A statistician's control theory cheat sheet

Tiffany Ding

February 18, 2025

Why should a statistician study control theory?

1. Control $\to$ Stats: We can use control ideas to design new statistical methodologies (e.g., Conformal PID Control)
2. Stats $\to$ Control: We can apply statistical ideas to improve methods for control

1 Introduction

What is control?

Control theory is the study of how to take actions to get a dynamical system to behave in a desired way.

Notation. It is standard in control to use x to denote the state, u to denote the control input (i.e., the action we get to choose), and w to denote perturbations (i.e., state transition noise). x, u, w can be of arbitrary dimension.
Definition 1. A *dynamical system* is describe by an initial state x_0 and function describing the time dependence (this can be continuous or discrete, but we will focus on discrete systems):

$$x_{t+1} = f_t(x_t, u_t, w_t) \tag{1}$$

For simplicity, we will assume that we can observe the state x directly. A more general version of the control problem is to assume we observe $y_t = g_t(x_t)$, where g_t is a (possibly noisy) measurement function.

A generic control problem looks like

$$\min_{u_{1:T}} \sum_{t=1}^{T} c_t(x_t, u_t)$$
$$\text{subject to (1)}$$

where c_t is a cost function chosen to encourage the desired behavior. Some examples:

1. To keep the state close to x^*, we can use $c_t(x_t, u_t) = \|x_t - x^*\|^2$
2. To avoid states $x \in A$ (e.g., locations of obstacles), can use $c_t(x_t, u_t) = \infty \cdot \mathbb{1}\{x_t \in A\}$
3. To avoid large-valued control inputs, we can add a control penalty $\|u_t\|^2$

We can create weighted combinations to encourage multiple behaviors.

Examples of control.

1. Wright brothers: famous not just for flight but specifically *controlled flight* — their plane involved movable surfaces (fins and wings) that allowed them to control the trajectory of flight so they could stay at the desired height and head towards the right direction
2. Autonomous vehicles: want to adjust steering, gas, brakes to stay in lane, avoid obstacles, and go at a safe but not too slow speed.

3. Economics: we can view the economy as a very complicated dynamical system. Governments want to promote economic growth while managing inflation by setting taxes, regulations, and interest rates appropriately.

Control vs. ____

- *Control vs. online learning* (à la Hazan et al. (2016)): Online learning doesn't consider states and transitions. Control does.
- *Control vs. reinforcement learning*: Ben Recht describes "reinforcement learning as optimal control when the dynamics are unknown" (Recht, 2019). Indeed, whereas the most classical control formulations do not allow for unknown dynamics (e.g., optimal and robust control), *adaptive control* does. Sutton, Barto, and Williams have a paper called "Reinforcement learning is direct adaptive optimal control" Sutton et al. (1992), so it seems accurate to view RL as a particular type of control.

Basic vocabulary

Open vs. closed loop systems.

- *Open-loop* = no feedback = "feedforward". Cannot update action based on output of system. Example: sprinkler system that turns on every morning, regardless of whether it is raining
- *Closed loop* = actions are adjusted based on feedback.

We will focus on closed-loop control.

A successful control system should have these properties:

1. **Controllability** = ability to move a system from any given state to any desired state by choosing the right control input
2. **Stability** = bounded input $\to$ bounded output. Formally, there exists a sequence of control inputs that can make $\lim_{t\to\infty} x_t = 0$.
3. **Observability** = ability to uniquely identify the state vector x_t given the observation vector $y_t = g(x_t)$. For linear systems, observability is determined by matrix rank conditions.

2 Control formulations

(A non-exhaustive list.)

(Stochastic) optimal control. Solve

$$\min_{u_{1:T}} \mathbb{E}_{w_{1:T}} \left[\sum_{t=1}^{T} c_t(y_t, u_t) \right] \qquad \text{subject to (1)},$$

usually assuming the transition function $f_t \equiv f$ is known or already estimated.

Robust control. Given a constraint set $\mathcal{K}$ for the noise, solve

$$\min_{u_{1:T}} \max_{w_{1:T} \in \mathcal{K}} \sum_{t=1}^{T} c_t(y_t, u_t) \qquad \text{subject to (1)},$$

usually assuming the transition function $f_t \equiv f$ is known or already estimated.

Adaptive control. Solve a generic control problem, but now the transition functions f_t are unknown and must be estimated online. This is usually accomplished by assuming a parametric form for f and running recursive least squares or gradient descent to do parameter estimation.

Online nonstochastic control. This formulation of control was proposed relatively recently in Agarwal et al. (2019). " Nonstochastic" here means that instead of assuming a probabilistic model for the noise $w_{1:T}$, it is allowed to be an arbitrary/adversarial sequence.

The goal is to devise an algorithm A for choosing action u_t for each time t that achieves low *regret* with respect to a baseline class Π of policies, defined as

$$\text{regret}_T(\mathcal{A}, \Pi) = \max_{w_{1:T}:\|w_t\|\leq 1} \sum_{t=1}^{T} c_t(x_t, u_t) - \min_{\pi \in \Pi} \sum_{t=1}^{T} c_t(x_t^\pi, u_t^\pi)$$

where x_t^π, u_t^π are the counterfactual state sequence and controls under the policy π.

