Optimal Pacing Strategies for Cyclists in Time Trials

A Research Proposal for the Degree of
Doctor of Philosophy, Mechanical and Aerospace Engineering
June 11, 2012

Gilbert Gede
Department of Mechanical and Aerospace Engineering
University of California Davis
email: ggede@ucdavis.edu

This document first provides background information for the research proposal, the goals of the research proposal, and a proposed timeline.

1 Background & Motivation

Many different sports have events which involve "racing against the clock" - running, swimming, rowing, and cycling are examples. Some competitions might involve other athletes but not direct competition; swimming competitions, regattas, and some running events involve different lanes, and cycling time trials do not permit drafting which removes the influence of other competitors.

In these types of events victory is determined by an individual's athletic abilities and pacing strategy, not by interactions with other competitors. This leaves limited options to improve performance: equipment selection, training/conditioning, and pacing strategy. While these factors are all controllable to some degree, this research only considers the pacing strategy.

Listening to athletes and coaches, different proposed strategies on race openings, middle sections, and closings will be heard. Starting slowly or quickly, increasing speed throughout the race, finishing with a kick - these are all different strategies people advocate, but there is no agreed on answer. For some events - the 100 m dash for example - all-out for less than 10 seconds is the obvious strategy. but for longer events, or those with variations in the course or environment, the answer is not obvious.

Examining the world of optimal control, we see that the optimal solution to control problems can be very non-intuitive; one example is the minimum time to climb problem for the f-4 jet fighter [6]. In this example, in order to reach an higher altitude in the minimum time, it is fastest to actually dive first, then perform a zoom climb. This is very counter-intuitive, but also serves as an example of what information can be gained by applying optimal control to a problem such as a minimum time athletic event.

A number of authors have examined human pacing strategy previously. Keller examined the optimal strategy for a runner at different distances and compared the results to the current (at the time) world records [7]. Keller finds an optimal solution using a energy model which uses the oxygen debt, an outdated theory. Since then, Ward-Smith has investigated how anaerobic metabolism affects energy production [8], but in a fashion which is not extensible to longer distance events or events of variable power output. Additionally, Morton has studied the minimal time problem for running using a modified critical power model, and has found that the optimal solution is all-out for the entire race [10]. All of these efforts have limitations though, in model fidelity or extensibility.

The goal of this research is to expand upon previous work in the field, and apply it to cycling. Cycling time trials have been chosen due to the ease of modelling the physical system and measuring actual riders. Time trials are ideal because they remove the other riders from the problem, and thus remove the issue of drafting (and optimal position within the peloton). By formulating a more realistic model of human bioenergetics, in addition to a more flexible framework for defining the course to allow for more variability, it is hoped that new insights into the minimum time problem can be found.

If an athlete's energy production capabilities can be accurately modeled, the course and environment represented, and the physical dynamics simulated, this research hopes to be able to determine the optimal strategy for the event. This could be in the form of time-histories of power output, thrust, or speed. This would serve as a guide, or map, for the athlete to follow during the event. This could allow for further reductions in event times, possibly allowing new records or giving athletes an edge in a stage of a tour.

2 Research Goals

There are 4 components which are not specific to the cyclist/time trial that are part of this research proposal: creation of a generic framework which applies direct collocation methods to optimal control problems, improvements on automatic analytic calculations of Jacobian and Hessian matrices for optimal control problems, implementation of a nonlinear programming algorithm, and development of a validated bioenergetics model. All of these components will be used in answer the question: what is the optimal strategy for a cyclist in a time trial. Before discussing the other parts of the research proposal, a description of the cyclist time trial problem will be given. Please note that all code which will be written will be available under an open source license.

2.1 Problem Definition

In cycling, a time trial is an event in which a cyclist attempts to cover a specified distance in a minimum time. This almost always occurs on a predetermined course with fixed starting and finishing points. What makes a time trial different than other cycling events is that each cyclist is on the course independent from the influence of other cyclist; they all start and finish the race at different times and are not allowed to draft. Therefore, in order to win the event, the only relevant parameters are the athlete's abilities and current condition, equipment, and the strategy they use.

I would like to point out that time trials can exist in many situations. They can be part of a cycling tour, either at the start or between other stages. They can also be part of triathlons, where the cycling portion happens between the swimming and running stages. Time trials can also be events independent of another competition. In the first two cases, tours and triathlons, there are some conditions which must be discussed. In cycling tours, the time trial stage is just one of many stages. The competitor could have had an event the day before and another the day after the time trial. This can limit their performance, both involuntarily and voluntarily: they might not be in an ideal, well-rested state going into the event, and they might not want to induce too much fatigue that day as they have another event soon. Triathlons have their three stages in one day, back-to-back, but are made of different sports: swimming, cycling, and running. I do not intend to study time trials as part of a tour or triathlon at this point in time, due to these complexities. I will instead study the time trial as an independent event.

By removing these complex (but real-world) constraints, I have narrowed the cyclist/time trial problem down to two parts, the physical dynamics and the bioenergetic dynamics; when put together they define the optimal control problem. These are both dynamic systems of time and need to be described by a set of differential equations and state variables. The details of the bioenergetic dynamics will be discussed in a later section; I will now describe the physical dynamics of the cyclist.

2.1.1 Physical Dynamics

The bicycle as a multibody system has been studied for some time. Modelling the bicycle and rider as a multibody system leads to a large number of states. For optimal control problems, especially when solved numerically, the solution is harder to calculate as the number of states increases. Quantities such as roll angle, roll rate, steer angle, steer rate, wheel rotation angles, rider lean angle, rider lean rate, etc. All change on timescales which are relatively fast compared to the duration of the event (seconds vs. hundreds of seconds).

An assumption will thus be made: there will not be any significant lateral maneuvers during the event, nor significant braking events. These justify reducing the dynamics of the multibody system down to those of a particle, following a predefined path. However, I still intend on modelling the inertia of the wheels and legs, with a simplification: the cyclist will have a desired cadence, and the geartrain can be modeled as a continuously variable transmission. The wheel rates will already be a function of the ground speed and by making these assumptions the rider's pedaling rate can also be written as a function of ground speed.

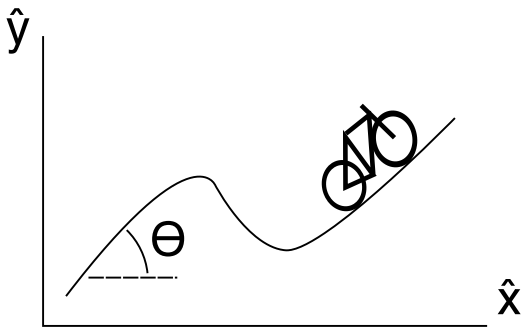

Figure 1: A "side view" of the cyclist's course

Figure 1 is a simplified representation of the courses that will be modeled; lateral motion is already being ignored leaving only longitudinal and vertical motion. The cyclist is clearly not free to move in both dimensions though; they have to stay on the ground. This reduces the two coordinates down to one. The vertical displacement will be defined as a function of the longitudinal displacement, leaving only 1 coordinate to define the position of the cyclist on the course, q_x. The speed also needs to be considered, so longitudinal speed will be represented as $\dot{q_x}$.

