Department of Mathematics and Computer Science

Heidelberg University

Master thesis

in Scientific Computing

submitted by

Yu Xiang

born in November 10, 1989

2022

Gradient method of solving parameterized optimal control problems, with a case study in state constrained rocket car

This master thesis has been carried out by Yu Xiang

at the

Heidelberg University

under the supervision of

Professor Dr. Ekaterina A. Kostina

Contents

1 Introduction

Many real life problems, can be modeled as parameterized optimization problems, such as the therapy design of Cerebral Palsy (CP) problem described in Schlöder [2022]. In this paper, we focus on using gradient method to solve parameterized optimization problems, with a case study in state constrained rocket car.

Without giving a rigorous condition and definition[1], a general optimization problem is typically of the form

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ s.t.\quad & g(x) = 0, \\ & h(x) \leq 0 \end{aligned} \tag{1.1}$$

where $f(x)$ is the objective or cost function, $g(x) = 0$ and $h(x) \leq 0$ are the constraints. Some optimization problems may have uncertain parameters whose value are priori unknown, and the optimal objective value depends on the parameter value. This kind of problem is called the parameterized optimization problems and is of the form

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x,p) \\ s.t.\quad & g(x,p) = 0, \\ & h(x,p) \leq 0 \\ & x = x(p) \\ & x = x(p^0)\ if\ p = p^0 \\ & p \in \Omega_P \end{aligned} \tag{1.2}$$

where p^0 is a fixed value in the feasible uncertainty set Ω_P, where the parameter p can take value from.

Parameterized optimization problems are very difficult to solve due to the uncertainty in the parameter p. In the paper Schlöder [2022], multiple methods of solving the parameterized optimization problem have been discussed. The main focus (of solving the Cerebral Palsy problem) of the paper Schlöder [2022], is the "worst-case treatment planning by bilevel optimal control", i.e. a bilevel optimization problem. The bilevel optimisation method in paper Schlöder [2022] solves the parameterized optimization problems, e.g. the Cerebral Palsy problem, in a conservative way.

One method of solving the original CP problem in a conservative way is to transform the problem 1.2 into another form. Assuming that the parameter $\tilde{p}$ lies in an uncertainty set Ω_P, we can firstly reach one objective, i.e. identifying a worst possible solution with respect to $\tilde{p}$, i.e. solving a lower level problem. Based on the result of lower level, we can continue to find the best solution with respect to x, i.e. solving a upper level problem. The "worst-case treatment planning by bilevel optimal control", i.e. a bilevel optimization problem, is an optimization problem in which another optimization problem enters the

[1]We do not give a rigorous definition on purpose so that the problem we have described here can be applied to more general cases when such condition and definition are more clearly defined.

constraints. Mathematically, the problem 1.2 is transformed into another form, and can be formulated in a simplified notation, as following

$$\begin{aligned} &\underset{x}{min}\ \underset{p\in\Omega_P}{max}\ f(x,p) \\ &\qquad s.t.\ g(x,p)=0,\ h(x,p)\leq 0 \end{aligned} \tag{1.3}$$

Due to the *min max* notation, this classical approach of solving the bilevel problem can also be called *minmax* approach.

As stated in Schlöder [2022], many different methods can be used to solve a bilevel problem, three approaches have been discussed in detail, i.e. a transformation of the bilevel problem to a single level problem, a classical approach and a training approach. A intuitive approach is to transfer the bilevel problem into a single level problem, however, in general the resulting single level problem is not equivalent to the original bilevel problem and this approach is also out of the focus of the paper Schlöder [2022] as well as this paper at hand. A classical approach, aka a robust optimization appraoch, is consistent with the *minmax* appoach, which will be discussed in more detail in Chapter 2.

The paper Schlöder [2022] introduces the "Training Approach". It is based on the idea that in the real world, during the training period, an intervention is introduced and a certain, but a priori unknown, parameter $p \in \Omega_P$ is realized. What follows the training period (during which the parameter p is realized), the patient is able to react to it in an optimal manner, i.e. an optimal value $f(x,p)$ will be obtained given the realized parameter p. The paper Schlöder [2022] call this approach "worst case modeling Training Approach", and it can be written as

$$\begin{aligned} &\underset{p\in\Omega_P}{max}\ \underset{x}{min}\ f(x,p) \\ &\qquad s.t.\ g(x,p)=0,\ h(x,p)\leq 0 \end{aligned} \tag{1.4}$$

Due to the *max mix* notation, this approach of solving the bilevel problem can also be called *maxmin* approach.

The paper Schlöder [2022] use a derivative free method in the Training Approach. This paper at hand will focus on a gradient method to solve the *maxmin* problem. In particular, we are interested in how to compute the derivatives theoretically and numerically. We would like to apply the quasi-Newton and multiple shooting method when solving the problem numerically. The approaches discussed in this paper at hand will be demonstrated with a case study in state constrained rocket car.

We choose this rocket car case for two reasons: firstly, the case is relatively easy to understand and is quite representative of the general usage in real life; secondly, the case has theoretical solution and we can compare the numerical results with the theoretical value so that we can check whether our gradient method can find the optimal solution and how fast it converges.

The structure of this paper is as follows: in Chapter 1, i.e. this chapter, we give an introduction on what problems this paper intends to address. In the Chapter 2, we introduce the case of the state constrained rocket car. In the Chapter 3, we discuss the classical approach and training approach, and show the theoretical value of the chosen case. In Chapter 4, we give the mathematical background of the quasi-Newton and multiple shooting method. In Chapter 5, we show how we can solve the case numerically using the methods described in Chapter 4. In Chapter 6, we compare our theoretical and numerical results and conclude the paper.

2 Rocket car case

Since the approaches we are going to use in this paper will be demonstrated with the case of rocket car, we decide to describe the rocket car case first. So that, when we are discussing our approaches, we can directly describe how they can be used in solving the rocket car case. The description of the rocket car case is mostly coming from the paper Schlöder [2022], with content either verbatim or in a modified form.

