16-745 Optimal Control Lecture 7

Reid Graves

February 6, 2025

1 Last Time

- Regularization + Duality
- Merit Functions + Line Search
- Control History

2 Today

- Deterministic Optimal Control
- Pontryagin
- LQR

3 Deterministic Optimal Control

- Continuous Time:

$$\min_{x(t),u(t)} \quad J(x(t), u(t)) = \int_{t_0}^{t_f} l(x(t), u(t))dt + l_F(x(t_F))$$

  Cost function is J, Stage cost is $l(x(t), u(t))$ and is a function of the state and input trajectories: $x(t), u(t)$, the terminal cost is l_F

$$s.t. \quad \dot{x}(t) = f(x(t), u(t))$$

  $\dot{x}$ is dynamics constraint. Usually have other constraints too. This is the optimal control problem. x, u are trajectories.

- This is an "infinite-dimensional" optimization problem. The answer is a single trajectory input. Solution is not a feedback policy.
- Solutions are open-loop trajectories
- There are a handful of problems with analytical solutions- like LQR, but not many
- We will focus on the discrete-time setting

3.1 Discrete Time

$$\min_{x_{1:N}, u_{1:N-1}} \quad J(x_{1:N}, u_{1:N-1}) = \sum_{n=1}^{N-1} l(x_k, u_k) + l_F(x_N)$$
$$s.t. \quad x_{k+1} = f(x_k, u_k)$$
$$u_{min} \leq u_k \leq u_{max} \quad \forall\ k \text{ torque limits}$$
$$c(x_n) \geq 0 \quad \forall\ k \text{ obstacle/safety constraints}$$

- This is a finite-dimonsional problem
- Samples x_k, u_k are often called "knot points"
- Continuous $\rightarrow$ discrete uses integration (e.g. Runge-Kutta)
- Discrete $\rightarrow$ continuous uses interpolation- common to use cubic splines

3.2 Pontryagin's Minimum Principle

- Also called the "Maximum Principle" if you maximize a reward instead of a cost
- It is the First-Order Necessary conditions for a deterministic optimal control problem
- In discrete time, just a special case of KKT conditions
- Given:

$$\min_{x_{1:N}, u_{1:N-1}} \quad \sum_{k=1}^{N-1} l(x_k, u_k) + l_F(x_N)$$
$$s.t. \quad x_{k+1} = f(x_k, u_k)$$

- We can form the Lagrangian:

$$L = \sum_{k=1}^{N-1} l(x_k, u_k) + \lambda_{n+1}^T (f(x_k, u_k) - x_{k+1}) + l_F(x_N)$$

- This result is usually stated in terms of the "Hamiltonian"

$$H(x, u, \lambda) = l(x, u) + \lambda^T f(x, u)$$

- Plug it into L:

$$L = H(x_1, u_1, \lambda_2) + \left[\sum_{k=2}^{N-1} H(x_k, u_k, \lambda_{k+1} - \lambda_k^T x_k \right] + l_F(x_N) - \lambda_N^T x_N$$

- Take derivatives w.r.t. x, λ:

$$\frac{\partial L}{\partial \lambda_k} = \frac{\partial H}{\partial \lambda_k} - x_{k+1} = f(x_k, u_k) - x_{k+1} = 0$$
$$\frac{\partial L}{\partial x_k} = \frac{\partial H}{\partial x_k} - \lambda_k^T = \frac{\partial l}{\partial x_k} + \lambda_{k+1}^T \frac{\partial f}{\partial x_k} - \lambda_k^T = 0$$
$$\frac{\partial L}{\partial x_N} - \frac{\partial l_F}{\partial x_N} - \lambda_N^T = 0$$

- For u we write the min explicitly to handle torque limits:

$$
\begin{aligned}
u_k = \arg\min_u H(x_k, u, \lambda_{k+1}) \\
s.t. \quad u \in \mathcal{U} \quad \text{Shorthand for “in feasible set”}
\end{aligned}
$$

- In summary:

$$
\begin{aligned}
x_{k+1} &= \nabla_\lambda H(x_k, u_k, \lambda_{k+1}) \\
\lambda_k &= \nabla_x H(x_k, u_k, \lambda_{k+1}) \\
u_k &= \arg\min_u H(x_k, u, \lambda_{k+1}) \\
&\quad s.t. \quad u \in \mathcal{U} \\
\lambda_N &= \frac{\partial l_F}{\partial x_N}
\end{aligned}
$$

- These can be stated in continuous time:

$$
\begin{aligned}
\dot{x} &= \nabla_\lambda H(x, u, \lambda) \\
-\dot{\lambda} &= \nabla_x H(x, u, \lambda) \\
u &= \arg\min_u H(x, u, \lambda) \\
&\quad s.t. \quad u \in \mathcal{U} \\
\lambda(t_F) &= \frac{\partial l_F}{\partial x}
\end{aligned}
$$

3.3 Some Notes

- Historically many algorithms were based on integrating ODEs forward + backward to do gradient descent
- These are called “indirect” and/or “shooting” methods
- In continuous time $\lambda(t)$ is called “costate” trajectory
- These methods have largely fallen out of favor as computers have improved.

4 LQR Problem

Linear Quadratic Regulator (LQR)

$$
\begin{aligned}
\min_{x_{1:N}, u_{1:N-1}} \quad & J = \sum_{k=1}^{N-1} \frac{1}{2} x_k^T Q_k x_k + \frac{1}{2} u_k^T R_k u_k + \frac{1}{2} x_k^T Q_k x_k \\
s.t. \quad & x_{k+1} = A_k x_k + B_k u_k \\
Q \geq 0, \quad & R > 0
\end{aligned}
$$

- Can (locally) approximate many nonlinear problems
- Computationally Tractable
- Many extensions e.g. infinite horizon, stochastic, etc.
- “Time invariant” if $A_k = A$, $B_k = B$, $Q_k = Q$, $R_k = R$ $\forall k$. Time varying otherwise

4.1 LQR with Indirect Shooting:

$$
\begin{aligned}
x_{k+1} &= \nabla_\lambda H(x_k, u_k, \lambda_{k+1}) = Ax_k + Bu_k \\
\lambda_k &= \nabla_x H(x_k, u_k, \lambda_k) = Qx_k + A^T \lambda_{k+1} \\
\lambda_N &= Q_N x_N \\
u_k &= \nabla_u H(x_k, u_k, \lambda_{k+1}) = 0 \Rightarrow -R^{-1} B^T \lambda_{k+1}
\end{aligned}
$$

- Procedure
  1. Start with initial guess $u_{1:N-1}$
  2. Simulate/rollout to get $x_{1:N}$
  3. Backward pass to get $\lambda, \Delta u$
  4. Rollout with line search on Δu
  5. Go to 3. until convergence

4.2 Example

- "Double Integrator"

$$
\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
$$

- Think of this as a brick sliding on ice with no friction

$$
\begin{aligned}
x_{k+1} &= \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_k \\ \dot{q}_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}h^2 \\ h \end{bmatrix} u_k \\
x_{n+1} &= Ax + Bu
\end{aligned}
$$