University of Edinburgh

School of Physics and Astronomy

Senior Honours Project Report

Cycling Performance in Alpe d'Huez

Author
Lewis Trainer

Supervisor
Miguel Martinez-Canales
November 27, 2020

Abstract

The purpose of this study is to create an accurate model of elite level cyclists in a racing environment. The aim is to combine both physical and physiological aspects accurately in order to return a model that can be used by cycling teams to formulate technical tactics and racing strategies, with the objective of maximising their chances of winning both individual races and over prolonged racing events, i.e. a Grand Tour event that lasts 3 weeks. The model will be considering elements of racing such as reduction of drag due to pelotons, as well as relying on a model indicating the maximum power outputs that a rider can achieve in any racing environment and implementing various rider profiles. Successful employments of the model include calculating the base power input of a peloton needed to ensure a solo breakaway is not possible and identifying which rider profile applying maximum powers at certain legs of a race will result in the largest reduction of time.

"Life is like a bicycle. To keep your balance, you must keep moving."
- Albert Einstein

Contents

1 Introduction

In order to move, the human body must access its energy stores and do work. A certain amount of power must be produced by the human body in order to do this work. Elite athletes operate at the limits of how much work the human body can do and when doing so the human body must obey the laws of thermodynamics. This is the core of what restricts the physical limits of elite level athletes. In addition to these limits, one must also understand how to maximise performance within these limits, elite level athletes and their respective teams can better strategise plans for competition and prioritise the specific areas in which an athlete can carry out the most work in order to produce the best results.

This paper will look specifically into modelling the performance of elite level cyclists in tour like conditions, and will provide examples of how such examples can be used by a rider and their team in a tactical context. The model will be concerned with both the physical and physiological constraints on a rider at any time point during a race.

Physically, cycling can be modelled very accurately, only considering three forces to calculate the power of our cyclist; Aerodynamic Drag, Rolling Resistance and Gravity. Both rolling resistance and Gravity can be found by simply analysing data about the cyclist and their environment. Aerodynamic drag is slightly more complex as is proportional to the square of the velocity of the cyclist so this velocity must be known in order to calculate the aerodynamic drag. Only needing these three forces is useful as it allows us to work in multiple ways: if a rider supplies a certain amount of power in certain conditions, it is possible to calculate how fast they will travel. And if a rider is to move at a certain velocity in certain conditions, it is possible to calculate how much power they must expend in order to do so. It is worth noting that only gravity relies on the mass of the cyclist. This is why riders from different disciplines within the sport have slightly different builds.

Physiologically, modelling a cyclist is no different from modelling any other athlete. There is a maximum power output that every athlete can achieve and there is a limit to the time that this power can be sustained. The Critical Power (CP) model[1] states that there is a hyperbolic relationship between power output and the time it can be sustained. The asymptote of this hyperbola is known as the Critical Power, and can theoretically be taken as the power output that can be sustained by an athlete, indefinitely, without exhaustion. The CP model predicts that there is a limited amount of work that can be done above the CP (W'), and once this limit is achieved, no more work can be attained above CP (the athlete can no longer output power greater than the CP). Figure 1 shows the hyperbolic relationship between power output and duration of sustainability, along with an indicator of the CP asymptote. Given accurate data of a rider and their maximum sustainable power output over any duration of time, along with a reasonable estimate for their personal W' reserve, this model can be applied to real time specific events in order to calculate the speed of riders either instantaneously or over a specified period of time. Along with having differing physical appearances, riders whom specialise in varying disciplines will also have varying physiological differences, and the time to exhaustion for certain power outputs differs from discipline to discipline. For example, an elite sprinter will have much higher maximum

power capacities for short periods of time (< 5"), whereas an elite endurance rider will be able to sustain slightly higher powers for a much longer periods.

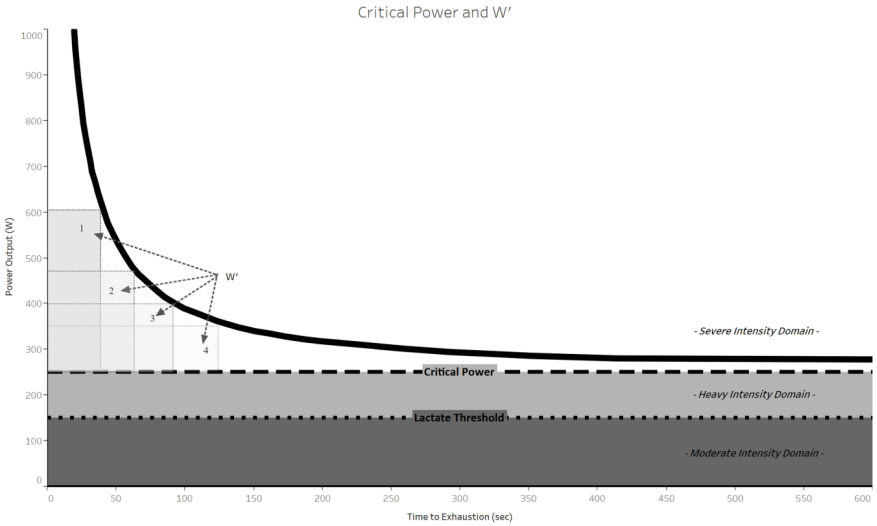

Figure 1: Schematic representation of the CP model of the power-duration relationship. Note the hyperbolic shape of the power-duration curve (thick black line) and that the four rectangles demarcating W' have the same area, reflecting that W' is a constant amount of work that can be done above CP.[2]

Cycling grand tours are not only widely regarded within the sport as the pinnacle of cycling racing, but are also some of the most supported and beloved sporting events in the world, with the Tour de France being the most widely attended sporting event worldwide.[3] A Grand Tour event will last 3 weeks. Being able to create a plan that maximises success whilst minimising fatigue for each individual race is of the upmost importance to riders and their respective teams.

2 Background

Developing and implementing a model to effectively formulate racing strategies involves separating the into two categories; the *physical* model and the *physiological* model. The physical model involved modelling the forces acting on the rider and their environment in a race-like context. The physiological aspect of the model was a direct implementation of the CP model. The combination of these two models allows for an accurate tool to simulate potential outcomes of various cycling contexts.

