Optimal Control, Lecture 11: Numerical Solutions

Anders Hansson

Division of Automatic Control
Linkoping University

Contents

- Gradient method
- Shooting method
- Discretization method
- Collocation method
- Multiple shooting method

Two Categories of Methods

We must often resort to numerical solutions when solving optimal control problems.

1. **Indirect methods:** Solving the necessary conditions of PMP by integrating the differential equations in a recursive manner.
2. **Direct methods:** Solving a discretization of the optimal control problem directly.

Indirect Methods

1. **Shooting method:** The TPBVP is solved by integrating the dynamical equations forward in time using an ODE solver.
2. **Gradient method:** The dynamical equation relating the control signal and the state is integrated forward in time, and the adjoint equation is integrated backward in time, using an ODE solver.

Direct Methods

All methods based on approximating the control signal as, *e.g.*, piecewise constant or as a polynomial.

1. **Discretization method:** Euler forward difference for approximating state trajectory.
2. **Collocation method:** Use polynomial approximations to approximate the state trajectory.
3. **Multiple shooting method:** Explicitly make use of an ODE solver to compute the state trajectory.

All methods rely on an efficient nonlinear programming solver at the top level to optimize the control signal parameters.

Numerical Algorithms Needed

- Integration of ODEs, e.g. `ode45`
- Finding roots of systems of nonlinear equations, e.g. `fsolve`
- Solving finite-dimensional optimization problems, e.g. `fmincon`

The Gradient Method

$$\begin{aligned} \text{minimize} \quad & \phi\left(x(T)\right) + \int_0^T f\left(t, x(t), u(t)\right) dt \\ \text{subject to} \quad & \dot{x}(t) = F\left(t, x(t), u(t)\right) \end{aligned} \tag{1}$$

It can be shown that the gradient of the objective function with respect to $u(t)$ is given by

$$\frac{\partial H(t, x(t), u(t), \lambda(t))}{\partial u}$$

for any $x(t)$ as long as $\lambda(t)$ satisfies the adjoint equation

$$\dot{\lambda}(t) = -\frac{\partial H(t, x^\star(t), u^\star(t), \lambda(t))}{\partial x}, \qquad \lambda(T) = \frac{\partial \phi(x^\star(T))}{\partial x}$$

Algorithm for Gradient Method

1. Guess an initial value for the control signal $u(t)$, $t \in [0, T]$.
2. Solve
$$\dot{x}(t) = F\left(t, x(t), u(t)\right), \quad x(0) = x_0$$
using an ODE solver.
3. Solve
$$\dot{\lambda}(t) = -\frac{\partial H(t, x(t), u(t), \lambda(t))}{\partial x}, \qquad \lambda(T) = \frac{\partial \phi(x(T))}{\partial x}$$
using an ODE solver.
4. Update $u(t)$ as
$$u(t) \leftarrow u(t) - \alpha \frac{\partial H(t, x(t), u(t), \lambda(t))}{\partial u}$$
5. Repeat steps 2–4 until
$$\int_0^T \left| \frac{\partial H(t, x(t), u(t), \lambda(t))}{\partial u} \right|^2 dt$$
is small enough.

Pros and cons

- $(+)$ It gives good improvement in the first iterations.
- $(+)$ Stability is good since integration of x and λ performed in stable directions.
- $(+)$ Control constraints can be taken into account by projecting onto the control constraint set.
- $(+)$ It was used to solve a large number of aeronautical problems in the 1960s.
- $(-)$ The step size α has to be chosen with care.
- $(-)$ Convergence tends to be slow.

The Shooting Method

Solves TPBV problem resulting from PMP:

$$\dot{x} = F(x, \mu(x, \tilde{\lambda})), \quad x(0) \in S_0,\ x(T) \in S_T$$

$$\dot{\lambda} = -\frac{\partial H(x, \mu(x, \tilde{\lambda}), \tilde{\lambda})}{\partial x}, \quad \lambda(0) \perp S_0,\ \lambda(T) - \frac{\partial \phi(x(T))}{\partial x} \perp S_T \tag{2}$$

The Shooting Method ctd.