We consider the rocket car case with state constraints, i.e. the one-dimensional movement of a mass point under the influence of some constant acceleration/deceleration, e.g. modeling head-wind or sliding friction, which can accelerate and decelerate in order to reach a desired position. The mass of the car is normalized to 1 unit[1] and the constant acceleration/deceleration enters the model in form of an unknown parameter $p \in \Omega_P \subset \mathbb{R}$ suffering from uncertainty, with the uncertainty set Ω_P convex and compact. We consider a problem in which the rocket car shall reach a final feasible position and velocity in a minimum time:

$$
\begin{aligned}
\min_{T,u(\cdot),x(:,p)} \quad & T && && (2.1a)\\
s.t. \quad & x = (x_1, x_2) && && (2.1b)\\
& \dot{x} = T \begin{pmatrix} x_2(t;p) \\ u(t) - p \end{pmatrix}, && t \in [0,1], && (2.1c)\\
& x(0,p) = 0, && && (2.1d)\\
& x_1(1;p) \geq 10, && && (2.1e)\\
& x_2(t;p) \leq 4, && t \in [0,1], && (2.1f)\\
& x_2(1;p) \leq 0, && && (2.1g)\\
& T \geq 0, && && (2.1h)\\
& u(t) \in [-10, 10], && t \in [0,1]. && (2.1i)
\end{aligned}
$$

where x represents the variables of the rocket car, and it has two components $x = (x_1, x_2)$. The first component x_1 is the (time-transformed) position of the rocket car. The second component x_2 is (time-transformed) velocity of the rocket car. The condition 2.1d, i.e. $x(0,p) = 0$, indicates that at $t = 0$, both the position and velocity of the car is 0. The condition 2.1e, i.e. $x_1(1;p) \geq 10$, indicates that the position of the car at $t = 1$ must be greater or equal to 10. The condition 2.1, i.e. $x_2(t;p) \leq 4$, indicates that the velocity of the car is always smaller or equal to 4 across the whole period. The condition 2.1, i.e. $x_2(1;p) \leq 0$, indicates that the velocity of the car at $t = 1$ is always smaller or equal to 0. Here, a negative velocity means that the car is moving in a direction that

[1]We do not specify the unit on purpose since the actual unit, either one kilogram or meter, does not play a role in the modeling. We are more concerned about the scale.

decreases the position. To make the rocket car case even simpler, we can limit the size of the uncertainty set, as following

$$p \in \Omega_P = [p_l, p_u] \subset [0, 9], \tag{2.2}$$

where $p_l < p_u$, with p_l and p_u the lower and upper boundary of the parameter p.

The decision variable in the problem 2.1 is the controllable parameter T, which encodes the process duration of the corresponding problem with free end time. The control function $u : [0, 1] \to \mathbb{R}$ represents the acceleration/deceleration value, and is dependent on the unknown parameter p, as shown in the condition 2.1c. The second component of the condition 2.1c, i.e. $\dot{x_2} = T(u(t) - p)$, indicates the change in the velocity of the car at time t is subject to the value of $T, u(t)$ and p. The first component of the condition 2.1c, i.e. $\dot{x_1} = Tx_2(t; p)$, indicates the position of the car at time t is subject to the value of T and the velocity $x_2(t; p)$ at time t. The variable $x(t : p)$ is a dependent variable, and is uniquely determined by $T, u(\cdot)$ and p. The goal is to minimize T such that the variable $x(t : p)$ satisfies all the conditions in 2.1.

2.1 Theoretical solution to rocket car case

As explained in Chapter 1, we choose the rocket car case for two main reasons, i.e. the easyness of understanding and the existence of theoretical solution, which will be shown in this section.

The optimization problem 2.1 has a unique global solution, and no further local solution exists. The optimal controllable parameter is given by

$$T^\star = T^\star(p) = 2.5 + \frac{40}{100 - p^2}, \tag{2.3}$$

and the optimal control function $u^\star(\cdot)(= u^\star(\cdot; p))$ by

$$u^\star(\cdot) = \begin{cases} 10, & for\ 0 \leq t < \frac{4}{(10-p)T^\star} \\ p & for\ \frac{4}{(10-p)T^\star} \leq t < 1 - \frac{4}{(10+p)T^\star} \\ -10 & for\ 1 - \frac{4}{(10+p)T^\star} \leq t \leq 1 \end{cases} \tag{2.4}$$

In words, we accelerate as strongly as possible (the acceleration value $u^\star(t) = 10$) until the velocity $x_2^\star(t; p) = 4$, and then keep the velocity $x_2^\star(t; p)$ constant for a certain periofd of time[2], and eventually decelerate as as strongly as possible until the velocity is $x_2(1; p) \leq 0$,. The moment $x_2(1; p)$ reaching the value of 0 is the moment that we finds the optimal/smallest T that all the conditions are satisfied.

The proof of the theoretical solution is given in Appendix B of Schlöder [2022]. Because of the simplicity nature of the rocket car case, we can find the theoretical solution of the nominal/original problem for our case 2.1. But for many real life problems, it is very difficult to find a direct solution to the original problem, and for some cases not feasible, due to the uncertainty in the parameter p. That is why in the paper Schlöder [2022], a classical (in the form *minmax*) approach and a training (in the form *maxmin*) approach have been discussed, and both approaches will lead to a conservative solution to the

[2]The acceleration value cancels out with a inherent deceleration value so that the velocity can stay constant. The inherent deceleration value can be result of a friction or head wind.

original problem. A conservative solution to the CP problem is a acceptable (or desired) result since less risk should be taken regarding the therapy design of CP problem. In the next chapter, we discuss, in details, the classical approaches and training approach for the rocket car problem. After that, we focus on the quasi-Newton and multi shooting approach to the same problem.

3 The Classical and Training Approach

The paper on hand focuses on using quasi-Newton and multi-shooting method for the Training Approach. In this chapter, we shortly introduce the Classical Approach first and then we discuss the Training Approach in greater detail. In the next chapter, we can introduce the quasi-Newton and multi-shooting method, and elaborate in detail how they can be used for the Training Approach.

3.1 The Classical Approach

As stated in the introduction part, the classical approach is consistent with the *minmax* approach, during which, two level optimization problems are solved.

In the lower level, we solve an optimization problem ($max\ f(x,p)$) with respect to p, and in the upper level, we continue to find the best solution with respect to x, as shown in 1.3. In the case of the rocket car, the classical approach will be expressed in the following form

$$\begin{aligned}
\underset{T,u(\cdot)}{min}\ \underset{p\in\Omega_P,x(\cdot,p)}{max}\quad & T && && (3.1a)\\
s.t.\quad & x=(x_1,x_2) && && (3.1b)\\
& \dot{x}=T\begin{pmatrix}x_2(t;p)\\ u(t)-p\end{pmatrix}, && t\in[0,1], && (3.1c)\\
& x(0,p)=0, && && (3.1d)\\
& x_1(1;p)\geq 10, && for\ all\ p\in\Omega_P, && (3.1e)\\
& x_2(t;p)\leq 4, && t\in[0,1],\ for\ all\ p\in\Omega_P, && (3.1f)\\
& x_2(1;p)\leq 0, && for\ all\ p\in\Omega_P, && (3.1g)\\
& T\geq 0, && && (3.1h)\\
& u(t)\in[-10,10], && t\in[0,1]. && (3.1i)
\end{aligned}$$

In the Classical Approach, the set of feasible controllable parameters and control functions are given by those T and $u(\cdot)$, which yield feasible trajectories $x(\cdot,p)$ for all $p\in\Omega_P$. The value of the objective function in the lower level does not depend on p and $x(\cdot,p)$. In other words, in this approach, the driver has no prior knowledge about the value of the parameter p and gets no feedback during the process and has to set up the driving strategy in advance.

