018m", "data_c2018f", "data_2018m", "data_2018f", "data_
      2017m")
  while(!(sum(data.frame == frame.names)==1)){
    data.frame <- readline(prompt = "Invalid data frame name. Try again: ")
  }

  # ------------------------------------------------------------------------
  if(data.frame == "data_c2018m"){ # 2018 CLASSIC MEN 15 km
    # Gender and year
    data_c2018m['gender'] = "male"
    data_c2018m['year'] = 2018
    data_c2018m['mu'] = 0.02845 # Avg. of the two estimated values for the particular
        race day

    # Mass
    # NOTE: ?77 means no mass is known for the skier and 77 kgs is assumed
    # NOTE: ID 39 is wrong, is actually ID 55. No weight known, 77 kg assumed
    #[1] H15CL_50=74  H15CL_10=77   H15CL_1003=?77 H15CL_13=82   H15CL_14=75   H15CL_
        27=?77   H15CL_28=?77
# H15CL_29=?77   H15CL_31=80   H15CL_32=86  H15CL_33=82   H15CL_34=76   H15CL_36=85
#[14] H15CL_37=75   H15CL_(39=55!)=?77   H15CL_41=75   H15CL_42=83   H15CL_45=72
    H15CL_47=68   H15CL_49=76   H15CL_6=75

    lenM <- length(data_c2018m[data_c2018m$skier == "H15CL_50",]$time)
    data_c2018m['mass'] = c(rep(74,lenM), rep(77, lenM), rep(77, lenM), rep(82, lenM)
        , rep(75, lenM), rep(77,lenM), rep(77, lenM), rep(77, lenM),
                        rep(80, lenM), rep(86, lenM),rep(82, lenM), rep(76,lenM), rep
                            (85,lenM), rep(75, lenM), rep(77, lenM),rep(75, lenM),
                            rep(83, lenM),
                        rep(72, lenM),rep(68, lenM) ,rep(76,lenM), rep(75, lenM)) + 3
                             #+3kg in mass is for equipment

    levels(data_c2018m$skier) <- gsub("H15CL_39", "H15CL_55", levels(data_c2018m$
        skier)) # Change wrong ID from 39 to 55

    #Return
    return.frame <- data_c2018m


```

```
43
44  # ------------------------------------------------------------------------
45  } else if (data.frame == "data_c2018f") { # 2018 CLASSIC WOMEN 10 km
46    # Gender and year
47    data_c2018f['gender'] = "female"
48    data_c2018f['year'] = 2018
49    data_c2018f['mu'] = 0.02845
50
51    # Mass
52    lenW <- length(data_c2018f[data_c2018f$skier == "D10CL_46",]$time)
53    data_c2018f['mass'] = c(rep(52, lenW), rep(55,lenW), rep(53,lenW), rep(58, lenW),
          rep(65, lenW), rep(56,lenW),
54                        rep(63, lenW)) + 3 #+3kg in mass is for equipment
55
56    # Return
57    return.frame <- data_c2018f
58
59
60  # ------------------------------------------------------------------------
61  } else if (data.frame == "data_2018m"){ # 2018 SPRINT SKATE MEN
62    # Gender and year (+ technique and lap)
63    data_2018m['year'] = 2018
64    data_2018m['gender'] = "male"
65    data_2018m['technique'] = "free/skate"
66    data_2018m['lap'] = 1
67    data_2018m['mu'] = 0.02499
68
69    # Mass
70    # Sprint_10=77 Sprint_12=80 Sprint_13=82 Sprint_15=78 Sprint_31=80 Sprint_33=82
          Sprint_34=76 Sprint_35=74 Sprint_36=85 Sprint_37=75 Sprint_42=83 Sprint_
          50=74 Sprint_6=75
71    lenM <- length(data_2018m[data_2018m$skier == "Sprint_10",]$time)
72    data_2018m['mass'] = c(rep(77,lenM), rep(80,lenM), rep(82,lenM), rep(78,lenM),
          rep(80,lenM),
73                         rep(82,lenM), rep(76,lenM), rep(74,lenM), rep(85,lenM), rep
                               (75,lenM),
74                         rep(83,lenM), rep(74,lenM), rep(75, lenM)) + 3 # +3kg for
                               equipment
75
76    # Return
77    return.frame <- data_2018m
78
79  # ------------------------------------------------------------------------
80  } else if (data.frame == "data_2018f"){ # 2018 SPRINT SKATE WOMEN
81    # Gender and year (+ technique and lap)
82    data_2018f['year'] = 2018
83    data_2018f['gender'] = c("female")
84    data_2018f['technique'] = "free/skate"
85    data_2018f['lap'] = 1
86    data_2018f['mu'] = 0.02499
87
88    # Mass
```

