AA203: Optimal and Learning-based Control Course Notes

James Harrison*

May 16, 2019

5 Direct Methods for Optimal Control

In the previous section we considered indirect methods to optimal control, in which the necessary conditions for optimality were first applied, yielding a two-point boundary value problem that was solved numerically. We will now consider the class of direct methods, in which the optimal control problem is first discretized, and then the resulting discrete optimization problem is solved numerically.

5.1 Direct Methods

We will write our original continuous optimal control problem,

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \int_0^{t_f} c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t), t \in [0, t_f] \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ & \boldsymbol{x}(t_f) \in \mathcal{M}_f \\ & \boldsymbol{u}(t) \in \mathcal{U}, t \in [0, t_f] \end{aligned} \tag{1}$$

where $\mathcal{M}_f = \{\boldsymbol{x} \in \mathbb{R}^n : F(\boldsymbol{x}) = 0\}$ and where we have, for simplicity, assumed zero terminal cost and $t_0 = 0$. We will use forward Euler discretization of the dynamics. We select a discretization $0 = t_0 < t_1 < \ldots < t_N = t_f$ for the interval $[0, t_f]$, and we will write $\boldsymbol{x}_{i+1} \approx \boldsymbol{x}(t), \boldsymbol{u}_i \approx \boldsymbol{u}(t)$ for $t \in [t_i, t_{i+1}]$, and $\boldsymbol{x}_0 \approx \boldsymbol{x}(0)$. Denoting $h_i = t_{i+1} - t_i$, the continuous time optimal control problem is transcibed into the nonlinear constrained

*Contact: jharrison@stanford.edu

optimization problem

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u}} \quad & \sum_{i=0}^{N-1} h_i c(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) \\ \text{s.t.} \quad & \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h_i f(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i), i = 0, \dots, N-1 \\ & \boldsymbol{x}_N \in \mathcal{M}_f \\ & \boldsymbol{u}_i \in \mathcal{U}, i = 0, \dots, N-1 \end{aligned} \tag{2}$$

5.1.1 Consistency of Time Discretization

Having performed this discretization, a reasonable (and important) sanity check on the validity of the direct approach is whether we recover the original problem in the limit of $h_i \to 0$. For simplicity, we will drop the time-dependence of the cost and dynamics. We will write the Lagrangian for (2) as

$$\mathcal{L} = \sum_{i=0}^{N-1} h_i c(\boldsymbol{x}_i, \boldsymbol{u}_i) + \sum_{i=0}^{N-1} \boldsymbol{\lambda}_i^T (\boldsymbol{x}_i + h_i f(\boldsymbol{x}_i, \boldsymbol{u}_i) - \boldsymbol{x}_{i+1}). \tag{3}$$

Then, the KKT conditions are

$$0 = h_i \frac{\partial c}{\partial \boldsymbol{x}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i) + \boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i-1} + h_i \frac{\partial f}{\partial \boldsymbol{x}_i}^T (\boldsymbol{x}_i, \boldsymbol{u}_i) \boldsymbol{\lambda}_i \tag{4}$$

$$0 = h_i \frac{\partial c}{\partial \boldsymbol{u}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i) + h_i \frac{\partial f}{\partial \boldsymbol{u}_i}^T (\boldsymbol{x}_i, \boldsymbol{u}_i) \boldsymbol{\lambda}_i \tag{5}$$

Rearranging, we have

$$\frac{\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i-1}}{h_i} = -\frac{\partial f}{\partial \boldsymbol{x}_i}^T (\boldsymbol{x}_i, \boldsymbol{u}_i) \boldsymbol{\lambda}_i - \frac{\partial c}{\partial \boldsymbol{x}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i) \tag{6}$$

$$0 = \frac{\partial f}{\partial \boldsymbol{u}_i}^T (\boldsymbol{x}_i, \boldsymbol{u}_i) \boldsymbol{\lambda}_i + \frac{\partial c}{\partial \boldsymbol{u}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i). \tag{7}$$

Let $\boldsymbol{p}(t) = \boldsymbol{\lambda}_i$ for $t \in [t_i, t_{i+1}], i = 0, \dots, N-1$ and $p(0) = \lambda_0$. Then, the above are direct discretizations of the necessary conditions for (18),

$$\dot{\boldsymbol{p}}(t) = -\frac{\partial f}{\partial \boldsymbol{x}}^T (\boldsymbol{x}(t), \boldsymbol{u}(t)) \boldsymbol{p}_i - \frac{\partial c}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{8}$$

$$0 = \frac{\partial f}{\partial \boldsymbol{u}}^T (\boldsymbol{x}(t), \boldsymbol{u}(t)) \boldsymbol{p}(t) + \frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{x}(t), \boldsymbol{u}(t)). \tag{9}$$

5.2 Transcription Methods

A fundamental choice in the design of numerical algorithms for direct optimization of the discretized optimal control problem is whether to optimize over the state and action variables (a method known as collocation or simultaneous optimization) or strictly over the action variables (known as shooting).