3.2 The Training Approach

Contrast to the Classical Approach, in the Training Approach it is assumed that the driver of the rocket car is able to perform optmially for every p because of a preceding training period. Thus the worst possible optimal performance is given by a solution of the problem

$$
\begin{aligned}
\underset{p\in\Omega_P,T,u(\cdot),x(\cdot,p)}{max} \quad min \;\; & T && && (3.2a)\\
s.t. \;\; & x=(x_1,x_2) && && (3.2b)\\
& \dot{x}=T\begin{pmatrix}x_2(t;p)\\ u(t)-p\end{pmatrix}, && t\in[0,1], && (3.2c)\\
& x(0,p)=0, && && (3.2d)\\
& x_1(1;p)\geq 10, && && (3.2e)\\
& x_2(t;p)\leq 4, && t\in[0,1], && (3.2f)\\
& x_2(1;p)\leq 0, && && (3.2g)\\
& T\geq 0, && && (3.2h)\\
& u(t)\in[-10,10], && t\in[0,1]. && (3.2i)
\end{aligned}
$$

The solution of the Training Approach in paper Schlöder [2022] is given by a gradient-free method, more precisely, a so-called model-based Derivative-Free Optimization (DFO) approach for box-constrained optimization problems is used. The BOBYQA algorithm is chosen for such approach to solve problems of the form

$$
\begin{aligned}
&\underset{x\in\mathcal{R}^n}{min}\; F(x)\\
&\;\; s.t.\; a_i\leq x_i\leq b_i, i=1,...,n
\end{aligned}
\tag{3.3}
$$

The name BOBYQA is an acronym for "Bound Optimization BY Quadratic Approximation", and is used to solve lower level problem of 3.2. In the general DFO method, the objective function $F(\cdot)$ is considered a black box. For a given p, the parametric lower level OCP of the Training Approach 3.2 is solved with a direct DFO approach and the resulting (finite dimensional) solution is viewed as dependent variable. Furthermore, the uncentainty set Ω_P is box-shaped, and hence the BOBYQA algorithm is applicable to the problem in the Training Approach. The BOBYQA algorithm has been introduced in details in the paper Powell [2009], and we reiterate the main idea in the text that follows.

The method of BOBYQA is iterative, k and n being reserved for the iteration number and the number of variables, respectively. Further, m is reserved for the number of interpolation conditions that are imposed on a quadratic approximation $Q_k(x)\rightarrow F(x),\ x\in\mathcal{R}^n$, with m is a chosen constant integer from the interval $[n+2,\frac{1}{2}(n+1)(n+2)]$.

The approximation is available at the beginning of the k-th iteration, the interpolation equations have the form

$$
Q_k(y_j)=F(y_j),\ j=1,2,...,m. \tag{3.4}
$$

We let x_k be the point in the set $\{y_j : j=1,2,...,m\}$ that has the property

$$
F(x_k)=min\ \{F(y_j),\ j=1,2,...,m\}, \tag{3.5}
$$

with any ties being broken by giving priority to an earlier evaluation of the least function value $F(x_k)$. A positive number Δ_k, called the "trust region radius", is also available at the beginning of the k-th iteration. If a termination condition[1] is satisfied, then the iteration stops. Otherwise, a step d_k from x_k is constructed such that $\|d_k\| \leq \Delta_k$ holds, $x = x_k + d_k$ is within the bounds of equation 3.3, and $x_k + d_k$ is not one of the interpolation points $y_j : j = 1, 2, ..., m$. Then the new function value $F(x_k + d_k)$ is calculated, and one of the interpolation points, y_t say, is replaced by $x_k + d_k$, where y_t is different from x_k. It follows that x_{k+1} is defined by the formula

$$x_{k+1} = \begin{cases} x_k, & F(x_k + d_k) \geq F(x_k) \\ x_k + d_k, & F(x_k + d_k) < F(x_k) \end{cases} \tag{3.6}$$

Further, Δ_{k+1} and Q_{k+1} are generated for the next iteration, Q_{k+1} being subject to the constraints

$$Q_{k+1}(\hat{y}_j) = F(\hat{y}_j),\ j = 1, 2, ..., m, \tag{3.7}$$

at the new interpolation points

$$\hat{y}_j = \begin{cases} y_j, & j \neq t, \\ x_k + d_k, & j = t \end{cases},\ j = 1, 2, ..., m. \tag{3.8}$$

The operations of BOBYQA algorithm requires the user to provide an initial vector of variables $x_0 \in \mathcal{R}^n$, the initial trust region Δ_1, and the number m of interpolation conditions where $n + 2 \leq m \leq \frac{1}{2}(n + 1)(n + 2)$. Two different ways have been proposed for constructing the step d_k from x_k and updating procedures from the k-th iteration to the $k + 1$-th iteration in the paper Powell [2009], with both methods having utilized the "quadratic" nature of the approximation function $Q(\cdot)$.

The lower level OCP of the Training Approach 3.2 can be solved with the BOBYQA algorithm for a given p, since the lower level problem can be re-written into the form of 3.3 and the constraints are box-shaped. With the BOBYQA algorithm computing local extrema, the upper level problem still needs to be solved globally. In our rocket car case, this is straight-forward, i.e. maximizing over all p.

Nevertheless, the BOBYQA algorithm has limitations with several strong assumptions being made. Firstly, it has been assumed the uncertainty set is of moderate size and is box-shaped. Secondly, it has been assumed that there is only one local extrema, i.e. the lower level problem has only one solution for each $p \in \Omega_P$. In general, we cannot expect the second assumption to be valid. The BOBYQA algorithm is a gradient-free method with repect to the objective function $F(\cdot)$, it still utilizes the gradient of the approximation function $Q(\cdot)$ while updating the iteration. Therefore, this BOBYQA algorithm, or a general DFO approach, is still subject to the numerical errors and computational costs while calculating the gradients of the approximation function $Q(\cdot)$ and updating them in each iteration.