```
89      # Sprint_11=58 Sprint_16=62 Sprint_38=55 Sprint_39=65 Sprint_40=58 Sprint_43=68
            Sprint_48=57 Sprint_51=56 Sprint_52=63 Sprint_8=60  Sprint_9=63
90      lenW <- length(data_2018f[data_2018f$skier == "Sprint_11",]$time)
91      data_2018f['mass'] = c(rep(58,lenW),rep(62,lenW), rep(55,lenW), rep(65,lenW), rep
            (58,lenW), rep(68,lenW),
92           rep(57,lenW), rep(56,lenW), rep(63,lenW), rep(60,lenW), rep(63, lenW)) + 3
                #+3kg for equipment
93
94      # Return
95      return.frame <- data_2018f
96
97
98
99    # ------------------------------------------------------------------------
100   } else if (data.frame == "data_2017m"){ # 2017 SPRINT CLASSIC MEN
101     # Gender and year (+ technique and lap)
102     data_2017m['year'] = 2017
103     data_2017m['gender'] = c("male")
104     data_2017m['technique'] = "classic"
105     data_2017m['lap'] = 1
106     data_2017m['mu'] = 0.025
107
108     # Mass
109     #mass_2017m <- c(79,82,78,82,76,74,75,83,80) + 3 # +3kg is for equipment
110
111     len <- length(data_2017m[data_2017m$skier =="HSprintP_40",]$time)
112     data_2017m['mass'] = c(rep(79, len), rep(82, len), rep(78, len), rep(82,len), rep
            (76, len), rep(74, len), rep(75, len), rep(83, len), rep(80, len)) + 3 # +3
            kg for equipment
113
114     # Return
115     return.frame <- data_2017m
116
117   } else {
118     print("Something wrong happened. Check 'SetMassGenderYear.R'")
119   }
120
121   return(return.frame)
122
123 }
```

./../utils/angle_and_slope_calc_functions.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

## INCLINE ANGLE AND SLOPE CALCULATION

AngleCalculation <- function(data, window.size.h){
  # Calculate slope angle in race track
  #
  #   Args:
  #     data: Data frame to compute angle from
  #     window.size.h: Width of window in central differences algorithm
  #
  #   Returns:
  #     data: Input data frame and calculated 'angle' and 'slope' column

  data <- CentralDifferences("distance", "elevation", data, window.size.h, "slope")
  data['angle'] <- atan(data$slope)
  return(data)
}
```

./../utils/central_difference.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019


# For all skiers simultaneously
CentralDifferences <- function(x.name,y.name, data, window, col.name){
  # Approximates derivative by amended central differences scheme
  #
  #   Args:
  #     x.name, y.name: Column names for variables x and y
  #     data: Data frame to extract variables x,y data from
  #     window: Window size 'h' in central differences scheme
  #     col.name: Name of resulting column after use of central differences.
  #       Ex. x.name = time, y.name = speed -> col.name = acc
  #
  #   Returns:
  #     data: Data frame including new column with approximated derivative
  #
  # Note: Central differences scheme needs 'h' data points on each side when
      calculating for data point 'i'.
  #   Amendments are therefore made in each end of the vector where this is not the
      case, the scheme is regular in the middle.