Assume we can summarize what we know about the initial and final values in nonlinear equations

$$G_0(x(0), \lambda(0)) = 0, \quad G_T(x(T), \lambda(T)) = 0$$

Define $G : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n \times \mathbf{R}^n$ that takes as input the initial values $(x(0), \lambda(0))$, integrates the differential equations, and outputs the final values $(x(T), \lambda(T))$. (Can be implemented in Matlab using an ODE solver like `ode45`)

Solve

$$\begin{aligned} G_0(x(0), \lambda(0)) &= 0 \\ G_T(G(x(0), \lambda(0))) &= 0. \end{aligned}$$

using e.g. `fsolve`

Minimization of Hamiltonian

In case minimum of Hamiltonian obtained when partial derivative is zero, then this equation can be added to the differential equations in (2), *i.e.*, we consider

$$\begin{aligned}
&\dot{x} = F(x,u), \quad x(0) \in S_0,\ x(T) \in S_T \\
&\dot{\lambda} = -\frac{\partial H(x,u,\tilde{\lambda})}{\partial x}, \quad \lambda(0) \perp S_0,\ \lambda(T) - \frac{\partial \phi(x(T))}{\partial x} \perp S_T \\
&\frac{\partial H\left(x,u,\tilde{\lambda}\right)}{\partial u} = 0
\end{aligned} \tag{3}$$

instead of (2) when we define the function G. The equation above is not an ODE, but a *differential algebraic equation* (DAE) which in MATLAB can be solved with, *e.g.*, `ode15i`.

Pros and cons

- $(+)$ Conceptually simple. It was used to launch sattelites in the 1950s.
- $(+)$ Control constraints can be taken care of, but we need to carry out the minimization of the Hamiltonian at each step of the ODE/DAE solver, and this would require us to write special purpose code.
- $(-)$ It can be crucial to find a good initial estimate of $\lambda(0)$.
- $(-)$ Integration of the differential equations can be severely unstable.

The Discretization Method

$$
\begin{aligned}
\text{minimize} \quad & \phi\left(x(T)\right) + \int_0^T f\left(t, x(t), u(t)\right) dt \\
\text{subject to} \quad & \dot{x}(t) = F\left(t, x(t), u(t)\right) \\
& x(0) \in S_0, \quad x(T) \in S_T \\
& u(t) \in U \subset \mathbf{R}^m \\
& T \geq 0
\end{aligned}
\tag{4}
$$

Approximations

Define a partitioning of time interval $[0, T]$ as

$$0 = t_0 \leq t_1 \leq \cdots \leq t_N = T$$

in N intervals $[t_i, t_{i+1}]$, and let $h_i = t_{i+1} - t_i$. Often $h_i = T/N$.

Approximate $u(t)$ for $t \in [t_i, t_{i+1}]$ as $u(t) = u_i \in \mathbf{R}^m$.

Approximate the derivative of the state with

$$\dot{x}(t) \approx \frac{x(t_{i+1}) - x(t_i)}{h_i}, \quad t \in [t_i, t_{i+1}].$$

resulting with $x_i = x(t_i)$ in

$$x_{i+1} = x_i + h_i F(t_i, x_i, u_i),$$

Approximate objective function as

$$\phi(x_N) + \sum_{i=0}^{N-1} h_i f(t_i, x_i, u_i).$$

Approximate Discrete-Time Optimal Control Problem

$$
\begin{array}{ll}
\text{minimize} & \phi(x_N) + \displaystyle\sum_{i=0}^{N-1} h_i f(t_i, x_i, u_i) \\
\text{subject to} & x_{i+1} = x_i + h_i F(t_i, x_i, u_i), \quad i \in \mathbf{Z}_{N-1} \\
& x_0 \in S_0, \quad x_N \in S_T \\
& u_i \in U \subset \mathbf{R}^m.
\end{array}
$$

with variables $x = (x_0, \ldots, x_N)$, and $u = (u_0, \ldots, u_{N-1})$.

Finite Dimensional Nonlinear Program (NLP)

Define $\mathcal{F}_0(x,u) = \phi(x_N) + \sum_{i=0}^{N-1} h_i f(t_i, x_i, u_i)$.

Assume $u \in U \times U \times \cdots \times U$ can be described as $\mathcal{F}(u) \preceq 0$.

