Finding the Optimal Power Function for Minimizing Time around a Bike Course

Team 2229096

February 19, 2022

Abstract

We model the power expenditure of bikers and time taken to finish a race to find the optimal times to use their power that minimizes race time. We utilize python coding to produce the best power function as a Fourier series of 20 terms and we run 10,000 different functions to find the optimal one. We found out that the best way to minimize time depends on a variety of conditions including drag, exhaustion, maximum attainable power, number and steepness of turns, and slope of the track. This model can be applied to different race tracks in advance to give bikers a hint on when to use their power to finish the race in the least time. We tested our model on the time trial courses of the 2021 Olympic Time Tokyo, 2021 UCI World Championship in Flanders, and a track of our design.

Contents

1 Introduction **3**

1.1 Variables and general model 3

2 Analytical Approaches **3**

2.1 The Lagrangian approach 3

3 Numerical Approaches **4**

3.1 The Energy Approach 4

3.1.1 Introduction to energy approach 4

3.1.2 Our Model 4

3.1.3 Why the energy approach did not work 6

3.2 The Power Approach 6

3.2.1 variables and functions 6

3.2.2 The code associated with it 8

3.3 Results and reflection 12

3.3.1 Different Types Of Bikers 12

4 Advice to the Directeur Sportif **14**

4.1 Different power profiles 14

4.2 Genders in cycling 14

4.3 Modelling tracks 15

1 Introduction

1.1 Variables and general model

When it comes to decision making in biking, it is crucial to put in mind so many things. Finding the optimal way to finish a race considering the varying conditions of the weather and the tack features as well as the human exhaustion factors is very difficult to account for analytically.

Basic human intuition is enough to solve the simplest case of this problem, by going as fast as you can. However, we try to find the most optimal way to spend energy of the biker. And, in a real world, energy does not go directly into speeding the biker and that there is a lot of dynamics in this system. For example, the energy input can be used to climb a hill which is dependent on x, go against air drag force which is dependant on velocity,and it is subjected to depletion due to exhaustion and extended power usage.

Not only that, but the energy spent going uphill is gained again when going downhill, the wind could be blowing in the same direction and adding kinetic energy to biker, and taking rests also adds more to the maximum attainable power of the biker.

We develop an analytical approach to the problem that requires a lot of simplifications in Section 3, and we then move to finding a numerical result in Section 4. By the end, we apply our model on real race tracks, give a script to approximate a race track for later use and conclude by an advice to directur sportif.

2 Analytical Approaches

2.1 The Lagrangian approach

In physics it is established that

$$P = Fv \tag{1}$$

Applying separation of variables, we can see that

$$t = \int \frac{F}{P} dx \tag{2}$$

Substituting the relationship we have deduced in Net Force and Power, we find that the time required to finish the race is defined by

$$\int_0^x \frac{b + g(x)}{P_d - P_p} dx. \tag{3}$$

Therefore, our problem becomes a simple minimization of t problem with the functional inside of the integral in 3.

$$\mathcal{L} = \frac{b + g(x)}{P_d - P_p} \tag{4}$$

Recalling that to minimize a functional, we use the Euler-Lagrange equation described as

$$\frac{\partial \mathcal{L}}{\partial P_d} - \frac{d}{dx}\frac{\partial \mathcal{L}}{\partial(\frac{\partial P_d}{\partial x})} = 0 \tag{5}$$

Looking closely at the Euler Lagrange relation we derived, we can see the Lagrange we defined earlier in eq. 4 has no dependence on $\frac{\partial P_d}{\partial x}$. Therefore, the second term is zero and the first term is also zero. Moreover, upon investigation, the functions for P_d prove to be complex, and thus a direct numerical method is more efficient.

3 Numerical Approaches

Realizing that tackling this problem is going in difficulty with increasing the number of factors we outline, we decided to take a numerical approach.

