Department of Mathematics and Computer Science

Heidelberg University

Master thesis

in Scientific Computing

submitted by

Yu Xiang

born in November 10, 1989

2022

Numerical methods for 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 an optimal control problem (OCP), for example, launching a rocket to the moon with minimum fuel expenditure as the objective, or maximizing the profit from the factory production, with constraints in resources available and uncertain market demand. This paper focus on solving optimal control problems with multiple shooting and quasi Newton method.

In general, optimal control deals with the problem of finding the control over the state for a dynamic system over a period of time such that an objective function is optimized. Generally, an optimal control problem can be formulized as follows:

$$\begin{aligned} \min_{x(t),u(t)} \quad & F(x(t),u(t)) \\ s.t. \quad & \dot{x}(t) = f(x(t),u(t)) \\ & x(t) \in \Omega \\ & u(t) \in \mathbb{U} \\ & t \in [t_0, t_f] \end{aligned} \tag{1.1}$$

Here t is the independent variable (generally speaking, time), usually using t_0 and t_f representing the initial and terminal time respectively. $x(t)$ is the state variables, and $u(t)$ is the control variables, $F(\cdot)$ is the objective function, also called the cost function. $\dot{x}(t) = f(x(t), u(t))$ represents the underlying dynamic system, $x(t) \in \Omega$ and $u(t) \in \mathbb{U}$ represents the constraints for which the state variables $x(t)$ and control variable $u(t)$ must satisfy respectively. Sometimes, the constraints are expressed as functions of $x(t)$ and $u(t)$ together.

Generally speaking, there are three basis approaches to address optimal control problems (a) dynamic programming (b) indirect, and (c) direct approaches. (ref Moritz Diehl [2005]). This paper on hand, focus on the direct approaches, which are one of the most widespread and successfully used techniques. Direct approaches transform the original infinite optimal control problem into a finite dimensional nonlinear programming problem (NLP). This NLP is then solved by variants of state-of-the-art numerical optimization methods, and the approach is therefore often sketched as "first discretize, then optimize."

Mutilple shooting method can be used in the "first discretize" part of the direct approaches. The main idea is to divide the whole interval $t \in [t_0, t_f]$ into multiple subintervals, and introduce initial guess for each subinterval, solve the problem in each subinterval with the initial guess, and impose additional matching conditions at the boundary of each subinterval to form a solution of the whole interval. In each subinterval, with mutiple shooting methods applied, the system of differential equations will be turned into a system of ordinary non-linear algebraic equations. Therefore, the original OCP is transferred into piecewise nonlinear programming problem (NLP). We can use the Karush–Kuhn–Tucker (KKT) approach to combine the constraints and the original objective function into a new Lagrange function.

This new function is an optimization problem without any constraints and its optimum value can be determined via derivative methods. To solve any unconstrained NLP, we

can work within an iterative sequential quadratic programming (SQP), or Newton-style framework. Newton-style methods are second-order derivatives iterative optimization algorthm and generally, but not always, converges faster than first-order derivatives mehtod (e.g. gradient descent). The Newton method needs to calculate the second order derivatives, i.e. the Hessian matrix and its inverse in each iteration, which is very expensive to compute. Quasi Netwon methods employs an approximation to the original Hessian matrix and takes an efficient way to update the approximation matrix. Therefore, quasi Newton method is generally faster than Newton method.

Besides the control variables u and state variables x, some optimal control 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 optimal control problems and is of the form

$$\begin{aligned} \min_{x(t),u(t)} \quad & F(x(t;p),u(t)) \\ s.t. \quad & \dot{x}(t;p) = f(x(t;p),u(t)) \\ & x(t;p) \in \Omega \\ & u(t) \in \mathbb{U} \\ & p \in \mathbb{P} \\ & x = x(;,p^\star) \; if \; p = p^\star \\ & t \in [t_0, t_f] \end{aligned} \tag{1.2}$$

where $p^\star$ is a fixed value in the feasible uncertainty set $\mathbb{P}$, where the parameter p can take value from. Equation 1.2 derives from the equation 1.1, with the paramter p added.