Assume $x_0 \in S_0$ and $x_N \in S_T$ can be described as

$$G_0(x_0) = 0, \quad G_T(x_N) = 0$$

Let

$$\mathcal{H}(x,u) = \begin{bmatrix} x_1 - x_0 - h_0 F(t_0.x_0, u_0) \\ \vdots \\ x_N - x_{N-1} - h_{N-1} F(t_{N-1}, x_{N-1}, u_{N-1}) \\ G_0(x_0) \\ G_T(x_N) \end{bmatrix}.$$

Then with variables x and u

$$\begin{array}{ll} \text{minimize} & \mathcal{F}_0(x,u) \\ \text{subject to} & \mathcal{F}(u) \preceq 0, \quad \mathcal{H}(x,u) = 0 \end{array}$$

Pros and cons

- $(+)$ Exists many good software packages for nonlinear programming.
- $(+)$ Lot of structure in NLP that can be utilized .
  - The objective and constraint derivatives in the NLP will be sparse and have block structure.
  - Can use Riccati recursions to compute search directions.
- $(+)$ More sophisticated discretizations can be used, e.g. Runge-Kutta.
- $(-)$ There are many variables and constraints in the NLP.
- $(-)$ The approximate solution may not converge to the true solution as $h \to 0$.
- $(-)$ Physical insight we have in continuous time formulation might get lost.

The Multiple Shooting Method—The Control Signal

Use similar ideas as in the discretization method, but more accurate approximations.

Let $\varphi_i : \mathbf{R} \times \mathbf{R}^{k_i} \to \mathbf{R}^m$ and

$$u(t) = \varphi_i(t, a_i), \quad t \in [t_i, t_{i+1}].$$

The vector a_i parameterizes the function, and these vectors will be optimization variables.

We collect them in the vector $a \in \mathbf{R}^k$, where $k = \sum_{i=0}^{N-1} k_i$.

The functions φ_i could be constants as in the previous subsection, and then $a_i = u_i$, but we can also define affine functions, general polynomials or orthogonal basis functions.

The Shooting

We define initial values s_i, $i \in \mathbf{Z}_{N-1}$, and we then use an ODE solver to integrate the N differential equations

$$\dot{x}(t) = F(t, x(t), \varphi_i(t, a_i)), \quad x(t_i) = s_i, \quad i \in \mathbf{Z}_{N-1}$$

over the intervals $[t_i, t_{i+1}]$.

We denote with abuse of notation the solutions by $x(t, a_i, s_i)$ for $t \in [t_i, t_{i+1}]$.

In order to ensure continuity of the solutions we introduce matching conditions

$$h(s_i, a_i, s_{i+1}) = s_{i+1} - x(t_{i+1}, a_i, s_i) = 0.$$

Objective Function

We also use an ODE solver to solve the differential equations

$$\dot{J}(t) = f(t, x(t, a_i, s_i), \varphi_i(t, a_i)), \quad J(t_i) = 0$$

over the intervals $[t_i, t_{i+1}]$. We denote with an abuse of notation the solutions by $J_i(t, s_i, a_i)$ for $t \in [t_i, t_{i+1}]$.

Objective function in (4) approximated as

$$\mathcal{F}_0(s, a) = \phi(s_N) + \sum_{i=1}^{N-1} J_i(t_{i+1}, s_i, a_i)$$

Constraints

The constraint $u(t) \in U$ for all $t \in [0, T]$ is sampled at each time t_i, and expressed as $\mathcal{F}^i(a_i) \preceq 0$ and collected in

$$\mathcal{F}(a) = \begin{bmatrix} \mathcal{F}^0(a_0) \\ \vdots \\ \mathcal{F}^{N-1}(a_{N-1}) \end{bmatrix}. \tag{5}$$

As for previous method assume that the functions G_0 and G_T can be used to describe the constraints on initial state and final state. All equality constraints can be described using the function

$$\mathcal{H}(s, a) = \begin{bmatrix} h(s_0, a_0, s_1) \\ h(s_1, a_1, s_2) \\ \vdots \\ h(s_{N-1}, a_{N-1}, s_N) \\ G_0(s_0) \\ G_T(s_N) \end{bmatrix}.$$

Finite Dimensional NLP

$$\begin{array}{rl} \text{minimize} & \mathcal{F}_0(s,a) \\ \text{subject to} & \mathcal{F}(a) \preceq 0 \\ & \mathcal{H}(s,a) = 0 \end{array}$$

with variables s and a.