5.2.1 Collocation Methods

Collocation methods optimize both the state variables and the control input at a fixed, finite number of times, $t_0, \ldots, t_i, \ldots, t_N$. Moreover, the dynamics constraints are enforced at these points. As such, it is necessary to choose a finite-dimensional representation of the trajectory between these points. This rough outline leaves unspecified a large number of algorithmic design choices.

First, how are the dynamics constraints enforced? Both derivative and integral constraints exist. The derivative approach enforces that the derivative of the state with respect to time of the parameterized trajectory is equal to the given system dynamics. The integral approach relies on integrating the given dynamics and enforcing agreement between this and the trajectory parameterization. In these notes, we will focus on the derivative approach.

Second, a choice of trajectory parameterization is required. We will primarily discuss Hermite-Simpson methods in herein, which parameterize each subinterval of the trajectory (in $[t_i, t_{i+1}]$) with a cubic polynomial. Note that the choice of a polynomial results in integral and derivative constraints being relatively simple to evaluate. However, a wide variety of parameterizations exist. For example, pseudospectral methods represent the entire trajectory as a single high-order polynomial.

We will now outline the Hermite-Simpson method as one example of direct collocation. Having selected a discretization $0 = t_0 < t_1 < \ldots < t_N = t_f$, we denote $h_i = t_{i+1} - t_i$. In every subinterval $[t_i, t_{i+1}]$, we approximate $\boldsymbol{x}(t)$ with a cubic polynomial

$$\boldsymbol{x}(t) = \boldsymbol{c}_0^i + \boldsymbol{c}_1^i(t - t_i) + \boldsymbol{c}_2^i(t - t_i)^2 + \boldsymbol{c}_3^i(t - t_i)^3 \tag{10}$$

which yields derivative

$$\boldsymbol{x}(t) = \boldsymbol{c}_1^i + 2\boldsymbol{c}_2^i(t - t_i) + 3\boldsymbol{c}_3^i(t - t_i)^2. \tag{11}$$

Writing $\boldsymbol{x}_i = \boldsymbol{x}(t_i), \boldsymbol{x}_{i+1} = \boldsymbol{x}(t_{i+1}), \dot{\boldsymbol{x}}_i = \dot{\boldsymbol{x}}(t_i), \dot{\boldsymbol{x}}_{i+1} = \dot{\boldsymbol{x}}(t_{i+1})$, we may write

$$\begin{bmatrix} \boldsymbol{x}_i \\ \dot{\boldsymbol{x}}_i \\ \boldsymbol{x}_{i+1} \\ \dot{\boldsymbol{x}}_{i+1} \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ I & h_i I & h_i^2 I & h_i^3 I \\ 0 & I & 2h_i I & 3h_i^2 I \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_0^i \\ \boldsymbol{c}_1^i \\ \boldsymbol{c}_2^i \\ \boldsymbol{c}_3^i \end{bmatrix} \tag{12}$$

which in turn results in

$$\begin{bmatrix} \boldsymbol{c}_0^i \\ \boldsymbol{c}_1^i \\ \boldsymbol{c}_2^i \\ \boldsymbol{c}_3^i \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ -\frac{3}{h_i^2} I & -\frac{2}{h_i} I & \frac{3}{h_i^2} I & -\frac{1}{h_i} I \\ \frac{2}{h_i^2} I & \frac{1}{h_i^2} I & -\frac{2}{h_i^3} I & \frac{1}{h_i^2} I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_i \\ \dot{\boldsymbol{x}}_i \\ \boldsymbol{x}_{i+1} \\ \dot{\boldsymbol{x}}_{i+1} \end{bmatrix}. \tag{13}$$

