4
Optimal Control and Trajectory Optimization

Previously, the idea of using *optimal control*[1] techniques (also referred to as *trajectory optimization*) for robot motion planning and control was presented. In this chapter, the optimal control problem is revisited in more detail, including a brief discussion on the use of both *indirect* and *direct* methods.

[1] D. E. Kirk. *Optimal Control Theory: An Introduction*. Dover Publications, 2004

Optimal Control and Trajectory Optimization

Consider an optimal control problem (OCP) formulated as the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{u},\boldsymbol{x}}{\text{minimize}}\ & h(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt, \\ \text{s.t. } & \dot{\boldsymbol{x}}(t) = a(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \\ & \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \end{aligned} \tag{4.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the robot state, $\boldsymbol{u} \in \mathbb{R}^m$ is the control input, x_0 is a known robot initial condition, $a(\boldsymbol{x}, \boldsymbol{u}, t)$ is a function describing the robot's dynamics, and the functions $h(\boldsymbol{x}(t_f), t_f)$ and $g(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$ define the cost function[2]. The goal is to solve the optimal control problem (4.1) in order to define an *optimal* open-loop control law of the form

$$\boldsymbol{u}^*(t) = f(\boldsymbol{x}(t_0), t).$$

[2] State constraints $\boldsymbol{x}(t) \in \mathcal{X}$ and control constraints $\boldsymbol{u}(t) \in \mathcal{U}$ are also often included in practice, but for simplicity are not included here.

Unfortunately, this optimization problem is particularly challenging to solve since it is *infinite-dimensional*[3]. Methods for solving (4.1) can be categorized as either *indirect* or *direct*. Both types of methods (almost always) require some form of discretization, such that the problem can be solved numerically. However, the way in which the problem is discretized is what makes each method unique.

[3] It is referred to as infinite-dimensional because it is an optimization over functions and not just a finite set of parameters.

1. *Indirect methods* follow a "first optimize, then discretize" approach. These methods first derive conditions for optimality of the original infinite-dimensional problem. A solution is then recovered by discretizing the optimality conditions.

2. *Direct methods* follow a "first discretize, then optimize" approach. These methods first discretize the original problem into a finite-dimensional problem (called a *nonlinear program*), which is then solved numerically to recover an optimal solution.

4.1 Indirect Methods

As previously mentioned, indirect methods solve the optimal control problem (4.1) by deriving *necessary optimality conditions* (NOC). A numerical procedure is then used to find solutions that satisfy these conditions of optimality, thereby "indirectly" solving the original OCP. As a brief example, for unconstrained finite-dimensional optimization problems the classic first-order necessary optimality condition[4] is that the gradient of the function must be zero (e.g. minimize $f(x) = x^2$ with $x \in \mathbb{R}$ has NOC $\frac{df}{dx} = 0$).

[4] It is important to note that these conditions are called necessary because they are "necessary", but they may not be "sufficient". In other words there may exist solutions that satisfy the NOCs but do not solve the original problem.

4.1.1 Constrained Finite-Dimensional Optimization

Before discussing techniques to derive necessary optimality conditions for the infinite-dimensional OCP (4.1), it is useful to briefly examine analogous conditions in finite-dimensional optimization[5]. Consider the equality-constrained finite-dimensional optimization problem:

[5] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}}\ f(\boldsymbol{x}), \\ &\quad \text{s.t. } h_i(\boldsymbol{x}) = 0, \quad i = 1, \dots, m \end{aligned} \tag{4.2}$$

with variable $\boldsymbol{x} \in \mathbb{R}^n$.

Necessary optimality conditions for (4.2) are derived by first forming a function called the Lagrangian $L(\boldsymbol{x}, \boldsymbol{\lambda})$, which augments the objective function with a weighted sum of the constraint functions:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i h_i(\boldsymbol{x}), \tag{4.3}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is a vector of *Lagrange multipliers*. The NOCs are then given as:

$$\begin{aligned} \nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = 0, \\ \nabla_{\boldsymbol{\lambda}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = 0, \end{aligned} \tag{4.4}$$

which are the gradients of the Lagrangian with respect to the variables $\boldsymbol{x}$ and the multipliers $\boldsymbol{\lambda}$. Note that the NOCs (4.4) are a set of $n + m$ *algebraic* equations with $n + m$ unknowns. In contrast, it will be seen next that the NOCs for infinite-dimensional problems are not algebraic, but rather differential.