How is this different from stochastic optimal control and robust control? If our goal is to do control in an "optimal" way, a key insight from online nonstochastic control is that the "optimal" control depends on the realized sequence of noise $w_{1:T}$. Robust control provides a solution for the worst-case, when $w_{1:T}$ take on values that result in the largest cost for us. Indeed, when faced with the worst-case $w_{1:T}$, playing the robust control solution will achieve lower cost than any other policy. However, when faced with a different sequence $w_{1:T}$, it is no longer guaranteed that the robust control solution achieves the lowest cost. Online nonstochastic control asks the question of whether we can construct an algorithm that, for *any* sequence $w_{1:T}$, we can achieve costs that are close to the costs that would have been achieved by choosing actions according to a policy in the baseline policy class. See Chapter 1.6 of Hazan et al. (2016) for further discussion.

3 Common controllers

Bang-Bang (aka On-Off)

- *Goal:* Keep state $x_t \in \mathbb{R}$ in $[x_{\min}, x_{\max}]$ by choosing control inputs u_t in $[u_{\min}, u_{\max}]$
- *Control rule:*

$$u_t = \begin{cases} u_{\max} & \text{if } x_t < x_{\min} \\ u_{\min} & \text{if } x_t > x_{\max} \end{cases}$$

- *Pros.* (1) Bang-bang control is actually optimal control for some problems, especially "minimum time" problems (e.g., the fastest way to get a car to a point is to apply max acceleration until a certain point then apply max braking). (2) Easy to implement.
- *Cons.* Can lead to oscillations. Proportional control addresses this.

PID (Proportional-Integral-Derivative) This is the most commonly used controller.

- *Goal:* Keep state x_t close to x^* (*set point*).
- *Control rule:* Define error term $e_t = x_t - x^*$. For constants K_P, K_I, K_D, set action according to

$$u_t = K_P \underbrace{e_t}_{P} + K_I \underbrace{\sum_{i=1}^{t-1} e_{t-i}}_{I} + K_D \underbrace{(e_t - e_{t-1})}_{D}$$

  Practitioners have developed a variety of heuristics to tune K_P, K_I, K_D.

  - "P" = proportional to current error. This is intuitive: when we are far from the set point, we should take a larger action than when we are close to it (this mitigates overshooting and oscillations).
  - "I" = integral of past errors. This is necessary to ensure that the system has the right steady state. If the sum of past errors was always positive or always negative, integral control would allow us to correct for this bias.

- "D" = derivative of the error. This is an attempt to preemptively correct for future errors. One way to see this is by observing that a first-order approximation of e_{t+1} is $\hat{e}_{t+1} = e_t + (e_t - e_{t-1})$. e_t is already included in the "P" term, so we just have to add $(e_t - e_{t-1})$.

- *Practical notes.* With only proportional control, the system is not guaranteed to converge to the set point, so you need at least P and I. In fact, many PID controllers are actually just PI controllers (with no D component).

Model Predictive Control

- *Control rule:*
  1. Measure the current state x_t.
  2. Compute the optimal trajectory $u^*_{0:N}$ for next N time steps starting from x_t.
  3. Execute the first step of the optimal trajectory u^*_0.
  4. Repeat.
- *Pros:* Good for strongly nonlinear systems with constraints (can be directly added to the optimization problem) and time delays (which PID struggles with).
- *Cons.* Need to solve a new optimization problem at every time step, so potentially expensive (contrast with LQR, for which we can just do the optimization offline),
- *Remarks.*
  1. System identification is an important prerequisite for running MPC
  2. MPC is an instantiation of optimal control

LQR (Linear-Quadratic Regulator)

- *Setting.*
  - Linear (time-invariant) dynamical system
    $$x_{t+1} = Ax_t + Bu_t \tag{2}$$
  - Quadratic costs
    $$c_t(x,u) = x^T Qx + u^T Ru$$
    where Q and R are PSD matrices.
- *Control rule:* $u^*(x) = Kx$ where $K = -(R+B^TSB)^{-1}(B^TSA)$ and S solves $S = Q+A^TSA-A^TSB(R+B^TSB)^{-1}B^TSA$.
- *Claim:* This control rule is optimal.
  Proof:
  1. Optimality[1] means that $u^*(x)$ solves
     $$\min_{u(x)} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} x_t^T Qx_t + u_t^T Ru_t$$
     $$\text{subject to } x_{t+1} = Ax_t + Bu_t$$
  2. The Bellman optimality equation says that $v^*(x)$, which gives the value of starting at state x and behaving optimally, must satisfy
     $$v^*(x) = \min_u \left\{x^T Qx + u^T Ru + v^*(Ax+Bu)\right\} \tag{3}$$

[1]Remark: We consider optimality over an infinite horizon, but there are also proofs for optimality of LQR over a finite horizon H. Hazan et al. (2016) presents both.