2.1 The Physical Model

Implementing the physical model involved modelling the physical forces felt by a rider at any given time. Modelling these forces required information about the course and its environment, as well as physical attributes of the rider. The three forces that act on a rider at any given time are the aerodynamic drag, the frictional force due to rolling and gravity.

2.1.1 Forces Acting on the Rider

Aerodynamic drag, also known as wind resistance or the drag force, is a force in fluid dynamics which acts opposite to the relative motion of a moving object with respect to the fluid that surrounds it. Specifically it is caused by the density of the fluid and the speed and shape of the object which is moving through it. In the context of the cyclist, this aerodynamic drag is caused by the density of the air (fluid) in which the cyclist is moving, the relative motion of the cyclist (velocity), and the effective frontal area of the cyclist and their bike. The drag equation is given as the following:

$$F_D = \frac{1}{2}\rho v^2 C_D A_f \tag{1}$$

where F_D is the drag force, ρ is the density of the fluid (in this case the air density), v is the velocity of the object (cyclist), C_D is the drag coefficient, and A_f is the effective frontal area of the object (rider).

The rolling resistance, or rolling friction, is the force that resists the motion of a body rolling on a surface and generally acts in a direction opposite to the relative motion of the moving object. It is caused by the weight of the moving body and the coefficient of rolling friction between the surfaces of contact of the body and the surface. In the context of a cyclist the rolling resistance is caused by the weight of the rider and the coefficient of rolling friction between the tyres of the bicycle and the surface of the track on which the cyclist is riding. The rolling resistance is expressed as the following:

$$F_r = C_f m a_g \tag{2}$$

where F_r is the force of rolling resistance, C_f is the coefficient of rolling friction, m is the mass of the object(rider) and a_g is the acceleration of the object due to gravity.

The weight of an object can be thought of as the force exerted on a body due to gravity and it acts in the direction of the acceleration of the body due to gravity. This is often seen expressed as the following:

$$W = m a_g \tag{3}$$

where W is the weight of the object, m is the mass of the object and a_g is the acceleration of the object due to gravity.

A stationary object on a flat surface when acted on by a constant acceleration due to gravity, will not be moved by the weight. However, if the object is on a surface that is not flat, then the force exerted on the body due to gravity will have components that act in multiple directions. On an inclined plane the weight has two components: (1) the perpendicular force, which is the component of the force that pushes the object into the surface and acts perpendicularly to the surface; and (2) the parallel force, which is the component of the force that pulls the object down the inclined surface.

In the context of a cyclist, the parallel force is felt by the rider. The parallel force is expressed as the following:

$$F_g = ma_g sin\theta \tag{4}$$

where F_g is the parallel component of the weight due to gravity, m is the mass of the object(rider), a_g is the acceleration of the object due to gravity, and θ is the angle of incline of the surface.

Combining these three forces with the relationship:

$$P = F_{tot}v \tag{5}$$

where P is the power needed to overcome an applied force F_{tot} in order to move at velocity v and $F_{tot} = F_D + F_r + F_g$, provides a model which can both return the speed of a rider given a power and also the power needed to move at a specific speed. Using this equation, allows an unknown speed to be calculated from a known power and vice versa.

2.1.2 The Peloton

A cycling peloton is the main pack of riders in a race. The riders form closely packed formations, in order to conserve energy by creating various slip streams or drafts to reduce drag. A paper from Blocken et al. (2018)[4] discusses the use of Computational Fluid Dynamics (CFD) simulation and wind tunnel testing in order to accurately model the reduction in aerodynamic drag on each cyclist in the peloton, exploring varying sizes and formations of riders, from 4-rider to 121-rider variations, shown in figure 2. In the context of racing, the speed of the pack is dictated by the front rider, as it is this rider that will feel the smallest reduction in aerodynamic drag, and therefore the most work is being done by this rider. Throughout a race, cyclists will switch positions in the peloton in order to allow for recovery. Because of this, using the front rider as an average for our peloton is reasonable. The Blocken and colleagues study shows that the overall effects of a peloton on the drag of the front-most rider is significant, with a aerodynamic drag reduction of 14% being experienced by the leading rider of the 121-rider peloton shown in figure 2. This reduction is also significant for smaller groups of riders, with the front-most rider of a 4-rider peloton in the formation shown in figure 2 experiencing a drag reduction of 4%.

At racing speeds ($\approx 54km/h$) aerodynamic drag accounts for $\approx 90\%$ of the total drag force felt by the rider.[5] Because of this, the aforementioned reductions display the importance of the peloton in long, multistage races such as Grand Tours, as they allow athletes to preserve energy for critical legs of races, and limit energy consumption.

2.2 The Physiological Model

The physiological model involves a direct implementation of the power output of the rider based on the CP model. This model states that every rider has a specific capacity of work that can be done above their CP. When the CP is exceeded, the remaining