Parameterized optimal control problems are very difficult to solve due to the uncertainty in the parameter p. Since the paramter p can take different values, so does the corresponding objective function $F(\cdot)$. Then, the solution of 1.2, i.e. $min\ F(\cdot;p)$ can be regarded as a function of parameter p. One p value corresponds to one solution. Since p is priori unknown, then it makes sense to solve the parameterized optimal control problems in a conservative way. In the paper Schlöder [2022], multiple methods of solving the parameterized optimal control problem have been discussed. The main idea of solving the parameterized optimal control problem in a conservative way is to transform the problem 1.2 into another form. Two different ways of solving the parameterized optimal control problem have been discussed in details, i.e. the classical approach and the training approach. Both are in the form of a bilevel optimization roblem, i.e. an optimization problem in which another optimization problem enters the constraints. Details about these two approaches will be discussed in Chapter 3.

The approaches discussed above will be demonstrated with a case study in state constrained rocket car, with the description of the case, and its numerical solution given in Chapter 4.

The structure of this paper is as follows. Following current introduction Chapter 1, in next Chapter 2, we focus on expaining in details how to solve optimial control problems with direct approaches mainly using mutiple shooting and quasi Newton method. In Chapter 3, we discuss the approaches for solving parameterized optimal control problem, i.e. the classic approach and the training approach. In Chapter 4, we give the description of our case study, i.e. the state constrained rocket car case, and compare the numerical solutions of the classical approach and training approach. In the final Chapter 5, we conclude the analysis with the numerical results.

2 Numerical methods

Optimal control theory deals with systems that can be controlled, i.e whose evolution can be influenced by some external agent. In this paper we only consider the case that the control variable u is a function of only time t, and not function of the state variable x. This type of problem is known as open loop, or controllability problem. The system dynamics of the optimal control problem can be generalized as a system of differential equations

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.1}$$

We would like to have such dynamic system to run in an optimal way, subject to the constriants that are applicable in real life. This indicates that the problem will have a clearly defined objective function, with the dynamic system and the constraints expressed in explicit formula. With the example of launching the rocket to the moon, the objective function can be mimizing the fuel used or minimize the time horizon of the system, subject to the contraints, e.g. gravity, fuel efficiency, speed limitation etc. The trajactory (dynamic system), as well as some of the contraints, can be expressed in differential equations.

In Chapter 1, equation 1.1 gives a high-level formulation of an optimal control problem. Here we augument equation 1.1 with mathematical details, with the objective function and the constraints expressed in explicit formulas. For real-life problems, the optimal control formulation can typically be generalized in the following form

$$\min_{x(t),u(t)} \; F(x(t), u(t)) = \int_{t_0}^{t_f} L(x(t), u(t))dt + E(x(t_f)) \tag{2.2a}$$

$$s.t. \;\; \dot{x}(t) = f(x(t), u(t)), \quad (system\ dynamics) \tag{2.2b}$$

$$g(x(t), u(t)) = 0, \quad (path\ equality\ constraints) \tag{2.2c}$$

$$h(x(t), u(t)) \leq 0, \quad (path\ inequality\ constraints) \tag{2.2d}$$

$$x(t_0) = x_0, \quad (initial\ value) \tag{2.2e}$$

$$r(x(t_f)) \leq 0, \quad (terminal\ constraints) \tag{2.2f}$$

$$x^{lower} \leq x(t) \leq x^{upper} \tag{2.2g}$$

$$u^{lower} \leq u(t) \leq u^{upper} \tag{2.2h}$$

$$t \in [t_0, t_f] \tag{2.2i}$$

Here $L(\cdot)$ and $E(\cdot)$ are called the running cost and end cost, with their sum $F(\cdot)$ the cost/objective function. $g(\cdot)$, $h(\cdot)$ and $r(\cdot)$ are functions representing equaltity constraints, inequality constraints, and terminal constraints respectively. The time horizon is $[t_0, t_f]$, with the initial value known as x_0.

Depending on nature of the underlying optimal control problems, the mathematical expression can take a modifed form of equation 2.2. For certain problems, some of the constraints defined in equation 2.2 may not play a role, while for other problems,

addtional constraints may be needed or existing constraints need to be modified. Nevertheless, Equation 2.2 gives us a general mathematical formulation of the typical optimal control problems in real life, and we do not go further discussion with possible (minor) modification to Equation 2.2.