  # Find columns matching x and y input
  index <- which(colnames(data) == x.name | colnames(data) == y.name)
  data[paste(col.name)] = 0
  h <- window


  for(skier in unique(data$skier)){
    df <- data[data$skier == skier, ] # Extract data for one skier

    for (lap in unique(data$lap)){
      x <- df[df$lap == lap, index[1]] # Calculate for each lap
      y <- df[df$lap == lap, index[2]]

      # Allocate
      deriv <- rep(0,length(x))


      ###  Approximate derivative  ###
      #Middle
      for(i in (h+1):(length(x)-h)){
        deriv[i] <- (y[i+h]-y[i-h])/(x[i+h]-x[i-h])
      }
      #Beginning
      for(i in 1:h){
        deriv[i] <- (y[i+h]-y[1])/(x[i+h]-x[1])
      }
```

```
50        #End
51        for(i in (length(x)-h+1):length(x)){
52          deriv[i] <- (y[length(x)]-y[i-h])/(x[length(x)]-x[i-h])
53        }
54        #
55        data[(data$lap==lap & data$skier==skier), paste(col.name)] = deriv
56
57        } # end for lap
58    } # end for skier
59
60    return(data)
61
62  } # end function
```

./../utils/smoothing_splines.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019


## Smoothing splines

SmoothingSplines <- function(x, y, data.frame, df){
  # Smooth y(x) by cubic smoothing splines
  #
  #   Args:
  #     x, y: String names of columns x and y to smooth, ex. x = "distance", y = "speed"
  #     data.frame: Data frame containing data x and y
  #     df: Degrees of freedom in the smoothing spline algorithm. Minimum 1, maximum 'n' = no. of observations
  #
  #   Returns: data.frame with smoothed data added in separate column

  data.frame[paste(y,"Smoothed",sep="")] = 0 # if y = "speed" then data.frame["speedSmoothed"] = 0 is set
  index <- which(colnames(data.frame)==x | colnames(data.frame)==y)
  #i=1
  for (skier in unique(data.frame$skier)){
    data <- data.frame[data.frame$skier == skier, ] # Data for one skier only
    print(data$skier[1])
    for (lap in unique(data$lap)){
      print(lap)
      xin <- data[data$lap==lap ,index[1]]
      yin <- data[data$lap==lap ,index[2]]
      fit <- smooth.spline(x=xin, y=yin, df=df) # Loop through laps because smooth.spline is sensitive to duplicate values / demands unique values
      data.frame[(data.frame$lap == lap & data.frame$skier == skier),paste(y,"Smoothed",sep="")] <- fit$y
    }
  }
  return(data.frame)


}
```

./../utils/drag_area_values.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

# TODO: Update documentation
# TODO: Decide how to include testing

GetDragAreaValueClassical <- function(subtechnique, testing = FALSE){
  # Determines drag contribution (drag area value CdA) based on subtechnique
  #   Values chosen based on article by Ainegren and Jonsson
  #   ("Drag area, frontal area and drag coefficient in cross-country skiing techniques")
  #
  #   Args:
  #     subtechnique: Vector with values 0,1,2,3
  #       0: Other (= tuck + turn)
  #       1: DP = Double poling
  #       2: DIA/DS = Diagonal stride
  #       3: DPK = Double pole kick
  #     testing: TRUE/FALSE
  #       If TRUE, drag area values are drawn from a normal distribution
  #       with given mean and standard deviation
  #
  #   Return: Vector of drag area values for classical subtechniques

  drag.area.values <- rep(0, length(subtechnique))
  subtechnique <- round(subtechnique)

  # Set CdA value
  if (testing == TRUE){
    # Draw from distribution
    drag.area.values[subtechnique == 0] <- rnorm(1, mean = 0.23, sd = 0.025) # Other (=tuck/turn)!!
    drag.area.values[subtechnique == 1] <- rnorm(1, mean = 0.44, sd = 0.025) # Double poling
    drag.area.values[subtechnique == 2] <- rnorm(1, mean = 0.54, sd = 0.025) # Diagonal stride
    drag.area.values[subtechnique == 3] <- rnorm(1, mean = 0.46, sd = 0.025) # Double poling w/kick

