LECTURE 3 : TRAJECTORY OPTIMISATION

27/10
28/10

$$\min_{x(\cdot)\,u(\cdot)} \quad \underbrace{\phi(x(T)) + \int_{t_0}^{T} l(t, x(t), u(t))\, dt}_{J}$$

Terminal cost — Running cost

Total accumulated cost

subject to

$$b(x(t_0), x(T)) = 0$$

boundary constraints

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

continuous dynamics

$$x_{min} \le x(t) \le x_{max}$$

$$u_{min} \le u(t) \le u_{max}$$

State and control bounds

$$g(t, x(t), u(t)) \le 0$$

path constraints

$$h(t, x(t), u(t)) = 0$$

path constraints

$$x(t_0) = x_0, \quad x(T) = x_T$$

Initial and final constraints

Three types of algorithms to solve this

Dynamic Programming	Indirect Methods	Direct methods
Solve HJB over the entire state space Analytic : LQR Numerical : Value iteration	Mathematical conditions that an optimum must satisfy (CoV / PMP)	Transcribe into finite-dimensional optimisation problem ① Shooting ② Transcription ③ Collocation

I Dynamic programming has been used to compute cost-to-go functions over some sequence of states. In CS, DP provides fundamental insight in the development of algorithms that compute optimal solutions to problems. In its original form, however, DP was developed to solve the optimal control problem. In this setting (optimal control) a discrete set of equations is replaced by a continuum of stages, known as 'time'. The dynamic programming recurrence is instead a PDE called the Hamilton-Jacobi Bellman (HJB) equation. The HJB equation is solved numerically, in some cases it can be solved analytically, eg. LQR.

Pontryagin's Minimum Principle, which is derived from DP principle, generalises optimisation performed in Hamiltonian mechanics.

HJB equation discrete case

Recall when we had X, U discrete, optimal planning can be performed by using Dijkstra on the search graph.

Let be the optimal cost to go function and the dynamics be

States S, actions a, time n

$$S[n+1] = f_d(S[n], a[n])$$

$$\sum_{n=0}^{\infty} l_d(S[n], a[n])$$

$$\forall S,\ J^*(S) = \min_a \left[l_d(S,a) + J^*(f_d(S,a)) \right]$$

$$\pi^*(S) = \arg\min_a [\ \uparrow\ ]$$

HJB equation continuous case

states x, actions u, time t

$$\dot{x}(t) = f_c(x(t), u(t))$$

$$\int_{t=0}^{\infty} dt\, l_c(x,u)$$

$$\pi^*(x) = \arg\min_u [\ ]$$

$$\forall x \quad 0 = \min_u \left[l(x,u) + \frac{\partial J^*}{\partial x} f_c(x,u) \right]$$

LQR special solution for HJB analytically

$\dot{x} = Ax + Bu$ with quadratic cost

$$J(x_0, u) = \int_0^{\infty} \underbrace{(x^T Q x + u^T R u)}_{l(x,u)}\, dt.$$

Optimal cost-to-go is $V(x_0) = \min_{u(\cdot)} J(x_0, u)$

Substituting into stationary HJB

$V(x_0) = x^T P x$

$$0 = \min_u \{ x^T Q x + u^T R u + \nabla V(x)^T (Ax + Bu) \}$$

Assume quadratic $V(x_0)$, substitute and differentiate wrt u,

$u^* = -R^{-1} B^T P x \Rightarrow$ back into HJB $A^T P + PA - PBR^{-1}B^T P + Q = 0$. (ARE)

II INDIRECT METHODS

PMP provides conditions that an optimal trajectory must satisfy.

Consider $\min_u \Phi(x(T)) + \int_0^T l(x(t), u(t))\, dt$

subject to $\dot{x} = f(x,u)$. Define the Hamiltonian as

$H(x,u,\lambda) = l(x,u) + \lambda^T f(x,u)$, $\lambda(t)$: auxiliary variable. For unconstrained control

$$\dot{x} = \frac{\partial H}{\partial \lambda} \qquad \dot{\lambda} = -\frac{\partial H}{\partial x} \qquad 0 = \frac{\partial H}{\partial u}$$

Necessary but not sufficient

The costate $\lambda(t)$ represents sensitivity of optimal cost-to-go to a change in the state along the optimal trajectory.

