Mathematical Modelling of Optimal Road Cycling Strategies

Peter James Nee

10-08-2020

Contents

1 Abstract

We study the strategies and theory behind long-distance cycling, and build a model to describe the motion of an athlete moving along some of a range of tracks employing a variety of strategies. The objective is to compare different strategies along these routes, and determine optimal pedalling strategies that should be used given a certain track. We consider pedalling and breaking, road geometry, and forces that arise when in motion. We compare simulations from in-house code with the DynOpt toolbox.

2 Introduction

Long distance road cycling is typically performed on complex routes, consisting of bends, curves and hills. Typical racing strategies employed by athletes today consist of constant pedalling/power output, or constant speed. Anecdotal advice such as "entering a turn slowly so you can leave it faster" gives a qualitative strategy but lacks a quantitative component based on speed and local route gradient and curvature. The reasoning behind this is the centripetal force required to take bends (which we will define analytically later) has a square proportionality with the speed that the bend is taken at, and as such athletes are required to slow down to cope with this.

Many races take places on routes of length up to 300km, and as such are performed by teams. Teams are groups of 6-9 cyclists that specialise in certain parts of the race (i.e. climbing, descending, sprints). This often comes down to the athlete's physiology, as we will see later with some of the forces we identify, eg. different masses and power outputs will cater to different parts of the race more effectively.

These teams will often travel in what's referred to as a "peleton", where they will bunch together as they move along the race. The reasoning is that the wake behind the lead rider will create a slipstream, and as such the riders behind will experience a much smaller amount of drag. This in turn means less power output to stay with the group, and avoid the effects of fatigue too early in the race.

We simplify the problem to one of pedalling strategies that should be employed. We do not consider positioning on the road, or delve too deeply into the physiology and fatigue of an athlete. Such considerations have been considered by others, in the breakout strategies employed in road cycling [1], along with position for taking turns in a velodrome [2]. As such, our model assumes the athlete has already chosen (whether optimally or not) the *line* they will trace out along the route.

Our goal, first, is to identify an appropriate frame of reference in which to solve a force-balance equation, and secondly to develop a code to convert our data into that frame. Following this we identify forces considered integral to our problem, and determine their representations in our frame. Finally, we develop a program to run simulations along these routes, given a specific

pedalling strategy, and compare a number of different strategies.

3 Reading in of data

One of the main aspects of this project was to not only analyse strategies along idealised parametrised routes such as sine and Gaussian curves, but also look at cycling along more complicated, real-world routes. We use simplified routes to test our models before applying them to more complex routes. For a real route, we must read in a GPX file for different routes and convert these to Cartesian co-ordinates.

GPX files typically consist of 3 columns of data: the latitude, longitude and elevation of certain points along the road. We convert the latitude and longitude for each point into a set of x and y co-ordinates, and take the elevation to be the z co-ordinate for these points. Although in doing this we are essentially mapping a curved plane onto a flat, 2 dimensional plane. Due to the scale of these routes the error incurred is negligible.

We convert our latitude, longitude and elevation to Universal Transverse Mercator (UTM) co-ordinates, which are composed of an x and y direction, along with a *zone* that describes a global (ellipsoid) correspondence [5]. There are 60 such zones, however we shall ignore this component of the system as it is unnecessary for our calculations. The details of this calculation are omitted.

4 Frenet-Serret Frame

The Frenet-Serret formulas are a commonly used set of formulas in differential geometry and vector calculus. The frame is used to completely describe the kinematic properties of a particle moving along a continuous, differentiable curve in a Euclidean space $\mathbb{R}^3$. It is made up of 3 orthonormal vector-valued functions $\boldsymbol{T}$, $\boldsymbol{N}$, and $\boldsymbol{T}$, as well as two scalar functions τ and κ. While these may be functions of any subset of the real numbers, a common convenient parameterisation is as functions of arc-length s. The Frenet-Serret Frame is only valid for non-degenerate curves (non-zero curvature), and is described by:

$$\boldsymbol{\alpha}'(s) = \boldsymbol{T}(s)$$
$$\boldsymbol{T}'(s) = \kappa(s)\boldsymbol{N}(s)$$
$$\boldsymbol{T}(s) \times \boldsymbol{N}(s) = \boldsymbol{B}(s)$$
$$|\boldsymbol{B}'(s)| = -\tau(s)$$

