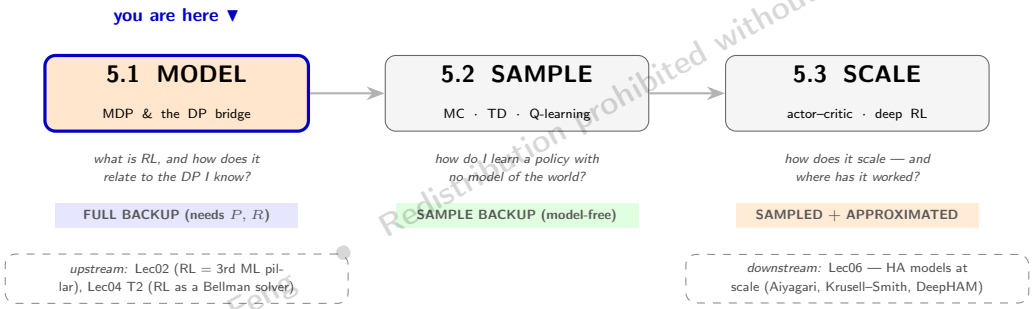
AI for Economic Research: Dynamic Models, Language, and Agents
Lecture 5.1: The RL Framework and Its Link to Dynamic Programming

Zhigang Feng

2026-07-08 19:03:50

Lecture 5 in One Map

One lecture, three decks, one idea: replace the model you must specify with experience you can sample — then scale it with function approximation.



Act 5.1 (here) *runs in five stops: what RL is → your problem in MDP language → what classical DP quietly demands → where it breaks → the departures that become Acts 5.2–5.3. Running anchor throughout: the optimal growth model. New to dynamic macro? A self-contained appendix (Solow → Bellman) backs this deck — the next slide says how to use it.*

This Act in Five Stops — and What It Assumes

What the main track assumes: you have met the **Bellman equation** and value function iteration (first-year macro). If that is you, this deck is self-contained — stops 2–3 are home turf, revisited with RL eyes.

New to dynamic macro? Appendix A1–A11 builds everything from the Solow model up — self-contained, intuition first. **Minimum before stop 3:** A5 (the Bellman equation, read as one sentence) + A8 (why iteration finds it). Best read in full after the session; buttons throughout the deck jump there and back.

▸ Appendix roadmap

What Is Reinforcement Learning?

Intuition and everyday examples, before any formalism

Reinforcement Learning: Learning by Trial and Error

- **Reinforcement learning (RL)**: an **agent** learns to make good decisions by **interacting** with an **environment** and observing a **reward**.
- No teacher hands it the right answer — it learns from **consequences**: try, observe, adjust, repeat.

How RL differs from the other ML pillars:

- **Supervised** learning: data come with *labels* (the correct answer).
- **Unsupervised** learning: find *structure* in unlabeled data.
- **Reinforcement** learning: no labels — only a *scalar reward*, often *delayed*, to be maximized over time.

What Is RL? Three Everyday Examples

Setting	Action	Reward	What it learns
Playing a game (Go, Atari, chess)	a move	win / lose, score	a winning **policy**
A child / robot learning to walk	motor commands	progress; didn't fall	a stable **gait**
A/B testing (web page, price)	which version to show	click / purchase	the best **option**

Common thread: decisions are **sequential**, rewards can be **delayed**, and the agent must **explore** (try things) — not just exploit what looks best so far.

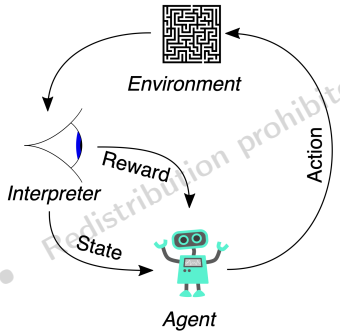
The Same Loop in Economics

- **Dynamic pricing:** a firm sets a *price* (action), observes *demand / profit* (reward), learns a **pricing rule**.
- **Monetary policy:** a central bank sets the *interest rate* (action), observes *inflation / output* (reward), learns a **policy rule**.
- **Consumption–savings:** a household chooses *how much to save* (action), enjoys *utility now* and carries *wealth forward* (state), learns a **savings rule**.

These are exactly the **sequential decision problems** economists have long solved with **dynamic programming**. RL is a different way to attack the *same* problems — by learning from experience.

The Common Pattern: Agent, Environment, Reward

At each time step, the agent observes state S_t, chooses action A_t. The environment returns reward R_{t+1} and transitions to S_{t+1}.



Trajectory: $s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots$ The **return** (cumulative reward) from time t: $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$.

The built-in difficulty: rewards arrive *late* — which of the many past decisions earned them? This is the **credit-assignment problem** (Minsky); every algorithm in this lecture is, at heart, an answer to it.

Two Ways to Solve a Dynamic Problem: Map vs. Explore

Dynamic Programming

"model first, then solve"

- Specify the environment fully (P, R).
- Compute V at **every** state — draw the complete map.
- Then act by the map.
- Like an **omniscient planner**.

Reinforcement Learning

"interact and learn"

- No complete map.
- **Walk** the environment repeatedly; update route judgments after each trip.
- Estimates approach what the map would say — without ever drawing it.
- Like an **explorer**.

Same destination (the optimal policy), two routes. RL shines exactly when drawing the full map is **infeasible** — *when* and *why* that happens is stops 3–4. First, we make both sides precise.