Choosing intermediate times $t_i^c = t_i + \frac{h_i}{2}$ (collocation points), we can define interpolated

controls $\boldsymbol{u}_i^c = \frac{\boldsymbol{u}_i + \boldsymbol{u}_{i+1}}{2}$. From the above, we have

$$\boldsymbol{x}_i^c := \boldsymbol{x}(t_i + \frac{h_i}{2}) = \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_{i+1}) + \frac{h_i}{8}(f(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) - f(\boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1}, t_{i+1})) \tag{14}$$

$$\dot{\boldsymbol{x}}_i^c := \dot{\boldsymbol{x}}(t_i + \frac{h_i}{2}) = -\frac{3}{2h_i}(\boldsymbol{x}_i + \boldsymbol{x}_{i+1}) - \frac{1}{4}(f(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) + f(\boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1}, t_{i+1})). \tag{15}$$

Thus, we can write our discretized problem as

$$\begin{aligned} \min_{\boldsymbol{u}_{0:N-1}, \boldsymbol{x}_{0:N}} \quad & \sum_{i=0}^{N-1} h_i c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}_i^c - f(\boldsymbol{x}_i^c, \boldsymbol{u}_i^c, t_i^c) = 0, i = 0, \ldots, N-1 \\ & F(\boldsymbol{x}_N) = 0 \\ & \boldsymbol{u}(t) \in \mathcal{U}, i = 0, \ldots, N-1 \end{aligned} \tag{16}$$

5.2.2 Shooting Methods

Shooting methods solve the discrete optimization problem via optimizing only over the control inputs, and integrating the dynamics forward given these controls. A simple approach to the forward integration is the approach we have discussed above, in which forward Euler integration is used. Single-shooting methods directly optimize the controls for the entire problem. These approaches are fairly efficient for low dimension, short horizon problems, but typically struggle to scale to larger problems. Multiple shooting methods, on the other hand, optimize via shooting over subcomponents of the problem, and enforce agreement between the trajectory segments generated via shooting within each subproblem. These methods are therefore a combination of shooting methods and collocation methods. Generally, numerical solvers for shooting problems will, given an initial action sequence, linearize the trajectory and optimize the objective function with respect to those linearized dynamics to obtain new control inputs.

5.2.3 Sequential Convex Programming

Direct optimization of the discretized nonlinear control problem typically results in a non-convex optimization problem, for which finding a good solution may be difficult or impossible. The source of this non-convexity is typically the dynamics (and sometimes the cost function). The key idea of sequential convex programming (SCP) is to iterative re-linearize the dynamics (and construct a convex approximation of the cost function, if it is non-convex) around a nominal trajectory.

First, we will assume for this outline that the cost c is convex. Let $(\boldsymbol{x}_0(\cdot), \boldsymbol{u}_0(\cdot))$ be a nominal tuple of trajectory and control (which is not necessarily feasible). We linearize the dynamics around this trajectory:

$$f_1(\boldsymbol{x}, \boldsymbol{u}, t) = f(\boldsymbol{x}_0(t), \boldsymbol{u}_0(t), t) + \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_0(t), \boldsymbol{u}_0(t), t)(\boldsymbol{x} - \boldsymbol{x}_0(t)) + \frac{\partial f}{\partial \boldsymbol{u}}(\boldsymbol{x}_0(t), \boldsymbol{u}_0(t), t)(\boldsymbol{u} - \boldsymbol{u}_0(t)). \tag{17}$$

We can then solve the linear optimal control problem (with $k = 0$, initially),

$$\begin{aligned} \min \quad & \int_0^{t_f} c(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = f_{k+1}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), t \in [0, t_f] \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 \qquad\qquad\qquad\qquad \boldsymbol{x}(t_f) = \boldsymbol{x}_f \\ & \boldsymbol{u}(t) \in \mathcal{U}, t \in [0, t_f] \end{aligned} \tag{18}$$

where the dynamics are linear and the cost function is quadratic. Discretizing this continuous control problem yields a tractable convex optimization problem with dynamics $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h_i f(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i), i = 0, \ldots, N-1$. We then iterate this procedure until convergence is achieved with the new trajectory.

5.3 Further Reading

A broad introduction to direct methods for trajectory optimization is presented in [Kel17b]. This tutorial also features a discussion of trajectory optimization for hybrid systems, which we have not discussed in this section, as well as numerical solver features. For a more comprehensive review of direct methods for trajectory optimization by the same author with an emphasis on collocation methods, see [Kel17a].

References

[Kel17a] Matthew Kelly. An introduction to trajectory optimization: how to do your own direct collocation. *SIAM Review*, 2017.

[Kel17b] Matthew Kelly. Transcription methods for trajectory optimization: a beginners tutorial. *arXiv:1707.00284*, 2017.