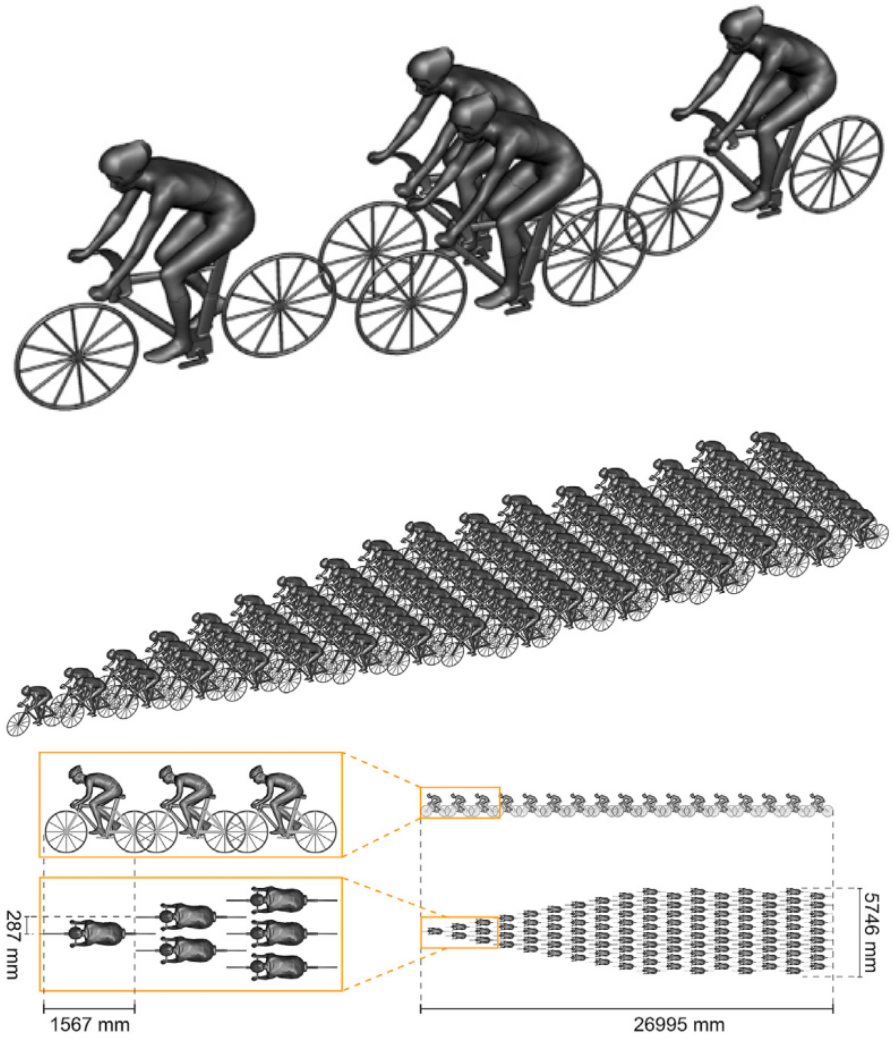

Figure 2: Graphical depiction of peloton configurations used in Blocken et al. study to determine reduction in aerodynamic drag due to peloton. Images taken directly from Blocken et al. paper.[4]

W' of the rider is reduced. When the capacity of W' reaches 0 the rider can no longer sustain any work above their CP. In theory once this has been achieved the rider should still be able to function at their CP, but in reality it is markedly reduced, although there is little reliable data on how much it is reduced by.[2] The hyperbolic relationship shown in figure 1 can be represented by the following equation:

$$P = \frac{W'}{t} + CP \tag{6}$$

in which P = power output, t = time to exhaustion, CP is the critical power (the asymptote shown in Figure 1), and W' is the remainder of work that can be performed above CP.[6]

2.3 Rider Profiles

As mentioned previously, riders who specialise in different disciplines, have different attributes, both physically and physiologically. A rider can fall under a number of categories within a racing team; sprinter, climber, puncheur, all-rounder, time trialist and domestique. This study will focus specifically on the all-rounder, sprinter and climber.

The typical all-rounder has a mass of $72kg$ with bike included, their height is approximately $1.83m$. There is no obvious correlation between a riders height and their speciality discipline, although there is a correlation between their mass. As the climbers goal is to climb, their biggest enemy is gravity. As a result they typically have less mass than their counterparts, with an average mass of $69kg$ with bike included. A sprinter requires more explosive power than their counterparts, which results in a larger muscle mass, resulting in an average mass of $75kg$ with bike included. As we have assumed that our riders will be approximately the same size, we will also assume that they have the same frontal area and as a result the same coefficient of drag. Various studies have been conducted to find the best methods to calculate these numbers, with Blocken et al.[4] and Chowdury et al.[7] studies giving reasonable experimental evaluations of $A_f \approx 0.40m^2$ and $C_D \approx 0.86$. These are the values which will be used throughout this study.

Phsyiologically, these rider profiles differ. For the purpose of this study we are concerned with the record power profiles of specific riders. Pinot and Grappe's 2011 study[8] into assessing the power profile of elite cyclists provides useful values for the record power obtained by a selection of riders over specific periods of time. The riders assessed in the study included amateur and professional riders, therefore the upper bound of the standard deviation will be taken for any power results, as this will most likely give a more accurate representation of performance of a professional cyclist. This study provides our numbers for the CP of our all-rounder (360W). As this paper includes a number of riders with different specialties, it will form the basis of our all-rounder. The paper states that climbers have the highest record power output over times of $30" - 45"$ (375W) whilst sprinters have the highest record power output over smaller time intervals ($1' - 5"$). The other physiological aspect which must be considered for the purpose of this study is the riders W' reserve. A 2006 study[9] found that a reasonable estimate for the W' of a cyclist is $18.5kJ$. There is little reliable experimental data on elite level athletes in assessing this number, so for the purpose of this study the value of $W' = 18.5kJ$ will be used for the all-rounder. The Typically a sprinter will have a much larger W' of $\approx 22.0kJ$ (with a reduced CP of 350W), with a climber sharing the value of $W' = 18.5kJ$ with the all-rounder.

2.4 Alpe d'Huez

Alpe d'Huez is considered by many to be the most iconic climb in all of cycling. The route is $13.1km$ long, setting off from the French commune of Le Bourg-d'Ouisans at an altitude of $730m$ and writhing up the Alps into the ski resort of Alpe d'Huez at an altitude of $1803m$, giving a total altitude gain of almost $1.1km$. The route wriggles through a total of 21 hairpin turns, resulting in one of the most scenic cycling routes in the world. The climb has seen some of the most famous cycling performances in Tour

de France history, cementing the routes place as a holy grail of competitive cycling.[10]

Alpe d'Huez lends itself particularly well for modelling as it has a near constant slope rating from start to finish, with the initial, steepest leg of the route having a slope of 10.6% and the final, shallowest leg of the route have a slope of 4.3%, with the average slope over the entire course being 8.19%.

The model assumes a number of constants based on the environment of Alpe d'Huez; gravitational acceleration = $9.807ms^{-2}$, air density = $1.226kgm^{-3}$ and frictional coefficient of an asphalt road = 0.005.