3.1 The Energy Approach

3.1.1 Introduction to energy approach

We then shifted gears to analyze the system numerically using Python code of our creation. In Listing 1 we used an Energy approach to analyze the system. Energy should be a viable solution since we are analyzing an open system with known conservative and neoconservative forces. The conservative force is going to cause no change of the mechanical energy, so energy is going to be either stored in the rider, spent in the form of gaining kinetic energy, or spent in gaining potential energy by going uphill. Non conservative forces include but are not limited to the drag.

3.1.2 Our Model

```
import math

# Relevant physical quantities
E = 2000 # The total energy that a rider can use pedaling the bike forward; unit of KJ
D = 10000 # The total distance of a race track; unit of m
g = 9.81 # The acceleration due to gravity on the surface of the earth; unit of m/s^2
x = 0 # The distance the rider has cycled; starts at 0; unit of m
dt = .01 # The increment of time added in each iteration; unit of s

# Initializing lists to keep track of different physical quantities
E_ = [E,E] # energy
X_ = [0.0, 0.001] # distance cycled
F_ = [0,0] # External forces acting on the rider-bike system,
# i.e. gravity, rolling friction, air resistance
V_ = [0.0,0.1] # Velocity of the rider-bike system
```

```
T_ = [0,0] # Time, each element will be dt greater than the previous one
A_ = [0,0] # Acceleration of the rider-bike system

# Relevant physical constants
G = 0 # gradient of the road, negative values denote downhill, positive ones denote uphill
M = 85 # assumed mass of the rider-bike system; in unit of kg
mu = 0.02 # coefficient of rolling friction
v = 0.1 # m/s
A = 0.38 # frontal surface area of the rider-bike system; in unit of m^2
rho = 1.2 # air density at sea level; in unit of kg/m^3
Cd = 0.5 # drag coefficient
Constant = Cd * A * rho # variable to make the code more readable


# Returns the net force acting against the rider-bike system
# @param  x, the distance traveled by the rider as of the last time increment
# @param  v, the velocity of the rider as of the last time increment
# @return F_net, the net force acting against the rider-bike system

def F(x, v):
    F_g = g * math.sin(math.atan(G)) * M
    F_r = F_g * mu
    F_a = 0.5 * Constant * v**2
    F_net = -(F_g+F_r+F_a)
    return F_net

def W(x):
    return x**2

# while the distance traveled is less than the total distance of the course
while X_[-1] < D:
    # distance traveled since the last time increment
    delta_x = X_[-1] - X_[-2]
    # work done by the rider since the last time increment
    delta_w = W(X_[-1])
    # net force working against the rider-bike system during the last time increment
    delta_f = F(X_[-1],V_[-1])
    # work that was translated to velocity since the last time increment
    # calculated by subtracting work used to overcome the resistive forces
    # from work put in by the rider
    work_vel = delta_w - delta_f * delta_x

    # acceleration the rider underwent during the last time increment
    accel_cur = work_vel / (M * delta_x)
    # appending the latest acceleration to the acceleration list
    A_.append(accel_cur)
    # velocity of the rider during the last time increment
    vel_cur = V_[-1] + A_[-1] * dt
        # appending the latest velocity to the velocity list
    V_.append(vel_cur)

    # distance traveled since the last time increment
    x_cur = X_[-1] + V_[-1]*dt
        # appending the latest position to the position list
    X_.append(x_cur)

    # increment dt
    T_.append(T_[-1]+dt)

print("The time it took to complete the course is: ", T_[-1], "seconds")
```

3.1.3 Why the energy approach did not work

This approach does not produce sensible results. Changing dt between 0.1 and 1.0, which is the amount of time incremented in each iteration, causes the time it takes to complete the course to change on the orders of magnitude of 5-25. This does not make any intuitive sense. Therefore, our attempt in solving the problem in an energy approach did not succeed and we had to pivot to a different solution.

