Cycling Pace Optimization

Aaron Vliet

November 2024

1 Introduction

The goal of this project is to find the optimal pacing in terms of 'normalized power' (8) for a cyclist to complete a segment of road (e.g. a time trial course) in a specified time. Segment data (e.g. grade vs distance) is sourced from Strava. The goal is to create workout files such that optimal pacing can be performed in real life. The optimizer results can also be used to evaluate how 'optimal' pacing was in past efforts.

In the future, a more complicated optimizer may be formulated that takes into consideration how a cyclist's C_DA changes with respect to power (e.g. a cyclist doing 200w can be in a much more aero position than one sprinting at 1000w), and perhaps even how multiple riders should pace a team time trial given some assumptions on drafting and pace-line rotation.

1.1 Preface

Note that cycling pace optimization has already been done in the past - see bestbikesplit and this paper. Therefore, my goal with this project is instead to learn more about trajectory optimization (and a general feel for time trial pacing), as well as to practice general modeling of real-world systems. However, the exact way I'm framing the problem (in terms of normalized power and a specified segment time) maybe hasn't been done yet, as far as I can tell from my brief literature review.

1.1.1

This document was originally written to record my progress for later reference, as I don't have a lot of time for personal projects and thus often can't work on them consistently. However, since I wrote most of it up already, I figured I might as well add a few more details and post it to my portfolio website. I am hoping to further improve/fix the optimizer, at which point I may make my GitHub repo public as well.

2 Dynamics

For the equations of motion, we will be using a "body axes," where (x, y, z) corresponds to (forward, left, up) relative to the bike — slightly different than typical aircraft body axes. Let $v = \dot{x}$. The dynamics of a rider in the $\hat{x}$ direction can be approximately described by the following equations for gravitational force (1), rolling resistance (2), and aerodynamic drag (3):

$$F_{gx} = -mg\sin(\theta) \tag{1}$$

$$F_{Crr} = -C_{rr}mg\cos(\theta) \tag{2}$$

Where $\theta = 0$ is a flat road, and $\theta > 0$ is a incline.

$$F_d = -\frac{1}{2}C_DA\rho(v)^2 \tag{3}$$

However, the above equation doesn't take into account wind, so we will instead consider:

$$F_d = cos(\beta)\frac{1}{2}C_DA\rho(|\vec{v} + \vec{w}|)^2 \tag{4}$$

Where $\vec{w}$ is the wind velocity (relative to the ground), with $w > 0$ being a headwind. β is the yaw angle, e.g. the angle between the velocity vector and the wind vector. Drag is aligned with the direction of incident airflow, but we only really care about the component aligned with the velocity vector of the rider, hence the $\cos\beta$ term. However, this isn't necessarily the best way to model drag in crosswind conditions, since the C_DA of a rider will change with the yaw angle. Additionally, for highly aero-optimized bike setups (e.g. time trial bikes), parts of the bike can act as symmetric airfoils such that when there is a moderate yaw angle (e.g. on the order of

$\approx 10°$), lift will be generated. That lift is orthogonal to the incident airflow, meaning it will have a component in the velocity direction. This is often referred to as the "sail effect," and can reduce the resultant aero force of individual components (e.g. wheels) significantly. In the future it could be interesting to try to model this, but for now it will be assumed to be a very small higher order effect.

Force produced from power:

$$F_P = \frac{P}{|v+w|} \tag{5}$$

We can now sum the horizontal forces to get the equation of motion:

$$\Sigma F_x = F_P + F_d + F_{gx} + F_{Crr} \implies \Sigma F_x = \frac{P}{|v+w|} - mg\sin(\theta) - C_{rr}mg\cos(\theta) - \frac{1}{2}CdA\rho(|v+w|)^2 = ma \tag{6}$$