In HJB, $0 = \min_u \{ \ell(x,u) + \nabla V(x)^T f(x,u) \}$

In PMP, $H(x,u,\lambda) = \ell(x,u) + \lambda^T f(x,u)$

along an optimal trajectory $\lambda(t) = \nabla_x V(t, x^*(t))$

For finite horizon optimal control, value function depends on time $V(t,x)$

For stationary infinite horizon it reduces to $V(x)$

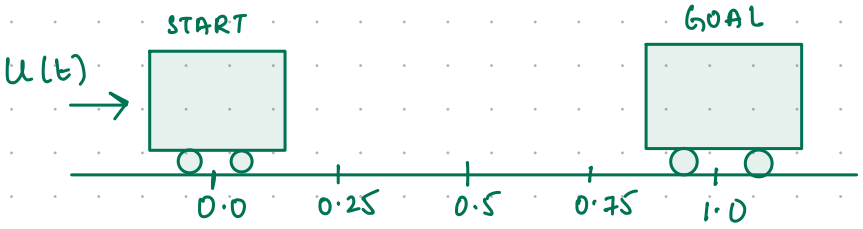
Solution: Minimise u^* by minimising the Hamiltonian

u^* as a solution to λ

Substitute u^* into $\dot{x}$ & $\dot{\lambda}$ equations

Imagine a robot moving along a straight line from rest at position 0 to position 1m

We are trying to break curse of dimensionality by limiting to only x(0) to x(T).



$p(0) = 0$ $\qquad$ $p(T) = 1$

$T = 2s$

Our states are $x(t) = \begin{pmatrix} p(t) \\ v(t) \end{pmatrix}$, $u(t) = a(t)$

The dynamics are $\dot{p}(t) = v(t)$

$\dot{v}(t) = u(t)$

In state space $\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$

$$\dot{x}(t) = A\, x(t) + B\, u(t)$$

Our goal is

→ Start from $p(0) = 0$, $v(0) = 0$

→ End at $p(2) = 1$, $v(2) = 0$

→ Use as little acceleration as possible

Continuous time optimal control problem

$$\min_{x(\cdot), u(\cdot)} \quad J = \frac{1}{2} \int_0^2 u(t)^2 \, dt$$

$$\text{s.t} \quad \dot{p}(t) = v(t)$$

$$\dot{v}(t) = u(t)$$

$$p(0) = 0, \quad v(0) = 0$$

$$p(2) = 1, \quad v(2) = 0$$

We will first derive optimal trajectory using PMP, then solve the same problem numerically using direct shooting, transcription and collocation.

OPTIMAL CONTROL PROBLEM
(Minimise cost subject to dynamics and constraints)

- DP / HJB (Optimal value fun^n)
- PMP (Optimality conditions)
- Direct methods (Numerical trajectory optimisa^n)

PMP :

$$H = \frac{1}{2}u^2 + \lambda_p v + \lambda_v u$$

The necessary conditions are

$$\dot{\lambda}_p = -\frac{\partial H}{\partial p} = 0 \qquad \dot{\lambda}_v = -\frac{\partial H}{\partial v} = -\lambda_p \qquad u + \lambda_v = 0$$

Since λ_p = constant, λ_v is first order in t

$$u^* = c_1 t + c_2$$

$$\Rightarrow \dot{v}^* = c_1 t + c_2 \qquad \Rightarrow p^* = c_1 \frac{t^3}{6} + c_2 \frac{t^2}{2}$$

$$v^* = c_1 \frac{t^2}{2} + c_2 t$$

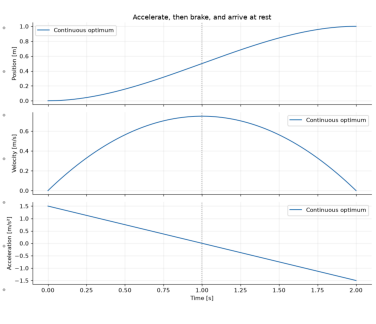
$$1 = c_1 \frac{8}{6} + c_2 \frac{4}{2}$$