3.2 The Power Approach

3.2.1 variables and functions

Consider the following relationship between power, force, and velocity:

$$P = Fv, \tag{6}$$

where P is the power put into the system, F is the force of resistance, and v is the velocity of the cyclist. The main objective of the cyclists is to complete the track in as little as time as possible, that is, we want to minimize time. Next, note that the only control knob that the cyclist has is the input power, or "driving power," P_d, which will be a function of position on the track. Altogether, we simply want to find a function $P_d(x)$ that minimizes the time it takes to complete the race. Turning to Eq. 6 gives us instant velocity as:

$$\frac{dx}{dt} = \frac{P}{F}. \tag{7}$$

In our models, we are going to treat P and F as functions of position, energy, and velocity, but not time. This allows a separation of variables, and integration

$$t = \int_{x_s}^{x_f} \frac{F}{P} dx, \tag{8}$$

where x_s is the start of the race and x_f is the finish line.

In order to analyze the time function and thus minimize it, using some method, we need to find its constituent parts. First, let's analyze F. At first sight, one might be inclined to add a force due to gravity here (thinking of uphill vs. downhill riding). However, Eq. 6 specifies F as the resistance force. If the cyclist is going downhill, then the force of gravity is not resisting him or her. For this reason, gravity (or the gradient of the track) will be analyzed as a part of the power function P.

The simplest resistance force F_c is to imagine a constant resistance force against the rider. This might include things such as friction in the bike and wheels. We can simply model this as:

$$F_c = C, \tag{9}$$

where C is some constant.

Next, we can model air resistance. For massive objects, drag is linearly dependant on velocity, so we have a force that is

$$F_d(v) = dv, \tag{10}$$

where d is some drag coefficient.

The main way weather could play a role is through the wind. The wind velocity v_w which will be dependant on a direction and possibly time as well. Once this parameter is defined, it will act as a parameter for the frag force. Hence, Eq. 11 becomes

$$F_d(v, v_w) = d(v - v_w), \tag{11}$$

where $v_w > 0$ is when the wind is moving the same direction as the cyclist.

Finally, we can similarly construct the centripetal force acting against the cyclist as the cyclist turns a corner. To this end, first recall that centripetal force $F_{\text{centripetal}}$ is given by

$$F_{\text{centripetal}}(v) = Sv^2, \tag{12}$$

where S is some constant induced by the bend in the track, (typically m/r. Next, we need to model the location of where this bend is in the track. For this, we are going to a narrow Gaussian as the location. Two features of the bend are relevant, namely, the sharpness of the bend and how long the bend lasts. Setting S and W as the sharpness and width of the bend respectively, and combining it with the centripetal force yields

$$F_{\text{bend}}(x, v) = Sv^2 \exp[-(\frac{x - x_0}{.2W})^2], \tag{13}$$

where x_0 is where the bend is centered. Combining each of the mentioned forces simply gives us

$$F(x, v, v_w) = F_{\text{bend}}(x, v) + F_d(v, v_w) + F_d(v) + F_c \tag{14}$$

Next, let's consider the function for power P. This of course will be a function of the driving power P_d. However, we also consider that the each cyclist will have a maximum power, and will be penalized for going over this threshold. Furthermore, this maximum power changes depending on how much energy they have spent in the race (or equivalently how long they've been pedaling at a high power). Letting $P_{0\max}$ and $E_{\max}$ be the initial power threshold and energy of the cyclist respectively, we assume that the power threshold as a function of is

$$P_{\max}(E) = -\frac{P_{0\max}}{E_{\max}^2}x^2 + P_{0\max} \tag{15}$$

This gives us the intuitive result that at the beginning of the race the cyclist will be able to use the most power, but as they get tired (lose energy) the available power expenditure decreases quadratically. Additionally, the cyclist can use no power if they reach $E_{\max}$ before the end of the race. So we can express the power input into the system by the cyclist with a sigmoid function, $\sigma(P_d, P_{\max})$, defined as