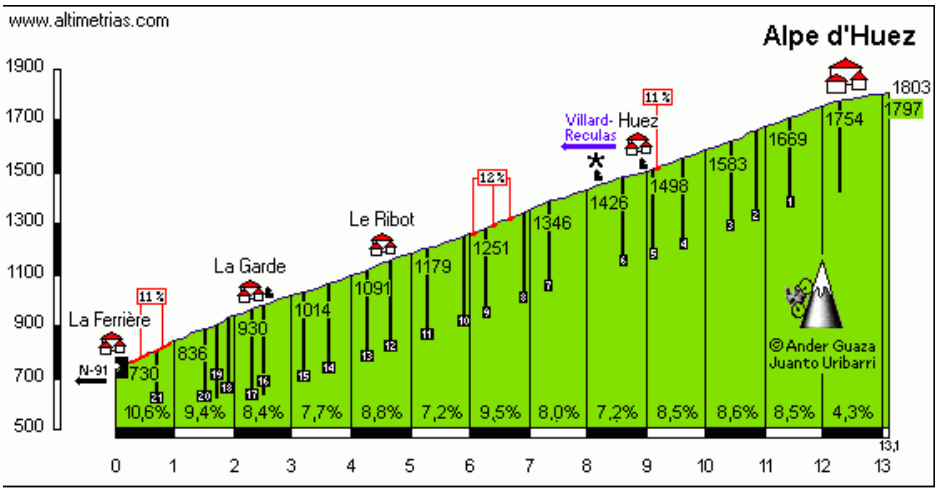

Figure 3: Break down of average slope rating per km over the entirety of Alpe d'Huez route.[10]

This near constant slope is useful for analysis, as an almost steady power output must be supplied by riders in order to maintain a steady pace throughout the climb. There are no sections of the climb in which the rider is not exerting energy, meaning results will give a clear correlation between power exertion and success.

All results gathered in this study were done using Alpe d'Huez as the course, unless explicitly stated otherwise.

3 Implementation

The model was implemented via an object-oriented based program written in Python. There are three main files in which the model was implemented, including a class for the rider being modelled, a class for the route being modelled, and a file containing functions in which quantitative analysis is completed. The user inputs the data of the cyclist and the course via .txt files, and this data is assigned to the respective classes as attributes. The cyclist input files are formatted as such: All data is displayed in one row, with each column giving a different attribute of the rider; Height(m), Mass(kg), Effective Frontal Area(m^2), Coefficient of Drag, W' (J), and CP (W). The route input files are formatted as such: Each row represents a division or leg of a race, with the first column giving the slope rating of the leg, and the second column giving the distance

(m) of the leg. (See appendix C for example of how a new instance of the class is created and see appendices A and B for class initialisers.)

3.1 Cyclist.py

The cyclist class contains the following basic functions to calculate the total combined force that a rider must provide in order to move at a certain speed, thus allowing for a calculation of the power needed to do so:

- A function that returns the aerodynamic drag that a rider must overcome in order to travel at a given speed. (1)
- A function that returns the frictional rolling force felt by the rider. (2)
- A function that returns the gravitational force felt by the rider at any point on the course. (4)
- A function that returns the total combined force that the rider must provide in order to move.
- A function that returns the power needed to provide the force needed to move at a specific speed. (5)

In order to calculate how fast the rider will be moving given specific power involves rearranging equation (5) for v. Doing so and combining with our eqautions of our three forces results in a v^3 polynomial. Therefore the following functions were implemented:

- A function that returns the coefficient of v^3 of our polynomial based on (1).
- A function that returns the coefficient of v of our polynomial based on (2) (4).
- A function that uses numpy.roots to solve our v^3 polynomial in order to return the speed that a rider would travel for a given power. (5)

The main functions of the class are those which return a speed given a power and power given a speed, so for functionality purposes if the user wishes to use these functions they can be called independently and contain the other functions within the class.

The Cyclist.py class can be seen in appendix A.

3.2 Route.py

The route class contains only the initialiser function and serves the sole purpose of storing information about the route and to provide structure to the code.

The Route.py class can be seen in appendix B.

3.3 Main.py

The main.py file contains a number of functions, with the majority of the functions implemented with the purpose of plotting various graphs from data files and calling

on functions from the cyclist class in order to obtain specific data for specific contexts.

3.3.1 Power vs. Time to Finish

A function created in order to return a list of times to completion of a course, when given a list of different steady powers to be supplied. For each power in the list, the time to complete each leg of the route is calculated and combined with the previous leg times, resulting in a total time to completion for that power.

3.3.2 Unsteady Leg Times

A function was implemented to allow the user to input an unsteady power for the rider. The user inputs a list of powers with the list length corresponding to the number of legs in the course, i.e. the power the rider will expend over each individual length. This allows the user to choose which legs to prioritise power out and eventually W' expenditure.

The function operates as follows:

The user passes in the cyclist and route instances it wishes to evaluate, along with the list of power inputs and filenames it wishes to save the data to. A list to store the remaining W' after each leg is created, along with a list to store the total time taken after each leg and the power output of each leg. These lists are initialised with the initial W' of the rider, the initial power from the list of inputs and zero respectively. The function then loops through the number of legs in the course and for each leg carries out the following pseudo-code calculations:

```
calculate speed from power input
calculate time t to complete leg at speed calculated
if power input > critical power:
    possible time to sustain = W' / (power - critical power)
    if t > possible time to sustain:
        possible distance = speed * possible t
        calculate remaining distance
        using critical power calculate time to complete
            remaining distance, rt
        leg time = rt + possible t
        set W' to 0
        append critical power to list of actual power outputs
    elif t == possible time to sustain:
        leg time = t
        set W' to 0
        append critical power to list of actual power outputs
    else:
        leg time = t
        set W' to W' - W' usage
        append power input to list of power outputs
    add W' to W' list
```

```
else:
    add W' to W' list
    leg time = t
    append power input to list of power outputs
```

This pseudo code checks if the rider is operating above CP. If this is the case, the time that this power can be sustained for is calculated using W' and the remaining W' and leg times are calculated. Once W' reaches 0 the riders power is limited to a maximum of their CP for the remainder of the course. In reality this may be reduced considerably below their CP but as their is no reliable numbers for this the theoretical assumption is instead implemented.

The code implemented in main.py can be seen in appendix C.

4 Results and Discussion

This section will present results of the model implemented in the context of racing tactics, and discuss how the accuracy of these results can be improved.

4.1 Steady Power

Figure 4 shows the effects of aerodynamic drag reduction of a large peloton (121 rider) when compared to a solo rider, with power being the function of time to completion of the demonstrated course. For each power value it is assumed that this power is steady throughout completion of the course. At a power of 450 W the times to completion were the following; 31mins 47 secs for the peloton and 32 mins 14 seconds for the solo rider.