2.1 Solution

Assuming steady state, the symbolic expression for velocity in terms of power can be found using either the cubic formula or a symbolic solver in something like MATLAB or Mathematica — I used MATLAB. The result is, of course, quite nasty looking:

$$v(p) = \left(\sqrt{\frac{P^2}{C_D A^2 \rho^2} + \frac{(2\,g\,m\,\sin(\theta)+2\,C_{rr}\,g\,m\,\cos(\theta))^3}{27\,C_D A^3 \rho^3}} + \frac{P}{C_D A \rho}\right)^{1/3} - \frac{2\,g\,m\,\sin(\theta)+2\,C_{rr}\,g\,m\,\cos(\theta)}{3\,C_D A \rho\left(\sqrt{\frac{P^2}{C_D A^2 \rho^2}+\frac{(2\,g\,m\,\sin(\theta)+2\,C_{rr}\,g\,m\,\cos(\theta))^3}{27\,C_D A^3 \rho^3}}+\frac{P}{C_D A \rho}\right)^{1/3}}$$

On the other hand, the symbolic expression for power in terms of velocity is quite nice:

$$p(v) = (|v+w|)\left(\frac{\cos(\beta)C_D A\rho(|v+w|)^2}{2} + C_{rr}mg\cos(\theta) + mg\sin(\theta)\right) \tag{7}$$

2.2 Examples

For the following examples, we will use a fairly average $C_D A$ of .32, rolling resistance of $\frac{5}{1000}$, $1.225\frac{\text{kg}}{\text{m}^3}$, and a combined bike and rider mass of 75kg. Using the above equations, we can find a rider producing 200 watts will move at roughly 21.2 mph, and if the rider wants to move at 20 mph on a flat road, roughly 170 watts will be required. Compared to real life, this seems slightly optimistic, but plausible. It is close to the Strava Sauce Performance Predictor tool.

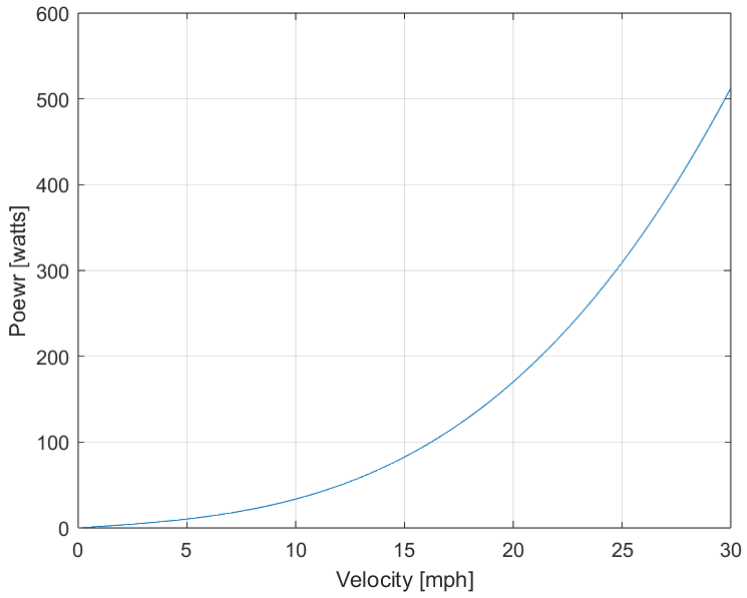

Figure 1: Power vs speed

3 Additional notes

3.1 Normalized Power

Normalized power is a common metric among cyclists that describes how difficult an effort is. It is similar to average power, but weighs high power output more heavily, which more accurately describes the physiological tax of an effort. TrainingPeaks describes it as "an estimate of the power that you could have maintained for the same physiological 'cost' if your power had been perfectly constant, such as on an ergometer, instead of variable power output."

Let $\mathbf{P}_{30}$ be an array containing the average power for every 30 second period. Let N be the number of periods.

$$\mathrm{NP} = \left(\frac{\sum \left({\mathbf{P}_{30}}^4 \right)}{N} \right)^{\frac{1}{4}} \tag{8}$$

3.2 Data source

From the analysis tab on a Strava activity, the StravaSauce browser extension allows one to download raw data, including things like speed, power, altitude, and grade. There is one data point per second.