Following the previous assumptions and course definition, the system can be broken down into various parts to be modelled: the center of mass of the bicycle/rider system, the rotational inertia of the wheels, and the rotational inertia of the riders legs. The velocity of the mass center is defined as:

$$\underline{v}^b = \dot{q}_x \hat{x} + \dot{q}_x \tan(\theta(q_x))\hat{y} \tag{1}$$

The tangent speed can be defined (after some trigonometric transformations) as:

$$v = \frac{\dot{q}_x}{\cos(\theta(q_x))} \tag{2}$$

The angular velocity of each wheel is defined as:

$$\underline{\omega}^w = -v/r\hat{z} \tag{3}$$

The mass of the complete bike is m, and the inertia of the wheels is I_w.

The rider's legs pose a difficult problem; the speed will be derived from the earlier assumptions. The crank rate will be a function of ground speed, approximated as a continuously variable transmission instead of as a set of discrete gears. Below a "first gear speed" the angular rate will be the product of ground speed and a constant, above a "top gear speed" the angular rate will again be the product of ground speed and another constant, and in between the angular rate will be constant (over the transmission's continuously variable range). The inertia of the cyclist's legs will be lumped/averaged into I_l. The angular rate of the rider's legs is then defined by:

$$\underline{\omega}^l = \begin{cases} -c_1 v \hat{z} & \text{if } v < v_1 \\ -w_0 \hat{z} & \text{if } v_1 \leq v \leq v_2 \\ -c_2 v \hat{z} & \text{if } v_2 < v \end{cases} \tag{4}$$

Where $c_1$and c_2 are functions of the bicycle's geartrain, w_0 is a function of the rider, and v_1 and v_2 is a function of the rider and bicycle's geartrain.

The forces on the system are: rider leg torque, aerodynamic drag, rolling resistance, and gravitational force (when on a slope). These are defined as:

$$T = -M_l \hat{z} \tag{5a}$$

$$F_d = -c_d v \underline{v}^b \tag{5b}$$

$$F_{rr} = -c_r f_n \frac{\underline{v}^b}{v} \tag{5c}$$

$$F_g = -mg\hat{y} \tag{5d}$$

Lagrange's equations can be used to find the equation of motion for this system. First, I found the kinetic energy of each body. The bike included the mass of the rider's legs and the wheels, and the rotational kinetic energies were found for each rotating body. Next, I used Lagrange's equations to find the equation of motion. Note that a small substitution has been made: the inertia of the wheels is assumed to be that of a thin ring, $I_w = m_w r^2$, which I believe is a reasonable approximation.

$$\ddot{q}_x = \frac{1}{2(m + 2m_w + I_l c^2)}[cM_l \cos\theta \\ -c_d \frac{\dot{q}_x^2}{\cos\theta} - c_{rr}\dot{q}_x^2 \frac{\partial\theta}{\partial q_x} - mg\cos^2\theta(\tan\theta + c_{rr})] \tag{6}$$

Note that c will change with speed, as shown in the definition of $\underline{\omega}^l$ above; between v_1 and v_2 $c = \frac{\omega_0}{v}$ and I_l will be 0, as that term will drop out when forming Lagrange's equations with the legs at a constant speed.

In summary, I am using two states to define the physical dynamics: horizontal speed and horizontal position. The dynamics will follow the equation of motion presented above. To solve the optimal control problem, I will use the direct collocation method which will now be discussed. Also, the physical dynamics are one half of the system in the optimal control problem; the other half of the system is in the human's bioenergetic capabilities which will be discussed in a later section.

2.2 Direct Collocation Framework

In this section, I will first provide a brief overview of optimal control and how I think about it in relation to my problem. Next, I will discuss a particular method used to solve optimal control problems, and what I have currently accomplished with it. Finally, I will discuss where I plan to go in the future with this research.

2.2.1 Optimal Control Background

There are many different types of optimal control problems; some involve minimal fuel consumption for a maneuver, maximum distance traveled in a fixed time, maximum load to orbit, etc. In a general sense, a basic optimal control problem could be defined as follows (although many variations are possible):

$$J = \phi + \int_{t_0}^{t_f} \mathcal{L} \tag{7a}$$

subject to

$$\psi = 0 \tag{7b}$$

$$\dot{y} = f[y(t), u(t), t] \tag{7c}$$

where $y(t)$ is a vector of state variables and $u(t)$ is a vector of controls. J is the cost function (what is being minimized or maximized), ϕ is a cost based on the beginning or ending states and controls, $\mathcal{L}$ is a cost based on a function integrated over time, ψ is a vector of beginning and/or ending constraints on the states or controls, and the state time dynamics are represented by f. This is a relatively general cost function which could be used in most situations, and the rest of the formulation can be used in most cases without path constraints. Bryson [15] provides many different optimal control problem formulations which are variations on the above.

For simple cases, the optimal control problem can be solved analytically using calculus of variations and the Euler-Lagrange equation. This can involve augmenting the cost function with the constraints and state dynamics and solving for a stationary position, indicating an extrema. Unfortunately, for more complex problems, an analytical approach is not feasible.

Specifically, for the cyclist, the analytical approach is only feasible for courses with a non-variable slope and a simple bioenergetics model.

A numerical approach is required for a more complex case, including the more general and realistic cycling time trial cases I wish to study. Most of the numerical approaches for these complex problems involve transcription of the equations into a form which can be provided to a generic optimization code (one not specifically designed for optimal control problems).

More details of optimization codes will be discussed later, but in general they work as follows: a cost function, a function which gives equality constraints, a function which gives inequality constraints, and an initial guess are provided; the input (which is what is being optimized) is a vector, the cost function returns a scalar, and the equality and inequality constraints are vectors. Transcription of an optimal control problem involves a definition of each of these 3 functions, as well as a selection of a starting point.

A simple approach to this problem is the shooting method. With this method, the time range of the problem is discretized and forward integration is performed. The input vector consists mainly of the controls, although initial state values, system parameters, and the final time can be part of this vector in some problems. The equality and inequality constraints are then evaluated at discrete time points, and the cost function is computed with the results of the time integration.

While straightforward, the shooting method has some downsides. One downside is the "disconnect" between the input vector and the cost and constraint functions. The values of these functions are computed from the output of the time integration, which has its own numerical characteristics. Computation of the Jacobians of these functions becomes quite difficult and when performed numerically, can be inaccurate. This leads to a failure to find optimal solutions.

2.2.2 The Direct Collocation Method

The direct collocation method provides an alternative approach to the problem. In this approach, the first step is to add to the input to the NLP code. Time discretization still occurs, but instead of only having the controls defined at each time step, the controls and the states defined at each time step within the input vector. Immediately, it is clear that the size of the optimization problem has increased; instead of the input vector being approximately the size of (no. of controls × no. of time steps), it is now (no. of states + no. of controls × no. of time steps).