Table 1 shows a number of results calculated of the time to finish for Alpe d'Huez for a large selection of riders and peloton configurations. Some notable results that can be obtained from this data set are as follows; We can see that if a solo climber wishes to break from the pack and perform at their CP, the risk of them being overtaken by the pack is significantly small, despite the pack performing at their maximum W'. We can also see that if a solo climber wishes to break from the pack, then the most effective tactic to reduce the deficit would be to allow a 4-rider peloton of climbers to also break away from the pack, although allowing another solo climber will still be favourable to doing nothing. This will reduce the time deficit of the solo climber by 8 seconds.

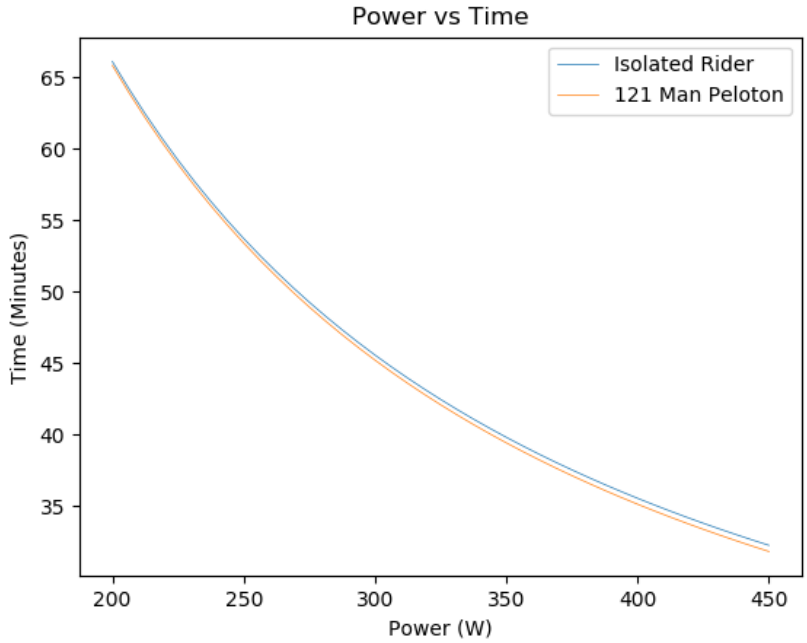

Figure 4: Power vs. Time to Finish for a number of steady power inputs, for both a solo rider and a large peloton (121 rider).

Rider Profile	Riders	Steady Power (W)	Time to Completion (mins/secs)
All-Rounder	1	360 (CP)	41' 00"
	1	367 (W')	40' 21"
	4	360 (CP)	40' 53"
	4	367 (W')	40' 13"
	121	360 (CP)	40' 33"
	121	367 (W')	39' 53"
Climber	1	375 (CP)	38' 22"
	1	382 (W')	37' 48"
	4	375 (CP)	38' 14"
	4	382 (W')	37' 36"

Table 1: Time to Finish for a number of rider profiles and peloton configurations @ both CP and maximum W' on Alpe d'Huez.

4.2 Unsteady Power

In reality, it is unlikely that a rider/rider team will maintain a steady power throughout an entire race. It is important to analyse which areas of the race are most effective to exert power.

Figure 5 displays the relationship between W' of a rider as a function of time as well as the riders unsteady power input as a function of time. These results were gathered over a flat 1000m track. It is clear that a greater increase in power above the riders critical power results in a notable reduction in remaining W' for the rider. This is a

real time representation of the CP model of an athlete.

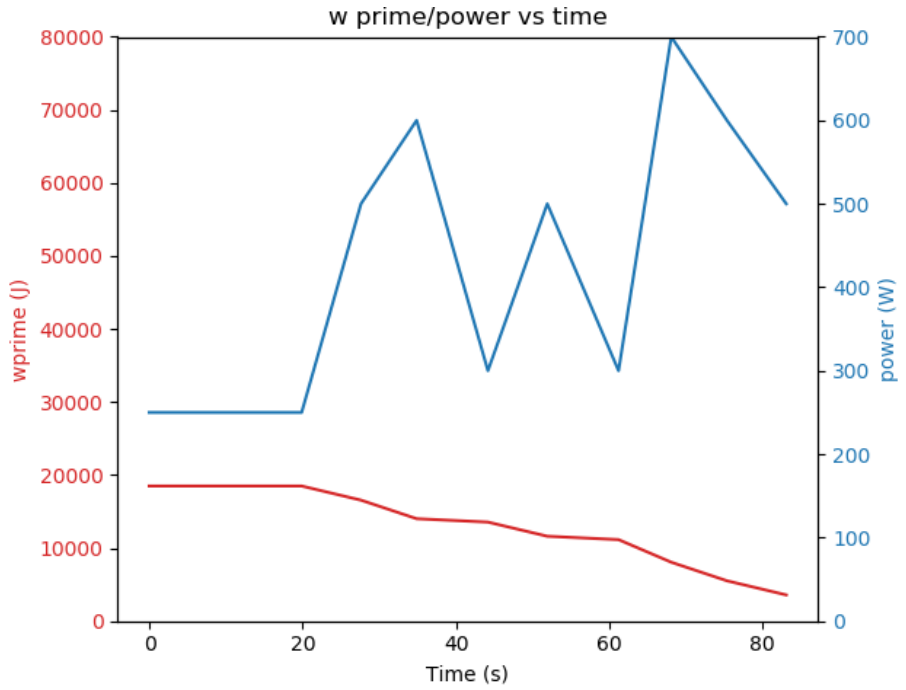

Figure 5: Power input vs Time plotted over Remaining W' vs Time shows the relationship that working above the riders CP has on the riders W'. In this example the riders CP is 250W. Flat 1000m course.

Five sections of the Alpe d'Huez course with the steepest slope were identified 3. A pair of solo climbers were pitted against each other, with one rider applying a constant power of 382W (max W'), whilst the second climber maximises W' on only the 5 sections identified, and operates at CP throughout the remaining legs. Table 2 displays the time to completion for both riders, whilst Figure 6 compares the time to completion of each individual leg in the race. The significant result is that the rider supplying an unsteady power input will in fact complete the course in a reduced time, despite only completing 5 legs of the race in a reduced time to their steady counterpart.

Rider Profile	Riders	Power	Time to Completion (mins/secs)
Climber	1	Unsteady	38' 22"
	1	Steady	37' 48"

Table 2: Time to Finish for a solo climber operating at a steady power (maximum W') vs applying max W' over 5 identified legs on Alpe d'Huez.