3.2.1 A note on wind direction

As it turns out, the raw data that StravaSauce provides does not include the heading/the direction of travel. This means that it isn't so easy to implement wind direction. In the future I will likely write a function that calculates this based on the change in lat/long coordinates, but for now we will assume no wind for simplicity.

3.3 Parameter modeling

3.3.1 CDA calculation

To accurately optimize pace, we need to have a reasonably accurate model of the system. While things like mass and air density are trivial to measure, C_DA is slightly more involved. Therefore, I've written a simple MATLAB script that calculates C_DA from a Strava ride. Specifically, the script calculates C_DA for time steps of either 1 second or 10 second duration, and then takes C_DA as the average. This method — particularly with 10 second time steps — should be slightly more accurate than calculating the C_DA over the entire segment at once. However, due to braking, coasting, noise in the data, etc., sometimes there will be brief spikes in C_DA. For these points, I replace them with the StravaSauce Performance Predictor C_DA estimation, which is the C_DA over the entire segment based on net change in altitude. Ideally, a wind speed sensor would be used rather than GPS groundspeed, but in the absence of that, the function should be instead limited to use on calm days, or at least on loops or out and back segments.

Figure 2: CDA estimation

3.3.2 Rolling Resistance

The coefficient for rolling resistance (C_{rr}) is also a bit hard to estimate, but has a much smaller effect on the overall system dynamics and thus also on the optimization. However, C_{rr} could be estimated using a tow test/roll down test, but this is fairly difficult, especially if $C_D A$ is not precisely known. To achieve reasonably accurate real-world C_{rr} estimates, a lot of data would need to be collected and from that some sort of regression (least squares would probably suffice?) to estimate the rolling resistance.

4 Optimization problem transcription

The goal here is to transcribe the trajectory optimization problem into a "constrained parameter optimization" problem — that is, a finite dimensional problem where the "decision variables" are numbers rather than functions. Basically, this means we will be finding an array of optimal power values rather than a continuous function for power.

4.1 Minimization function

We will use our normalized power function (8) as our objective function. $N = \frac{\text{target time}}{30}$ However, we will consider each power parameter to correspond to a timestep of 10 seconds rather than 30 seconds ($\implies length(\mathbf{P}) = 3N$). I think 10 seconds is roughly the shortest period that a cyclist can vary their power. However, longer time steps would obviously improve runtime significantly.

4.2 Constraint functions

We will have two constraint functions. Let $v[n]$ and $\theta[k]$ be arrays, both of length $3N$ (equal to the length of $\mathbf{P}$).

4.2.1 Dynamics

We will approximate each discrete section as being under steady state. Therefore, the

$$\frac{\mathbf{P}[n]}{|v[k]|} - mg\sin(\theta[k]) - C_{rr}mg\cos(\theta[k]) - \frac{1}{2}CdA\rho(v[k])^2 = m\dot{v} = 0$$

4.2.2 Velocity constraint

We want the average velocity to be equal to $\frac{\text{Total distance}}{\text{Target time}}$

$$\implies v_{avg} = \frac{1}{3N}\sum_{n=1}^{3N} v[n]$$

4.3 Implementation

Now that the optimization problem defined, we can use MATLAB's fmincon() function to find the optimal power. In the future, I may switch to python, but fmincon is very convenient for the time being.

4.4 Time dependence and position dependence

However, there's a slight complication: the dynamics equation has both time dependence and position dependence, since power is calculated for each time step, and theta is a function of position. This is a fundamental flaw in the way the optimization problem was posed. If we only have power, we can't determine velocity directly, since we also need theta. But... to calculate theta, we need velocity.