$$\sigma(P_d, P_{\max}) = [1 + \exp[-10(P_d - P_{\max})]]^{-1} \tag{16}$$

Then, we can express the power into the system as:

$$P_d - P_d\sigma(P_d, P_{\max}). \tag{17}$$

Equation 17 ensures that the $P_{\max}$ is indeed a maximum. In addition to the power input by the cyclist, we can model the force of gravity in the function for power. Note from Eq. 6 that a gradient function $g(x)$ can be related to power by multiplying by velocity:

$$P_g(x) = g(x)v. \tag{18}$$

So for our power function, we have :

$$P(x, P_d, P_{\max}) = P_d - P_d\sigma(P_d, P_{\max}) - g(x)v \tag{19}$$

Therefore, the function for time is:

$$t = \int_{x_s}^{x_f} \frac{F_{\text{bend}}(x, v) + F_d(v, v_w) + F_d(v) + F_c}{P_d - P_d\sigma(P_d, P_{\max}) - g(x)v} dx \tag{20}$$

To minimize this function, we deploy an analytical method where we use a Fourier series to "guess" the function that minimizes time. We are able to brute force our way through the problem by guessing a wide plethora of functions for $P_d(x)$.

3.2.2 The code associated with it

```
import numpy as np
import random
import matplotlib.pyplot as plt
```

```python
track1File = open('track1.txt', 'r')
track1 = str.split(track1File.readline()[1:-1], ',')

def taylor(x, c):
    ts = [1]
    n = 1
    while len(ts) < ncoef:
        ts.append(x**n)
        if len(ts) < ncoef:
            ts.append(n*np.cos(x))
        n += 1
    return np.dot(np.array(c), np.array(ts))

def fourier(x, c):
    fs = [1]
    n = 1
    while len(fs) < ncoef:
        fs.append(np.sin(n*x/distance))
        if len(fs) < ncoef:
            fs.append(np.cos(n*x/distance))
        n += 1
    return np.dot(np.array(c), np.array(fs))

def sigmoid(z, E):
    return 1/(1+np.exp(-10*(z-Pmax(E))))

def Pd(x, c):
    return fourier(x, c)

def g(x):
    return G*(x-.5)

# Returns the power generated by the rider
# @param  x, the position of the rider in the course
# @param  v, the velocity of the rider
# @param  c, physical constant
# @param  E, the Energy list
# @return PD or 0
def P(x, v, c, E):
    PD = Pd(x, c) - .95*sigmoid(Pd(x, c), E) * \
        Pd(x, c) - g(x)*v  # pay attention to sign
    if PD >= 0:
        return PD
    if PD < 0:
        return 0
```

```
def F(x, v):
    gaussum = 0
    i = 0
    while i < len(xturn):
        gaussum += s*(v**2)*np.exp(-((x-xturn[i])/(.2*wturn[i]))**2)
        i += 1
    return b*v + C + gaussum

def Pmax(E):
    return -(P0max/Emax**2)*x**2 + P0max

xturn = [.25, .75]
wturn = [.05, .05]

s = .3
Emax = 100
P0max = 50
distance = 1
b = .2
C = 7
G = 2
bestcoeffs = 'none'
bestt = 2
ncoef = 20
dt = .001
attempt = 1
nattempts = 1000
while attempt <= nattempts:
    print(attempt)

    c = [random.uniform(0, 1.5*P0max)]
    while len(c) < ncoef:
        rand = random.uniform(-1.5*P0max, 1.5*P0max)
        c.append(rand)

    x = .001
    v = .001
    E = 0
    t = 0
    stopped = False
    noE = False
    while x < distance:
        f = F(x, v)
        p = P(x, v, c, E)
        v = p/f
        x += v*dt
        E += Pd(x, c)*dt
        t += dt

        if v < 0:
            stopped = True
            break
        if E >= Emax:
            noE = True
            break
        if t > bestt:
            break

    attempt += 1
    if (t < bestt) and (not stopped) and (not noE):
        bestt = t
        bestv = v
        bestcoeffs = c
        attempt = 0
        print(bestt)

print(bestcoeffs)
print(bestt)

F_ = []
P_ = []
V_ = []
X_ = []
E_ = []
T_ = []
G_ = []
c = bestcoeffs
x = .001
v = .001
E = 0
t = 0
dt = .0001
stopped = False
noE = False
while x < distance:
    f = F(x, v)
    F_.append(f)

    p = P(x, v, c, E)
    P_.append(Pd(x, c))

    v = p/f
    V_.append(v)

    x += v*dt
    X_.append(x)

    E += Pd(x, c)*dt
    E_.append(E)

    t += dt
    T_.append(t)

    G_.append(g(x)*v*dt)
    # plt.scatter(x,Pd(x,c), color = "b", marker=".")
    # plt.scatter(x,Pd(x,c), color = "b", marker=".")
    #plt.scatter(x, 10*np.sin(10*x) + 9, color = "r", marker="." )
    if v <= 0:
        stopped = True
        break
    if E > Emax:
        noE = True
        break
    if t > bestt:
        break

xx = plt.plot(X_, P_, label="P(x)")
plt.xlabel("Distance")
plt.ylabel("Power expendure")

#yy= plt.plot(X_,G_, label = "g(x)")
plt.legend()
plt.xlim(0, 1)
plt.ylim(-200, 700)
plt.grid()

# plt.title("Energy spent is{}".format(sum(E_), "right"))
# plt.fill_between(xx,0,yy)
plt.legend()
plt.show()
```