The paper at hand, instead, is utilizing the gradient of the objective function $F(\cdot)$ directly, with some approximation applied as well. We have used the multiple shooting and quasi-Newton method for solving the lower level problem of the Training Approach

[1]Typically, a termination condition is satified when the objective value can not be improved further after some iterations. for the termination condition of BOBYQA algorithm, please refer the paper Powell [2009] for more details.

3.2. In the chapter that follows, we introduce the quasi-Newton and multiple shooting method.

4 quasi-Newton and Multiple Shooting Method

In this chapter, we first introduce the classical Newton method used for solving optimization problems and then focus on the quasi-Newton and multiple shooting method.

As stated in the introduction chapter 1, a general optimization problem is typically of the form

$$\begin{aligned} &min\ f(x) \\ s.t.\ &x \in \Omega \end{aligned} \tag{4.1}$$

Here $x \in \Omega$ represents the constraints for which x must satisfy, it may be in the form of $\{g(x) = 0, h(x) \leq 0\}$ as in the problem 1.1, i.e. the feasible set $\Omega = \underset{x}{arg}\ \{g(x) = 0, h(x) \leq 0\}$. Various methods exist for handling the constraints, such as Karush–Kuhn–Tucker (KKT) method discussed in Section 5.2. After transforming the constrained problem into an unconstrained form, we can, therefore, directly apply algorithms that are suitable for unconstrained minimization. For the sake of simplicity, we focus on explaining the Newton and quasi-Newton method without constraints in this chapter. How the quasi-Newton can be expanded for problems with constraints, will be explained in the more details in Section 5.2.

4.1 Newton method

The problem 4.1 without constraints, i.e. $min\ f(x)$ can be solved via Newton's method, which attempts to solve this problem by constructing a sequence $\{x_k\}$ from an initial guess (starting point) x_0 that converges towards a minimizer $x^\star$ of $f(x)$ by using a sequence of second-order Taylor approximations of $f(x)$ around the iterates. The second-order Taylor expansion of $f(x)$ around x_k is

$$f(x_k + \delta_x) \approx h(x_k) := f(x_k) + f'(x_k)\delta(x_k) + \frac{1}{2}f''(x_k)\delta(x_k)^2$$

where δ represents a small change (with respect to x), and f', f'' are the first and second order derivatives of the original function $f(x)$. The notation f', f'' are usually expressed as ∇f and H (the Hessian matrix) respectively when x is a vector of variables. In the text that follows, we will use the symbol ∇f and H directly. Therefore, the Talyor expansion can be written as

$$f(x_k + \delta_x) \approx h(x_k) := f(x_k) + \nabla f(x_k)^T\delta(x_k) + \frac{1}{2}H(x_k)\delta(x_k)^2$$

The next iterate x_{k+1} is defined so as to minimize this quadratic approximation $h(\cdot)$. The function $h(\cdot)$ is a quadratic function of $\delta(x)$, and is minimized by solving $\nabla h(\cdot) = 0$. The gradient of $h(\cdot)$ with respect to $\delta(x_k)$ at point x_k is

$$\nabla h(x_k) = \nabla f(x_k) + H(x_k)\delta(x_k)$$

We are motivated to solve $\nabla h(x_k) = 0$, which turns out to solve a linear system

$$\nabla f(x_k) + H(x_k)\delta(x_k) = 0 \tag{4.2}$$

Therefore, for the next iteration point x_{k+1}, we can just add the small change $\delta(x_k)$ to the current iterate, i.e.

$$x_{k+1} = x_k + \delta(x_k) = x_k - H^{-1}(x_k)\nabla f(x_k),$$

here $H^{-1}(\cdot)$ represents the inverse of the Hessian matrix $H(\cdot)$. The Newton method performs the iteration until the convergence, i.e. x_k and $f(x_k)$ converge to $x^\star$ and $f(x^\star)$, respectively [1]. The details of the Newton method is as follows:

Newton method steps

- Step 0, $k = 0$, choose an initial value x_0
- Step 1, $\delta(x_k) = -H^{-1}(x_k)\nabla f(x_k)$, if $\delta(x_k) = 0$, then stop
- Step 2, choose a step-size α_k (typically $\alpha_k = 1$)
- Step 3, set $x_{k+1} = x_k + \alpha_k\delta(x_k)$, let $k = k + 1$. Go to Step 1

The parameter α_k is introduced to augment the Newton method such that a line-search of $f(x_k + \alpha_k\delta(x_k))$ is applied to find an optimal value of the step size parameter α_k.

Though the Newton method is straightforward and easy to understand, it has two main limitations. Firstly, it is sensitive to initial conditions. This is especially apparent if the objective function is non-convex. Depending on the choice of the starting point x_0, the Newton method may converge to a global minimum, a saddle point, a local minimum or may not converge at all. In another word, due to the sensitivity with respect to the initialization, the Newton method may be not able to find the global solution. Secondly, the Newton method can be computationally expensive, with the second-order derivatives, aka, the Hessian matrix $H(\cdot)$ and its inverse very expensive to compute. It may also happen that the Hessian matrix is not positive definite, therefore, Newton method can not be used at all for solving the optimization problem. Due to these limitations of the Newton method, instead, we have chosen the quasi-Netwon method for solving our rocket car problem.

4.2 quasi-Newton method

We have stated that one limitation or the downside of the Newton method, is that Newton method can be computationally expensive when calculating the Hessian (i.e. second-order derivatives) matrix and its inverse, especially when the dimensions get large. The quasi-Newton methods are a class of optimization methods that attempt to address this issue. More specifically, any modification of the Newton methods employing an approximation matrix B to the original Hessian matrix H, can be classified into a quasi-Newton method.

[1] In another word, the Newton mehtod has converged when the small change $\delta(x_k) = 0$ or $\delta(x_k)$ is small enough that the change in the objective function is below a pre-defined tolerance level.