In order to ensure that the time dynamics of the system are not violated, additional equality constraints are defined. The states have already been discretized over the time interval, now the trajectory of the states will be approximated using a series of piecewise cubic polynomials. An assumption in the direct collocation method is that this will be able to adequately represent the states (higher order polynomials can also be used). By using cubic polynomials, the states at the midpoint of a time interval and the derivatives of the states at the midpoint can be found analytically, from the implicit definition of these values. What is done next is to compare the analytical derivative of the midpoint of the polynomial to the state derivative function evaluated at the polynomial midpoint; when these are equal, the claim is that the changes in the states over time do not violate the system dynamics. See the following equations:

$$\text{given } \dot{y} = f(y,t) \tag{8a}$$

$$\text{and } t \in (t_1, t_2) \tag{8b}$$

$$\text{we define}$$

$$y_1 \triangleq y \text{ at } t_1 \tag{8c}$$

$$y_2 \triangleq y \text{ at } t_2 \tag{8d}$$

$$f_1 \triangleq f(y_1, t_1) \tag{8e}$$

$$f_2 \triangleq f(y_2, t_2) \tag{8f}$$

$$\text{and compute} \tag{8g}$$

$$y_c = \frac{1}{2}(y_1 + y_2) + \frac{\Delta t}{8}(f_1 - f_2) \tag{9a}$$

$$t_c = \frac{1}{2}(t_1 + t_2) \tag{9b}$$

$$\dot{y_c} = -\frac{3}{2\Delta t}(y_1 - y_2) - \frac{1}{4}(f_1 + f_2) \tag{9c}$$

$$f_c = f(y_c, t_c) \tag{9d}$$

Above, y is a vector of all the states. Once $\dot{y_c}$ and f_c are defined, a "defect" is defined as the difference between the two.

$$\Delta \triangleq f_c - \dot{y_c} \tag{10}$$

Note that Δ is a vector of the same size as y. A defect vector is defined for each polynomial; if there are N time steps, there will be $N - 1$ polynomial arcs and $N - 1$ defect vectors. Now, these Δ vectors are included in the equality constraints; when they are equal to 0, the system dynamics are being adequately represented. Compared to the shooting method, the rest of the transcription is the same; the existing equality constraints and inequality constraints exist, and the objective function is unchanged (although if the objective function relied upon integration, an extra state might need to be added). Note that there is no more numerical integration.

2.2.3 Current Results

Previously, I have used direct collocation to solve a simplified model of the time trial problem. The energetics are based on the critical power model (discussed later), and the dynamics involve modelling a point mass with thrust as an input. The course is flat in this example and is 2500 meters in length.

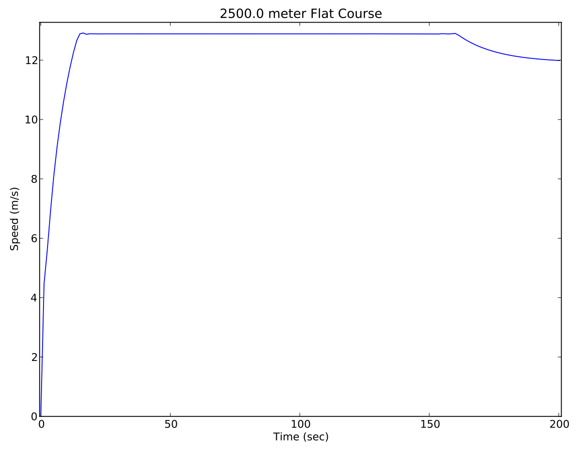

Figure 2: Old results achieved with the direct collocation method and the critical power model.

Examining figure 2, 3 distinct phases are present: a ramp up in velocity, a cruise section, and a “coast” section at the end. This is very similar to the optimal strategy found by Keller for a runner, which he calculated analytically [7].

2.2.4 Proposed Work

I have shown that the direct collocation method can be used to compute optimal pacing strategies for a basic time-trial/cyclist model. There is other work which can be done using direct collocation though. First, I intend to continue to use this method and apply it to the more complicated physical and energetic models I plan on using. I would also propose development of open source code to facilitate solutions of optimal control problems with direct collocation.

Currently, there are existing commercial codes, one example is DIRCOL [16], which transcribes optimal control problems into nonlinear programming problems. I intend to take user-defined analytic equations (represented symbolically) for system dynamics and constraints, and transform them into functions for a NLP code to use.

This is similar to the currently available options, and I propose a number of improvements. First, automatic determination of the initial guess for the states from the initial control guess, reducing the amount of initial information to be provided. Second, by comparing the results of numerical integration to the polynomial approximations, determine two things - the degree of polynomial required or the amount of time-discretization required. Automatic refinement of the time discretization is the third facet; after initial results are returned from the NLP code, examining the results and identifying areas which would benefit from a finer time grid allows for the problem to be redefined and run again, to allow for more accuracy. Fourth, automatic creation of compiled C functions from the problem definition; while the use of Python or MATLAB allows for easy programming, there is a significant performance disadvantage compared to compiled codes. There also needs to be a way for user-defined functions which are not analytic to be supplied and nicely integrated with these compiled functions. The final improvement I propose is in a combination of automatic/minimal-computation Hessian and Jacobian matrices.

2.3 Efficient Hessian and Jacobian Calculation

Nearly all nonlinear programming codes require the Jacobian matrices of the constraints and of the objective function in addition to the Hessian of the Lagrangian. Codes which do not require this information are not considered here. NLP codes which do not explicitly require these matrices almost always approximate them numerically. The ability of the NLP code to reach an optimal solution depends on the accuracy of these matrices.

In this section, I will mainly discuss the Hessian matrix. The computational cost of numerically computing it is quadratic relative to the Jacobian matrices. I have found that an accurate Hessian matrix can significantly reduce the number of iterations required to solve a problem, which is the motivation for this part of my research. Furthermore, most of what I propose is applicable to the Jacobian matrices and is simpler in such an application.

For most NLP algorithms, while the Lagrangian used in the formulation may vary, the following components are relevant to the Hessian calculation:

$$\mathcal{L} = \phi(x) + \lambda_{eq} c_{eq}(x) + \lambda_{ineq} c_{ineq}(x) \tag{11}$$

where the λ's are vectors of Lagrange multipliers, the c's are vectors of equality and inequality constraints, and x is the input vector. While there may be other terms in the Lagrangian, they do not show up in the Hessian. The Hessian matrix is defined as:

$$H = \begin{bmatrix} \frac{d^2}{dx_1 dx_1}\mathcal{L} & \frac{d^2}{dx_1 dx_2}\mathcal{L} & \cdots & \frac{d^2}{dx_1 dx_n}\mathcal{L} \\ \frac{d^2}{dx_2 dx_1}\mathcal{L} & \frac{d^2}{dx_2 dx_2}\mathcal{L} & \cdots & \frac{d^2}{dx_2 dx_n}\mathcal{L} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{d^2}{dx_n dx_1}\mathcal{L} & \frac{d^2}{dx_n dx_2}\mathcal{L} & \cdots & \frac{d^2}{dx_n dx_n}\mathcal{L} \end{bmatrix} \tag{12}$$

Within most NLP codes, the Hessian matrix is calculated with a BFGS update [5]. This involves examining the change in the Jacobian of the Lagrangian with the change in the input vector. While this provides acceptable performance, it requires a large number of iterations to reach accurate values (typically, $H = I$ is the initial values).

For more accurate Hessian computations, there are other options. The Hessian can be computed from finite differences, but there can be problems with numerical accuracy with this approach (as ϵ^2 terms are in the expression). Yokoyama et. al [17] provide a few alternatives, based on using the structure of the Hessian and sparse techniques. Examining the majority of the equality constraints which are the defects, Δ, it is clear that only two time points (and the states at each) are involved in each constraint; this will lead to a block diagonal structure for the majority of the Hessian matrix. Sparse matrix operations would most certainly be of benefit in this case, but it is not possible to take advantage of this sparseness with the finite difference computations.