Various methods exist for solving optimal control problems. The paper Moritz Diehl [2005] summarizes three general approaches to address optimal control problems: (a) dynamic programming, (b) indirect, and (c) direct approaches. Dynamic programming uses the principle of optimality to compute recursively a feedback control for all time t and all x_0. Indirect methods use the necessary conditions of optimality of the infinite problem to derive a boundary value problem (BVP) in ordinary differential equations.

This paper on hand, focus on the direct approaches, which are one of the most widespread and successfully used techniques. Direct approaches transform the original infinite optimal control problem into a finite dimensional nonlinear programming problem (NLP). This NLP is then solved by variants of state-of-the-art numerical optimization methods, and this approach is therefore often sketched as "first discretize, then optimize." One of the most important advantages of direct approaches is that they can easily treat inequality constraints, e.g. the inequality path constraints as in equation 2.2d. This is because structural changes in the active constraints during the optimization procedure are treated by well developed NLP methods, which can deal with inequality constraints and active set changes. (ref Moritz Diehl [2005]).

In the text that follows, we first explain the mutilple shooting method, which can be used in the "first discretize" part of the direct approaches. Afterwards, we explain the KKT conditions, which can easily treat equaltity and inequality constraints of the optimal control problems. With the background knowledge of mutiple shooting methods and KKT conditions, we continue to explain the Newton-style method and sequential quadratic programming. These methods, together, can serve as the "then optimize" part of the direct approaches, and solve the optimal control problems of the form as in equation 1.1, or in more mathematical details as in equation 2.2.

2.1 Multiple shooting

Multiple shooting method was initially introduced to solve boundary value problem (BVP) in differential equation scope David Morrison [1962]. This method, with some adjustment, therefore, is well suited for solving optimal control problem with constraints in differential equations. 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 optimal control problem of the form as in equation 2.2.

To illustrate the concept of shooting method to solve boundary value problem (BVP), we use 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 subintervals. We define a general differential equation with boundary value of the following form

$$\begin{aligned} &\dot{y} = f(t, y, p) \\ &y(t_f) = y_f \end{aligned} \tag{2.3}$$

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:

$$\begin{aligned} &\dot{y} = f(t, y, p), t \in [\tau_k, \tau_{k+1}],\ k = 0, 1, ..., m-1 \\ &y(\tau_k) = \hat{y}_k, k = 0, 1, ..., m-1 \end{aligned} \tag{2.4}$$

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.

$$\begin{aligned} &y(\tau_{k+1}; \hat{y}_k) = \hat{y}_{k+1},\ k = 0, 1, ..., m-1 \\ &\hat{y}_m (i.e.\ \hat{y}_{\tau_m} = \hat{y}_{t_f}) = y_f \end{aligned} \tag{2.5}$$

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 comprehensive 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 based on our initial knowledge of the problem. 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 also applied to the twice differential system of the following form

$$y''(t) = f(t, y(t), y'(t)) \quad y(t_0) = y_0, \quad y(t_f) = y_f, \quad t \in [t_0, t_f] \tag{2.6}$$

The problem 2.6 is similar to the problem 2.3. 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.

2.1.1 Mutiple shooting method for OCP

Mutiple shooting method is also well suited for solving optimal control problems as in equation 2.2, it can serve as the "first discretize" part of the direct approaches. For problems in the form as in equation 2.2, we can discretize the time $t \in [t_0, t_f]$ 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 as in equation 2.2 into piecewise OCPs with augmented parameters and constraints.