Here we have that:

- $\boldsymbol{T}$ is the unit vector tangent, and points in the direction of motion
- $\boldsymbol{N}$ is the normal unit vector, and points in the direction that the curve is *curving* in

- $\boldsymbol{B}$ is the binomial unit vector, the cross-product of $\boldsymbol{T}$ and $\boldsymbol{N}$
- κ is the curvature, which describes how much the curve is curving in the plane spanned by $\boldsymbol{T}$ and $\boldsymbol{N}$, and only takes positive values
- τ is the torsion, and measures how much the curve is *twisting* out of the aforementioned plane. (positive torsion is in the direction of $\mathbf{B}$, negative in the opposite.)

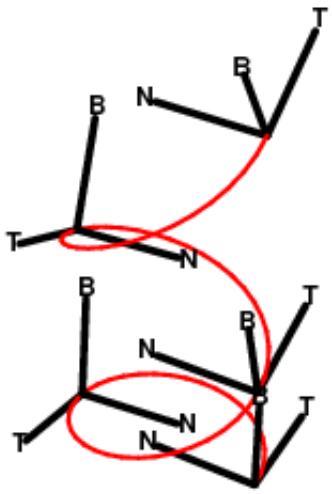

Figure 1: Diagram demonstrating $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{B}$ moving along the curve [3].

These vectors move along with the curve, as is demonstrated in Figure 1. By the fundamental theorem of space curves the two scalar functions, κ and τ, completely determine the shape of the regular curve, with the vectors $\boldsymbol{T}$, $\boldsymbol{N}$, and $\boldsymbol{B}$ determining the orientation of the curve. The position of said curve is arbitrary: for parametrised curves we tend to start near the origin, while for routes we read in we will take the positions after converting from latitude and longitude to an x-y plane.

5 Interpolation of data

We require a continuous representation for all of the functions in Section 4. However, GPX files only provide discrete values along any route. As such,

we must interpolate our data along the route. In the interpolation process, we suppress any oscillations that may arise due to describing the route as a piecewise polynomial. We use cubic Hermite splines.

On a unit interval (x_k, x_{k+1}), we build a piecewise cubic polynomial $\mathbf{P}_k(x)$ to approximate f with the requirements that:

- $\mathbf{P}_k(x_k) = f(x_k)$ and $\mathbf{P}_k(x_{k+1}) = f(x_{k+1})$
- $\mathbf{P}'_k(x_k) = f'(x_k)$ and $\mathbf{P}'_k(x_{k+1}) = f'(x_{k+1})$

This procedure ensures that our polynomial en-captures the position of the road that we have specified, is differentiable and preserves the shape of the road. This suppresses any oscillations that may normally occur (such as in Figure 2), respecting the monotonicity that the route typically exhibits.

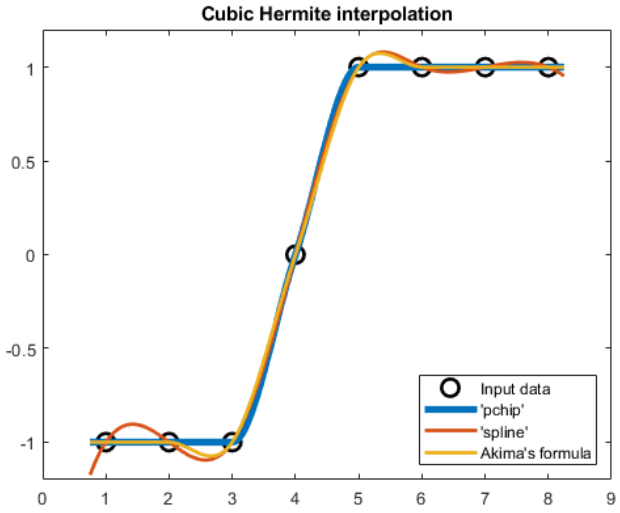

Figure 2: Illustration of different interpolation techniques on a simple function [4].