Automatic or symbolic differentiation is another option, but it requires analytic knowledge of the problem. Fortunately, this information will be part of the direct collocation code I propose implementing. What is more likely in reality is a mostly analytic definition of a problem with some numerical functions; e.g. a rocket going through the atmosphere whose drag is represented by a complex function, possibly involving lookups and interpolation, but where the rest of the dynamics are relatively simple. In such a case, I propose that the code I develop uses a mixture of symbolic differentiation and minimal finite difference calculations to find the Hessian; the necessary calculations will arise from the application of the chain rule. The number of finite differences involved would be of order $O(n)$, rather than $O(n^2)$, and would be applied to much simpler functions. This same approach can also be applied to equality and inequality constraints not based off enforcement of the system dynamics; e.g. a path defined by discrete data points which must be followed.

2.3.1 Proposed Work

I propose creating a general mathematical formulation for the Hessian matrix for direct collocation optimal control problems, and an implementation of this to go with my proposed direct collocation code. Furthermore, I plan on applying these same techniques to the calculations of the Jacobian matrices for the equality and inequality constraints, in addition to performance tests for computing these matrices.

I also intend on running numerical experiments comparing the performance of this against alternative methods for Hessian calculations. Additionally, the most nonlinear programming codes require a positive definite Hessian matrix. I will examine current solutions which are used to ensure a positive definite Hessian and determine a method to use.

2.4 Nonlinear Programming Code

There are many different classifications of optimization problems. Objective (cost) functions can be linear or nonlinear, there can be equality and/or inequality constraints, each of which could be linear or nonlinear. Specifically, the methods I will discuss are for local optimization and are not integer programming or mixed-integer/nonlinear methods. Additionally, there are methods which do not require the derivatives of functions; these methods are not discussed or considered.

A relatively large number of nonlinear programming codes currently exist - at least dozens of different solvers. For this research two different codes have been applied to the cyclist's problem, and a third is under development. In MATLAB's Optimization Toolbox, the `fmincon` function provides an interface to code written by Mathworks [3]. Within the NumPy/SciPy libraries, `fmin_slsqp` provides an interface to the code SLSQP written by Dieter Kraft [4]. Other code under development is based on the work present by Byrd [5].

Most NLP methods fall under the category of active-set or interior-point. To understand the distinction, first look at the simple equality constrained case:

$$\min J(x) \tag{13a}$$

$$\text{subject to } h(x) = 0 \tag{13b}$$

In order to solve this, we minimize the Lagrangian of this problem: $\mathcal{L} = J(x) + \lambda^T h(x)$, and we claim that a minimum has been found when the derivative of the Lagrangian with respect to x is 0 (actually, this is a stationary position; there are further conditions for a minimum). In the above Lagrangian, λ represents the

Lagrange multipliers; these values can be interpreted as how much each constraint is affecting the solution. When there are inequality constraints, in the form $g(x) \leq 0$ the problem becomes more challenging.

For the active-set method, while iterating, we examine which inequality constraints are close to 0. It is assumed that constraints which are very near 0 would actually be at 0 in an exact solution. We then group these inequality constraints with the equality constraints, call these the active-set, and treat them as equality constraints. The other inequality constraints are ignored (not part of the active-set).

In the interior-point (barrier) method, the NLP problem is written differently:

$$\min\ J(x) \tag{14a}$$
$$\text{subject to } h(x) = 0 \tag{14b}$$
$$g(x) + s = 0 \tag{14c}$$
$$s \geq 0 \tag{14d}$$

Above, s is a slack variable, used to transform the inequality constraint into an equality constraint. The Lagrangian is now defined as:

$$\mathcal{L} = J(x) + -\mu \sum_i lns_i + \lambda_h^T h(x) + \lambda_g^T (g(x) + s)$$

Where μ is introduced as a "barrier parameter". As the iterations continue, we slowly drive μ to 0; this allows for an asymptotic approach to the inequality constraints (if they will be near 0).

Compared to the active-set method, I have observed that the interior-point method will handle problems with a large amount of active inequality constraints better. I believe there are two reasons which make the inter-point method better for solving direct collocation optimal control problems.

First, with many active-set methods, at most one constraint is added and one constraint is removed from the active-set at each iteration. This actually places a lower bound on the minimum number of iterations to reach an optimal point: the number of constraints which are not part of the active-set at the first iteration but are part of the optimal active set. This would be assuming that each iteration in fact added one of these "optimal" active-set constraints, which is unlikely to happen in reality. This clearly makes selection of the initial active-set important.

Second, when using active-set solvers, I have observed switching behavior at active inequality constraints; this is theorized to correspond to inequality constraints dropping in and out of the active-set at each iteration - the code cannot take stable steps around the inequality boundary, so constraints are switched on and off at every iteration. The interior-point method seems to produce smoother solutions. Figure 3 displays previous results exhibiting the switching computed by active-set methods.

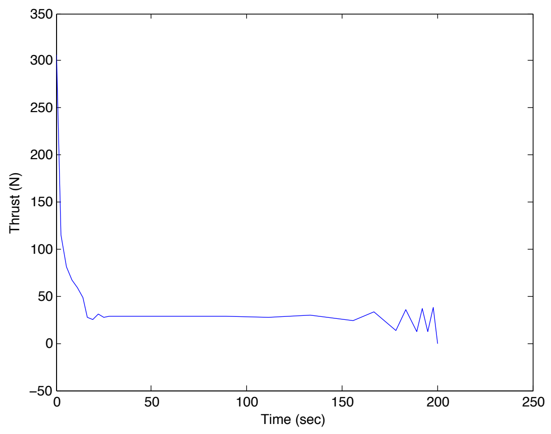

Figure 3: Previous results showing "switching" behavior

2.4.1 Current Results

At the moment, a NumPy/SciPy [18] implementation of the interior-point method as described in [5] is being written in Python. The goal of this is to provide an open-source NLP solver, based off of an algorithm which has been successfully demonstrated to work in an existing commercial code.

The implementation I have written currently allows for users to supply objective, equality and/or inequality constraints, Jacobian matrices, and a Hessian matrix. If these matrices are not supplied, they are approximated; the Jacobian matrices are approximated by finite differences (with selectable perturbation tolerance and finite difference order) and the Hessian matrix is approximated using the BFGS method.

In order to test this code, I have used two approaches. The first is a number of example problems with known solutions. One specific example I have used is the displacement constrained double integrator, first presented by Bryson and solved numerically with direct collocation by von Stryk [26]. The problem is a essentially a particle moving in one dimension, with force as an input, and a number of constraints. The particle starts with an initial position and velocity, and at the terminal time, must be at the same position, but moving in the opposite direction (with the same initial speed); furthermore, the particle is limited in the maximum displacement from it's initial position. The objective function is minimizing the time integral of the thrust. Figure 4 shows the solution to this problem, as calculated by the NLP code I have written. It provides a reasonably accurate solution to the problem (the objective value computed was 4.000105 vs. 4 for the analytic solution using 80 timesteps).

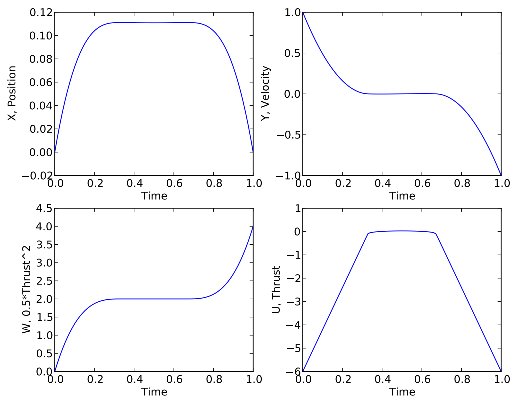

Figure 4: Results of applying my NLP code to a test problem.