3. We assume that the value function can be written as $v^*(x) = x^T Sx$ for some matrix S (and will later show that this is indeed a solution). Plugging into (3) yields

$$x^T Sx = \min_u \left\{ x^T Qx + u^T Ru + (Ax + Bu)^T S(Ax + Bu) \right\} \tag{4}$$

Taking the gradient with respect to u of the right-hand side and setting equal to 0 gives

$$2Ru + 2B^T S(Ax + Bu) = 0$$

So the optimal u is

$$u = \underbrace{-(R + B^T SB)^{-1}(B^T SA)}_{K} x$$

However, we still don't know S. To solve for S, we plug this optimal u back into (4):

$$x^T Sx = x^T Qx + x^T K^T RK^T u^T + (Ax + BKx)^T S(Ax + Kx)$$

Since this must hold for all x, we must have

$$S = Q + K^T RK + (A + BK)^T S(A + BK)$$

which can be simplified to

$$S = Q + A^T SA - A^T SB(R + B^T SB)^{-1} B^T SA \tag{5}$$

by plugging in the definition of K. This last equation is called the *discrete time algebraic Ricatti equation*, which Elad Hazan describes as "one of the most important equations in control theory."

LQG (Linear Quadratic Gaussian) = Kalman filter + LQR

- *Setting.*
  - Linear (time-invariant) dynamical system, with noise

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{6}$$

  - Quadratic costs

$$c_t(x, u) = x^T Qx + u^T Ru$$

  - Linear observations, with noise

$$y_t = Cx_t + v_t$$

  where w_t and v_t are independent Gaussian noise

- *Control rule.* Estimate the state using Kalman filter, then apply LQR. This is optimal due to the "separation principle," which says that under some assumptions, an optimal controller can be obtained by first constructing an optimal state estimator and then applying an optimal controller.

Gradient Perturbation Controller.

Setting. We assume linear transitions as in (6) but we now make no assumptions on the distribution of w_t. Cost functions c_t have to be convex, bounded, and have bounded gradients wrt x_t and u_t. We will choose a policy from the class of policies Π^{DAC} called *disturbance action controllers (DAC)*. A disturbance action policy $\pi_{K,M_{1:h}}$ is parameterized by the matrices $M_{1:h} = [M_1, \ldots, M_h]$ and stabilizing[2] controller K. It outputs control u_t^π at time t according to the rule

$$u_t^\pi = Kx_t + \sum_{i=1}^{h} M_i w_{t-i}.$$

[2](Definition 2.4 of Hazan et al. (2016)) Formally, a control policy π for a dynamical system with $f_t \equiv f$ is γ-stabilizing if the sequence of states $x_t^\pi = f(x_t^\pi, \pi(x_t^\pi), w_t)$ satisfies $\|x_t\| \leq \gamma$ for any sequence of perturbations w_t's with $\|w_t\| \leq 1$. If such a policy π exists, f is said to be *γ-stabilizable*

Algorithm. Given A and B, choose a stabilizing K. Choose window length h, initial values $M^1_{1:h}$, and learning rate η. Then at each time t,

1. Use control $u_t = Kx_t + \sum_{i=1}^h M_i^t w_{t-1}$ and incur cost $c_t(x_t, u_t)$
2. Observe state x_t and compute noise $w_t = x_t - Ax_{t-1} - Bu_{t-1}$
3. Update DAC matrices:
   - Define $\ell_t(M_{1:h}) = c_t(x_t(M_{1:h}), u_t(M_{1:h}))$ where $x_t(M_{1:h})$ is the counterfactual state arrived at by playing a DAC policy $M_{1:h}$ from the beginning of time and $u_t(M_{1:h})$ is defined similarly.
   - Update $M^{t+1}_{1:h} = M^t_{1:h} - \eta \nabla \ell_t(M^t_{1:h})$

Guarantee. Compared against the baseline policy class Π^{DAC}, GPC achieves $O(\sqrt{T})$ regret. At a high level, this comes from applying the guarantee for online gradient descent (with some additional arguments necessary). For details, see Theorem 7.2 and the following discussion in Hazan et al. (2016).

4 Recommended resources

A good general resource is Chapter 7–9 (especially Chapter 8) of *Data Driven Science & Engineering: Machine Learning, Dynamical Systems, and Control* [link], by Brunton and Kutz, and the accompanying videos [link]. For a classic control theorist's view, see *Feedback Systems: An Introduction for Scientists and Engineers* [link] by Åström and Murray. For more on online nonstochastic control, see *Introduction to Online Nonstochastic Control* [link] by Hazan. For a whole book on PID control, see *PID Controllers: Theory, Design, and Tuning* [link]. Chapter 17 of *Robotic Systems* [link] by Kris Hauser has a nice presentation of optimal control, and Section 17.6 in particular does a good job describing Model Predictive Control, with a focus on practical considerations.

References

Naman Agarwal, Brian Bullins, Elad Hazan, Sham Kakade, and Karan Singh. Online control with adversarial disturbances. In *International Conference on Machine Learning*, pages 111–119. PMLR, 2019.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4): 157–325, 2016.

Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.

Richard S Sutton, Andrew G Barto, and Ronald J Williams. Reinforcement learning is direct adaptive optimal control. *IEEE control systems magazine*, 12(2):19–22, 1992.