The first quasi-Newton algorithm, i.e. the Davidon–Fletcher–Powell (DFP) method, was proposed by William C. Davidon in 1959 Davidon [1959], which was later popularized by Fletcher and Powell in 1963 Fletcher R. [1963]. Some of the most common used quasi-Newton algorithms currently are the symmetric rank-one (SR1) method A. R. Conn [1991] and the Broyden–Fletcher–Goldfarb–Shanno(BFGS) method. The family of the quasi-Newton algorithms are similar in nature, with most of the difference arising in the part how the approximation Hessian matrix is decided and the updating distance $\delta(x_k)$ is calculated. One of the main advantages of the quasi-Newton methods over Newton method is that the approximation Hessian matrix B can be chosen in a way that no matrix needs to be directly inverted. The Hessian approximation B is chosen to satisfy the equation 4.2, with the approximation matrix B replacing the original Hessian matrix H, i.e.

$$\nabla f(x_k) + B_k \delta(x_k) = 0 \tag{4.3}$$

In the text that follows, we explain how the iteration is performed in the BFGS method, as an example illustrating the quasi-Newton method. In the BFGS method, instead of computing B_k afresh at every iteration, it has been proposed to update it in a simple manner to account for the curvature measured during the most recent step. To determine an update scheme for B, we will need to impose additional constraints. One such constraint is the symmetry and positive-definiteness of B, which is to be preserved in each update for $k = 1, 2, 3, ...$ Another desirable property is that B_{k+1} is sufficiently close to B_k at each update $k + 1$, and such closeness can be measured by the matrix norm, i.e. the quantity $\|B_{k+1} - B_k\|$. We can, therefore, formulate our problem during the $k + 1$ update as

$$\begin{aligned} &\underset{B_{k+1}}{min} \ \|B_{k+1} - B_k\| \\ &s.t. \ \ B_{k+1} = B_{k+1}^T, \ \ B_{k+1}\delta(x_k) = y_k \end{aligned} \tag{4.4}$$

where $\delta(x_k) = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. In the BFGS method, the norm is chosen to be the Frobenius norm:

$$\|B\|_F = \sqrt{\sum_i^m \sum_j^n |b_{ij}|^2}$$

Solving the problem 4.4 directly is not trivial, but we can prove that problem ends up being equivalent to updating our approximate Hessian B at each iteration by adding two symmetric, rank-one matrices U and V:

$$B_{k+1} = B_k + U_k + V_k$$

where the update matrices can then be chosen of the form $U = auu^T$ and $V = bvv^T$, where u and v are linearly independent non-zero vectors, and a and b are constants. The outer product of any two non-zero vectors is always rank one, i.e. U_k and V_k are rank-one. Since u and v are linearly independent, the sum of U_k and V_k is rank-two, and an update of this form is known as a rank-two update. The rank-two condition guarantees the "closeness" of B_k and B_{k+1} at each iteration.

Besides, the condition $B_{k+1}\delta(x_k) = y_k$ has to be imposed.

$$B_{k+1}\delta(x_k) = B_k\delta(x_k) + auu^T\delta(x_k) + bvv^T\delta(x_k) = y_k$$

Then, a natural choice of u and v would be $u = y_k$ and $v = B_k\delta(x_k)$, we then have

$$\begin{aligned} B_k\delta(x_k) + ay_ky_k^T\delta(x_k) + bB_k\delta(x_k)\delta(x_k)^TB_k^T\delta(x_k) &= y_k \\ y_k(1 - ay_k^T\delta(x_k)) &= B_k\delta(x_k)(1 + b\delta(x_k)^TB_k^T\delta(x_k)) \\ \Rightarrow a = \frac{1}{y_k^T\delta(x_k)},\ b &= -\frac{1}{\delta(x_k)^TB_k\delta(x_k)} \end{aligned}$$

Finally, we get the update formula as follows:

$$B_{k+1} = B_k + \frac{y_ky_k^T}{y_k^T\delta(x_k)} - \frac{B_k\delta(x_k)\delta(x_k)^TB_k}{\delta(x_k)^TB_k\delta(x_k)}$$

Since B is positive definite for all $k = 1, 2, 3, ...$, we can actually minimize the change in the inverse B^{-1} at each iteration, subject to the (inverted) quasi-Newton condition and the requirement that it is symmetric. Applying the Woodbury formula, we can show (see the Appendix for more details) that the updating formula of inverse B^{-1} is as follows

$$B_{k+1}^{-1} = (I - \frac{\delta(x_k)y_k^T}{y_k^T\delta(x_k)})B_k^{-1}(I - \frac{y_k\delta(x_k)^T}{y_k^T\delta(x_k)}) + \frac{\delta(x_k)\delta(x_k)^T}{y_k^T\delta(x_k)} \tag{4.5}$$

As shown in the formula 4.5, at each iteration, we update B^{-1} by using $\delta(x_k) = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. Since an update of B^{-1} depends on the previous value, we need to initialize B^{-1}, either as an identity matrix or as the true Hessian matrix $H(x_0)$, calculated based on the starting point x_0.

The problem 3.2 involves constraints in the partial differential equation form and in a fixed time horizon (after the time transformation from $[0, T]$ to $t \in [0, 1]$). Therefore, we can apply multiple shooting method discretize the original OP problem into mutiple OP problems in different subintervals, with the constraints enforced at the boundary of subintervals to guarantee the continuity. Together with the matching conditions, we can then aggregate the subproblems and apply quasi-Newton method to get the final optimal solution. In the next section, we explain the multilpe shooting method first and in the next chapter we then focus on how the multilpe shooting and the quasi-Newton method, together, can be used for solving the rocket car problem.

4.3 Multiple Shooting Method

Multiple shooting method was initially introduced to solve boundary value problem (BVP) in differential equation scope David Morrison [1962]. This method, therefore, is well suited for solving optimal control problem with constraints in differential equations. However, some modification is necessary so that the multiple shooting method can be applied to solve optimal control problem. In the text that follows, we first explain how the multiple shooting can be used for solving BVP in the differential equation scope. After that, we explain how mutilple shooting can be applied to a general optimal control problem, and particularly how it can be used for solving the rocket car problem 3.2.

To illustrate the concept of shooting method to solve boundary value problem (BVP), we use the the following example.

$$\dot{x} = x(t), t_0 \leq t \leq t_f$$

The analytical solution of above equation is

$$x(t) = x(t_0)e^{t-t_0}$$

where e is the exponential number. Then $x(t_0) = x_0$ will be determined such that it will satisfy $x(t_f) = b$ for a given value b. Therefore, the equation $x(t_f) - b = 0$ or $x_0 e^{t-t_0} - b = 0$ is obtained. This derivation is called as shooting method. Generally, the shooting method can be summarized as follow

shooting method

- Step 1, choose an initial value $x_0 = x(t_0)$
- Step 2, form a solution of the differential equation from t_0 to t_f
- Step 3, evaluate the error at the boundary, if $x(t_f)-b = 0$, then stop, otherwise continue to Step 4
- Step 4, update the guess for x_0 based on some updating schema, go to Step 2