4.1.2 Necessary Optimality Conditions

Analogously to the Lagrangian (4.3) in finite-dimensional optimization, the first step to defining the NOCs for the infinite-dimensional OCP (4.1) is to define a

function called the *Hamiltonian*:

$$H(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}(t), t) := g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \boldsymbol{p}^\top(t) a(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \tag{4.5}$$

where $\boldsymbol{p}(t) \in \mathbb{R}^n$ is a multiplier referred to as a *costate*. The NOCs are then given by a set of differential and algebraic equations:

$$\begin{aligned} \dot{\boldsymbol{x}}^*(t) &= \frac{\partial H}{\partial \boldsymbol{p}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t), \\ \dot{\boldsymbol{p}}^*(t) &= -\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t), \\ 0 &= \frac{\partial H}{\partial \boldsymbol{u}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t), \end{aligned} \tag{4.6}$$

which must be satisfied for all $t \in [t_0, t_f]$. These NOCs consist of $2n$ first order differential equations and m algebraic equations. Identifying unique solutions to the $2n$ differential equations requires $2n$ boundary conditions (actually $2n+1$ if the final time t_f is not fixed). The initial condition $\boldsymbol{x}^*(t_0) = \boldsymbol{x}_0$ specifies n of these conditions, and the remaining conditions are given by

$$\begin{aligned} &\Big(\frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t_f), t_f) - \boldsymbol{p}^*(t_f))\Big)^\top \delta \boldsymbol{x}_f \\ &+ \big(H(\boldsymbol{x}^*(t_f), \boldsymbol{u}^*(t_f), \boldsymbol{p}^*(t_f), t_f) + \frac{\partial h}{\partial t}(\boldsymbol{x}^*(t_f), t_f)\big)\delta t_f = 0, \end{aligned} \tag{4.7}$$

where $\delta \boldsymbol{x}_f$ and δt_f are referred to as *variations*. If either the final time or final state is fixed in the optimal control problem the corresponding variation is forced to be zero, which changes the boundary conditions (4.7). The resulting boundary conditions for the four possible scenarios are now summarized:

Fixed Final Time and Fixed Final State: If both t_f and $\boldsymbol{x}(t_f)$ are fixed, both variations δt_f and $\delta \boldsymbol{x}_f$ are set to zero. In this case the boundary conditions (4.7) are trivially satisfied, and the remaining boundary conditions on the NOCs (4.6) are given by:

$$\begin{aligned} \boldsymbol{x}^*(t_0) &= \boldsymbol{x}_0, \\ \boldsymbol{x}^*(t_f) &= \boldsymbol{x}_f. \end{aligned}$$

Fixed Final Time and Free Final State: If only t_f is fixed, then only the variation $\delta t_f = 0$. In this case the conditions (4.7) simplify and the boundary conditions for the NOCs (4.6) are given by:

$$\begin{aligned} \boldsymbol{x}^*(t_0) &= \boldsymbol{x}_0, \\ \frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t_f), t_f) - \boldsymbol{p}^*(t_f) &= 0. \end{aligned}$$

Free Final Time and Fixed Final State: If only $\boldsymbol{x}_f$ is fixed, then only the variation $\delta \boldsymbol{x}_f = 0$. In this case the conditions (4.7) simplify and the boundary conditions

for the NOCs (4.6) are given by:

$$\begin{aligned}
&\boldsymbol{x}^*(t_0) = \boldsymbol{x}_0, \\
&\boldsymbol{x}^*(t_f) = \boldsymbol{x}_f, \\
&H(\boldsymbol{x}^*(t_f), \boldsymbol{u}^*(t_f), \boldsymbol{p}^*(t_f), t_f) + \frac{\partial h}{\partial t}(\boldsymbol{x}^*(t_f), t_f) = 0.
\end{aligned}$$

Note that in this case since the final time is free an additional boundary condition is added, so there are now $2n + 1$ total conditions.