Sprinters will typically reserve their W' until the final moments of a race. Figure 7 displays the typical Power vs Time and W' vs Time functions for a sprinter throughout a race. It is assumed that a sprinter will never be the front rider of a peloton due to their reduced CP. For this result it was assumed that the sprinter will always occupy the position in the peloton experiencing the least aerodynamic drag.[4] This allows the

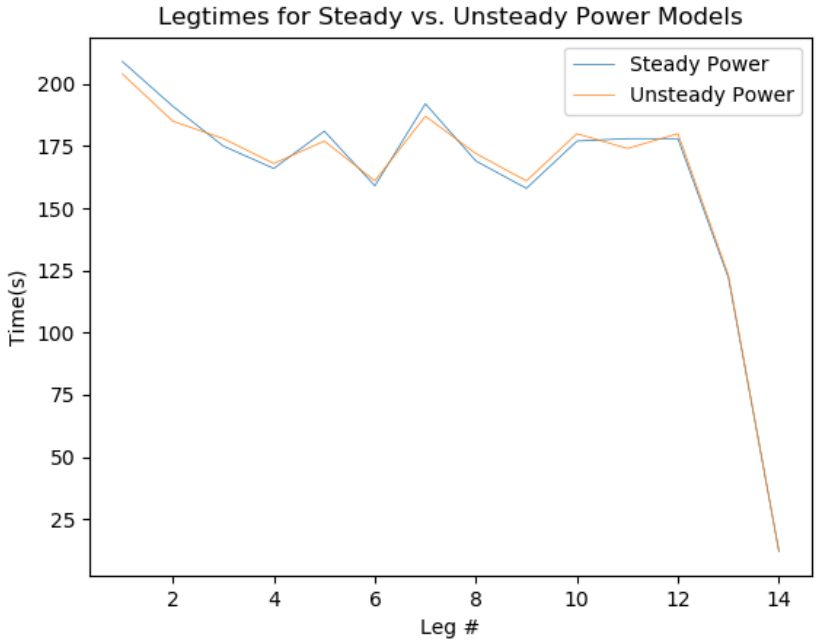

Figure 6: Legtimes for Steady vs. Unsteady Power Inputs of a solo climber on Alpe d'Huez.

entirety of their W' to be reserved for the final sprint. The final 2 legs of the Alpe d'Huez course were identified for the sprint, and the sprinters W' was maximised in order to find the time to completion of the final 2 legs. It is assumed that when the sprint is taking place, the rider is isolated from any pelotons. Table 3 displays various times to completion of the final 2 legs of the course for a number of rider configurations. From this data, we can deduce that if, for example, a 4-rider peloton were to break away, containing 3 climbers and a sprinter and operated for the initial 12 legs of the course at the climbers max W' with the sprinter remaining in the back position of the peloton, they could potentially gather a 16 second advantage over a 4-rider peloton breakaway containing only climbers.

Rider(s) and Power Model	Time to Completion (mins/secs)
SS (Max W' for final 2 Legs only)	113"
SC (Max W', Steady)	134"
4PC (Max W', Steady)	129"
121P (CP, Steady)	139"

Table 3: Time to Completion of final 2 legs for a number of rider profiles and peloton configurations @ both CP and maximum W' on Alpe d'Huez. (SS = Solo Sprinter, SC = Solo Climber, 4PC = 4-Rider Peloton of Climbers, 121P = 121-Rider Peloton)

4.3 Discussion

The overall effectiveness and usability of the model relies heavily on the accuracy of data present on the rider and their opponent. This set of results prove that with ac-

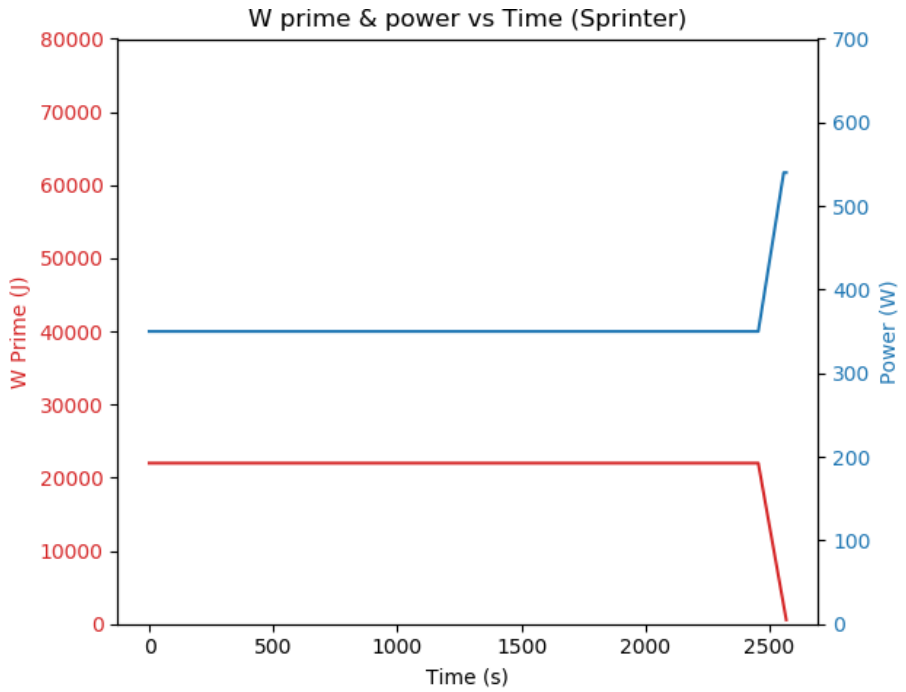

Figure 7: Power input vs Time plotted over Remaining W' vs Time for a solo sprinter expending all W' in the final 1.1km.

curate and reliable data, the model can be effectively and systematically implemented in the planning of a single race or grand tour event. The results specified in this study may not be specifically relevant to a rider/rider team, but they do demonstrate that the model can be used and implemented by tactical planners, coaches and sports analysts etc. in many different varieties in order to simulate potential outcomes and racing plans.

There are a number of improvements and additions that could be made to the model, in order to increase the accuracy of results and specific race scenarios. The model does not take into account track bends and the forces acting on a rider in these moments. The model does not take into account cross-winds and other possible adverse weather effects.