The other approach I have used to validate and experiment with my NLP code is the CUTEr test suite of optimization problems [19]. The CUTEr test suite is comprised of approximately one thousand test problems for constrained optimization. The problems are defined in a Standard Input Format (SIF), which can then be given to a variety of interfaces which compile it into appropriate formats. An interface to Python already exists which I have been using to test my code [20].

2.4.2 Proposed Work

There are many different paths for development (with varying levels of depth) for a tool like this. My focus on optimal control problems will determine the additional work I plan on doing with this code. The different categories that I see existing are: ease-of-use and interfaces, numerical performance/accuracy, scalability, and algorithmic enhancements.

The accessibility of the code I write is an important consideration in its future development. While I would benefit from any experience I gain working on this NLP code, I would like to make an actual contribution to existing options for optimization. Part of this will involve flexibility in how users can supply their problem and how easy it is for them to install and use the code I write. I plan on adding the ability for a problem to be defined in more than just Python: most likely C, FORTRAN, and the SIF format will be able to be used to supply problems.

Numerical performance and accuracy issues relate to how long computations will take and what numerical precision can be maintained. Clearly, when doing millions of operations in each iteration, code which runs quickly is ideal. Implementation of the NLP code in another language than Python, perhaps a compiled language, could give better performance. This can mean sacrificing some of the readability and accessibility of the open-source code though. Other options also exist; using combinations of compiled languages and Python is possible, where more expensive operations are written in a lower-level language. Use of graphics processing units (GPU's) in computers is also a possibility for speeding up matrix operations. Current GPU's can have hundreds of processing cores. Oak Ridge National Laboratory is using GPU's as part of its most powerful supercomputer [21]. For either of these approaches though, there are downsides such as an increased need to manage computer memory and resources at lower levels to realize any performance gains. Just as critical as computational speed is the numerical precision which can be maintained. As discussed earlier, finite difference computations for mixed partial derivatives involve division by a ϵ^2 term. The result of such computations can easily suffer from reduced accuracy. I intend to examine both requirements and the ideal ways to address these needs.

Scalability is closely related to the numerical performance and accuracy issues. Currently, the code I have written implements sparse matrix operations, as many of the matrices operated on are mostly empty, especially for optimal control problems. Determining what the maximum problem size that I expect the code to be used for and at what size problems see massive slow downs (when the operations can't fit into the CPU cache or even the main memory) will help answer some of the questions about further performance enhancements.

Finally, I would like to study algorithmic enhancements I could make to the existing code and algorithm presented by Byrd et al. The main topic I plan on studying is improvements on solution convergence time, and specifically, barrier parameter update strategies. Again, for the interior-point method, there is a barrier parameter which limits how close inequality constraints can be to 0. This has already been studied by Liu [22] for the algorithm I am using; it has only been studied in a general sense though. I would like to study what strategies can be used to update the barrier parameter for optimal control problems, specifically those formulated as direct collocation problems, to improve solution convergence time. Currently, the barrier parameter update strategy used in the algorithm does not actually depend on the values of the inequality constraints, either in identifying when the barrier parameter updates or how it updates. I believe a more intelligent barrier parameter update strategy could consider more error information, and I intend on studying its performance when used with optimal control problems.

Of these four parts, the first three involve work on my part, but will draw from current knowledge in computing. The fourth topic, regarding barrier parameter update strategies, will hopefully produce new knowledge in the optimization field.

2.5 Human Energetics Model

As stated previously, the physical dynamics of the cyclist are only half of the system in the optimal control problem. The other half of the system is the energy production capabilities of the human cyclist. Just as the physical dynamics are modeled by a set of differential equations, the energy production dynamics must also be modeled by a set of differential equations. The physical dynamics ultimately came down to 1 input, rider leg torque, which gave the acceleration. For the energetic dynamics there is a similar goal: the equations must take in 1 input, which should then allow for calculation of the actual torque produced by the rider and model how fatigue changes.

I will describe the first model I used, the critical power model, as well as some of its shortcomings. I will then describe a model I am developing - its underlying assumptions, its origin, what it is currently capable of, and its shortcomings. I will then describe the next steps I plan to take in modelling human energy production abilities.

2.5.1 The Critical Power Model

The basis for the critical power model is the hyperbolic relationship between constant power output and the time until exhaustion. The parameters for the model are determined by observing the total energy output produced from multiple constant power tests to exhaustion, each for a time t. For each power level tested, the total energy output is computed and plotted against time. We define the slope of this plot as the critical power and the y-intercept as the anaerobic work capacity. See the equation below:

$$E = C_{aw} + C_p t \tag{15}$$

Dividing both sides of the equation by time, an expression for the constant power level that can be sustained over the given time is obtained:

$$P = \frac{C_{aw}}{t} + C_p \tag{16}$$

In order to apply this to non-steady state cases, a differential equations which models this behavior is needed. Morton [9] presented a hydraulic analogy, in which one state represents the amount of energy remaining. The differential equation representing the critical power model can be written as:

$$\frac{d}{dt}E = -P + C_p \tag{17}$$

where the power (P) is the input. The interpretation of this model is that there is an unlimited amount of energy generated at the limited rate denoted by the critical power, C_p, and a finite amount of energy, C_{aw} that can be used at any rate. One interpretation of the percentage of remaining anaerobic energy is as an indicator of fatigue.

It should be noted that this model represents whole-body energy production abilities, but only for a given task, i.e. the values from cycling ergometer and rowing ergometer tests will most likely not give the same critical power and anaerobic work capacity.

The critical power model, as presented above, has some limitations. It was not explicitly created for use in a variable power system and does not model recovery correctly. There is also no rate limitation on anaerobic energy expenditure. These limitations can be seen by considering a numerical example. Consider a hypothetical athlete with parameters $C_{aw} = 10000$ J and $C_p = 300$ W (somewhat realistic values). Using the critical power model, the athlete could produce 10,300 W for 1 second, then rest for approximately 34 seconds, and then be at full energy capacity again, able to produce 10,300 W for 1 second. This is clearly unrealistic. Alternatively, this athlete could start an event and produce 300 W for 10 days straight (actually, for an infinite amount of time).

Morton [9] does list some of the assumptions which are important to the critical power model, but because the equations are so simple, these have to be taken into consideration externally, i.e. when computing the optimal strategy there need to be numerical constraints bounding the power output. This is undesirable from a computational point of view, as it increases the complexity of the optimization problem.

Conversely, some experimental evidence supports the critical power part of the model (a power output level which can be sustained for a long period of time). Jones [12] has experimentally computed critical power values for subjects, and then measured biological markers at various time points; this was also compared to previous experimental data. Subjects produced power just above (+5%) and at the critical power level. Observed values of blood lactate rose initially, then were nearly stable for the critical power output; for the same power output, observed PCr levels dropped and were then nearly stable. For the power output just above the critical power level, blood lactate continued to rise and PCr levels continued to decrease until exhaustion. These markers are not necessarily the cause of fatigue, but their presence is associated with the onset of fatigue. Jones' experiment shows that the concept of critical power, but not necessarily the finite anaerobic work capacity concept, has some validity within the time frame we are considering (around 30 mins).