Free Final Time and Free Final State: If neither t_f or $\boldsymbol{x}(t_f)$ is fixed, then the boundary conditions for the NOCs (4.6) are given by:

$$\begin{aligned}
&\boldsymbol{x}^*(t_0) = \boldsymbol{x}_0, \\
&\frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t_f), t_f) - \boldsymbol{p}^*(t_f) = 0, \\
&H(\boldsymbol{x}^*(t_f), \boldsymbol{u}^*(t_f), \boldsymbol{p}^*(t_f), t_f) + \frac{\partial h}{\partial t}(\boldsymbol{x}^*(t_f), t_f) = 0.
\end{aligned}$$

Again, since the final time is free an additional boundary condition is added such that there are $2n + 1$ total. Note that last two conditions are both extracted from (4.7) because the variations $\delta \boldsymbol{x}_f$ and δt_f are independent.

4.1.3 Two-Point Boundary Value Problems

Finding solutions that satisfy the necessary optimality conditions (4.6) for the optimal control problem is challenging. In particular, any solution must satisfy a set of $2n$ differential equations with boundary conditions specified at both t_0 and t_f. The problem of finding solutions to differential equations with boundary conditions specified at two points is called a *two-point boundary value problem*. Luckily, numerical procedures have been developed for solving these types of problems. For example the `scikits.bvp_solver` package in Python or the function `bvp4c` in Matlab implement schemes for solving these problems.

Most solvers for two-point boundary value problems typically assume the NOCs (4.6) and their boundary conditions are expressed in the standard form:

$$\dot{\boldsymbol{z}} = g(\boldsymbol{z}, t), \quad l(\boldsymbol{z}(t_0), \boldsymbol{z}(t_f)) = 0. \tag{4.8}$$

However, some types of problems may not directly fit into this standard form. For such instances, it is sometimes possible to convert a non-standard form problem into the standard form (4.8) [6].

In optimal control settings one common case where the two-point boundary value problem cannot directly be expressed in standard form is free final time problems, where t_f needs to be determined but does not have any associated dynamics. A useful trick in this case is to define a new variable $\tau = \frac{t}{t_f} \in [0, 1]$ to replace the time variable t (since before t_f wasn't known but now $\tau_f = 1$ is known). With this new variable the following changes can be made:

[6] U. Ascher and R. D. Russell. "Reformulation of boundary value problems into "standard" form". In: *SIAM Review* 23.2 (1981), pp. 238–254

1. Replace all derivatives with respect to t with derivatives with respect to τ, using $\frac{d(\cdot)}{d\tau} = t_f \frac{d(\cdot)}{dt}$ (chain rule).

2. Introduce a "dummy" state r that corresponds to t_f with dynamics $\dot{r} = 0$.

3. Replace t_f with r in all NOCs and in all boundary conditions.

The "dummy" state r can then be included in the vector $\boldsymbol{z}$ and the NOCs expressed in the standard form (4.8). In summary, this approach can be thought of as "tricking" the standard-form solver to think that the final time is 1 and that t_f is actually a state with dynamics (although the dynamics are $\dot{t}_f = 0$).

Example 4.1.1 (Free Final Time OCP)**.** Consider a double integrator system

$$\ddot{x} = u,$$

where $x \in \mathbb{R}$ is the state and $u \in \mathbb{R}$ is the control input where the control task is to find a trajectory that minimizes the cost function

$$J = \frac{1}{2}\alpha t_f^2 + \int_0^{t_f} \frac{1}{2}\beta u^2(t)\mathrm{d}t,$$

and satisfies the boundary conditions

$$x(0) = 10, \quad \dot{x}(0) = 0, \quad x(t_f) = 0, \quad \dot{x}(t_f) = 0.$$

This problem is a free final time problem with a fixed final state, and the cost is formulated to find a trajectory that minimizes a combination of the time to reach the final state and the amount of control effort required to get there. A trade-off between minimizing final time and minimizing control effort is made by adjusting the weighting parameters[7] α and β. From the cost function it is apparent that:

[7] What does intuition suggest the optimal behavior would be for $\alpha = 0$ or for $\beta = 0$?