However we do not necessarily have the exact gradients of the road at each (or in fact any) of the points along the road. As such, we use Newton's divided-difference to approximate derivatives at each point. This will provide reasonable approximations along straighter parts of the route, however may cause some errors in very sharp, short turns. If such issues arise, natural cubic spline interpolation (with natural or clamped ends) may be more appropriate, and as such our model may easily adapted to use this method instead. We find cubic Hermite splines appropriate in the cases we consider.

While the interpolated function may only be C^2, we perform this interpolation on each function (and component of each function). This ensures that each function generated in the Frenet-Serret frame is differentiable.

6 Force balance equation

We consider the four dominant forces present in our model. For an athlete of mass (inclusive of their equipment) m, moving in direction $\boldsymbol{T} = (x, y, z)$:

- A number of models for the pedalling Force may be applied, such as constant pedalling and linear change with a maximum speed $V_{\max}$

$$\text{Pedalling} = \begin{cases} Pf & \text{(constant)} \\ Pf \cdot \frac{V_{\max} - v}{V_{\max}} \cdot H(V_{\max} - v) & \text{(linear change)} \end{cases}$$

  where Pf is a constant power, v the velocity, and H the Heaviside function

- The gravitational force is given by

$$\text{Gravity} = m \cdot g \cdot (\boldsymbol{T} \cdot (0, 0, -1))$$

  where g is the usual gravity constant. The dot product extracts the vertical component of $\boldsymbol{T}$ in the direction of gravity.

- The aerodynamic drag force is given by

$$\text{Drag} = -0.5 \cdot C_d \cdot A \cdot v^2,$$

  where C_d is the drag coefficient, typically 0.3 for an athlete, A is the surface area in the direction of motion, and v is the speed of the athlete. This always opposes the direction of motion.

- The friction force due to the bicycle tyres in contact with the road is given by

$$\text{Friction} = -\mu \cdot m \cdot g \cdot \frac{\boldsymbol{T} \cdot (x, y, 0)}{|(x, y, 0)|},$$

  where μ is the coefficient of friction, and g is as described as above. As the gradient changes, the normal force varies from $m \cdot g$ when flat to a theoretical 0 for a vertical road. As such, we need to calculate what angle our incline makes with the x-y plane. Here we make use of vector calculus results to determine the angle between two vectors, where our two vectors are the tangent and a projection of our tangent to the x-y plane.

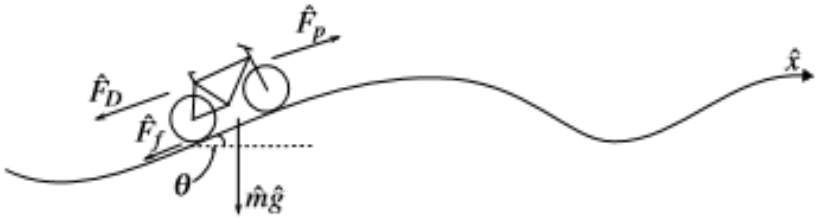

Figure 3: Illustration of the direction of the forces acting on the athlete [1]

We can see in Figure 3 that all our forces, save gravity, act in the tangent direction $\boldsymbol{T}$. We extract the gravity component in that direction. We solve Newton's second law for acceleration ($\dot{v}$), with the above forces on the right hand side:

$$m\cdot\dot{v} = \text{“Pedalling”} + m\cdot g\cdot(\boldsymbol{T}\cdot(0,0,-1)) - 0.5\cdot C_d\cdot A\cdot v^2 - \mu\cdot m\cdot g\cdot\frac{\boldsymbol{T}\cdot(x,y,0)}{|(x,y,0)|} \quad (1)$$

While we solve (Eq. 1) equation for v, the arc-length $s = \frac{dv}{dt}$ determines our position along the curve, and hence determines the input tangent vector $\boldsymbol{T}$ on traversing the route. This creates a coupled system.