2.5.2 The Developed Model

The critical power model was the first model I used to represent an athlete's bioenergetic capabilities due to its simplicity and history. After realizing the shortcomings of the model, mainly its poor recovery modelling) I decided that I needed to use a "better" model. At this point in time, in order to define "better", I attempted to build a deeper understanding of the underlying exercise physiology. I will discuss some of the assumptions which form the basis of this new model, as well as its origin, and current status.

Brooks et al. [2] provide a great deal of information on exercise physiology; from that resource and discussions with one of the authors, assumptions have been collected for the human energetics system. It is important to note that these assumptions have been formed for events around 30 minutes in duration and relate to the phenomenon of peripheral fatigue. They are now presented as a list of assumptions applicable to both fiber types, the slow fiber type, and the fast fatiguable fiber type. Fast fatigue resistant fibers have a performance somewhere in between slow and fast fatiguable fibers; their inclusion will come later after conclusions about modelling the other fiber types are reached.

- Both Fiber Types:
  - Glycogen, through one pathway or another, will most likely provide all the energy for an event of this duration and intensity.
  - Energy stores of glycogen will not run out within 1 hour, or rather usage of energy stores will not be what impacts power production; this implies that glycogen depletion does not need to be considered as the cause of fatigue.
  - All fibers Use ATP/PCr to provide immediate energy, and neither will be (nor can be) completely depleted; but the rate at which both are used are and replenished is generally in balance.
  - The time dynamics of ATP/PCr depletion/replenishment are much faster than the length of the event and do not need to be considered.
  - For times less than 1 hr, fast and slow fibers are independent in their energy stores and fatigue/recovery mechanisms.
- Fast Fibers:
  - Glycolysis is what provides ATP turnover.
  - At maximal exercise, glycogen breakdown produces pyruvate (which is then turned into lactate), ATP, H+, and NADH.
  - As the fast fibers fatigue (after they have been recruited) their force output is lessened, until some sort of recovery.
- Slow Fibers:
  - Carbohydrates and lipids provide ATP turnover.
  - Maximal oxygen uptake corresponds to maximal carbohydrate oxidation, meaning there is no fat oxidation. Additional power will be generated by work done in other fibers.
  - Negligible fatigue (and reduction in power output) occurs in these fibers for an exercise duration of less than 1 hour.

From the above assumptions, it seems reasonable that a reduction in force production capability does not necessarily need to be estimated by tracking a physical quantity which causes fatigue directly, but rather that fatigue can be represented by some possibly non-measurable quantity, which is linked to fatigue. Examples of such quantities are blood lactate levels, calcium ion concentrations, PCr levels, etc. - these are all associated with the onset of fatigue, but either play no roll or are only part of causing fatigue. Essentially, the state variables representing fatigue might not necessarily have to represent a physical quantity which causes fatigue.

The change in this quantity can then be represented by a fatigue rate which is proportional to the current power output. Since I am assuming that fatigue is not caused by fuel depletion, I am going to claim that recovery from fatigue is also not related to fuel replenishment. Essentially, I am making the claim that peripheral fatigue is caused by muscle use. Furthermore, I am claiming that there is a link between muscle use, biochemical processes within the muscle, and a quantity which represents the current conditions within the muscle. Fatigue and recovery in some sense are simply the chemical reaction being driven to the right or left. I am also going to claim that peripheral fatigue and recovery, no matter what the biochemical processes which are the cause, can be represented by some differential equation.

The conclusions from these assumptions are present in the following model, where fatigue is represented independently for each fiber type, fatigue itself might not be a physical, measurable quantity, and there is the possibility for recovery. I will also show that in this model, fatigue increases at a rate proportional to the current force production.

Xia and Frey Law have presented a model which allows for significantly more detail than the critical power model. They proposed representing the force production capability of each muscle motor unit (MU) with 3 compartments, a fatigued compartment, a resting compartment, and an active compartment. The sum of these three compartments adds up to the total "size" of the motor unit in question. These values are denoted as M_f, M_r, M_a, and S. Note that for the compartments (M) the subscript refers to the compartment; if there are multiple fiber types being examined, they each have a unique superscript. For any given muscle group, one can write the following equation which relates the total "size" of the motor unit with all of its compartments:

$$S = M_f + M_r + M_a \tag{18}$$

To track the amount of muscle in each compartment, they suggested the following differential equations:

$$\frac{dM_r}{dt} = -C(t) + M_f R \tag{19a}$$

$$\frac{dM_a}{dt} = C(t) + M_a F \tag{19b}$$

$$\frac{dM_f}{dt} = M_a F - M_f R \tag{19c}$$

These equations show how the motor unit is dynamically apportioned between compartments. The resting muscle increases proportionally to the amount of fatigued muscle and decreases by the amount specified in the activation drive, $C(t)$. The amount of active muscle increases by the rate specified in the activation drive, and decreases proportionally to how much is currently active (the fatiguing mechanism). The amount of fatigued muscle grows at a rate proportional to the amount of active muscle and recovers at an amount proportional to the amount of currently fatigued muscle. Here $C(t)$ is the muscle activation-deactivation drive; it represents the rate at which muscles are being "turned on/off". It is also the input to the system. M_a is the output to the system; it is the amount of currently activated muscle, and it is proportional to the amount of force currently generated which is the output.

Note that in this model, the *rate* at which muscles activate and deactivate is considered; this will model the time delay between when force is requested and when force is produced. For each motor unit (which can be extended to represent muscle groups), the unit is split between activated, resting, and fatigued compartments; the dynamics of transport between the compartments are related to how much of the MU is currently active (the fatiguing mechanism), how much of the MU is currently fatigued (the recovery mechanism), and how much of the MU is continuing to activate/deactivate.

This model is then extended to multiple motor units of different fiber types; while the basic compartment equations are presented for a motor unit, when considering groups of different fiber types, it makes sense to instead view this model as representing a muscle group or joint. A mucle group will have 3 fiber types in each group: slow (s), fast fatigue-resistant (i), and fast fatiguable (f) fibers [1, 2]. In reality, there are not just 3 fiber types, although humans mainly have type I, IIa, and IIx, so I will treat muscles as being comprised of only 3 fiber types. After choosing to represent the muscle with 3 fiber types, 3 sets of states and differential equations can be written for each fiber type:

$$\frac{dM_r^k}{dt} = -C(t)^k + M_f^k R^k \tag{20a}$$

$$\frac{dM_a^k}{dt} = C(t)^k - M_a^k F^k \tag{20b}$$

$$\frac{dM_f^k}{dt} = M_a^k F^k - M_f^k R^k \tag{20c}$$

$$S^k = M_f^k + M_r^k + M_a^k \tag{20d}$$

$$\text{for } k = s, i, f$$

The total force output of the muscle group can be computed as the sum of M_a^s, M_a^i, and M_a^f. This then allows representation of the energy production abilities of the 3 fiber types independently, i.e. the fast fatiguable muscles can be modeled to be able to generate more force than the other types, but fatigue faster and recover more slowly than the other types. This is accomplished by specifying a different size, fatigue rate, and recovery rate for each fiber type.

As stated there is one output, force, which is the sum of the active muscle compartments for all fibers. Now, one input to the entire system is needed. Considering the size principle proposed by Henneman [14], we see that for a single input (the desired force output) we recruit only as many fibers as necessary, in the order of slowest to fastest (in reality, it is the ease of triggering an action potential in the soma, which generally has slower fibers being easier to trigger). For example, if our slow fibers can produce 100 N, and the desired force is 50 N, only some slow fibers will be used. If the desired force is 150 N, then all of the slow fibers will be used and some of the faster fibers will need to be recruited. From this size principle, logic arises which relates an input to the amount of force production potential in each fiber type, and then computes how much of each fiber type should be active. This will lead to the slow fibers always being used and the faster fibers used only when force demand is high enough.