3.3 Results and reflection

3.3.1 Different Types Of Bikers

There are various types of bikers that could affect this model. There are sprinters, climbers, riders, punchers, and all-rounders.

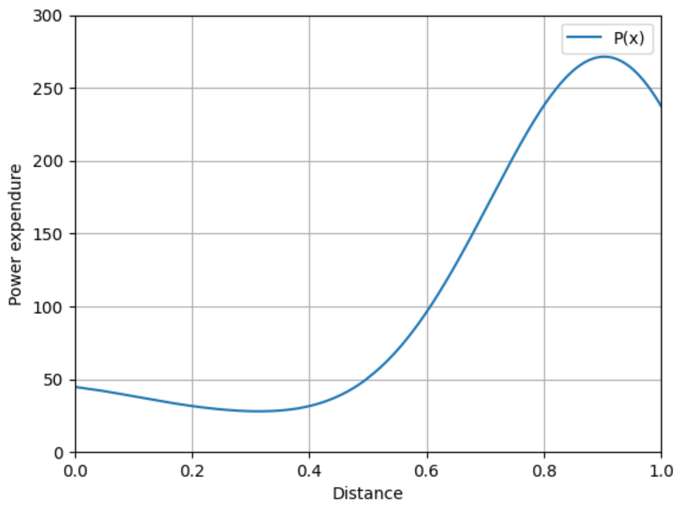

Figure 1: Power function of all-rounder in Fuji track.

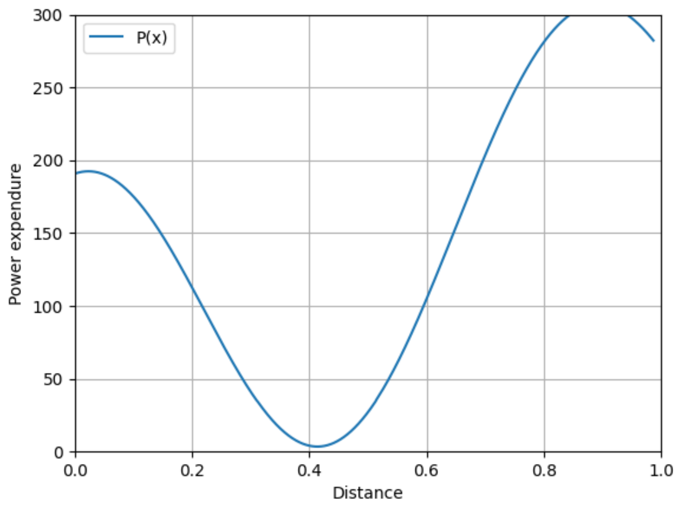

Figure 2: Power function of climber in Fuji track.