7 Centripetal Force

In the Frenet-Serret frame, acceleration is given in the $\boldsymbol{T}$ and $\boldsymbol{N}$ directions:

$$\boldsymbol{a}(t) = \dot{v}(t)\boldsymbol{T}(t) + \kappa(t)[v(t)]^2\boldsymbol{N}(t)$$

Eq (1) is the $\boldsymbol{T}$ component balance in Newton's second law. However, we must consider the centripetal acceleration in the $\boldsymbol{N}$ direction. Speed and curvature combine to direct the motion out of the curve. We aim to keep this acceleration below a certain threshold, determined by how much force the cyclist can withstand/output.

8 Numerical Solution to ODE

The motion along a route depends on the position, ascribed to the arclength s, with $\boldsymbol{T} = \boldsymbol{T}(s)$. The work done, W, is the total pedalling power exerted by the cyclist. The full coupled system reads:

$$\begin{aligned}
\frac{ds}{dt} &= v,\\
\frac{dv}{dt} &= \frac{\text{“Pedalling”} + m\cdot g\cdot(\boldsymbol{T}\cdot(0,0,-1)) - 0.5\cdot C_d\cdot A\cdot v^2 - \mu\cdot m\cdot g\cdot\frac{\boldsymbol{T}\cdot(x,y,0)}{|(x,y,0)|}}{m},\\
\frac{dW}{dt} &= \text{“Pedalling”}.
\end{aligned}$$

Due to the non-linearity of our governing equations and route, we must solve the system numerically. We use Matlab's in-built ode45, which employs a Runge-Kutta method. Note that the system is non-stiff (there is no rapid change in our solution for any set real-world pairings of our state variables).

We need to chose numerical parameter values for our constant parameters and systematic pedalling strategies. In our simulations we use either constant pedalling forces or one that varies linearly with a maximum speed. We also set conditions to *free wheel* (no pedalling) at certain speeds and input a braking force (if the centripetal force described earlier ever became too large).

9 Simulations

We consider several pedalling strategies employed across different routes (some parametrised, while some read in and interpolated as described previously). The first example we present is a sinusoidal route: a straight line in the x-y plane, but 2 periods of a sine curve with varying amplitude in the z direction,

$$
\begin{aligned}
& t \in [0, 100], \\
& x = t, \\
& y = 1, \\
& z = 15 + 15 \cdot \sin\left(\frac{2 \cdot \pi \cdot x}{\frac{L}{2}}\right),
\end{aligned}
$$

shown in Figure 4. We applied a constant pedalling force, but if the centripetal force, $\kappa(s)[v(s)]^2$, ever becomes too large, we cease pedalling and apply a braking force.

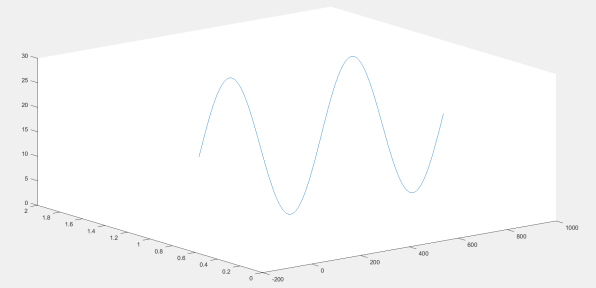

Figure 4: Rendering of sinusoidal route.

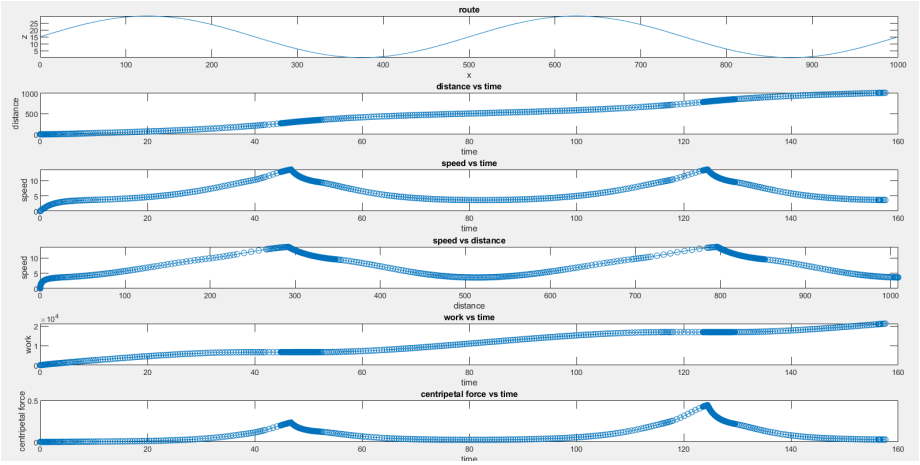

Figure 5: The athletes run along the route.