In multiple shooting method, the "shoot" interval is partitioned into some short intervals. We define a general differential equation with boundary value of the following form

$$\begin{aligned} \dot{y} &= f(t, y, p) \\ y(t_f) &= y_f \end{aligned} \tag{4.6}$$

where y denotes the differential variables, p is some parameter, $t \in [t_0, t_f]$, y_f is the boundary value at t_f. With multiple shooting method, one chooses a suitable grid of multiple shooting nodes $\tau_j \in [t_0, t_f]$, where $t_0 = \tau_0 < \tau_1 < ... < \tau_m = t_f$, i.e. m subintervals covering the whole interval. At the beginning of each subinterval, $[\tau_k, \tau_{k+1}], k = 0, 1, ..., m-1$, we have the initial guess of the starting value $\hat{y}_k$. Then in each subinterval, we have the initial value problem of the following form:

$$\begin{aligned} &\dot{y} = f(t, y, p), t \in [\tau_k, \tau_{k+1}],\ k = 0, 1, ..., m-1 \\ &y(\tau_k) = \hat{y}_k, k = 0, 1, ..., m-1 \end{aligned} \tag{4.7}$$

In each subinterval, we introduce the new unkown parameter $\hat{y}_k$, we solve an initial value problem and will get a solution $y(t), t \in [\tau_k, \tau_{k+1}]$. The piecewise solution is not necessary continuous and also not necessarily satisfy the boundary condition $y(t_f) = y_f$. The continuity has to be enforced by additional matching conditions at each subinterval boundary, i.e.

$$\begin{aligned} &y(\tau_{k+1}; \hat{y}_k) = \hat{y}_{k+1},\ k = 0, 1, ..., m-1 \\ &\hat{y}_m = \hat{y}_{\tau_m} = \hat{y}_{t_f} = y_f \end{aligned} \tag{4.8}$$

The procedure of multiple shooting method can then be summarized as

Mutilple shooting method

- Step 1, choose multiple shooting nodes $t_0 = \tau_0 < \tau_1 < ... < \tau_m = t_f$
- Step 2, choose initial guess for $\hat{y}_k, k = 0, 1, ..., m-1$
- Step 3, form solutions of the differential equation in each subinterval $[\tau_k, \tau_{k+1}], k = 0, 1, ..., m-1$

- Step 4, evaluate the error at the boundary of each subinterval. If $y(\tau_{k+1}; \hat{y}_k) - \hat{y}_{k+1} = 0,\ k = 0, 1, ..., m-1$ and $\hat{y}_m - y_f = 0$, then stop, otherwise continue to Step 5

- Step 5, update the guess for $\hat{y}_k, k = 0, 1, ..., m-1$ based on some updating schema, go to Step 3

In practice, there are many details to be decided when using (multiple) shooting methods, which we will discuss briefly without giving a complete description. In Step 1 & 2, when choosing the shooting nodes and the initial guess $\hat{y}_k$, how they are chosen usually depends on nature of the problem as well as a balance between accuracy and computational cost. For example, the nodes can be equally spaced and the $\hat{y}_k$ can be initialized with the same value, or they can be addressed in different methods. In Step 3, polynomial functions can be used as the approximate solutions, leveraging the fact that Taylor expansion can be used to approximate any continuous functions. In Step 4, "evaluate the error" is usually in the form of evaluating an objective function, which, e.g. can be defined as the sum of quadratic errors. In Step 5, the "updating schema" can be defined to so that the $\hat{y}_k$ can move in a direction that decreases the objective function. The (quasi-) Newton method, for example, can be used as an updating method in Step 5.

The multiple shooting method can be used for many problems. For example, it can be applied to the twice differential system of the following form

$$y''(t) = f(t, y(t), y'(t)) \quad y(t_0) = y_0, \quad y(t_f) = y_f, \quad t \in [t_0, t_f] \tag{4.9}$$

The problem 4.9 is similar to the problem 4.6. In each subinterval, a boundary value problem (BVP) is to be solved and matching conditions at the boundary of each subinterval are to be enforced so that the final solution is continuous and applicable to the whole interval.

With the same idea, we can use mutiple shooting to solve optimal control problems. For the rocket car problem 3.2, we can discretize the time $t \in [0, 1]$ and the original problem is split into multiple subintervals, with the constraints discretized and applied to each subinterval. We also need to introduce the initial guess for each subinterval and add the matching condition at each boundary. In the end, we turn the original problem 3.2 into piecewise OCPs with augmented parameters and constraints. But the piecewise OCPs can be aggregated together due to their non-overlapping properties and the same structure, i.e. they can be aggregated to one objective function with the constraints expressed in matrix form. The transformed problem can then be solved with quasi-Newton method. In the next chatper, we discuss in detail how the problem 3.2 can be solved with mutiple shooting and quasi-Newton method in details.

5 Solving Rocket Car Problem

Before we dive into the mathematics, we first re-introduce the physics of the rocket car case so that we can describe and model the problem more precisely.

5.1 Re-introduce the rocket car problem

Within the training approach, with a given p, we want to solve the lower level problem of the following form 5.1

$$
\begin{aligned}
& min\ T && && (5.1a)\\
& s.t.\ \ x = (x_1, x_2) && && (5.1b)\\
& \dot{x} = T \begin{pmatrix} x_2(t;p) \\ u(t) - p \end{pmatrix}, && t \in [0,1], && (5.1c)\\
& x(0;p) = 0, && && (5.1d)\\
& x_1(1;p) \geq 10, && && (5.1e)\\
& x_2(t;p) \leq 4, && t \in [0,1], && (5.1f)\\
& x_2(1;p) \leq 0, && && (5.1g)\\
& T \geq 0, && && (5.1h)\\
& u(t) \in [-10, 10], && t \in [0,1], && (5.1i)\\
& p,\ a\ given\ value && && (5.1j)
\end{aligned}
$$

In summary, we want to find the minimum time that the rocket car moves from the starting state (the position is at the origin point, and the speed is zero) to an ending state(the position is at least at point 10 or beyond, and the speed is less than or equal to zero). A negative speed indicates the car is moving in a direction that decreases the position. During the whole process, constraints on the acceleration/deceleration value and the speed are to be satisfied.