Pros and Cons

- $(+)$ The above optimization problem is also a finite dimensional optimization problem.
- $(-)$ Here we are not able to compute analytical derivatives of the functions defining the optimization problem.
- $(+)$ However, often ODE solvers can deliver also derivatives of the solutions with respect to (a, s).
- $(+)$ The fact that the differential equations can be solved in parallel can be used to speed up the solver.
- $(+)$ Straightforward to generalize to the case when there are inequality constraints related to the states x.

The Collocation Method

- Consider the optimal control problem in (4).
- Merge ideas from discretization method and from multiple shooting method.
- Same approximation of the control signal as in multiple shooting method.
- Numerical integration of differential equation is replaced by representing state as a cubic polynomial, which is more accurate than using forward Euler approximation of discretization method

State Approximation

Let

$$x_a(t) = \sum_{k=0}^{3} c_k^i \left(\frac{t - t_i}{h_i} \right)^k$$

for $t \in [t_i, t_{i+1}]$, where $c_k^i \in \mathbf{R}^n$ are coefficients of the vector valued polynomial.

Polynomial has to agree with the true state $x(t)$ and its derivative $\dot{x}(t)$ at the endpoints of the interval:

$$\begin{aligned} x_a(t_i) &= x_i \\ x_a(t_i + 1) &= x_{i+1} \\ \dot{x}_a(t_i) &= F(t_i, x_i, u_i) \\ \dot{x}_a(t_{i+1}) &= F(t_{i+1}, x_{i+1}, u_{i+1}), \end{aligned}$$

where

$$\dot{x}_a(t) = \sum_{k=1}^{3} \frac{k c_k^i}{h_i} \left(\frac{t - t_i}{h_i} \right)^{k-1}.$$

Solution

$$
\begin{aligned}
c_0^i &= x_i \\
c_1^i &= h_i F_i \\
c_2^i &= -3x_i - 2h_i F_i + 3x_{i+1} - h_i F_{i+1} \\
c_3^i &= 2x_i + h_i F_i - 2x_{i+1} + h_i F_{i+1},
\end{aligned}
$$

where $F_i = F(t_i, x_i, u_i)$, and where $x_i = x(t_i)$, $u_i = u(t_i) = \varphi_i(t_i, a_i)$ and $h_i = t_{i+1} - t_i$ as before.

The differential equation is satisfied at the endpoints of the interval by construction of the coefficients of the polynomials.

Midpoint Constraints and Objective Function

Depending on dimension of a_i we can try to enforce the differential equation to hold at one or several points inside the interval $[t_i, t_{i+1}]$.

Here we use center point $t_i^c = (t_i + t_{i+1})/2$, which results in the constraints

$$\dot{x}_a(t_i^c) = F(t_i^c, x_a(t_i^c), \varphi_i(t_i^c, a_i)).$$

This constraint involves the variables x_i, x_{i+1} and a_i and is just a nonlinear equation in these variables.

Objective function is approximated using center points as

$$\phi(x_N) + \sum_{i=0}^{N-1} h_i f(t_i^c, x_a(t_i^c), \varphi_i(t_i^c, a_i)).$$

Discrete-Time Optimal Control Problem

$$
\begin{aligned}
\text{minimize} \quad & \phi(x_N) + \sum_{i=0}^{N-1} h_i f(t_i^c, x_a(t_i^c), \varphi_i(t_i^c, a_i)) \\
\text{subject to} \quad & \dot{x}_a(t_i^c) = F(t_i^c, x_a(t_i^c), \varphi_i(t_i^c, a_i)) \\
& G_0(x_0) = 0, \quad G_T(x_N) = 0 \\
& \varphi(t_i, a_i) \in U \subset \mathbf{R}^m
\end{aligned}
$$

with variables $x = (x_0, \ldots, x_N)$ and $a = (a_0, \ldots, a_{N-1})$, where we as before assume that the constraint involving the control signal can be written as inequalities involving a function as in (5).

The variables x and a are implicitly present in x_a and $\dot{x}_a$.

Pros and Cons

- $(+)$ The above optimization problem is also a finite dimensional optimization problem.
- $(+)$ Here we are able to compute analytical derivatives w.r.t. (x, a) of the functions defining the optimization problem.
- $(-)$ The cubical approximation of the state might be less accurate as compared to the numerical integration of the state performed in the multiple shooting method.
- $(+)$ More general collocation methods involving orthogonal polynomials may be used.

Software

There are dedicated solvers for optimal control problems like ACADO, and CasADi.