As expected, we see in Figure 5 that when the athlete is about to reach the valley, they experience too much centripetal force from descending the hill and as such will not be able to cope with the force on the bend. So the athlete starts breaking, and the speed immediately decreases.

Let's consider a more complicated route, such as the Alpe d'Heuz, a popular climb in France. This climb is notorious for its steep slope, paired along with the constant winding and bending as we ascend the climb. In Figure 6 and Figure 7 we have our rendering and analysis of the route:

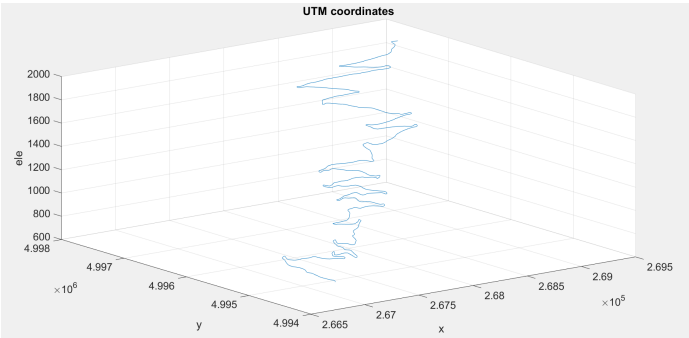

Figure 6: Our rendering of the Alpe d'Heuz.

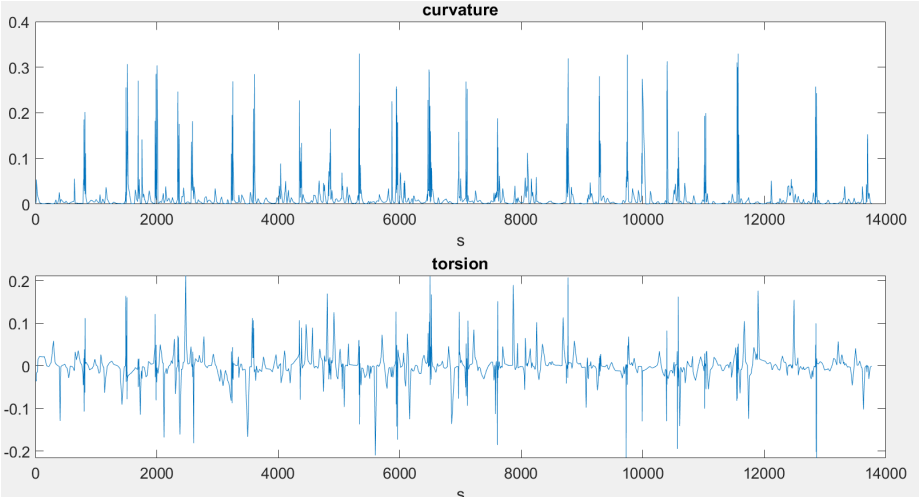

Figure 7: Graphs of the curvature and torsion along the Alpe d'Heuz.

We model an athlete along this route with two strategies: (1) a constant pedalling strategy for Figure 8, and (2) a linear change model with a maximum speed in Figure 9. This is run over a set period of time (as such, neither athlete will finish the race but we may still gleam some results).

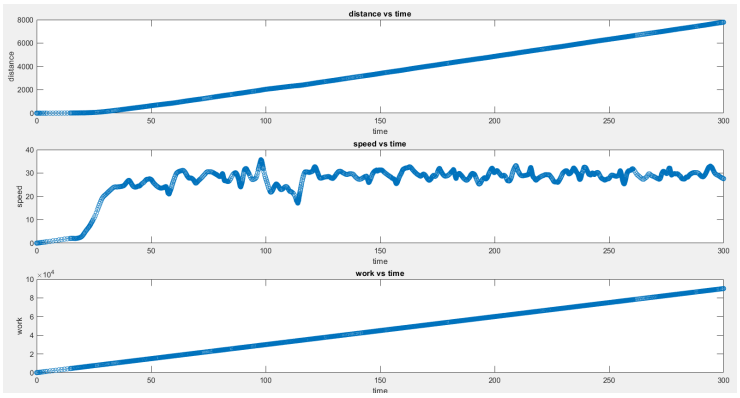

Figure 8: Constant pedalling along the Alpe d'Heuz.

