Dynamic Programming

1 Motivation: Why Dynamics Matter

- Many economic problems involve *decisions over time*.
- In static optimization, we choose a single vector x to maximize $f(x)$ once.
- In dynamic optimization, we choose an entire sequence $\{x_t\}_{t=0}^{T}$ to maximize

$$\sum_{t=0}^{T} \beta^t u(s_t, x_t)$$

  subject to evolving constraints.

The key feature of a dynamic problem is **intertemporal dependence** today's decision affects tomorrow's feasible set.

Examples

- Resource extraction (trade-off between current and future use)
  - Suppose a manager extracts a nonrenewable resource such as groundwater or oil. Each period, extraction reduces the remaining stock. The manager faces a trade-off between the *immediate profit* from extracting today and the *future value* of the remaining stock. Dynamic optimization determines the optimal extraction path balancing current and future gains — the or *shadow value* of the resource.
- Investment and capital accumulation
  - Firms decide how much to invest in physical capital (machinery, buildings) or human capital (training, education). Investment today increases the productive capacity tomorrow, affecting future profits. The firm must weigh the *cost of investment* against the *expected future returns*, leading to an optimal investment strategy over time.
- Crop rotations or livestock feeding
  - Farmers choose how to allocate land and resources over multiple periods. Planting a crop today may deplete soil nutrients, affecting yields in subsequent seasons. Similarly, feeding strategies for livestock impact their growth and productivity over time. Dynamic programming helps determine optimal crop rotations or feeding schedules that maximize long-term farm profitability.
- Forest rotation and timber harvesting
  - Forest managers decide when to harvest trees to maximize the value of timber while ensuring sustainability. Harvesting too early may yield lower quantities of wood, while waiting too long can lead to overgrowth where the marginal wood growth does not justify the cost. Dynamic optimization helps find the optimal rotation period that balances immediate revenue with future discounted value.
- Pollution control and climate policy
  - Governments and firms must decide how much to invest in pollution abatement technologies or carbon reduction strategies. Reducing emissions today can mitigate future environmental damage and associated costs. Dynamic programming helps evaluate the trade-offs between current abatement costs and long-term benefits, guiding optimal policy decisions over time.

2 A Simple Two-Period Example

Consider a groundwater extraction problem:

$$\max_{q_1, q_2} \pi_1(q_1) + \beta \pi_2(q_2)$$

subject to

$$s_2 = s_1 - q_1, \quad q_t \leq s_t, \quad s_t \geq 0.$$

It will not be optimal to leave any resource in the ground at the end, so $s_2 = 0$ and $q_2 = s_1 - q_1$. Substituting q_2 into the objective gives

$$\max_{q_1} \pi_1(q_1) + \beta \pi_2(s_1 - q_1).$$

The first-order conditions give

$$\pi_1'(q_1) - \beta \pi_2'(q_2) = 0,$$

which equates the **marginal benefit of current extraction** to the **discounted marginal benefit of future extraction**.

This trade-off defines a **user cost** or **shadow value** of the resource stock.

---

3 Backward Induction

- In a *finite-horizon sequential decision problem*, one can solve by starting at the last period and then working backward.
- At the final period, the decision is trivial (or known) because nothing is left to optimize afterward.
- Then one moves to the second-to-last period: given that the future is solved optimally, choose your action now to maximize current payoff plus continuation value.
- This process continues backward until reaching the initial period. This is the logic of dynamic programming in finite-horizon settings.
- Backward induction is prefered over brute force enumeration because it simplifies potentially complex problems. Starting from the beginning, the number of paths can explode combinatorially, making it infeasible to evaluate all possible sequences of decisions.
- This was the intuition in the problem above: we used the constraint to eliminate q_2 and reduce the problem to a single decision variable q_1. Because there was no salvage value (no value to not using all of the resource at the end), we could assume that $q_2 = s_2 = s_1 - q_1$.
- In a finite-horizon problem, backward induction works because there is a last period: after that, no decisions remain. We can anchor the recursion at T and move backward.
- In an infinite-horizon problem, there is no terminal period. We need a different anchor. What we look for is a stationary path (or policy) that is internally consistent over time: i.e. at every date, given the current state, the decision you make will also be optimal for the continuation. In other words, you want a path or policy such that there is no incentive to deviate at any point — because the same 'value' structure recurs over time. That is what gives the stability and time-independence of the optimal policy in many infinite-horizon problems.