The dynamic systems of the OCP in eqation 2.2 is defined in the subequation 2.2b, which we show here independently

$$\dot{x}(t) = f(x(t), u(t)), \quad t \in [t_0, t_f] \tag{2.7}$$

If the initial value x_0 for 2.7 is known, then equation 2.7 becomes an initial value problem (IVP) and the solution can be found via numerical method. If analytical solutions exist for equation 2.7 , then we can use the analytical solution directly. Nevertheless, for most real-life problems, the analytical solution either does not exist or is very difficult to find, and numerical method is the only pratical way to find the solution. Given the initial value x_0, equation 2.7 will have solution as

$$\begin{aligned} x(t) - x_0 &= \int_{t_0}^{t} f(x(\tau), u(\tau)) d\tau, \quad t \in [t_0, t_f] \\ x(t) &= x_0 + \int_{t_0}^{t} f(x(\tau), u(\tau)) d\tau, \quad t \in [t_0, t_f] \end{aligned} \tag{2.8}$$

The integral in equation 2.8 can be approximated by numerical method, one simple approximation to the integral in equation 2.8 can be obtained via Euler method. The Euler method (also called forward Euler method) is a first-order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value. Using a step size equal to 1, the approximation to equation 2.8 via Euler method is of the form

$$x(t) = x_0 + \int_{t_0}^{t} f(x(\tau), u(\tau))d\tau \;\approx\; x_0 + tf(x(t_0), u(t_0)) \tag{2.9}$$

Euler method is a first order and explicit method for approximating intergrals, another widely used method is the trapezoidal rule, which is an implicit second-order method. The trapezoidal rule can approximate equation 2.8 as

$$x(t) = x_0 + \int_{t_0}^{t} f(x(\tau), u(\tau))d\tau \;\approx\; x_0 + \frac{t}{2}[f(x(t_0), u(t_0)) + f(x(t), u(t))] \tag{2.10}$$

Notice, the soultion $x(t)$ becomes an input in the approximation in equation 2.10, and that is why this method is classified as an implicit method. The trapezoidal rule belongs to the family of Runge–Kutta method.,i.e. a family of implicit and explicity iterative methods, with various order of derivatives used.

To solve the OCP as in equation 2.2, we first discretize the time horizon $\in [t_0, t_f]$ into mutiple subintervals $t_0 = \tau_0 < \tau_1 < ... < \tau_m = t_f$, then we introduce the initial guess for x and u in each subinterval, i.e. we have $x_0, x_1, ..., x_m$ and $u_0, u_1, ..., u_m$. Then in each subinterval $\mathbb{I}_j = [\tau_{j-1}, \tau_j]$, the following step should be taken:

- Solve the dynamic system as in equation 2.7 with the initial guess (x_{j-1}, u_{j-1}) within subinterval $\mathbb{I}_j$, and obtain a solution $x(\tau; x_{j-1}, u_{j-1}),\ \tau \in \mathbb{I}_j = [\tau_{j-1}, \tau_j]$
- Evaluate the cost function 2.2a within subinterval $\mathbb{I}_j$ with the solution $x(\tau; x_{j-1}, u_{j-1})$. We use the notation F_j as the objective function value that is applicable in subinterval $\mathbb{I}_j$, with the definition given in equation 2.12.
- Evaluate the error with the constraints 2.2c, 2.2d, 2.2f, 2.2g, and 2.2h. We use the notation Θ_j as the collections of contrstraints that are applicable in subinterval $\mathbb{I}_j$.

Within subinterval $\mathbb{I}_j$, with initial guess $X_j = (x_{j-1}, u_{j-1})$, the result from solving the dynamic system equation 2.7 might be an infeasiable solution, i.e. a solution which does not satisfy the constraints 2.2c, 2.2d, 2.2f, 2.2g, and 2.2h. In this case, we can update the initial guess iteratively until feasiable solutions are found and one optimal solution is reached. The solution is optimal if a minimum cost function value in the subinterval $\mathbb{I}_j$ is obtained while satisfying all the constraints 2.2c, 2.2d, 2.2f, 2.2g, and 2.2h. In practice, we do not aim to find the optimal solution in each subinterval, instead we aim to find the optimal solution of the whole interval. With the matching codition added, then the original OCP is transferred into NLP of the following form