Because our objective is to minimize the time between starting state and ending state, i.e. the variable T, which is unknown, we cannot define a time horizon over which we will discretize and optimize. Therefore, a new variable t is defined as follows:

$$t = \frac{\tau}{T} \in [0,1] \quad \tau \in [0,T] \tag{5.2}$$

Where τ is the real time between starting time 0 and ending time T, and t is the relative time between 0 and 1. The equation 5.1c can be also written as

$$\dot{x} = \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = T \begin{pmatrix} x_2(t;p) \\ u(t) - p \end{pmatrix} = \begin{pmatrix} Tx_2(t;p) \\ T(u(t)-p) \end{pmatrix} \tag{5.3a}$$

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{pmatrix} = \begin{pmatrix} \frac{\partial x_1}{\partial \tau}\frac{\partial \tau}{\partial t} \\ \frac{\partial x_2}{\partial \tau}\frac{\partial \tau}{\partial t} \end{pmatrix} = \begin{pmatrix} \frac{\partial x_1}{\partial \tau}T \\ \frac{\partial x_2}{\partial \tau}T \end{pmatrix} = \begin{pmatrix} Tx_2(t;p) \\ T(u(t)-p) \end{pmatrix} \tag{5.3b}$$

$$\begin{pmatrix} \frac{\partial x_1}{\partial \tau} \\ \frac{\partial x_2}{\partial \tau} \end{pmatrix} = \begin{pmatrix} x_2(t;p) \\ u(t) - p \end{pmatrix} \tag{5.3c}$$

In summary, the equation $\frac{\partial x_1}{\partial \tau} = x_2(t;p)$ means the change in the position in real time is proportional to the speed at that moment. The equation $\frac{\partial x_2}{\partial \tau} = u(t) - p$ means the change in speed is proportional to the acceleration/deceleration value at that moment.

To use multiple shooting and quasi-Newton method, we discretize the interval $t \in [0,1]$ into subinterval $[t_k, t_{k+1}], k = 0, 1, 2, ..., m-1$, where $0 = t_0, t_1, t_2, ..., t_k, t_k, ..., t_m =$

1. We solve the OCP within each subinterval, and enforce the matching condition at the boundary of each subinterval. The equation 5.3a is equivalent to 5.3c, and $\partial x_1 = x_2(t;p)\partial\tau,\ \partial x_2 = (u(t)-p)\partial\tau$. We can, therefore, solve the partial equations within each subinterval directly. We have the following solution

$$x_1(t_{k+1}) - x_1(t_k) = \int_{t_k}^{t_{k+1}} x_2(t;p)d\tau \tag{5.4a}$$

$$x_2(t_{k+1}) - x_2(t_k) = \int_{t_k}^{t_{k+1}} (u(t)-p)d\tau \tag{5.4b}$$

$$t \in [t_k, t_{k+1}],\ k = 0,1,2,...,m-1,\ t_0 = 0, t_m = 1 \tag{5.4c}$$

The integral can be approximated by numerical method, one simple approximation method to the integral in equation 5.4a can be

$$\int_{t_k}^{t_{k+1}} x_2(t;p)d\tau \approx \frac{x_2(t_{k+1}) + x_2(t_k)}{2}(\tau_{k+1} - \tau_k) \tag{5.5}$$

There are many methods that can be used to approximate the integrals, one of the widely used method is the Runge–Kutta method. With initial value $x_1(t_k), x_2(t_k), u(t_k)$ given at each subinterval $[t_k, t_{k+1}]$, based on the differential equation 5.1c, we can find the (approximation) solution within this subinterval. However, in order to get the feasible optimal solution for the original problem (i.e. the whole interval), we need to enforce the matching condition at the boundary of each subinterval as well as the constraints defined in equations 5.1d-5.1i. These constraints can be addressed with the KKT method discussed in the next Section 5.2.

5.2 KKT Conditions

In order to get the feasible solution, we can take the constraints into consideration by Lagrange multipliers with the Karush–Kuhn–Tucker (KKT) condition incorporated. Before we explain the KKT condition, we need to introduce several definitions and theorems first. We consider a general optimal control problem of the following form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \le 0 \end{aligned} \tag{5.6}$$

Definition 1 *(Feasible set) The feasible set Ω is the set*

$$\Omega := \{x \in \mathbb{R}^n \mid g(x) = 0,\ h(x) \le 0\} \tag{5.7}$$

Definition 2 *(Active Constraints and Active Set) An inequality constraint $h_i(x) \le 0$ is called active at $x^\star \in \Omega$ iff $h_i(x^\star) = 0$ and otherwise inactive. The index set $\mathcal{A}(x^\star) \subset \{1,...,n_h\}$ of active inequality constraint indices is called the "active set".*

Often, the name active set also includes all equality constraint indices, as equalities could be considered to be always active.

Definition 3 *(LICQ) The linear independence constraint qualification (LICQ) holds at $x^\star \in \Omega$ iff all vectors $\nabla g_i(x^\star)$ for $i \in \{1, ..., n_g\}$ and $\nabla h_i(x^\star)$ for $i \in \mathcal{A}(x^\star)$ are linearly independent.*

To give further meaning to the LICQ condition, let us combine all active inequalities with all equalities in a map $\tilde{g}$ defined by stacking all functions on top of each other in a colum vector as follows:

$$\tilde{g}(x) = \begin{pmatrix} g(x) \\ h_i(x), \ i \in \mathcal{A}(x^\star) \end{pmatrix} \tag{5.8}$$

With the definitions above, we are ready to formulate the famous KKT condition.

Theorem 1 *(KKT Conditions) If $x^\star$ is a local minimizer of the problem 5.1 and the LICQ holds at $x^\star$, then there exist so called multiplier vectors $\lambda \in \mathbb{R}^n_g$ and $\mu \in \mathbb{R}^n_h$ with*

$$\nabla f(x^\star) + \nabla g(x^\star)\lambda^\star + \nabla h(x^\star)\mu^\star = 0 \tag{5.9a}$$
$$g(x^\star) = 0 \tag{5.9b}$$
$$h(x^\star) \leq 0 \tag{5.9c}$$
$$\mu^\star \geq 0 \tag{5.9d}$$
$$\mu_i^\star h_i(x^\star) = 0, \ i = 1, ..., n_h \tag{5.9e}$$

The "KKT Conditons" are also known as "First-Order Necessary Conditons".

Definition 4 *(KKT Point) We call a triple $(x^\star, \lambda^\star; \mu^\star)$ a "KKT Point" if it satisfies LICQ (Definition 3) and the KKT conditions (Theorem 1).*

Definition 5 *(Lagrangian Function) We define the so called "Lagrangian function" to be*

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x) \tag{5.10}$$

Here, we have used the so called "Lagrange multipliers" $\lambda \in \mathbb{R}^n_g$ and $\mu \in \mathbb{R}^n_h$. The last three KKT conditions 5.9c - 5.9e are called the complementarity conditions. For each index i, they define an $L-$shaped set in the (h_i, μ_i) space. This set is not a smooth manifold but has a non-differentiability at the origin, i.e., if $h_i(x^\star) = 0$ and also $\mu_i^\star = 0$. This case is called a weakly active constraint. Often we want to exclude this case. On the other hand, an active constraint with $\mu_i^\star = 0$ is called strictly active.