My approach to solve this issue is to first assume constant velocity. Broadly speaking, this should be vaguely optimal (depending on how much wind and/or how much the grade varies), so it should be a reasonable initial guess. From there, we can find the average theta value for each timestep. Now that we have a better estimate for theta, we can recalculate velocity. Through this iterative process, we can hopefully converge on the theta and velocity arrays that match our power array.

5 Results and evaluation

To 'validate' the optimizer, we need something to check it against. The obvious option is to use existing Strava data from Individual Time Trials (ITTs). Ideally we can also evaluate pace with one single "figure of merit" (FoM) value.

$$\text{FoM} = \frac{\text{Normalized power}}{\text{average velocity}}$$

So far the results have been generally promising, but not as insightful as I had perhaps hoped.

5.1 Example 1

The most notable example that I've analyzed so far is the 2024 Tour de France stage 21 ITT from Monaco to Nice, France. This was compared to Santiago Buitrago's effort, who seems to be one of the few top riders that day that posted to Strava with power publicly visible. The TdF stage 21 ITT is a good candidate for optimization and analysis since it features both a long, sustained climb in addition to shorter climbs and sustained flats and downhills. Additionally, since the riders were pros on time trial bikes, their C_DA is likely more consistent than a more amateur rider on a road bike.

To compare to Buitrago, I pulled his weight from ProCyclingStats and assumed about 10kg for the bike and any other miscellaneous items, putting the total mass at 70kg. From there, I approximated Buitrago's C_DA — see figure 3. The result was a fairly decent average C_DA of roughly $0.22m^2$. However, the C_DA clearly varies significantly between the climb, the descents, and the flats here, so assuming a constant C_DA is not particularly accurate.

Notably, figure 4 shows that the optimizer seems to prefer a harder pacing strategy up the climbs than what Buitrago rode. The optimizer was also able to achieve a slightly lower FoM of 29.36 joules/meter, in comparison to Buitrago's 30.51 joules/meter. However, Buitrago was also able to catch up to the supposed optimal solution by the second, much shorter climb — perhaps in part due to a reduced C_DA on the descent. The supposed optimal power is also a bit spikey, with a surge of over 800 watts near the beginning, and other unstable sections on the second climb. The optimizer also doesn't need to slow down for turns. Towards the end, Buitrago clearly slows dramatically for the 180° turn in Nice, while the optimizer continues to chug on by. Given the lack of braking coupled with the potential for slightly inaccurate constants (C_DA, mass, air density), it is hard to say that the slight decrease in normalized power (and the FoM) is significant.

Segments with more rolling terrain seem to perform notably worse. Figure 5 shows a significantly higher FoM over the 2024 Charlie Baker Time Trial (CBTT) course in Massachusetts. The optimizer's solution is compared to the power of one of my friends, Derek Schaadt, an amateur rider on a road bike with clip-on time trial bars. It isn't clear exactly why the optimizer's solution is so bad — it could be due to inaccuracy introduced from discretizing the problem, or perhaps the constants associated with the problem aren't quite right. Either way, the optimizer is clearly not returning anywhere near the optimal result. However, the power does seem to match the shape of Derek's power fairly closely if the vertical shift is ignored.