$$\min_{X_j} \sum_j F_j \tag{2.11a}$$

$$s.t.\quad X_j = (x_{j-1}, u_{j-1}) \tag{2.11b}$$

$$(X_j, x(\tau_j; x_{j-1}, u_{j-1})) \in \Theta_j \tag{2.11c}$$

$$x(\tau_j; x_{j-1}, u_{j-1}) - x_j = 0 \tag{2.11d}$$

$$t_0 = \tau_0 < \tau_1 < ... < \tau_m = t_f \tag{2.11e}$$

$$j = 1, 2, ..., m \tag{2.11f}$$

where $(X_j, x(\tau_j; x_{j-1}, u_{j-1})) \in \Theta_j$ represents that the initial guess x_{j-1}, u_{j-1} and the solution $x(\tau_j; x_{j-1}, u_{j-1})$ for subinterval $\mathbb{I}_j = [\tau_{j-1}, \tau_j]$ satisfy the constraints 2.2c, 2.2d, 2.2f, 2.2g, and 2.2h within subinterval $\mathbb{I}_j$. The solution $x(\tau_j; x_{j-1}, u_{j-1})$ comes from solving the dynamic system in equation 2.2b for the subinterval $\mathbb{I}_j$ with numerical method used. The numerical method can be one of the Runge–Kutta methods as discussed before. The objective function F_j of subinterval is calculated as followin

$$\begin{aligned} F_j &= \int_{\tau_{j-1}}^{\tau_j} L(x(t), u(t))dt \qquad\qquad if \quad j = 1, ..., m-1 \\ F_j &= \int_{\tau_{j-1}}^{\tau_j} L(x(t), u(t))dt + E(x(t_f)) \qquad\qquad if \quad j = m \end{aligned} \tag{2.12}$$

2.2 KKT condition

As stated in the introduction Chpater 1, direct approaches transform the original infinite optimal control problem into a finite dimensional nonlinear programming problem (NLP). We have shown in previous section 2.1.1 that an OCP of the form 2.2 can be transferred into NLP of the form 2.11. This NLP is then solved by variants of state-of-the-art numerical optimization methods. KKT condition is one of the widespread and sucessfully used techniques to address the constraints of NLP. Before we explain the KKT condition, we need to introduce several definitions and theorems first. We consider a general NLP of the following form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned} \tag{2.13}$$

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

$$\Omega := \{x \in \mathbb{R}^n \mid g(x) = 0,\ h(x) \leq 0\} \tag{2.14}$$

Definition 2 *(Active Constraints and Active Set) An inequality constraint $h_i(x) \leq 0$ 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 column vector as follows:

$$\tilde{g}(x) = \begin{pmatrix} g(x) \\ h_i(x),\ i \in \mathcal{A}(x^\star) \end{pmatrix} \tag{2.15}$$

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 2.13 and the LICQ holds at $x^\star$, then there exist so called multiplier vectors $\lambda \in \mathbb{R}_g^n$ and $\mu \in \mathbb{R}_h^n$ with*

$$\nabla f(x^\star) + \nabla g(x^\star)\lambda^\star + \nabla h(x^\star)\mu^\star = 0 \tag{2.16a}$$
$$g(x^\star) = 0 \tag{2.16b}$$
$$h(x^\star) \leq 0 \tag{2.16c}$$
$$\mu^\star \geq 0 \tag{2.16d}$$
$$\mu_i^\star h_i(x^\star) = 0, \ \ i = 1, ..., n_h \tag{2.16e}$$

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*

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x) \tag{2.17}$$

Here, we have used the so called "Lagrange multipliers" $\lambda \in \mathbb{R}_g^n$ and $\mu \in \mathbb{R}_h^n$. The last three KKT conditions 2.16c - 2.16e 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. If the original problem is convex, the local miminizer is, therefore, the global minimizer. Taking advantage of these properties, we can reformulate the NLP as in equation 2.13 as well as the optimal control problem as in equation 2.2, and then re-write in the Lagrangian form 2.17, optimize the (objective) Lagrangian function with quasi Newton method.

2.3 quasi Newton method

As explained in Section 2.2, KKT condtion can be applied to include the constraints into a Lagrangian function, and we can employ an updating schema with an initial guess to solve NLP as in equation 2.13 and optimal control problem as in equation 2.2. Newton and quasi Netwon method can be used as the updating method to find the optimal solution. For the sake of simplicity, we focus on explaining the Newton and quasi Newton method without constraints in this chapter. How the quasi Newton method, together with multiple shooting, can be used to solve optimal control problem, will be exlained in more details in Chapter 4 when we discuss our numerical solutions for the rocket car case.

