

EIOBook-6: Introduction to Dynamics

Fast, intuitive, and highly coherent review notes

Coherence spine (read this first)

Dynamic IO begins with a diagnosis: static models treat today's choice as depending only on today's environment, but many IO settings are **state-dependent**—today's action changes tomorrow's state (inventory, installed base, machine age, capacity, experience, etc.). Once behavior is forward-looking, the key missing object is the **continuation value**. The Bellman equation summarizes optimal intertemporal choice by combining current payoffs with the discounted value of future states. This immediately creates two linked tasks: (i) **solve** the dynamic program (to compute value and optimal policy) and (ii) **estimate** primitives (payoffs and transitions) from data. Because repeatedly solving a DP is computationally expensive, estimation methods are organized by their computational strategy: either solve the DP inside estimation (NFXP) or avoid repeated solutions by exploiting observed conditional choice probabilities (CCP/Hotz–Miller, NPL). The applications at the end (replacement, patent renewal, dynamic pricing) are not separate topics; they are instances of the same template with different state variables, transitions, and frictions.

The one mental picture

Dynamic IO = (state today) + (choice today) → (state tomorrow), and agents choose today using the value of tomorrow. Formally, with discount factor β (write it as b if desired), the dynamic problem is summarized by:

$$V(s) = \max_a \left\{ \pi(a, s) + \beta \mathbb{E}[V(s') \mid a, s] \right\}.$$

Everything in the chapter explains why this object is needed, how to compute it, and how to use it for identification and counterfactuals.

A 60–90 minute study plan

1) Why static models fail (motivation)

Connection tag: this section exists to identify the *missing state variable* and the *bias* created by ignoring it. The examples are not decorative; each points to a specific state dependence that a static model cannot capture.

- **Storable goods and promotions:** temporary discounts trigger stockpiling (buy now, store, buy less later), so a static model can confuse timing responses with true long-run substitution.
- **Durables:** forward-looking consumers wait for future price declines and the composition of remaining consumers changes over time, distorting static elasticity and willingness-to-pay estimates.
- **Supply-side dynamics:** adjustment costs and sunk costs imply long-run structure and markups respond to policy in ways that static analysis misses.
- **Dynamic pricing:** inventories, ordering costs, and menu costs generate state-dependent pricing and price cycles.

2) The dynamic template (state, action, payoff, transition)

Connection tag: once timing matters, you need a language that keeps behavior consistent over time. The template converts “dynamic intuition” into objects you can solve and estimate.

- **State:** s_t (inventory, capital stock, patent age, installed base, last price, etc.)
- **Action:** a_t (buy/hold, replace/not, renew/not, price/order decisions, etc.)
- **Per-period payoff:** $\pi(a_t, s_t)$ (profit/utility plus possibly private shocks)
- **Transition:** $s_{t+1} \sim f(\cdot \mid a_t, s_t)$
- **Objective:** maximize discounted value with discount factor $\beta \in (0, 1)$

3) Bellman equation (the continuation-value bridge)

Connection tag: the Bellman equation is exactly what static IO is missing: the value of tomorrow. It turns forward-looking behavior into a testable optimality restriction.

$$V(s) = \max_a \left\{ \pi(a, s) + \beta \mathbb{E}[V(s') \mid a, s] \right\}.$$

The expectation term $\mathbb{E}[V(s')]$ formalizes how agents trade off current payoffs against future consequences through the state transition.

4) Solving the DP (because estimation needs model predictions)

Connection tag: you cannot compare the model to data unless you can compute its implied values and policies. Thus, solving the Bellman equation is the operational step that makes dynamics usable.

- **Contraction logic:** with $\beta \in (0, 1)$, the Bellman operator is a contraction, implying a unique fixed point and convergence of iterative algorithms.
- **Value Function Iteration (VFI):** iterate $V^{k+1} = \Gamma(V^k)$ until convergence.
- **Policy Iteration:** iterate on the policy (often fewer iterations but heavier steps).
- **Integrated value / log-sum trick:** with i.i.d. extreme-value shocks, shocks can be integrated out to reduce dimensionality and simplify computation.

5) Estimation methods (computational strategies)

Connection tag: estimation links primitives to observed choices, but the bottleneck is that V depends on parameters. Methods differ primarily by whether they solve the DP repeatedly during estimation.

1. **NFXP (Rust):** an outer loop searches parameters; an inner loop solves the DP for each candidate parameter vector.
Connection: most transparent conceptually, most expensive computationally.
2. **CCP / Hotz–Miller:** use observed conditional choice probabilities (CCPs) to recover value differences, reducing the need to solve the DP repeatedly.
Connection: exploits the structure that optimal choices are functions of relative value.

3. **NPL (Aguirregabiria–Mira):** iterates between CCPs and implied value objects; often much faster than NFXP while targeting the same fixed point/likelihood optimum.
Connection: a computational compromise that uses iteration rather than repeated full DP solution.

6) Applications (the same template with different states/frictions)

Connection tag: these are not separate topics; they are plug-and-play instantiations of the same state-action-payoff-transition template.

A) Replacement / lumpy investment

- State: machine age/capital stock
- Action: replace or not (or discrete investment levels)
- Transition: depreciation and replacement law (e.g., $k_{t+1} = (1 - \delta)(k_t + a_t)$)
- Friction: replacement costs generate discrete adjustment

B) Patent renewal

- State: current profit flow and patent age
- Action: renew or let expire
- Friction: renewal fee implies a threshold rule (renew iff continuation value exceeds the fee)
- Identification intuition: renewal decisions reveal the value distribution via revealed preference

C) Dynamic pricing

- State: inventory, wholesale costs, last price, demand shocks
- Action: price changes and ordering decisions
- Frictions: menu costs and fixed ordering costs generate state-dependent pricing and price cycles

One-page memory anchors (what ties everything together)

- **Motivation** → identifies a missing state variable and static bias.
- **Template** → defines state/action/payoff/transition.
- **Bellman** → formalizes continuation value and optimality.
- **Solution** → makes the model computable (needed for estimation).
- **Estimation** → differs by how often you solve the DP (NFXP vs CCP/NPL).
- **Applications** → the same template instantiated for different states/frictions.

Five-minute self-test

1. In one sentence: why can static models overstate elasticity for storable goods under promotions?
2. List the four primitives of a dynamic structural model (state, action, payoff, transition).
3. Write the Bellman equation and interpret the expectation term.
4. Conceptually distinguish NFXP from CCP/NPL.
5. In patent renewal, why do renewal fees generate a threshold rule (revealed preference logic)?