$$0 = 2c_1 + 2c_2$$

$$6 = 8c_1 + 12c_2$$

$$6 = 4c_2 \Rightarrow c_2 = \frac{3}{2}$$

$$\Rightarrow c_1 = \frac{-3}{2}$$



$$p^*(t) = \frac{3}{4}t^2 - \frac{1}{4}t^3$$

$$v^*(t) = \frac{3}{2}t - \frac{3}{4}t^2$$

$$u^*(t) = \frac{3}{2} - \frac{3}{2}t$$

The minimum cost

$$J^* = \frac{1}{2}\int_0^2 [u^*(t)]^2 dt$$

$$= 0.75$$

III DIRECT METHODS : Transform a continuous-time optimal control problem into a finite-dimensional optimisation problem.

① Direct Transcription: Optimise state and controls simultaneo, subject to discretised dynamics.

② Direct Shooting: Optimise the control sequence and obtain the states through forward simulation

③ Direct Collocation: Approximate the state trajectory using polynomials and enforce the dynamics at selected points.

① Direct transcription

$$\min_{x(\cdot),u(\cdot)} \quad J = \phi(x(T)) + \int_0^T l(t,x(t),u(t))\,dt$$

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

$$x(0) = x_0 \;, \; x(T) = x_T$$

$$g(t,x(t),u(t)) \leq 0$$

→ Discretise time, divide $[0,T]$ into N segments

$$t_k = k\Delta t, \quad \Delta t = \frac{T}{N}$$

Define the discrete variables

$$x_k = x(t_k), \quad u_k = u(t_k)$$

→ Discretise the dynamics

for eg forward Euler : $x_{k+1} = x_k + \Delta t\, f(t_k, x_k, u_k)$

More generally we write

$$x_{k+1} = f_d(x_k, u_k)$$

→ Discretise the cost

$$J_n = \phi(x_N) + \Delta t \sum_{k=0}^{N-1} l(t_k, x_k, u_k)$$

$$\min_{\{x_k, u_k\}} \quad \phi(x_N) + \Delta t \sum_{k=0}^{N-1} l(t_k, x_k, u_k)$$

$$\text{s.t.} \quad x_{k+1} - f_d(x_k, u_k) = 0$$

$$x_0 = x_0, \quad x_N = x_T$$

$$g(t_k, x_k, u_k) \leq 0$$

Note the decision vector is $[x_0^T, \ldots, x_N^T, u_0^T, \ldots, u_{N-1}^T]$ for $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ there are $(N+1)n + Nm$ scalar variables.

② Direct shooting

In direct shooting, only the control trajectory is parameterised

→ Divide $[0, T]$ into N intervals

$$u(t) = u_k, \quad t \in [t_k, t_{k+1})$$

→ Integrate the dynamics

$$x_1 = f_d(x_0, u_0)$$

$$x_2 = f_d(x_1, u_1)$$

$$\vdots$$

$$x_N = f_d(x_{N-1}, u_{N-1})$$

Consequently $x_k = F_k(U, x_0)$

→ The cost is,

$$J_N(U) = \phi(F_N(U, x_0)) + \Delta t \sum_{k=0}^{N-1} l(t_k, F_k(U, x_0), u_k)$$

$$\min_U \quad J_N(U)$$

$$\text{s.t.} \quad F_N(U, x_0) = x_T$$

$$g(t_k, F_k(U, x_0), u_k) \leq 0$$

10.2.3 Computational Considerations

So is direct shooting uniformly better than the direct transcription approach? I think it is not. There are a few potential reason that one might prefer the direct transcription:

- Numerical conditioning. Shooting involves calculating $\mathbf{A}^n$ for potentially large n, which can lead to a large range of coefficient values in the constraints. This problem is sometimes referred to as the "tail wagging the dog", or in very deep neural networks as "vanishing gradients" and/or "exploding gradients". In trajectory optimization, it is not simply an artifact of the shooting transcription, but is a real property of the problem we have formulated: the control input $\mathbf{u}[0]$ really does have more opportunity to have a large impact on the total cost than control input $\mathbf{u}[N-1]$. The direct transcription approach combats the numerical issue by spreading this effect out over a large number of well-balanced constraints.
- Adding state constraints. Having $\mathbf{x}[n]$ as explicit decision variables makes it very easy/natural to add additional state constraints; and the solver effectively reuses the computation of $\mathbf{A}^n$ for each constraint. In shooting, one has to unroll those terms for each new constraint.
- Parallelization. For larger problems, evaluating the constraints can be a substantial cost. In direct transcription, one can evaluate the dynamics/constraints in parallel (because each iteration begins with $\mathbf{x}[n]$ already given), whereas shooting is more fundamentally a serial operation.