2.3.1 Newton method

A general optimization problem is typically of the form

$$\begin{aligned} &min \;\; f(x) \\ &s.t. \;\; x \in \Omega \end{aligned} \tag{2.18}$$

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 equation 2.13, i.e. the feasible set $\Omega = \underset{x}{arg}\ \{g(x) = 0, h(x) \leq 0\}$. Before discussing the optimization problem with constraints, we first focus on the problems without constraints, i.e. problem of the form

$$\min_{x \in \mathbb{R}^n} \qquad f(x) \tag{2.19}$$

The problem 2.19 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

$$\nabla f(x_k) + H(x_k)\delta(x_k) = 0 \tag{2.20}$$

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

- 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, the quasi Netwon method is usually preferred for solving optimal control or general optimisation problem.

2.3.2 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.

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

[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.

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 2.20, with the approximation matrix B replacing the original Hessian matrix H, i.e.

$$\nabla f(x_k) + B_k \delta(x_k) = 0 \tag{2.21}$$

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

$$\begin{aligned} &\underset{B_{k+1}}{min} \ \|B_{k+1} - B_k\| \\ &s.t. \ \ B_{k+1} = B_{k+1}^T, \ B_{k+1}\delta(x_k) = y_k \end{aligned} \tag{2.22}$$

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 2.22 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

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

Finally, we get the update formula as follows:

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

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

$$B_{k+1}^{-1} = (I - \frac{\delta(x_k) y_k^T}{y_k^T \delta(x_k)}) B_k^{-1} (I - \frac{y_k \delta(x_k)^T}{y_k^T \delta(x_k)}) + \frac{\delta(x_k) \delta(x_k)^T}{y_k^T \delta(x_k)} \tag{2.23}$$

As shown in the formula 2.23, 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.

We have shown that with quasi Newton method, with an initial guess x_0, we can interatively update x_k until convergence so that an optimal solution to the unconstrained optimization problem as in 2.19 can be obtained. In the next section, we explain how the quasi Netwon method can be applied to optmizations problem with contstraints as well as OCP problem of the form 2.2.

2.4 Sequential quadrative programming

With all the background knowlege of mutiple shooting, KKT conditions and Newton-style method, now we are ready to explain how we can solve the OCP with the form 2.2 in details, i.e. working within an iterative Sequential Quadratic Programming (SQP) approach.

As explained in section 2.1.1, the original OCP with the form 2.2 can be transferred into a NLP of the form 2.11. Within the NLP 2.11, the independent variables are the initial guess of x and u for each subinterval $\mathbb{I}_j = [\tau_{j-1}, \tau_j]$, i.e. $w = (x_0, u_0, x_1, u_1, ..., x_{m-1}, u_{m-1})$. The constraints 2.11c refers to equation 2.2c, 2.2d, 2.2f, 2.2g, and 2.2h, which are applied to the independent variables $w = (x_0, u_0, x_1, u_1, ..., x_{m-1}, u_{m-1})$. They are already in the form as equality constraints or inequality constraints. Together with the matching condtion 2.11d, then the NPL problem 2.11 can be re-written as the form in NLP 2.13, with the x in NLP 2.13 replaced by $w = (x_0, u_0, x_1, u_1, ..., x_{m-1}, u_{m-1})$, with the equality constraints $g(x) = 0$ coming from the matching codition 2.11d and the original equality constraints 2.2c of OCP 2.2. The other constraints of OCP 2.2, as in 2.2d, 2.2f, 2.2g, and 2.2h can be written into the $h(x) \leq 0$ part of NLP 2.13.