In summary, there are 2 parts to Xia and Frey Law's model. First, there is the ability to represent the current fatigue and force production for a given muscle motor unit, and the differential equations needed to model the time dynamics of this motor unit. Second, there is the ability to model and track these quantities across multiple fiber types, and to relate a desired force production amount with an output force amount, while still modelling the time dynamics of each fiber group. I will now discuss the modifications to this model which I have made, to make it suitable for my intended application.

The Xia and Frey Law model has a term, $C(t)$ which represents the activation/deactivation drive in the muscle, which is the rate at which the fibers start to generate force. The dynamics of this happen on a very short time scale, less than a second. In order to represent these dynamics, the time discretization of the event would need to go down to the second or sub-second level. This would result in over 1800 time points for an event 30 minutes in duration. Furthermore, I do not believe that modelling these dynamics are important to accurately model the energetic system of the human over this time scale; i.e. whether the muscle activates in a quarter of a second or half a second will not impact the performance of the athlete, or the ability to model a muscle. What follows are proposed changes to the Xia and Frey Law model.

One change is removal of the activation/deactivation rate, as the sub-second time dynamics are not relevant at the larger time scale we are dealing with. In addition, examination of Xia's equations reveals that there is a redundant state; only 2 of the 3 states for each fiber group are independent. As the rate of muscle activation is not going to be considered, there is no longer a need to specify the rate at which muscle activates. Instead the muscle is partitioned into two compartments, fatigued and non-fatigued sections; these are denoted as M_f and M_n (subscripts again representing the compartment and superscripts representing the fiber type). As activation/deactivation rates are no longer part of the model, and there is no time considered in switching muscle from resting to active, the new input to the system will be the amount of active muscle. The description is now:

$$M_f + M_n = S \tag{21a}$$

$$M_n = M_a + M_r \tag{21b}$$

Where M_n and S are the amount of non-fatigued muscle and total amount of muscle, and again M_a and M_r are the active and resting compartments. The time rate of change for each compartment can be defined as:

$$\frac{dM_f}{dt} = M_a F - M_f R \tag{22}$$

Which is identical to the previous differential equation for the fatigue compartment except now M_a is an input, not a state. There is also a limitation where the amount of activated muscle must not exceed the amount of non-fatigued muscle and it must not be less than 0. This gives the following bounds on M_a:

$$0 \leq M_a \leq S - M_f \tag{23}$$

Just as in the original Xia model, this can be extended from a motor unit of homogeneous fiber types to a muscle group made of multiple motor units.; each fiber type will have its own state and values for S, F, and R. The fiber types are slow (s), fast fatigue resistant (i), and fast fatiguable (f). The compartments are now described by:

$$M_f^s + M_n^s = S^s \tag{24a}$$

$$M_f^i + M_n^i = S^i \tag{24b}$$

$$M_f^f + M_n^f = S^f \tag{24c}$$

and the differential equations are now:

$$\frac{dM_f^s}{dt} = M_a^s F^s - M_f^s R^s \tag{25a}$$

$$\frac{dM_f^i}{dt} = M_a^i F^i - M_f^i R^i \tag{25b}$$

$$\frac{dM_f^f}{dt} = M_a^f F^f - M_f^f R^f \tag{25c}$$

I will now define a single input to all the fiber types: $\tilde{F}_d$ which is defined as the percent of maximum available force output at that point in time. Next, this needs to be related to actual muscle force:

$$F_{max} = M_n^s + M_n^i + M_n^f \tag{26}$$

and

$$F_d = F_{max}\tilde{F}_d \tag{27}$$

where $\tilde{F}_d$ is between 0 and 1.

In order to transform our one external input into 3 inputs (one for each differential equation), Henneman's size principle will be used; this gives the following logic. If the slow fibers can completely meet the force production demand, only slow fibers are used. If the combination of slow fibers and the fast fatigue resistant fibers can completely meet the force production demand, we use all of the slow fibers, and some of the intermediate fibers. If the all 3 fiber types are needed to meet the force demand, then we use all of the slow and fast fatigue resistant fibers, and as much of the fast fatiguable fibers as necessary to meet our force demand. This logic is described by:

$$\begin{aligned}
\text{if: } F_d \le & M_n^s \\
& M_a^s = F_d \\
& M_a^i = 0 \\
& M_a^f = 0 \\
\text{else if: } M_n^s < F_d \le & M_n^s + M_n^i \\
& M_a^s = M_n^s \\
& M_a^i = F_d - M_a^s \\
& M_a^f = 0 \\
\text{else if: } F_d > & M_n^s + M_n^i \\
& M_a^s = M_n^s \\
& M_a^i = M_n^i \\
& M_a^f = F_d - M_a^s - M_a^i
\end{aligned}$$

At this point, there is now the ability to have 1 input into the 3 differential equations which model fatigue. Note that the input is a percentage of the maximal available force at this instant in time and is in the interval [0, 1]; this does not allow for a specified force to be requested. It would only be marginally more complicated to allow any force input; if the input force was above the maximum available force, the input force would be reduced to that maximum available force level. However, this formulation is not beneficial for optimal control problems. In those situations, the response of the output is not proportional to the input, because the input will be truncated at the maximum available power, leading to sharp discontinuities in the functions; this is difficult for optimization codes. Furthermore, the input to the optimization code does not necessarily have to a value which is directly correlated to a real-world quantity such as a power output level or speed for the athlete to match at any point during their event. A non-dimensional normalized number (here, desired percentage of maximum available force) is usually an ideal input to an optimization problem.

Now the description of the new model is complete. From here on it will be referred to as the proposed model. Again, I have taken the model presented by Xia & Frey Law which was intended to represent a single motor unit, and now reformulated it to be used in a more general form. It will now be applied to the multi-muscle group actions used in cycling. Consider a hypothetical athlete with the following model parameters. Time constants were set to be about 1 hour, 15 minutes, and 1.5 minutes for each fiber type. Next, the maximum force output was set to 250 N for the slow fibers, 100 N for the intermediate fibers, and 350 N for the fast fibers. Recovery rates were set to 1/10 of the fatigue rates. A simulated maximal force test is shown in the figure 5.

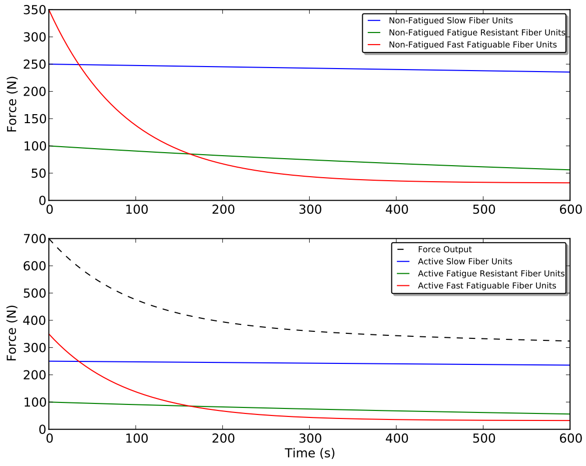

Figure 5: Response of the proposed model with hypothetical values under a maximal effort test.