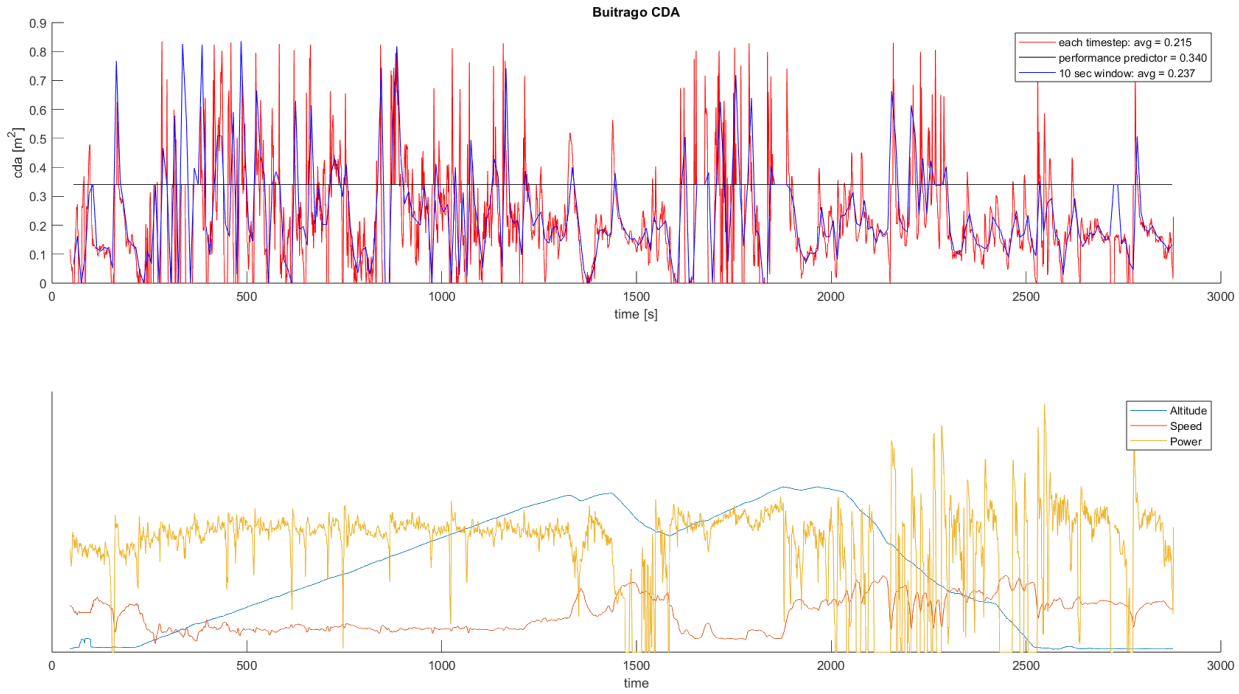

Figure 3: Buitrago CDA calculation

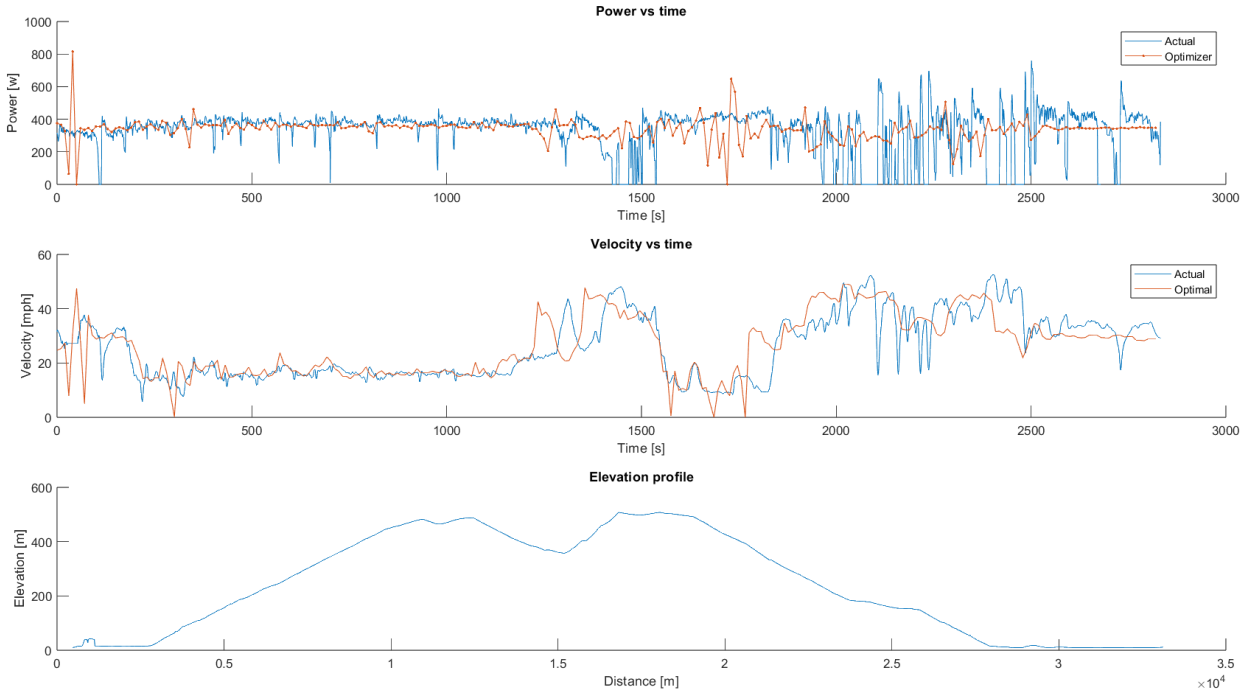