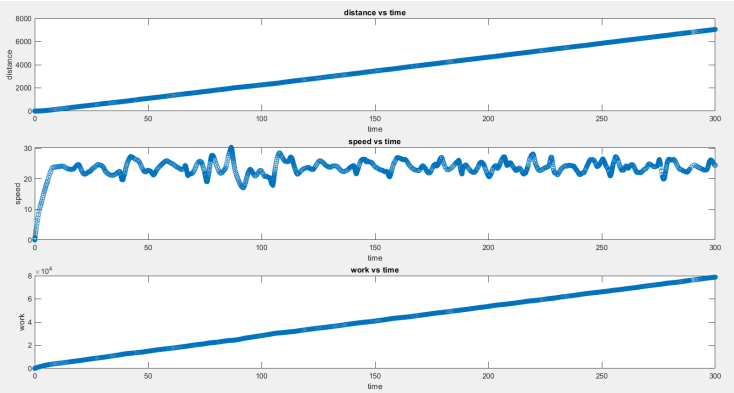

Figure 9: Linear change in pedalling along the Alpe d'Heuz.

Our models are:

- Constant $= Pf_1$
- Linear change $= Pf_2 \cdot \frac{V_{max}-v}{V_{max}} \cdot heaviside(v)$

Here $Pf_1 < Pf_2$, so that we can keep our average speed consistent. We can see in Figure 8 that constant pedalling model takes longer to accelerate. However, the maximum speed attained is greater such that roughly the same distance is covered as the second case, in the same time span. The notable piece is that we can see in Figure 9 that the linear change model makes it further, while the constant pedalling model uses a bit more work to not make it as far. We also note in Figure 9 that the linear change model fluctuates more in speed. We believe this is due to a *slow reaction* from the athlete.

It is important to note that while many of the parameters reflect real world cycling, some of the speeds and power outputs are not representative. As such, velocities and work done may reflect larger or smaller numbers that we would expect.

10 Dynamical optimal control trajectory

While the previous process works well, it has limitations. We may only compare strategies one at a time, and may only compare ones that we come up with. This can be quite timely, and it is not clear if we have found an *optimal* pedalling strategy. As such, we seek a method for comparing many different strategies at once.

DynOpt (MATLAB Dynamic Optimisation Code) is a toolbox developed to determine an optimal control trajectory, based on the descriptions of the system, a *cost function* to be minimised, and subject to different inequality and equality constraints. It uses an orthogonal collocation on finite element methods. Over

many iterations, a fitness is determined for each trajectory, until a convergence occurs and we are left with an optimal control trajectory.

In this problem, our system is a set of three differential equations:

$$\begin{aligned}
\frac{ds}{dt} &= v \\
\frac{dv}{dt} &= \frac{\text{“Pedalling”} \cdot u + m \cdot g \cdot (\boldsymbol{T} \cdot (0,0,-1)) - 0.5 \cdot C_d \cdot A \cdot v^2 - \mu \cdot m \cdot g \cdot \frac{\boldsymbol{T} \cdot (x,y,0)}{|(x,y,0)|}}{m}, \\
\frac{dW}{dt} &= \text{“Pedalling”} \cdot u
\end{aligned}$$

Here we treat $\boldsymbol{T}$ as a function of position arclength (s), and u is our control parameter. DynOpt computes different trajectories for this parameter, until it finds one with proper fitness. This is determined by the selection of a cost function, which we usually take to be work done, time taken or a weighted mix. For constraints, we have a few we most commonly use:

- $s(t_f) = l$, where l is the length of the route. Essentially, this ensures we only consider the athlete along the original route.
- $\kappa(t)[v(t)]^2 \leq \epsilon$, where ϵ is some constant. This makes sure that the athlete does not take any bend too quickly (as described previously).