  } else {
    # Regular
    drag.area.values[subtechnique == 0] <- 0.23 # Other (=tuck/turn)
    drag.area.values[subtechnique == 1] <- 0.44 # Double poling
    drag.area.values[subtechnique == 2] <- 0.54 # Diagonal stride
    drag.area.values[subtechnique == 3] <- 0.46 # Double poling w/kick
  }

  return(drag.area.values)
}

```

```
47
48
49 GetDragAreaValueSkate <- function(speed, acd.upright, acd.tuck, testing = FALSE){
50   # Set drag area value CdA for each value in the speed vector.
51   # Value is determined by speed.
52   #
53   #   Args:
54   #     speed: Vector with speed values
55   #     acd.upright: CdA for upright position. This position is assumed for v < 10 m/
         s
56   #     acd.tuck: CdA for tucked position, which is assumed for v > 10 m/s
57   #     testing: TRUE/FALSE
58   #       If TRUE, drag area values are drawn from a normal distribution
59   #       with given mean and standard deviation
60   #
61   #   Returns: Vector of drag area values for skating technique
62
63   drag.area.values <- rep(0, length(speed))
64
65   if (testing == TRUE){
66     drag.area.values[speed > 10] <- rnorm(n = 1, mean = acd.tuck, sd = 0.025) # > 10
           m/s tuck is assumed
67     drag.area.values[speed <= 10] <- rnorm(n = 1, mean = acd.upright, sd = 0.075) # <
           10 m/s upright is assumed
68
69   } else {
70     drag.area.values[speed > 10] <- acd.tuck # > 10 m/s tuck is assumed
71     drag.area.values[speed <= 10] <- acd.upright # < 10 m/s upright is assumed
72   }
73
74   return(drag.area.values)
75
76
77 }
```

./../utils/model_calc.R

```
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

## Power balance model calculation ------------------------------------------------

model <- function(data, parameters, speed.type = "original", acc.type = "original",
    classical.or.skate = "skate", testing = FALSE){
  # Calculating propulsive power based on the power balance model
  #   P = efficiency + friction + gravity + drag
  #
  #   Args:
  #     data: Data frame with race data to compute power from
  #     parameters: List of model parameters g, CdA, rho and mu
  #     speed.type: Smoothed or original speed. Determines what speed values to base
      calculations on.
  #         Options: 1) "speedSmoothed" (smoothed by cubic splines) or 2) "original".
      Default is "original".
  #     acc.type: Smoothed or original acceleration.
  #         Options:
  #           1) "accSmoothed" (acc by original v, then smoothed),
  #           2) "accSpeedSmoothed" (acc calculated from smoothed speed) or
  #           3) "original" (acc calculated from original speed)
  #     classical.or.skate: "classical" or "skate", determining how to find drag area
      value based on subtechnique.
  #         "skate" is default.
  #     testing: Simulation on/off
  #         If testing = FALSE, model parameters are constant and predetermined
  #         If tesing = TRUE, model parameters are drawn from a normal distribution
      with
  #           mean = predetermined value and sd = chosen uncertainty (Only done for
      mass within this function)
  #
  #   Returns: Data frame 'data', now including power calculations

  # Sources
  source("./utils/drag_area_values.R")


  # Model parameters
  mu <- parameters[[1]]; g.acc <- parameters[[2]]; acd.upright <- parameters[[3]]
  acd.tuck <- parameters[[4]]; acd.avg <- parameters[[5]]; rho <- parameters[[6]]
  CdA <- parameters[[7]]


  # Allocate
  data['power'] = 0
  data['friction'] = 0
  data['gravity'] = 0
  data['drag'] = 0
  data['rel.power'] = 0 # Power relative to mass
```