$$h(\boldsymbol{x}(t_f), t_f) = \frac{1}{2}\alpha t_f^2, \quad g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = \frac{1}{2}\beta u^2(t),$$

and the dynamics equation can be equivalently expressed as a first-order system of ODEs by setting $x_1 = x$ and $x_2 = \dot{x}$:

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = u,$$

such that $\boldsymbol{x} = [x_1,\, x_2]^\top$ and the boundary conditions become:

$$x_1(0) = 10, \quad x_2(0) = 0, \quad x_1(t_f) = 0, \quad x_2(t_f) = 0.$$

Now that the problem has been introduced, the first step is to derive the Hamiltonian:

$$H = \frac{1}{2}\beta u^2 + p_1 x_2 + p_2 u,$$

where p_1 and p_2 are the costates. Next, the NOCs (4.6) can be derived by taking the partial derivatives of H with respect to p, x, and u:

$$\begin{aligned}
\dot{x}_1^* &= x_2^*, \\
\dot{x}_2^* &= u^*, \\
\dot{p}_1^* &= 0, \\
\dot{p}_2^* &= -p_1^*, \\
0 &= \beta u^* + p_2^*.
\end{aligned}$$

The next step is then to determine appropriate boundary conditions for the NOCs. As mentioned before, this problem is a free final time and fixed final state problem. Therefore the boundary conditions are given by

$$\begin{aligned}
&x_1^*(0) = 10, \\
&x_2^*(0) = 0, \\
&x_1^*(t_f) = 0, \\
&x_2^*(t_f) = 0, \\
&\frac{1}{2}\beta u^*(t_f)^2 + p_1^*(t_f)x_2^*(t_f) + p_2^*(t_f)u^*(t_f) + \alpha t_f = 0.
\end{aligned}$$

Now, from the last NOC it can be seen that the optimal control u^* can be solved for in terms of the costate p_2^*:

$$u^* = -\frac{1}{\beta}p_2^*.$$

This expression can then be substituted into the second NOC and into the boundary conditions. At this point the resulting two-point boundary value problem can be expressed in the standard form (4.8) (by using the free final time trick previously discussed), and solved numerically. However, it also turns out that this problem is simple enough to solve analytically as well.

Analytical Solution: Integrating the differential equations for the costates p_1 and p_2 gives:

$$\begin{aligned}
p_1^* &= C_1, \\
p_2^* &= -C_1 t + C_2,
\end{aligned}$$

where C_1 and C_2 are constants. Therefore, the optimal control u^* can be expressed as $u^* = \frac{C_1}{\beta}t - \frac{C_2}{\beta}$ and the states x_1 and x_2 can be integrated to yield:

$$\begin{aligned}
x_2^* &= \frac{C_1}{2\beta}t^2 - \frac{C_2}{\beta}t + C_3, \\
x_1^* &= \frac{C_1}{6\beta}t^3 - \frac{C_2}{2\beta}t^2 + C_3 t + C_4,
\end{aligned}$$

where C_3 and C_4 are additional constants. There are now five unknown quantities, C_1, C_2, C_3, C_4, and t_f, which can be determined by leveraging the five

boundary conditions. In particular from the condition $x_1^*(0) = 10$ and $x_2^*(0) = 0$ it is easy to see that $C_3 = 0$ and $C_4 = 10$. The remaining boundary conditions can then be used to analytically solve for the remaining constants, and in particular:

$$t_f = (1800\frac{\beta}{\alpha})^{1/5}.$$

For a couple of interesting insights, it can be noted that as $\beta \to 0$ the cost function penalizes the final time more, and from the expression for t_f we can see that $t_f \to 0$. Additionally, as $\alpha \to 0$ the cost function penalizes control inputs, and correspondingly it can be seen in the expression for t_f that $t_f \to \infty$. Further, note that the optimal control takes the form

$$u^*(t) = \frac{C_1}{\beta}t - \frac{C_2}{\beta}.$$

Thus the control input is linear in time and its magnitude is inversely proportional to β.