Another way to improve the model is to implement other possible physiological models, such as a model for recovery mid race. With pelotons providing a large reduction in aerodynamic drag for certain rider positions, and many cycling courses containing large downhill sections, this would, in reality, allow many opportunities for a rider to recover W' mid race, allowing further leg time reductions. These elements would be very useful to implement into the model and would complement the current applications and models implemented effectively.

5 Conclusion

The development an accurate model of elite level cyclists in a racing environment was successfully demonstrated. Both physical and physiological aspects were combined in

order to produce a model that can be used by cycling teams to formulate technical tactics and racing strategies, with the objective of optimizing their chances of winning both individual races and over prolonged racing events. The model considers elements of racing such as reduction of drag due to pelotons, as well as implementing the CP model which indicating the maximum power outputs that a rider can achieve in any racing environment. The model also implements various rider profiles. Successful employments of the model include calculating the base power input of a solo breakaway needed to ensure a large peloton keeping pace is not possible, and to identifying which rider profile applies maximum powers at certain legs of a race. In turn, this will result in the largest reduction of time, and/or reduction in time deficits of opposition riders.

References

[1] David W Hill. The critical power concept. *Sports medicine*, 16(4):237–254, 1993.

[2] Michael J. Puchowicz, Eliran Mizelman, Assaf Yogev, Michael S. Koehle, Nathan E. Townsend, and David C. Clarke. The critical power model as a potential tool for anti-doping. *Frontiers in Physiology*, 9:643, 2018.

[3] Tour de France. https://www.letour.fr/en/ Accessed: 01/04/2020.

[4] Bert Blocken, Thijs [van Druenen], Yasin Toparlar, Fabio Malizia, Paul Mannion, Thomas Andrianne, Thierry Marchal, Geert-Jan Maas, and Jan Diepens. Aerodynamic drag in cycling pelotons: New insights by cfd simulation and wind tunnel testing. *Journal of Wind Engineering and Industrial Aerodynamics*, 179:319 – 337, 2018.

[5] Frederic Grappe, Robin Candau, Alain Belli, and Jean Denis Rouillon. Aerodynamic drag in field cycling with special reference to the obree's position. *Ergonomics*, 40(12):1299–1311, 1997.

[6] Toshio Moritani, Akira Nagata, Herrfrt A. DeVries, and Masuo Muro. Critical power as a measure of physical work capacity and anaerobic threshold. *Ergonomics*, 24(5):339–350, 1981. PMID: 7262059.

[7] Harun Chowdhury and Firoz Alam. Bicycle aerodynamics: an experimental evaluation methodology. *Sports Engineering*, 15(2):73–80, 2012.

[8] J Pinot and F Grappe. The record power profile to assess performance in elite cyclists. *International journal of sports medicine*, 32(11):839–844, 2011.

[9] Anni Vanhatalo, Jonathan H Doust, and Mark Burnley. Determination of critical power using a 3-min all-out cycling test. *Medicine & Science in Sports & Exercise*, 39(3):548–555, 2007.

[10] Alpe d'huez. https://www.altimetrias.net/aspbk/verPuertoF.asp?id=5 Accessed: 01/04/2020.

A Cyclist.py

```
import numpy as np

class Cyclist(object):

    def __init__(self, profile, peloton):
        if profile == 'a':
            file = "cyclist_files/all_rounder.txt"
        elif profile == 's':
            file = "cyclist_files/sprinter.txt"
        elif profile == 'c':
            file = "cyclist_files/climber.txt"
```

```
        self.height = np.loadtxt(file)[0]
        self.mass = np.loadtxt(file)[1]
        self.frontal = np.loadtxt(file)[2]
        self.drag = np.loadtxt(file)[3]
        self.w_prime = np.loadtxt(file)[4]
        self.critical_power = np.loadtxt(file)[5]

        if peloton == 'n':
            self.peloton_coefficient = 1.0
        elif peloton == 's':
            self.peloton_coefficient = 0.96
        elif peloton == 'l':
            self.peloton_coefficient = 0.86

    #returns force needed for cyclist to travel at
    # given speen
    def wind_force(self, air_density, speed):
        f = 0.5 * self.frontal * air_density *
            self.drag * (speed * speed)
            * self.peloton_coefficient
        return f

    #returns frictional rolling force
    def roll_force(self, friction, gravity):
        f = friction * self.mass * gravity
        return f

    #returns gravitational rolling force
    def grav_force(self, gravity, slope):
        f = self.mass * gravity * slope
        return f

    #returns total force needed for cyclist to move
    def total_force(self, wind, roll, gravity):
        t = wind + roll + gravity
        return t

    #returns wind coefficient of speed for given
    # power(v^3)
    def wind_power(self, air_density):
        p = 0.5 * self.frontal * air_density
            * self.drag * self.peloton_coefficient
        return p

    #returns frictional and gravitational coefficient
    # of speed for given power (v^1)
    def roll_grav_power(self, roll_f, grav_f):
```

```
        p = roll_f + grav_f
        return p

    #function to calculate power from speed
    def power(self, route, speed):

        c = cyclist
        r = route

        wind = self.wind_force(r.air_density, speed)
        grav = self.grav_force(r.gravity, r.slope)
        roll = c.roll_force(r.friction, c.mass, r.gravity)
        total_f = total_force(wind, grav, roll)

        p = total_f * speed

        return p

    #function to calculate speed from power
    def speed(self, route, power, i):

        r = route

        roll = self.roll_force(r.friction, r.gravity)
        grav = self.grav_force(r.slopes[i], r.gravity)
        wind = self.wind_power(r.air_density)
        rg = self.roll_grav_power(roll, grav)

        poly = [-wind, 0, -rg, power]
        speed = np.roots(poly)[2]
        sp = str(speed)[1:15]
        return float(sp)
```

B route.py

```
import math
import numpy as np
import scipy
from numpy.random import rand
import sys

class Route(object):

    def __init__(self, gravity, air_density, friction, file):
        self.gravity = gravity
        self.air_density = air_density
        self.friction = friction
```

```
        self.slopes = np.loadtxt(file)[:,0]
        self.distances = np.loadtxt(file)[:,1]
```