```
46  data['CdA'] = CdA # List of CdA values found from subtechnique
47  data['kin.energy'] = 0
48
49
50  # Go through all skiers
51  for (i in unique(data$skier)){
52
53    # Determine speed
54    if(speed.type == "speedSmoothed"){
55      v <- data[data$skier==i,]$speedSmoothed
56      #print("speedSmoothed chosen")
57    } else {
58      v <- data[data$skier==i,]$speed
59      #print("Original speed chosen")
60    }
61
62    # Determine acceleration
63    if(acc.type == "accSmoothed"){
64      acc <- data[data$skier==i,]$accSmoothed
65      #print("accSmoothed chosen")
66    } else if (acc.type == "accSpeedSmoothed") {
67      acc <- data[data$skier==i,]$accSpeedSmoothed
68      #print("accSpeedSmoothed chosen")
69    } else {
70      acc <- data[data$skier==i,]$acc
71      #print("Original acc chosen")
72    }
73
74    # Extract from data frame before calculation
75    alpha <- data[data$skier==i, ]$angle
76
77    if (testing == TRUE){
78      m <- rnorm(n=1, mean = data[data$skier==i,]$mass[1], sd = 1)
79    } else {
80      m <- data[data$skier==i,]$mass[1]
81    }
82
83    ### CALCULATE POWER ###
84
85    # Terms separately
86    data[data$skier==i, ]$friction <- mu*m*g.acc*cos(alpha)*v # Friction
87    data[data$skier==i, ]$gravity <- m*g.acc*sin(alpha)*v  # Gravity
88    data[data$skier==i, ]$drag <- (1/2)*rho*data[data$skier==i, ]$CdA*(v^3) # Drag
89    data[data$skier==i, ]$kin.energy <- m*v*acc # Change in kinetic energy
90
91    # Propulsive power
92    data[data$skier==i,]$power <- m*v*acc + data[data$skier==i,]$friction + data[data
          $skier==i,]$gravity + data[data$skier==i,]$drag
93    #data[data$skier==i, ]$power <- m*v*acc + mu*m*g.acc*v + m*g.acc*sin(alpha)*v +
          (1/2)*rho*acd.avg*(v^3) # Calculate with avg. CdA
94    data[which(data$power<0), ]$power = 0 # Set negative power to zero
95    data[data$skier==i, ]$rel.power <- data[data$skier==i,]$power/m  # Power relative
```

```
          to skier mass
 96
 97
 98   } # end for
 99   return(data)
100
101 } # end function
```

./../testing/ParameterSensitivity.R

```
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019


ParameterSensitivity <- function(parameter, data.frame, parameter.vector, classical.or.skate, speed.type, acc.type){
  # Calculating power with variations in a single parameter at a time to evaluate sensitivity
  #
  #     Args:
  #       parameter: Which model parameter to vary
  #           Options: drag.area, friction, air.density, mass
  #       data.frame: Data set to evaluate
  #       parameter.vector: Vector of values to test
  #       classical.or.skate: Skiing technique in race in data.frame
  #
  #     Returns:
  #       Data frame res.frame with calculated power corresponding to the variations in the 'parameter' argument

  source("./utils/model_calc.R")
  res.frame <- data.frame()

  ### Drag area
  if (parameter == "drag.area"){
    len <- length(parameter.vector[[7]])
    drag <- parameter.vector[[7]]

    #drag.area.vecs <- list(rep(0.20, len), rep(0.40, len), rep(0.60, len)) # Test one
    #drag.area.vecs <- list(rep(0.30, len), rep(0.40, len), rep(0.50, len)) # Test two
    drag.area.vecs <- list(rep(0.35, len), rep(0.45, len), rep(0.50, len)) # Test three


    for (par in drag.area.vecs){
      parameter.vector[[7]] <- par
      data <- model(data.frame, parameter.vector, speed.type = speed.type, acc.type = acc.type, classical.or.skate = classical.or.skate, testing = FALSE)
      res.frame <- rbind(res.frame,
          data.frame(power = data$power, rel.power = data$rel.power, distance = data$distance, drag.area = data$CdA,
          skier = data$skier, elevation = data$elevation, drag =data$drag,
              speedSmoothed = data$speedSmoothed))
    }
    return(res.frame)


