CSC_5RO01_TA

Hybrid Optimal Control

Lecture 3: Dynamic Programming

Mario Gleirscher (v0.1, 5/12/25)

3A/Master Course, 2 ETCS, 2025/26

Learning Goals

Questions:

1. How can we refine control beyond switches?
2. How can optimality be characterised inductively?
3. How can we compute optimal controllers numerically?

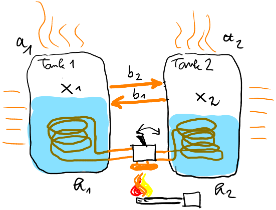
At the end of this lecture, you will …

1. know about two different ways to enact control in hybrid systems,
2. understand the Bellman optimality principle and dynamic programming (DP) as a powerful alternative to linear-quadratic (LQ) regulator (LQR),
3. learn about a simple algorithm to compute sampled-time/state controllers.

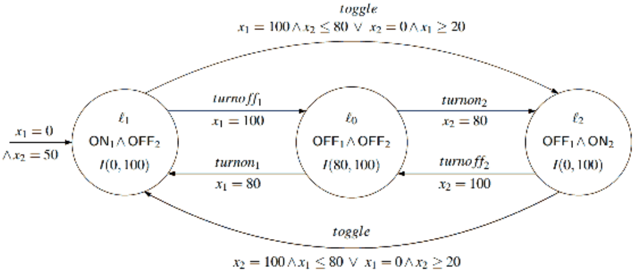
How can we refine control beyond switches?

Another Example: Two Heated Tanks

(Doyen et al. 2018, p. 8)

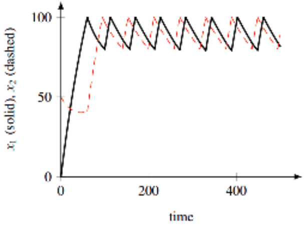


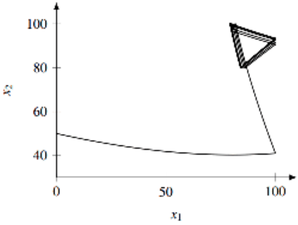
with implicit updates
$x_i^+ = x_i$ for $i \in 1, 2$



$\mathsf{ON}_1 \equiv \dot{x}_1 = h_1 - a_1 x_1 + b_1 x_2$
$\mathsf{OFF}_1 \equiv \dot{x}_1 = -a_1 x_1 + b_1 x_2$
$I(a, b) \equiv a \leq x_1 \leq b \wedge a \leq x_2 \leq b$

$\mathsf{ON}_2 \equiv \dot{x}_2 = h_2 - a_2 x_2 + b_2 x_1$
$\mathsf{OFF}_2 \equiv \dot{x}_2 = -a_2 x_2 + b_2 x_1$



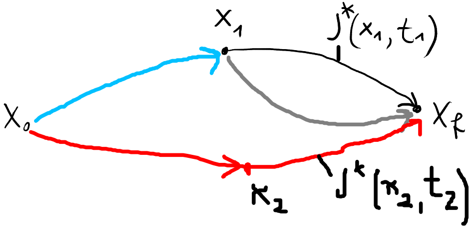


Is there an inductive characterisation of optimality?

Continuous Dynamic programming

(Bellman 1956)

Problem: Find $\mathbf{u}^*$ producing a trajectory from initial state/time $(\mathbf{x}_0, t_0)$ to final state/time $(\mathbf{x}_\mathrm{f}, t_\mathrm{f})$ with minimal cost $\tilde{J}^*(\mathbf{x}_0, t_0)$.



Bellman's Principle of Optimality

Any suffix of an optimal trajectory is necessarily optimal for the corresponding problem initiated at the start of that suffix.

Continuous Dynamic programming (Bellman 1956)

Given state $\mathbf{x}$, time t, control signal $\mathbf{u}$, stage and terminal cost L and Φ, decompose multi-stage into single-stage optimisation via the cost-to-go function

$$\tilde{J}(\mathbf{u}, \mathbf{x}(t), t) = \underbrace{\int_t^{t_\mathrm{f}} L(\mathbf{x}, \mathbf{u}; \tau)\, \mathrm{d}\tau}_{\text{running cost from } \mathbf{x}(t),\, t} + \underbrace{\Phi(\mathbf{x}; t_\mathrm{f})}_{\text{terminal cost}}$$

subject to $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}; t), \mathbf{x}(t_0) = \mathbf{x}_0, \mathbf{x}(t_\mathrm{f}) = \mathbf{x}_\mathrm{f}$, and $\mathbf{u} \in \overline{\mathcal{U}}_{\mathrm{admissible}}$.