Figure 4: Optimizer vs Buitrago, TdF stage 21

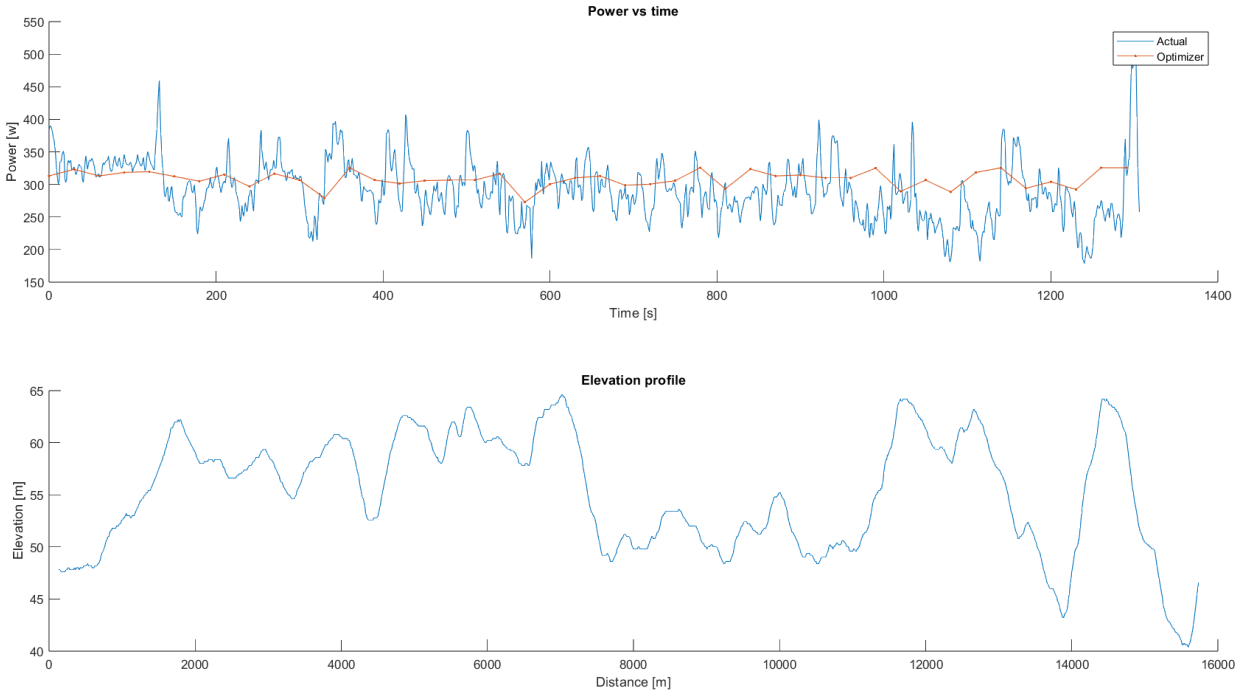

Figure 5: Optimizer vs Derek, CBTT