Along with many other settings, we also provide the partial derivatives of each of the equations and inequalities with respect to each parameter, state variable and control parameter.

We test this optimisation toolbox with our system along a few simple paths. One of the more interesting results is when we run the program over the first half of a scaled sine curve (running 100m in the x direction, and 30m in the z direction). We set the cost function to be the work done, and run the program over 1000 iterations. The result achieved is displayed in Figure 10.

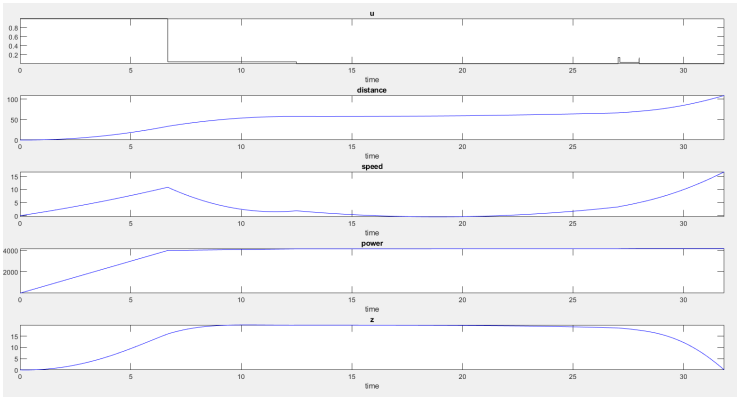

Figure 10: Pedalling strategy chosen by DynOpt after 1000 iterations.

Here $u \in [0, 1]$, and represents how much pedalling is being done (1 is pedalling as hard as possible, 0 being *coasting*). This result is realistic, as we would expect to pedal just hard enough to get over the crest of the hill, and then to ride the hill down the other end.

The interesting part is that it decides the best course of action is to pedal as hard as possible, for as short a period of time as possible. This is chosen as opposed to a strategy of exerting less effort over a longer period of time. As no analytical solution is available, it is difficult to determine the validity of this, however this was tested over several runs with different initial values for the control parameter, and was consistent in this strategy.

11 Next Steps

Our immediate next goal is to develop the optimisation software. While it works quite well along some routes, we have had some issues along more complicated routes, and also with the implementation of the restriction on the speed with which turns can be taken.

On top of this the programme still runs quite slowly. This is most likely due to the complexity of the process and its gradients, and as such we need to re-evaluate how we define the gradients to see if it can be streamlined.

We also need to look more at our interpolation procedure, as we still have some oscillations along tight turns that may cause issues with the curvature function κ.

And finally I want to look more into the representation of the data we do achieve. I would like to be able to analyse the effect a change in mass or pedalling capabilities might have, as well as a way of breaking down some of the current data we have efficiently. More time still needs to be put into some of the values for the parameters to try and accurately match real-life examples, and more work could be done to make the data more clear.

12 Conclusion

We successfully created a model for an athletes run along different long-distance cycling routes. These routes can either be parametrised or fed in via GPX file, and much of the data regarding the shape of the route can also be extracted. We also studied the centripetal force required to take different turns/bends, and its relation to the speed with which it's taken at. While our work with the optimising toolbox is not fully finished, we have been able to produce some clear results with it.

13 Acknowledgements

First I would like to thank the School of Mathematics UCD for allowing me the opportunity to undertake this research. It has been a fantastic opportunity

to gain more insight into the realm of research and mathematical modelling. However most importantly I would like to give my utmost thanks to Dr. James Herterich, for his supervision, guidance and input throughout the entirety of this project. It has been a huge honour to work with you.

References

[1] L. H. Gaul *Optimizing the breakaway position in cycle races Using mathematical modelling* 2018.

[2] GP Benham *Brachistochrone on a velodrome* 2020.

[3] A. Havens *Curvature, Natural Frames, and Acceleration for Plane and Space Curves* 2017.

[4] Matlab *https://www.mathworks.com/help/matlab/ref/pchip.html* 2020.

[5] Rafael Palacios *Conversion of lat/lon vectors to UTM vectors* 2007.