We are only going to cover two of these types in our paper, the all-rounder and the climber. The difference between them in this case is going to be that the climber is has got a lower power max but it is spread over a larger energy expenditure tank because they can climb more efficiently. On the other hand, the all-rounder has higher power, but they are limited to lower amount of energy.

In addition, this is the second track in Flanders. We did not have to analyze different types of riders since the track is mostly flat.

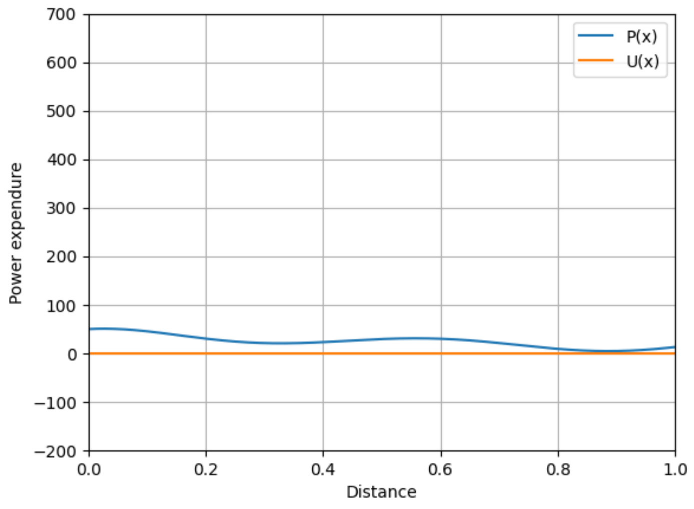

Figure 3: Power function of all-rounder in Flanders track.

4 Advice to the Directeur Sportif

4.1 Different power profiles

We target the numerical model we developed earlier for scientific purposes. However, the directeur sprotif of a team or the coach of any individual riders can extract very beneficial features from this model. For example, if the biker is going to compete in a known track, they can input the features of the track into the numerical solution to give them hints on when it is the most efficient to use their power and not to waste it.

Moreover, the directeur sprotif could run some test races to find the profile of the bikers in order to plug this in the program and tailor the best strategies they can use. Not only that, but then can also predict the most successful profile and develop the training plan for their trainees to work towards that biker profile.

4.2 Genders in cycling

Although our model did not account for the gender differences in terms of muscle mass, hormones, endurance, menstruation, injury risk, and weight, it should be

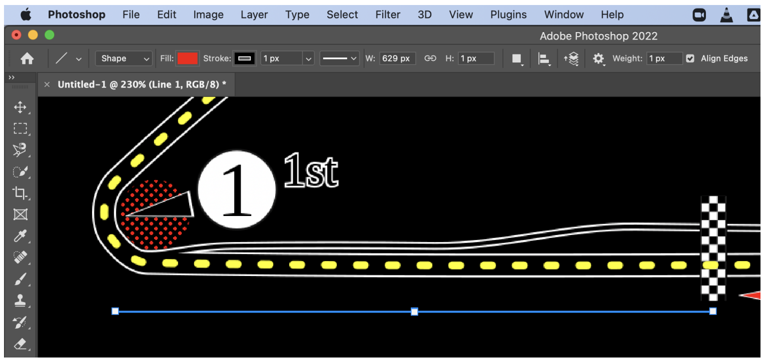

Figure 4: How to take measurements of track images

taken into account. The effects of the listed above factors play a crucial role in determining the possibility of winning.