Definition 6 *Regard a KKT point $(x^\star, \lambda^\star; \mu^\star)$. We say that strict complementarity holds at this KKT point iff all active constraints are strictly active.*

Theorem 2 *(Second Order Optimality Conditions) Let us regard a point $x^\star$ at which LICQ holds together with multipliers $\lambda^\star, \mu^\star$ so that the KKT conditions are satisfied and let strict complementarity hold. Regard a basis matrix $\mathbb{Z} \in \mathbb{R}^{n*(n-n_g)}$ of the null space of $\frac{\partial \tilde{g}}{\partial x}(x^\star) \in \mathbb{R}^{n_{\tilde{g}}*n}$, i.e., $\mathbb{Z}$ has full column rank and $\frac{\partial \tilde{g}}{\partial x}(x^\star)\mathbb{Z} = 0$. Then the following two statements hold:*

- If $x^\star$ is a local minimizer, then $\mathbb{Z}^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star)\mathbb{Z} \succeq 0$.(Second Order Necessary Condition, short : SONC)

- If $\mathbb{Z}^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star)\mathbb{Z} \succ 0$, then $x^\star$ a local minimizer. This minimizer is unique in its neighborhood, i.e., a strict local minimizer, and stable against small differentiable perturbations of the problem data. (Second Order Sufficient Condition, short: SOSC).

The matrix $\nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star)$ plays an important role in optimization algorithms and is called the Hessian of the Lagrangian, while its projection on the null space of the Jacobian, $\mathbb{Z}^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star)\mathbb{Z}$, is called the reduced Hessian. For an optimization problem, if we can start with an initial guess x_0 and find a updating schema which decreases the objective value while maintain the Hessian matrix positive, if such updating schema can converge, it will converge to a local minimizer. And if the original problem is convex, the local miminizer is, therefore, the global minimizer. Taking advantage of these properties, we can reformulate our rocket car problem into the form as in equation 5.6, and then re-write in the Lagrangian form 5.10, optimize the (objective) Lagrangian function with quasi-Newton method.

5.3 Rokect car problem in Lagrangian form

We discretize the original problem 5.1 based on time t into m sub-problems, i.e. we split the interval $t \in [0,1]$ into subinterval $[t_k, t_{k+1}],\ k = 0,1,2,...,m-1$, where $0 = t_0, t_1, t_2, ..., t_k, t_{k+1}, ..., t_m = 1$. We assume the parameter p is given, for example, we can assume $p = 0$, and find a corresponding optimal T. Later, we can find different optimal T with different p given. In each subinterval, we have inital guess for x, we introduce anther two variables, with s as the guess for x, and q as the guess for u. Therefore, in each subinterval, we have the initial value $x(t_k) = s(t_k), u(t_k) = q(t_k), k = 0,1,2,...,m-1$. When no confusion arises, we will write $s(t_k) = s_k$ and $q(t_k) = q_k$. Notice $s(t_k)$ has two component which corrsponds to x_1 (position) and x_2 (speed). When we want to specify the exact component at time t_k, we would write $s(t_k) = (s_1(t_k), s_2(t_k))$. We would like to find the solution for $x(t; q_k, s_k, p), t \in [t_k, t_{k+1}]$. Assume that such solution has been found, the matching condition at the boundary of each sub-interval will lead to the following equation

$$g(s,q) = \begin{bmatrix} s_0 - 0 \\ x(t_0; p) - s_0 \\ x(t_1; t_0, s_0, q_0, p) - s_1 \\ \vdots \\ x(t_{m-1}; t_{m-2}, s_{m-2}, q_{m-2}, p) - s_{m-1} \end{bmatrix} = 0 \tag{5.11}$$

Notice, we have the equality constraints up to the second to the last subinterval, i.e. $[t_{m-2}, t_{m-1}]$. This is because the terminal condition at time $t_m = 1$ (which the subinterval $[t_{m-1}, t_m]$ covers) has already been defined in the inequality constraints 5.1e and 5.1g. Moreover, the boundary point $s_k, u_k, k = 0,1,2,...,m-1$ is also subject to the inequality constraints 5.1f and 5.1i. All these constraints can be combined together in the following form

$$h(s,q) = \begin{bmatrix} s_2(t_k) - 4, k = 0,1,2,...,m-1 \\ 10 - x_1(t_m; t_{m-1}, s_{m-1}, q_{m-1}, p) \\ x_2(t_m; t_{m-1}, s_{m-1}, q_{m-1}, p) \\ q_k - 10, k = 0,1,2,...,m-1 \\ -10 - q_k, k = 0,1,2,...,m-1 \\ u(t_m) - 10 \\ -10 - u(t_m) \end{bmatrix} \leq 0 \tag{5.12}$$

Within equations 5.11 and 5.12, the solution of each subinterval

$$\begin{aligned} x(t; w_k) &= (x_1(t; w_k), x_2(t; w_k)), t \in [t_k, t_{k+1}] \\ w_k &= (s_k, q_k, p) \\ p&, \textit{ a given value} \end{aligned} \tag{5.13}$$

comes from numerically solving the partial differential equation 5.3. For example, using the Runge–Kutta method or simply using the approximation as used in equation 5.5.

The original objective function *min T* is to minimize the free final time, which is unknown, and we are discretizing the variable $t \in [0,1]$ as defined in equation 5.2. Then the original problem can be transformed, with the final time T as a variable to be optimized. Therefore, we can regard the triple $(s,q,T) := w$ as the new independent variables, and the original problem 5.1 is then transformed into the following form

$$\begin{aligned} \min_{w} \quad & 1 \\ \text{s.t.} \quad & g(w) = 0 \\ & h(w) \leq 0 \\ & w = (s,q,T) \end{aligned} \tag{5.14}$$

The objective function actually is gone, only the constraints play a role. Therefore, we have the Lagrangian function for the rocket car problem of the following form

$$\mathcal{L}(w,\lambda,\mu) = T(w) + \lambda^\top g(w) + \mu^\top h(w) \tag{5.15}$$

Part I

Appendix

A Lists

Bibliography

Ph. L. Toint A. R. Conn, N. I. M. Gould. Convergence of quasi-newton matrices generated by the symmetric rank one update. 1991.

John F. Zancanaro David Morrison, James D. Riley. Multiple shooting method for two-point boundary value problems. 1962.

William C. Davidon. Variable metric method for minimization. 1959.

Powell M.J.D. Fletcher R. A rapid convergent decent method for minimization. 1963.

Michael J. D. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. 2009.

Matthias Schlöder. Numerical methods for optimal control of constrained biomechanical multi-body systems appearing in therapy design of cerebral palsy. 2022.