Task: For some state $\mathbf{x}(t) \in \mathcal{X}$ at time t, compute the value function

$$V(\mathbf{x}; t) = \min_{\mathbf{u} \in \overline{\mathcal{U}}} \tilde{J}(\mathbf{u}, \mathbf{x}(t), t) = \tilde{J}(\mathbf{u}^*, \mathbf{x}(t), t)$$

with $\mathbf{u}^* = \arg\min_{\mathbf{u} \in \overline{\mathcal{U}}} \tilde{J}(\mathbf{u}, \mathbf{x}(t), t)$ where $V(\mathbf{x}; t_0) = J(\mathbf{u}^*)$.

Necessary condition: Hamilton-Jacobi-Bellman (HJB) equation

$$\underbrace{-\frac{\partial V(\mathbf{x}; t)}{\partial t} = \min_{\mathbf{u}(t) \in \mathcal{U}}}_{\text{point-wise minimisation}} \Big\{ \underbrace{L(\mathbf{x}, \mathbf{u}; t) + \frac{\partial V(\mathbf{x}; t)}{\partial \mathbf{x}} \overbrace{f(\mathbf{x}, \mathbf{u}; t)}^{\dot{\mathbf{x}}}}_{\text{control Hamiltonian } \mathcal{H}(\mathbf{x}, \mathbf{u}, \partial V(\mathbf{x}; t)/\partial \mathbf{x}; t)} \Big\}$$

How can we compute a digital optimal controller for sampled non-linear dynamics?

Discrete DP by Value Iteration

Idea: Derive discrete dynamics $x_{k+1} = f(x_k, u_k, k)$ and solve the HJB equation offline by recursion of the value function.

```
procedure DDP(in 𝒳, (f, 𝒰), J[L, Φ], [t_0, t_f]_δt; out V, u*)
    (V, u*) ← ⊥^{|𝒳|×[t_0, t_f]}                                   ▷ ⊥ undef., e.g., 0
    V(𝒳, t_f) ← Φ(𝒳)
    for k ∈ [t_0, t_f − δt] do                                       ▷ backwards
        for x_k ∈ 𝒳 do
            𝒳^{𝒰_k}_{k+δt} ← x_k + f'(x_k, 𝒰_k, k)δt                ▷ forward Euler set
            V(x_k, k) ← min_{u_k∈𝒰_k} {L(x_k, u_k, k) + V(x^{u_k}_{k+δt}, k + δt)}
            u*(x_k, k) ← arg min_{u_k∈𝒰_k} {L(x_k, u_k, k) + V(x^{u_k}_{k+δt}, k + δt)}
```

for discretised states $\mathcal{X}$, inputs $\mathcal{U}$, and finite horizon $[t_0, t_f]_{\delta t}$ and with $x^{u_k}_{k+\delta t} \in \mathcal{X}^{\mathcal{U}_k}_{k+\delta t}$.

Can hybrid automata be represented as programs?

Translation of Hybrid Automata into Hybrid Programs

Problem: (mutual) recursion after direct translation of HA
Solution: replace (mutual) recursion by iteration

Examples:

1. Mutual recursion, starting from A:

$$A := a; B$$
$$B := b; A$$

$$A' := a; b; A'$$

- ▶ $A' := (a; b)^\star$

2. Recursion with branching:

$$B := b; B \cup c; A$$

$$B' := b^\star; c; A$$

- ▶ $B' := b^\star; c; A$

3. Mutual recursion with branching, starting from A:

$$A := a; B$$
$$B := b; B \cup c; A$$

$$B' := b^\star; c; A$$
$$A' := a; b^\star; c; A'$$

- ▶ $A' := (a; b^\star; c)^\star$

with hybrid programs a, b, c and identifiers A, B, A', B'.

Summary

- DP parameterises cost functions as cost-to-go functions. Optimising the latter yields a state- and time-dependent value function and a corresponding optimal controller.
- The HJB equation yields a versatile induction principle for computing optimal controllers.
- The value change (cost reduction) by increasing search time corresponds to the stage cost L plus the minimal change (ideally, cost reduction) along the flow f.
- The HJB equation enforces any solution for V to be a Lyapunov function.

References I

Bellman, Richard E. (1956). Dynamic Programming. Princeton UP.

Doyen, Laurent et al. (2018). “Verification of Hybrid Systems”. In: Handbook of Model Checking. Springer, pp. 1047–1110.