  ### Friction
```

```
43  } else if (parameter == "friction"){
44    mu <- data.frame$mu[1]
45    friction <- c(mu-0.005, mu, mu+0.005, mu+0.010) # Test 1
46    #friction <- c(mu-0.0025, mu, mu+0.0025, mu+0.0050) # Test 2
47    #friction <- c(mu-0.0025, mu, mu+0.0025) # Test 3
48
49    for (par in friction){
50      parameter.vector[[1]] <- par
51      data <- model(data.frame, parameter.vector, speed.type = speed.type, acc.type =
            acc.type, classical.or.skate = classical.or.skate, testing = FALSE)
52      res.frame <- rbind(res.frame,
53          data.frame(power = data$power, rel.power = data$rel.power, distance = data$
                distance, friction = data$friction, mu = par,
54                     skier = data$skier, elevation = data$elevation))
55    }
56    return(res.frame)
57
58
59
60  ### Air density
61  } else if (parameter == "air.density"){
62    air.density <- c(0.9, 1.1, 1.3) # Test 1
63    #air.density <- c(1.0, 1.1, 1.2) # Test 2
64    #air.density <- c(1.05, 1.1, 1.15) # Test 3
65    for(rho in air.density){
66      parameter.vector[[6]] <- rho
67      data <- model(data.frame, parameter.vector, speed.type = speed.type, acc.type =
            acc.type, classical.or.skate = classical.or.skate, testing = FALSE)
68      res.frame <- rbind(res.frame,
69          data.frame(power = data$power, rel.power = data$rel.power, distance = data$
                distance, drag = data$drag, rho = rho,
70          skier = data$skier, elevation = data$elevation))
71
72    }
73    return(res.frame)
74
75
76
77  } else if (parameter == "mass"){
78    kgs <- data.frame$mass[1]
79    skier.mass <- c(kgs-2, kgs, kgs+2) # Test 1
80    #skier.mass <- c(kgs-1, kgs, kgs+1) # Test 2
81    #skier.mass <- c(kgs-0.5, kgs, kgs+0.5) # Test 3
82
83    for(mass in skier.mass){
84      data.frame$mass <- mass
85      data <- model(data.frame, parameter.vector, speed.type = speed.type, acc.type =
            acc.type, classical.or.skate, testing = FALSE)
86      res.frame <- rbind(res.frame,
87          data.frame(power = data$power, rel.power = data$rel.power, distance = data$
                distance, skier = data$skier, elevation = data$elevation, mass = data$
                mass))
```

```
    }

    return(res.frame)
  } # end if

} # end function
```

./../testing/MonteCarloSimulation.R

```r
# Master thesis, Gina Magnussen
# In collaboration with SenTIF Trondheim
# Spring 2019

MonteCarloSimulation <- function(num.runs, data.frame, classical.or.skate, speed.type, acc.type){
  # Simulating propulsive power by varying model parameters: Friction, drag area, rho,
  # and mass (done in model() (model_calc.R))
  #
  #     Args:
  #       num.runs: Number of simulation runs
  #       data.frame: Data frame with race data to base simulations on
  #       classical.or.skate: Technique of race analyzed - "classical" or "skate"
  #       speed.type: Type of speed used in model calculation,
  #           1) "original" (unfiltered speed) or 2) "speedSmoothed" (speed smoothed by smoothing splines)
  #       acc.type: Type of acceleration used in model calculation,
  #           1) "original" (acc calculated from unfiltered speed) or 2) "accSpeedSmoothed" (acc calculated from smoothed speed)
  #
  #     Returns:
  #       res.frame: Data frame with simulation data
  #



  source("./utils/model_calc.R")
  source("./utils/drag_area_values.R")
  library(tictoc)
  tic()
  res.frame <- data.frame()

  for(run in 1:num.runs){
    # Put together the parameter vector
      friction <- rnorm(n = 1, mean = data.frame$mu[1], sd = 0.0025)
      rho <- rnorm(n=1, mean=1.1, sd = 0.1)#??
      if (classical.or.skate == "classical"){
        CdA <- GetDragAreaValueClassical(data.frame$subtechnique, testing = TRUE)
      } else if (classical.or.skate == "skate"){
        if(speed.type == "original"){
          CdA <- GetDragAreaValueSkate(data.frame$speed, 0.5, 0.23, testing = TRUE)
        } else if (speed.type == "speedSmoothed"){
          CdA <- GetDragAreaValueSkate(data.frame$speedSmoothed, 0.50, 0.23, testing = TRUE)
        }
      }

      par <- list(friction, 9.81, 0.50, 0.23, 0.365, rho, CdA)
      # 9.81 = g.acc, 0.50 = acd.upright, 0.23 = acd.tuck, 0.365 = acd.avg


```

```
47      # Calculate power
48        model.calc <- model(data.frame,
49              par,
50              speed.type = speed.type,
51              acc.type = acc.type,
52              classical.or.skate = classical.or.skate,
53              testing = TRUE)
54      #print(run)
55      res.frame <- rbind(res.frame, data.frame(skier = model.calc$skier, power = model.
            calc$power,
56                          rel.power = model.calc$rel.power, distance = model.calc$
                                distance,
57                          drag.area = model.calc$CdA, run = run, lap=model.calc$lap))
58
59    } # end for
60
61    toc()
62    return(res.frame)
63
64  } # end function
```