Note that for this case, I have not specified a specific muscle or task; it could be considered similar to an ex vivo sample being tested in a lab. Also, note that in this formulation, the ratio of the recovery rate to the fatigue rate determines the force which can be produced when "exhaustion" has been reached:

$$M_{a,exh} = S\frac{R}{F+R} \tag{28}$$

I believe this will be useful in determining or validating actual model parameters for a real athlete.

2.5.3 Proposed Work

There are two directions that I propose to take my energetics model in. One is in validating the model and the other is in adding behavior to represent what actually happens in real life. Clearly work in these two areas will have to happen simultaneously, as in order to validate the model, the model has be capable of representing the behaviors which are displayed in testing.

I believe that right now, this model could be fit to data from a maximal effort test to obtain the coefficients needed. I believe such a model could replicate similar test results, under certain conditions: the athlete is not fatigued in either case, the athlete is warmed up with muscles at operating temperatures, and the cadence is identical, and there is not a recovery period. Recently, Sih et al. have used a compartment muscle model to perform similar simulations with good comparisons to experimental data [23].

Unfortunately, it would appear that cycling cadence can affect the rate at which fatigue occurs. Vanhatalo et al. showed that different cadences can result in slightly different peak powers and fatigue characteristics [24] - this is a behavior which I do not believe my model can represent; the cadence does not affect the ability to produce work currently.

It has also been shown by He et al. that ATP consumption and efficiency has a nonlinear relationship between muscle force and contraction speed [25]. This could potentially explain why there would be different fatigue rates at different cadences. The results He presented show that a slower muscle contraction velocity is generally more efficient up to a point. This would also correspond with the findings of Vanhatalo for cycling experiment.

I plan to consider additional experiments that have been performed which show ATP consumption at submaximal efforts (below the limiting force-velocity curve). I will also consider experiments, for both whole-body and ex vivo cases, which involve recovery periods or a duty-cycle in the muscle contractions. I believe examining experimental data for both of these cases will help me understand what additional behavior a model would need to encompass. From there, I will create a new mathematical framework and determine how to use experimental data to find the model parameters. If there is not enough existing data to validate the model against, but the model parameters could be measured with the appropriate whole-body exercise tasks, I will consider running my own experiments on human subjects to determine these parameters.

In summary, my previous approach has been to try and understand the underlying physiology of muscles during exercise, and then create a semi-empirical mathematical framework which is consistent with the physiology and "full-scale" human experiments. This will be my approach as I continue to attempt to accurately model human energy production abilities. I will continue to work to model fatigue accurately and include the effects of muscle contraction speed on fatigue - I will also validate the models I develop.

3 References

References

[1] Xia, T., Frey, L. A. (2008). A theoretical approach for modelling peripheral muscle fatigue and recovery. Journal of Biomechanics, 41, 3046-3052. doi:10.1016/j.jbiomech.2008.07.013

[2] Brooks, G. A., Fahey, T. D., Baldwin, K. M. (2005). Exercise physiology: Human Bioenergetics and Its Applications (Fourth Edi.). New York: McGraw-Hill.

[3] MATLAB. (2010). Natick, MA: The MathWorks, Inc.

[4] Kraft, D. (1994). Algorithm 733: TOMP-Fortran modules for optimal control calculations. ACM Transactions on Mathematical Software (TOMS), 20(3), 262-281. ACM. doi:10.1145/192115.192124

[5] Byrd, R. H., Hribar, M. E., Nocedal, J. (1999). An interior point algorithm for large-scale nonlinear programming. SIAM Journal on Optimization, 9(4), 877-900.

[6] Bryson, A. E. (1999). Dynamic Optimization. Addison Wesley Longman.

[7] Keller, J. B. (1974). Optimal Velocity in a Race. The American Mathematical Monthly, 81(5), 474-480.

[8] Ward-Smith, A. (1999). The kinetics of anaerobic metabolism following the initiation of high-intensity exercise. Mathematical biosciences, 159(1), 33-45.

[9] Morton, R. H. (2006). The critical power and related whole-body bioenergetic models. European journal of applied physiology, 96(4), 339-54. doi:10.1007/s00421-005-0088-2

[10] Morton, R. H. (2009). A new modelling approach demonstrating the inability to make up for lost time in endurance running events. IMA Journal of Management Mathematics, 20(2), 109-120. doi:10.1093/imaman/dpn022

[11] Hargraves, C. R., Paris, S. W. (1987). Direct trajectory optimization using nonlinear programming and collocation. Journal of Guidance, Control, and Dynamics, 10(4), 338-342. doi:10.2514/3.20223

[12] Jones, A. M., Vanhatalo, A., Burnley, M., Morton, R. H., & Poole, D. C. (2010). Critical Power: Implications for the Determination of V O2 max and Exercise Tolerance. Medicine and science in sports and exercise, 1876-1890. doi:10.1249/MSS.0b013e3181d9cf7f

[13] Wood, D. D., Fisher, D. L., & Andres, R. O. (1997). Minimizing Fatigue during Repetitive Jobs: Optimal Work-Rest Schedules. Human Factors: The Journal of the Human Factors and Ergonomics Society, 39(1), 83-101. doi:10.1518/001872097778940678

[14] Henneman, E, Somjen, G, & Do, C. (1965). Functional Significance of Cell Size in Spinal Motor Neurons. Electroencephalography and clinical neurophysiology, 19(5).

[15] Bryson, A. E., & Ho, Y.-C. (1975). Applied Optimal Control. Hemisphere Publishing Corp.

[16] von Stryk, O. (1999). User's Guide for DIRCOL. Darmstadt, Germany.

[17] Yokoyama, N., Suzuki, S., & Tsuchiya, T. (2008). Convergence Acceleration of Direct Trajectory Optimization Using Novel Hessian Calculation Methods. Journal of Optimization Theory and Applications, 136(3), 297-319. doi:10.1007/s10957-008-9351-0

[18] Jones, E., Oliphant, T., Peterson, P., & Others. (n.d.). SciPy: Open source scientific tools for Python.

[19] Gould, N. I. M., Orban, D., & Toint, P. L. (2005). General CUTEr documentation.

[20] Burmen, A. (2011). Python interface to CUTEr test problems for optimization.

[21] ORNL completes first phase of Titan supercomputer transition. (2012). `http://www.ornl.gov/info/press_releases/get_press_release.cfm?ReleaseNumber=mr20120229-00`

[22] Liu, G. (1999). PhD Thesis, Northwestern University. Design Issues in Algorithms for Large Scale Nonlinear Programming.

[23] Sih, B., Ng, L., & Stuhmiller, J. (2012). Generalization of a model based on biophysical concepts of muscle activation, fatigue and recovery that explains exercise performance. International journal of sports medicine, 33(4), 258-67. doi:10.1055/s-0031-1297958

[24] Vanhatalo, A., Doust, J. H., & Burnley, M. (2008). Robustness of a 3 min all-out cycling test to manipulations of power profile and cadence in humans. Experimental physiology, 93(3), 383-90. doi:10.1113/expphysiol.2007.039883

[25] He, Z., Bottinelli, R., Pellegrino, M., & Ferenczi, M. (2000). ATP consumption and efficiency of human single muscle fibers with different myosin isoform composition. Biophysical Journal, 79(2), 945-61. Elsevier. doi:10.1016/S0006-3495(00)76349-1

[26] Von Stryk, O. (1991). Numerical solution of optimal control problems by direct collocation. International Series of Numerical Mathematics (Vol. 111, pp. 1-13).