---

4 Stochastic Dynamic Programming

- In many problems, there is uncertainty about future states or payoffs.
- The state variable s_t may evolve according to a stochastic process, influenced by random shocks.
- The decision-maker must consider the expected value of future payoffs, integrating over possible future states.

4.1 Markov property

A problem is **Markovian** if the current state s_t contains all relevant information from the past. The future state depends only on the current state and decision.

Examples

- Stochastic resource growth model:
  Let fish stock evolve as
  $$s_{t+1} = G(s_t) + \varepsilon_{t+1} - h_t$$
  where ε_{t+1} is a random growth shock (possibly discrete). Then transition probabilities $P(s' \mid s, h)$ come from the distribution of ε. Note that s_{t+1} depends only on s_t and h_t, not on earlier states or actions.

- Crop yield shocks in agriculture:
  A farmer chooses input (fertilizer) x_t. The next soil state evolves as
  $$s_{t+1} = g(s_t, x_t)$$
  . The stochastic yield shock induces $P(s_{t+1} \mid s_t, x_t)$.

- Capital accumulation with productivity shocks:
  Suppose production is
  $$y_t = F(k_t, z_t)$$
  where z_t is a stochastic productivity state evolving as a Markov chain (discrete or discretized AR(1)). The decision is investment i_t, state vector is (k_t, z_t), and transitions are given by $\Pr(z_{t+1} \mid z_t)$ combined with deterministic $k_{t+1} = (1 - \delta)k_t + i_t$.

---

5 Vocabulary and the components of dynamic programming problem

- **State variable** s_t: summarizes all relevant information at time t needed to make decisions and predict future states. Examples: resource stock, capital stock, productivity level.
- **Control variable** x_t: the decision made at time t. Examples: extraction quantity, investment amount, input usage.
- **Transition function** $s_{t+1} = g(s_t, x_t, \varepsilon_{t+1})$: describes how the state evolves from t to $t+1$ given current state, action, and random shocks.
- **Reward function** $u(s_t, x_t)$: the immediate payoff from taking action x_t in state s_t. Examples: profit, utility, yield.
- **Policy** $\{x_t(s_t)\}$: a rule that specifies the action to take in each state at each time.
- **Value function** $V_t(s_t)$: the maximum expected discounted sum of future rewards starting from state s_t at time t.
- **Discount factor** $\beta \in (0, 1)$: reflects the preference for current rewards over future rewards. It reflects time preference or impatience: how much less future utility / profit is worth relative to now. In growth / resource models, discounting ensures that long-run payoffs are “worth less” the farther they are in the future, which prevents over-emphasis on infinitely deferred rewards.

  - It also enables contraction mappings in infinite-horizon problems.
  - We often assume a constant discount factor for simplicity, but in reality, discounting may vary over time or depend on circumstances.
- **Salvage value**: the value of remaining resources or capital at the end of the planning horizon. In finite-horizon problems, this can be zero or some terminal value function.

5.1 Examples

Here is a finite time discrete state and time example:

$$\max_{h_t} \left[\sum_{t=0}^{T-1} \beta^t \pi_t(h_t) + \beta^T V_T(s_T) \right]$$

subject to the simple deterministic transition dynamics:

$$s_{t+1} = s_t - h_t + G(s_t)$$

- State: s_t
- Control: h_t
- Transition: $s_{t+1} = s_t - h_t + G(s_t)$
- Reward: $\pi_t(h_t)$
- Policy: $h_t(s_t)$
- Value function: $V_t(s_t)$
- Discount factor: β
- Salvage value: $V_T(s_T) = 0$ or some terminal value

Identify the components of the problem

---

6 Looking Ahead

Topic	Focus	Reference
Finite-horizon DP	Backward recursion with discrete state spaces	M&F Ch. 7
Infinite-horizon DP	Stationary value functions; contraction mappings	M&F Ch. 8
Continuous-state models	Euler equations; collocation and projection	M&F Chs. 8–9
Applications	Resource extraction, investment, policy models	Applied readings

---