C main.py

```
from Cyclist import Cyclist
from route import Route
import math
import numpy as np
import matplotlib.pyplot as plt

def power_vs_times(cyclist, route, powers, savefile):

    c = cyclist
    r = route

    times = []

    for i in range(len(powers)):
        total_time = 0
        for j in range (len(r.slopes)):
            sp = c.speed(r, powers[i], j)
            distance = float(r.distances[j])
            total_time += distance / sp

        minutes = total_time/60
        times.append(minutes)

    np.savetxt(savefile, np.column_stack([powers, times]))

def unsteady_legtimes(cyclist, route, input, file1, file2):

    c = cyclist
    r = route
    w_primes = []
    times = []
    powers = []
    w_primes.append(c.w_prime)
    powers.append(input[0])
    times.append(0)

    for i in range(len(r.slopes)):
        power = float(input[i])
        #print(power)
        sp = c.speed(r, power, i)
        distance = float(r.distances[i])
        t = distance / sp
```

```
            if power > c.critical_power:
                possible_t = c.w_prime / (power -
                    c.critical_power)
                if t > possible_t:
                    d_possible = sp * possible_t
                    d_remain = r.distances[i] - d_possible
                    CP_sp = c.speed(r, c.critical_power, i)
                    t_remain = d_remain / CP_sp
                    leg_time = t_remain + possible_t
                    c.w_prime = 0.0
                    powers.append(c.critical_power)
                elif t == possible_t:
                    leg_time = t
                    c.w_prime = 0.0
                    powers.append(c.critical_power)
                else:
                    leg_time = t
                    c.w_prime = c.w_prime -
                        (t * (power - c.critical_power))
                    powers.append(power)
                w_primes.append(c.w_prime)
            else:
                w_primes.append(c.w_prime)
                leg_time = t
                powers.append(power)
            if i ==0:
                times.append(leg_time)
            else:
                times.append(leg_time + times[i])

        np.savetxt(file1, np.column_stack([times, w_primes]))
        np.savetxt(file2, np.column_stack([times, powers]))

    def plot_function(filename, graphname, title, x, y):

        X = np.loadtxt(filename)[:,0]
        Y = np.loadtxt(filename)[:,1]

        plt.plot(X, Y, linewidth = 0.5)
        plt.ylabel(y)
        plt.xlabel(x)
        plt.title(title)
        plt.savefig(graphname)
        plt.show()

    def plot_function_Y1Y2_vs_X(y1_filename, y2_filename,
                                graphname, title, x, y,
```

```
                                    y1, y2):

    Y1 = np.loadtxt(y1_filename)[:,1]
    Y2 = np.loadtxt(y2_filename)[:,1]
    X = np.loadtxt(y1_filename)[:,0]

    plt.plot(X, Y1, linewidth = 0.5, label = y1)
    plt.plot(X, Y2, linewidth = 0.5, label = y2)
    plt.legend()
    plt.ylabel(y)
    plt.xlabel(x)
    plt.title(title)
    plt.savefig(graphname)
    plt.show()

def plot_function_Y1Y2_vs_X_different_scales(y1_filename,
                                             y2_filename,
                                             graphname,
                                             title,
                                             x,
                                             y1,
                                             y2):

    Y1 = np.loadtxt(y1_filename)[:,1]
    Y2 = np.loadtxt(y2_filename)[:,1]
    X = np.loadtxt(y1_filename)[:,0]

    fig, ax1 = plt.subplots()

    color = 'tab:red'
    ax1.set_xlabel(x)
    ax1.set_ylabel(y1, color=color)
    ax1.set(ylim = (0, 80000))
    ax1.plot(X, Y1, color=color)
    ax1.tick_params(axis='y', labelcolor=color)

    ax2 = ax1.twinx()  # instantiate a second axes that
                       #shares the same x-axis

    color = 'tab:blue'
    ax2.set_ylabel(y2, color=color)
    ax2.set(ylim = (0, 700))
    ax2.plot(X, Y2, color=color)
    ax2.tick_params(axis='y', labelcolor=color)

    fig.tight_layout()
    plt.title(title)
```

```
    plt.savefig(graphname)
    plt.show()

def plot_function_w_errors(filename, graphname,
    title, x, y):

    X = np.loadtxt(filename)[:,0]
    Y = np.loadtxt(filename)[:,1]
    errors = np.loadtxt(filename)[:,2]

    plt.plot(X, Y, linewidth = 0.5)
    plt.errorbar(X, Y, linewidth = 0.5, yerr = errors,
        ecolor = 'red')
    plt.ylabel(y)
    plt.xlabel(x)
    plt.title(title)
    plt.savefig(graphname)
    plt.show()

def main():
    #route constants
    route_file  = "route_files/alpe.txt"
    gravity     = 9.807    # earth
    air_density = 1.226    # sea level
    friction    = 0.005    # asphalt road

    #cyclist constants
    p               = 'n' # n = isolated rider,
                          # s = small four man,
                          # l = large 121 man
    c_s             = 15  # speed of cyclist
    power           = np.linspace(200, 450, 41)

    solo = Cyclist('a', 'n')
    four_peloton = Cyclist('a', 's')
    large_peloton = Cyclist('a', 'l')
    r = Route(gravity, air_density, friction, route_file)

    legs = len(r.slopes)
    push = 550
    solo_powers = [push, push, 360, 360,
        push, 360, push, 360,
        360, 360, push, 360,
        360, 360]
    powers = []

    unsteady_legtimes(solo, r, solo_powers,
```

```
        'wPrime_vs_time/alpe/solo_unsteady.dat',
        'power_vs_time_wPrime/alpe/solo_unsteady.dat')
    unsteady_legtimes(four_peloton, r, solo_powers,
        'wPrime_vs_time/alpe/four_peloton_unsteady.dat',
        'power_vs_time_wPrime/alpe/four_peloton_unsteady.dat')
    #unsteady_legtimes(large_peloton, r, powers,
    #    'wPrime_vs_time/alpe/large_peloton.dat',
    #    'power_vs_time_wPrime/alpe/large_peloton.dat')



    '''
    power_vs_times(peloton, r, power,
        'power_vs_time/largePeloton.dat')
    plot_function_Y1Y2_vs_X('power_vs_time/solo.dat',
        'power_vs_time/largePeloton.dat',
        'power_vs_time/solo_vs_largePeloton.png',
        'Power vs Time', 'Power (W)', 'Time (Minutes)',
        'Isolated Rider', '121 Man Peloton')
    '''

main()
```