4.2 *Direct Methods*

Unlike indirect methods, direct methods do not require a derivation of the necessary optimality conditions. Instead these methods directly discretize the original optimal control problem (4.1) to turn it into a finite-dimensional constrained optimization problem called a *nonlinear programming problem*.

While several approaches for discretizing the OCP exist, one simple approach is to just use a forward Euler time discretization. Recall that the forward Euler time discretization method (the simplest of the Runge-Kutta methods) can be used to numerically solve differential equations. In particular, with the choice of a time step h_i the differential equations $\dot{\boldsymbol{x}} = a(\boldsymbol{x}, \boldsymbol{u}, t)$ are discretized as:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h_i a(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i), \tag{4.9}$$

where $\boldsymbol{x}_i = \boldsymbol{x}(t_i)$, $\boldsymbol{u}_i = \boldsymbol{u}(t_i)$, and $t_{i+1} - t_i = h_i$. With this recursive expression (4.9), an initial condition $\boldsymbol{x}(t_0)$, and a sequence of inputs $\boldsymbol{u}(t_i)$ for $i \geq 0$, the states $\boldsymbol{x}(t_i)$ can be computed easily. Suppose the optimal control problem (4.1) was defined over the time interval $[t_0, t_f]$. Applying a forward Euler time discretization essentially partitions this interval into a finite set of N times $\{t_0, t_1, \ldots, t_N\}$ where $t_N = t_f$ and the time step between each is $h_i = t_{i+1} - t_i$. Then the parameters of the optimization problem will simply become the state and controls at these times, $\boldsymbol{x}_i = \boldsymbol{x}(t_i)$ and $\boldsymbol{u}_i = \boldsymbol{u}(t_i)$ for $i = 0, \ldots, N$.

Rewriting the original OCP (4.1) as a function of the discrete set of parameters t_i, $\boldsymbol{x}_i$, and $\boldsymbol{u}_i$ will require modifications to both the constraints and to the cost function. First, the recursive formula (4.9) is used to replace the dynamics constraint $\dot{\boldsymbol{x}} = a(\boldsymbol{x}, \boldsymbol{u}, t)$ in the OCP[8]. Updating the cost function is going to require a numerical approximation of the integral, such as by using one of the

[8] The original dynamics model $\dot{\boldsymbol{x}} = a(\boldsymbol{x}, \boldsymbol{u}, t)$ is sometimes called the *continuous time* model and the recursive formula $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + ha(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i)$ is called the *discrete time* model.

Newton-Cotes formulas. The simplest of which would yield the approximation:

$$\int_{t_0}^{t_f} g(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \approx \sum_{i=0}^{N-1} h_i g(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i).$$

The OCP (4.1) can now be expressed completely as the finite-dimensional nonlinear program (NLP):

$$\begin{aligned} \underset{\boldsymbol{u}_i, \boldsymbol{x}_i}{\text{minimize}} \; & h(\boldsymbol{x}_N, t_N) + \sum_{i=0}^{N-1} h_i g(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i), \\ \text{s.t. } & \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h_i a(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i), \quad i = 0, \ldots, N-1, \\ & \boldsymbol{x}_0 = \boldsymbol{x}(t_0). \end{aligned} \tag{4.10}$$

4.3 *Consistency of Time Discretization*

The finite-dimensional problem (4.10) is only an *approximation* of the original problem (4.1), so it is important to justify that this approximation method is *consistent* with the original problem. This is accomplished by taking a look at the necessary optimality conditions for the NLP (4.10) and comparing them to the necessary optimality conditions for the original OCP (4.1).

Recall that the necessary conditions of optimality for equality-constrained finite-dimensional optimization problems have previously been discussed in Section 4.1.1. In particular, the Lagrangian is first formulated, which for (4.10) takes the form:

$$L = h(\boldsymbol{x}_N, t_N) + \sum_{i=0}^{N-1} h_i g(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) + \sum_{i=0}^{N-1} \boldsymbol{\lambda}_i^\top (\boldsymbol{x}_i + h_i a(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) - \boldsymbol{x}_{i+1}).$$