To solve NLP of the form 2.13, we can use the KKT apporach, i.e. including the constraints and the original objective function into a new objective function, defined as a Lagrangian function as 2.17, which we re-rewite here

$$\mathcal{L}(w, \lambda, \mu) = f(w) + \lambda^\top g(w) + \mu^\top h(w) \tag{2.24}$$

Based on our discussion regarding Newton-style method, to find the minimum of $\mathcal{L}(w, \lambda, \mu)$, we are motivated to solve a linear system as in equation 2.20, where the first

order derivatives and Hessian matrix is calculated for the objective function $\mathcal{L}(w, \lambda, \mu)$. Starting with an initial guess of (w_0, λ_0, μ_0), then we are motivated to update (w, λ, μ) so as to decrease the objective function $\mathcal{L}(\cdot)$ by solving a subproblem of quadrative programming of the following form

$$\min_{\Delta w} \quad \frac{1}{2}\Delta w^\top \nabla_w^2 \mathcal{L}(\cdot)\Delta w + \nabla_w f(w_i)^T \Delta w \tag{2.25a}$$

$$\text{s.t.} \quad g(w_i) + \nabla g(w_i)^T \Delta w = 0 \tag{2.25b}$$

$$h(w_i) + \nabla h(w_i)^T \Delta w \leq 0 \tag{2.25c}$$

where $\nabla_w^2 \mathcal{L}(\cdot)$ is the Hessian matrix of function $\mathcal{L}(\cdot)$, whose updating can be calculated with quasi Newton method. With a solution $\Delta w_k, \lambda_k^{QP}, \mu_k^{QP}$ found from the subproblem 2.25, we can therefore update the w, λ, μ as

$$w_{k+1} = w_k + \Delta w_k \tag{2.26a}$$

$$\lambda_{k+1} = \lambda_k^{QP} \tag{2.26b}$$

$$\mu_{k+1} = \mu_k^{QP} \tag{2.26c}$$

until convergence.

We have shown that the original OCP of the form 2.2 can be transferred into a NLP of the form 2.11 using mutiple shooting method. The NLP 2.11 can then be re-written in the form 2.13, and can be solved using Netwon-style framework, i.e. sequential quadrative programming and quasi Netwon method. The numerical implementation details to a case study of rocket car will be given in Chapter 4.

3 Optimal control under uncertainty

Besides the OCP as in the formulation 2.2, some OCP may have uncertain parameters whose value are priori unknown, and the optimal objective value depends on the parameter value, as shown in the formulation 1.2 in Chapter 1. This kind of problem is called the parameterized optimization problems and the formulation 1.2 can be augumented with mathematical details as well, leading to a OCP under uncertainty of the following form

$$\min_{x(t),u(t)} \; F(x(t;p),u(t)) = \int_{t_0}^{t_f} L(x(t;p),u(t))dt + E(x(t_f;p)) \tag{3.1a}$$

$$s.t. \;\; \dot{x}(t;p) = f(x(t;p),u(t)) \;\; (system\ dynamics) \tag{3.1b}$$

$$g(x(t;p),u(t)) = 0, \;\; (path\ equality\ constraints) \tag{3.1c}$$

$$h(x(t;p),u(t)) \leq 0, \;\; (path\ inequality\ constraints) \tag{3.1d}$$

$$x(t_0) = x_0, \;\; (initial\ value) \tag{3.1e}$$

$$r(x(t_f;p)) \leq 0, \;\; (terminal\ constraints) \tag{3.1f}$$

$$x^{lower} \leq x(t) \leq x^{upper} \tag{3.1g}$$

$$u^{lower} \leq u(t) \leq u^{upper} \tag{3.1h}$$

$$p \in \mathbb{P} \tag{3.1i}$$

$$x = x(;,p^\star)\ if\ p = p^\star \tag{3.1j}$$

$$t \in [t_0, t_f] \tag{3.1k}$$

In other words, the state variables x depenpds not only on the system dynamics 3.1b, the control variables u, but also an uncertain parameter p. Parameterized optimization problems are very difficult to solve due to the uncertainty in the parameter p. Since different parameter p will lead to different solutions, it makes sense to solve solve the parameterized optimal control problems in a conservative way. In the paper Schlöder [2022], multiple methods of solving the parameterized optimal control problem have been discussed. For simplicity, we use the more compact forumation 1.2 in Chapter 1 as the general representation of a parameterized optimal control problem, and we show the formulation here again

$$\begin{aligned} \min_{x(t),u(t)} \quad & F(x(t;p),u(t)) \\ s.t. \quad & x(t;p) \in \Omega \\ & u(t) \in \mathbb{U} \\ & p \in \mathbb{P} \\ & x = x(;,p^\star) \ if \ p = p^\star \\ & t \in [t_0, t_f] \end{aligned} \tag{3.2}$$

The main idea of solving the parameterized optimal control problem 3.2 in a conservative way is to transform the problem into another form. Two different ways of solving the parameterized optimal control problem will be discussed in details in the sebsequent sections, i.e. the classical approach and the training approach.