For linear convex problems, the solvers are mature enough that these differences often don't amount to much. For nonlinear optimization problems, the differences can be substantial. If you look at trajectory optimization papers in mainstream robotics, you will see that both direct transcription and direct shooting approaches are used. (It used to be that you could guess which research lab wrote a paper simply by the transcription they use!)

https://underactuated.mit.edu/trajopt.html

Informal derivation

$$x[n+1] \equiv x[n] + h\, f_c(x[n], u[n])$$ Euler integration

h: time step

$$l_d(x,u) \simeq h\, l_c(x,u)$$

Discrete cost to go: $$J^*(x) = \min_u \left[h\, l_c(x,u) + J^*(x + h\, f_c(x,u)) \right]$$

$$J^*(x + h\, f_c(x,u)) \approx J^*(x) + \frac{\partial J^*}{\partial x} h\, f_c(x,u)$$

$$J^*(x) = J^*(x) + \min_u \left[h\, l_c(x,u) + \frac{\partial J^*}{\partial x} h\, f_c(x,u) \right]$$

$$\begin{cases} 0 = \min_u \left[l_c(x,u) + \frac{\partial J^*}{\partial x} f_c(x,u) \right] \end{cases}$$

HAMILTON JACOBI BELLMAN EQUATION (HJB)

$J^*(x[n+1])$ where $x[n+1] = x[n] + h\, f_c(x[n], a[n])$

$$f(x) = f(x_0) + \left.\frac{\partial f}{\partial x}\right|_{x=x_0} (x - x_0)$$

For optimal policy u^*

$$0 = l_c(x, u^*) + \frac{\partial J^*}{\partial x} f_c(x, u^*)$$

$$\frac{\partial J^*}{\partial x} f_c(x, u^*) = -l_c(x, u^*)$$

$$\frac{dJ^*}{dt} = -l_c(x, u^*)$$

$\frac{dJ^*}{dt}$: cost to go; l_c: running cost

Taylor expansion:

$$g(y) = g(x) + \nabla g^T (y - x) + \frac{1}{2} (y-x)^T H(x) (y-x) + \dots$$

$$J^*(x + h\, f(x,u)) = J^*(x) + \frac{\partial J^*}{\partial x} h\, f(x,u)$$

Linear Quadratic Regulator

$\dot{x} = Ax + Bu$ $\quad Q \geq 0 \quad R > 0$

$l(x,u) = x^T Q x + u^T R u$

Infinite horizon,

$J^*(x) = x^T S x, \quad S \geq 0 \qquad \frac{\partial J^*}{\partial x} = 2x^T S$

$$0 = \min_u \left[x^T Q x + u^T R u + \frac{\partial J}{\partial x}(Ax + Bu) \right]$$

$\frac{\partial [\,]}{\partial u} = 0 \qquad u^* = -R^{-1} B^T S x = -Kx$

Substituting u^* back in, $0 = Q - SBR^{-1}B^TS + A^TS + SA$

$$\frac{\partial}{\partial u}\left[x^T Q x + u^T R u + 2x^T S(Ax + Bu) \right] = 0$$

$$2u^T R + 2x^T S B = 0$$

$$u^T R = -x^T S B$$

$$R^T u = -B^T S x$$

$$u = -R^{-1} B^T S x$$

$$x^T Q x + x^T S^T B R^{-1} R R^{-1} B^T S x + 2x^T S(Ax - BR^{-1}B^T S x) = 0$$

$$x^T Q x + x^T S^T B R^{-1} B^T S x + 2x^T S A x - 2x^T S B R^{-1} B^T S x = 0$$

$$\left[Q + S^T B R^{-1} B^T S + 2SA \right]$$

ARE

(Algebraic Riccati Equation)