Note that even though the initial condition constraint is included in (4.10) it can be ignored in the Lagrangian by simply assuming $\boldsymbol{x}_0$ is not actually a decision variable in the optimization problem (since it is fixed). The NOCs are then given by:

$$\begin{aligned} \nabla_{\boldsymbol{x}_i} L = h_i \frac{\partial g}{\partial \boldsymbol{x}}(\boldsymbol{x}_i, \boldsymbol{u}_i) + h_i \left(\frac{\partial a}{\partial \boldsymbol{x}}(\boldsymbol{x}_i, \boldsymbol{u}_i)\right)^\top \boldsymbol{\lambda}_i + (\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i-1}) = 0, \quad & i = 1, \ldots, N-1 \\ \nabla_{\boldsymbol{x}_N} L = \frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}_N) - \boldsymbol{\lambda}_{N-1} = 0, & \\ \nabla_{\boldsymbol{u}_i} L = h_i \frac{\partial g}{\partial \boldsymbol{u}}(\boldsymbol{x}_i, \boldsymbol{u}_i) + h_i \left(\frac{\partial a}{\partial \boldsymbol{u}}(\boldsymbol{x}_i, \boldsymbol{u}_i)\right)^\top \boldsymbol{\lambda}_i = 0, \quad & i = 0, \ldots, N-1 \\ \boldsymbol{x}_i + h_i a(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) - \boldsymbol{x}_{i+1} = 0, \quad & i = 0, \ldots, N-1 \end{aligned} \tag{4.11}$$

Now, from the indirect method with equations (4.5), (4.6), and boundary conditions (4.7) with fixed final time and free final state, the NOCs for the infinite-

dimensional OCP can be written as:

$$\begin{aligned}
\frac{\partial g}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),\boldsymbol{u}(t)) + \big(\frac{\partial a}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),\boldsymbol{u}(t))\big)^\top \boldsymbol{p}(t) + \dot{\boldsymbol{p}}(t) &= 0, \quad t \in [t_0, t_f] \\
\frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}(t_f)) - \boldsymbol{p}(t_f) &= 0, \\
\frac{\partial g}{\partial \boldsymbol{u}}(\boldsymbol{x}(t),\boldsymbol{u}(t)) + \big(\frac{\partial a}{\partial \boldsymbol{u}}(\boldsymbol{x}(t),\boldsymbol{u}(t))\big)^\top \boldsymbol{p}(t) &= 0, \quad t \in [t_0, t_f] \\
\dot{\boldsymbol{x}}(t) - a(\boldsymbol{x}(t),\boldsymbol{u}(t),t) &= 0, \quad t \in [t_0, t_f] \\
\boldsymbol{x}_0 - \boldsymbol{x}(t_0) &= 0.
\end{aligned} \tag{4.12}$$

The NOCs (4.11) for the discretized problem and the NOCs for the original OCP (4.12) are remarkably similar. In fact, the NOCs (4.11) can be seen as themselves simply the discretized versions of (4.12). To see this, simply perform a forward Euler discretization of the equations in (4.12) with:

$$\begin{aligned}
\dot{\boldsymbol{p}}(t) = \frac{\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i-1}}{h_i}, \quad \boldsymbol{p}(t_i) = \boldsymbol{\lambda}_i, \quad i = 0, \dots, N-1, \\
\dot{\boldsymbol{x}}(t) = \frac{\boldsymbol{x}_{i+1} - \boldsymbol{x}_i}{h_i}, \quad \boldsymbol{x}(t_i) = \boldsymbol{x}_i, \quad \boldsymbol{u}(t_i) = \boldsymbol{u}_i, \quad i = 0, \dots, N-1.
\end{aligned}$$

Therefore, as the time step $h_i \to 0$ the NOCs for the discretized (direct method) problem converge to the NOCs derived directly for the original infinite-dimensional OCP (indirect method)!

4.4 Exercises

4.4.1 Optimal Control and Trajectory Optimization

Complete *Extra Problem: Optimal Control and Trajectory Optimization* located in the online repository:

`https://github.com/PrinciplesofRobotAutonomy/AA274A_HW1`,

where you will compute a dynamically feasible and *optimal* trajectory for a unicycle robot by using an indirect method to set up the necessary optimality conditions and solve them using a two-point boundary value solver.