3.1 Classical approach

Assuming parameter $p^\star$ lies in an uncertainty set $\mathbb{P}$, we can firstly reach one objective, i.e. identifying a worst possible solution with respect to $p^\star$, 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 constraints. Mathematically, the problem 3.2 is transformed into another form, and can be formulated in a simplified notation, as following

$$\begin{aligned} \underset{u(\cdot)}{min} \ \underset{p\in\mathbb{P},\ x(:,p)}{max} \quad & F(x(t;p),u(t)) \\ s.t. \quad & x(t) \in \Omega \\ & u(t) \in \mathbb{U} \\ & p \in \mathbb{P} \\ & x = x(;,p^\star) \ if \ p = p^\star \\ & t \in [t_0, t_f] \end{aligned} \tag{3.3}$$

Due to the $min\ max$ notation, this classical approach of solving the bilevel problem can also be called $minmax$ approach. In the classical approach, the set of feasible controllable variables are given by $u(\cdot)$, which yield feasible trajectories $x(\cdot, p)$ for all $p \in \mathbb{P}$. 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 dynamic system has no prior knowledge about the value of the parameter p and gets no feedback during the process and has to decide the control strategy in advance.

3.2 Training approach

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 $p \in \mathbb{P}$ is realized. What follows the training period (during which the parameter p is realized), the dynamic system is able to react to it in an optimal manner, i.e. an optimal value $F(\cdot)$ 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

$$\begin{aligned} \max_{p \in \mathbb{P}} \min_{x(:,p),\ u(t)} \quad & F(x(t;p), u(t)) \\ s.t. \quad & x(t) \in \Omega \\ & u(t) \in \mathbb{U} \\ & p \in \mathbb{P} \\ & x = x(;, p^\star)\ if\ p = p^\star \\ & t \in [t_0, t_f] \end{aligned} \tag{3.4}$$

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

$$\begin{aligned} &\min_{x \in \mathcal{R}^n} F(x) \\ &s.t.\ a_i \leq x_i \leq b_i, i = 1, ..., n \end{aligned} \tag{3.5}$$

The name BOBYQA is an acronym for "Bound Optimization BY Quadratic Approximation", and is used to solve lower level problem of **??**. 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 **??** 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 $Q_k(x) \to F(x),\ x \in \mathcal{R}^n$, 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

$$Q_k(y_j) = F(y_j),\ j = 1, 2, ..., m. \tag{3.6}$$

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

$$F(x_k) = min\ \{F(y_j),\ j = 1, 2, ..., m\}, \tag{3.7}$$

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 3.5, 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

$$x_{k+1} = \begin{cases} x_k, & F(x_k + d_k) \geq F(x_k) \\ x_k + d_k, & F(x_k + d_k) < F(x_k) \end{cases} \tag{3.8}$$

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

$$Q_{k+1}(\hat{y}_j) = F(\hat{y}_j),\ j = 1, 2, ..., m, \tag{3.9}$$

at the new interpolation points

$$\hat{y}_j = \begin{cases} y_j, & j \neq t, \\ x_k + d_k, & j = t \end{cases},\ j = 1, 2, ..., m. \tag{3.10}$$

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 **??** can be solved with the BOBYQA algorithm for a given p, since the lower level problem can be re-written into the form of 3.5 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.

[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.

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

4 Numerical solution

4.1 Introduction to the rocket car case

4.2 Apply Classical (minmax) approach

4.3 Apply Training (maxmin) approach

5 Conclusion

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.

Holger Diedam Pierre-Brice Wieber Moritz Diehl, Hans Georg Bock. Fast direct multiple shooting algorithms for optimal robot control. *Fast Motions in Biomechanics and Robotics*, 2005.

Michael J. D. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Technical Report, Department of Applied Mathematics and Theoretical Physics*, 2009.

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

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den (Datum) .....