4.3 Modelling tracks

The process of taking the measurements of the track is not as straightforward. In our model, we had to use generalized units and the race track. That meant we had to stick to a race length of 1 and alter the units of everything so that they match up together. In the actual world, the directeur sprotif does not have to physically go to the race track and extract measurements. However, they can find the picture of the race track online, and use Photoshop or any software to measure the length of the track and normalize it to 1. Then, as shown in fig. 4, the radius and distance between each turn can be calculated in pixels which then can be transformed into meters and then to generalized units so that they match up with the units used throughout the model.

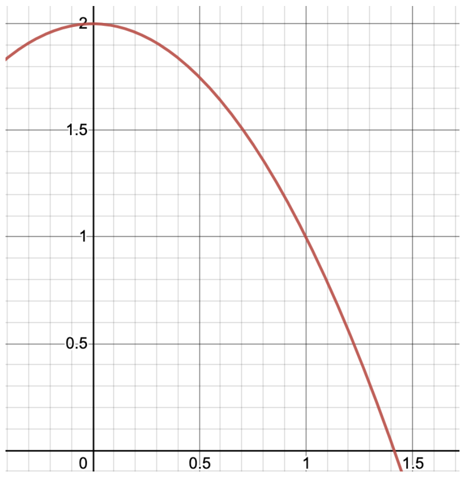

Figure 5: The maximum power graphed on the y-axis vs the total energy expended on the x-axis

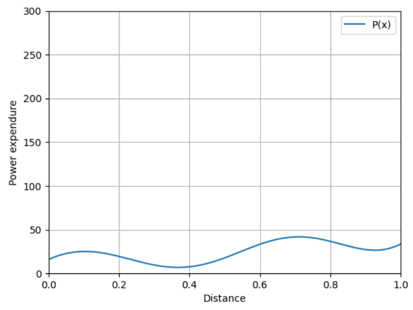

Figure 6: Our Own Track Design

Fig. 6 is our own track design. The four sharp turns occur at x = 0.2, 0.4, 0.8, and 0.9. The last turn is the sharpest.

In addition, directeur sprotif could also make the model more accurate by taking into account the humidity and effect of it on the maximum attainable function. It is also worth noting that we modeled the maximum attainable power of a biker as a quadratic decay that goes to zero as the biker spends all of their stored energy that is towards the end of the race. As shown in Fig. 5, this model does not represent all sorts of bikers. That is why extra work is needed to be done by the directeur sprotif to figure out the shape of this curve and how to represent it as a function in order to input it and produce more tailored results.

Weather effects play a very minor role in our modeling and don't affect the overall strategies of the cyclists. We recommend that the cyclists use the same strategies as normal as long as safety allows it.

Finally, directeur sprotif should be more knowledgeable in terms of extended usage of high power and its effect on cramps and injuries. This can be easily implemented into the program to prevent injuries and keep the bikers safe.

References

[1] HODSON-TOLE et al., During cycling what limits maximum mechanical power output at cadences above 120 rpm? 2019

[2] The Tokyo 2020 Olympics website https://olympics.com/en/featured-news/olympic-road-cycling-tokyo-2020-games-2021-five-things-preview

[3] The Flanders 2021 world championship https://www.cyclist.co.uk/in-depth/10089/flanders-world-championships-2021-time-trial-preview

[4] The Fuji speedway https://www.snaplap.net/fuji-speedway/

[5] The different types of cyclists https://digitsole.com/what-are-the-different-types-of-cyclists/

[6] Sigmoid function inspiration https://www.youtube.com/watch?v=v68zYyaEmEAt=1166s

[7] The computational cyclist https://www.gribble.org/cycling/power$_{vs}$*peed.html*

[8] Genders and cycling https://www.cyclingweekly.com/fitness/cycling-and-gender-how-and-why-male-and-female-cyclists-need-to-train-differently-344365