The MDP: Your Problem, Their Language

The formal RL toolkit — and why it is the recursive problem economists already solve

Sequential Decision Problems: What We Are Formalizing

- The economic problems we formalize here are **sequential**:
  - Consumption–saving · firm investment and pricing · monetary and fiscal policy.
- Three features make them genuinely dynamic:
  1. **Sequential**: decisions at every period, not once-and-for-all.
  2. **Intertemporal coupling**: today's consumption $\Rightarrow$ tomorrow's capital.
  3. **Uncertainty**: future shocks z_{t+1} unknown at time t.
- The objective is **long-term** — utility, profit, welfare — so every choice must weigh its whole future.
- Economists solve this class with **dynamic programming**; RL formalizes the *same* class as a **Markov decision process**. The dictionary between the two is next.

Markov Decision Process (MDP) Formulation

- An RL problem is formalized as a **Markov Decision Process (MDP)**:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle.$$

- $\mathcal{S}$: set of **states** (e.g., capital k, shock z, wealth distribution Φ).
- $\mathcal{A}$: set of **actions** (e.g., consumption c, investment i).
- $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, $P(s'|s,a)$: **transition kernel**.
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $R(s,a)$: **reward** ($=$ instantaneous utility $u(c)$). *Canonical form used throughout Lec05–Lec09.*
- $\gamma \in [0,1)$: **discount factor** ($= \beta$ in econ notation).
- **Objective:** Find an optimal policy π^* that maximizes

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right].$$

MDP Meets the Growth Model

▸ New to dynamic macro? Appendix: Solow → Bellman

Consider the **optimal growth model**:

$$\max_{\{c_t, k_{t+1}\}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \, u(c_t) \quad \text{s.t.} \quad c_t + k_{t+1} = z_t f(k_t) + (1-\delta)k_t$$

MDP element	**Growth model**
State s	(k_t, z_t) — capital + productivity
Action a	c_t (or equivalently k_{t+1})
Transition $P(s'\|s,a)$	$k_{t+1} = (1-\delta)k_t + z_t f(k_t) - c_t$; $\log z_{t+1} = \rho \log z_t + \varepsilon_{t+1}$
Reward $R(s,a)$	$u(c_t)$ — period utility
Discount γ	β — household discount factor
Policy $\pi(s)$	Consumption rule $c^*(k,z)$

The Bellman equation and MDP are two views of the **same** problem.

Policy and Value Functions

- **Policy** π: a mapping from states to actions.
  - Deterministic: $a = \pi(s)$ (e.g., $c^*(k, z)$ in econ).
  - Stochastic: $a \sim \pi(a|s)$ (useful for exploration).

- **State-value function** — expected return from state s under π:

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s\right].$$

- **Action-value function** — expected return for action a in state s:

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a].$$

- **Relationship:** $V^\pi(s) = \sum_a \pi(a|s)\, Q^\pi(s, a)$.

$V(s)$ vs. $Q(s,a)$: Economics vs. Engineering

Economics: Focus on $V(s)$

- Emphasis on **welfare/utility** from a given state.
- Optimal policies derived from planner's problem or equilibrium.
- DP solves for $V^{\pi}(s)$ directly.
- Rational agents optimize implicitly.

RL / Engineering: Focus on $Q(s,a)$

- Emphasis on **real-time action selection**.
- Learning $Q(s,a)$ allows decision-making **without a model**.
- $\pi^*(s) = \arg\max_a Q^*(s,a)$.
- Works in unknown environments.

Growing convergence: As economists tackle complex, partially-known environments (heterogeneous agents, financial markets), the Q-function approach becomes increasingly valuable.

The Bellman Connection

Economists' old friend is RL's backbone — plus the fine print classical DP relies on

Why Economists Specify, Then Solve

- Economists **write the model down first** — preferences $u(\cdot)$, technology $f(k)$, constraints, shock process — *then* solve it.
- **Why?** Each primitive carries economic meaning (β patience, α factor share, δ depreciation), so the solution inherits:
  - **Interpretability** — parameters are economic magnitudes.
  - **Welfare analysis** — integrate the same $u(c)$ that defines behavior.
  - **Equilibrium-consistent prices** — $r = zf'(k) - \delta$, $w = z[f(k) - kf'(k)]$ are *competitive factor prices*, not reduced-form fits.
- Disciplined by the **Lucas critique**: decision rules *shift* when policy shifts, so a rule fit to historical behavior can fail once deployed — better to solve a **structural** model.

The Bellman Equation — Economists' Old Friend

▸ Gentler build-up: Appendix A5

- Economists compute the value function $V(s)$ via the Bellman equation:

$$V(s) = \max_a \Big\{ R(s,a) + \gamma\, \mathbb{E}[V(s') \mid s, a] \Big\}.$$

- Define the **Bellman operator** T:

$$(TV)(s) = \max_a \left\{ R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V(s') \right\}.$$

- The value function is the **fixed point**: $V^* = TV^*$.

Theoretical Foundations of DP

▸ Why these hold: Appendix A8–A9

Three pillars guarantee that DP "works":

1. **Principle of Optimality** (Bellman):
   - Any tail of an optimal plan is itself optimal from the state it reaches.
   - Guarantees **equivalence** between the sequence problem and the recursive formulation.
2. **Contraction Mapping Theorem** (Banach fixed-point theorem):
   - $\|TV_1 - TV_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$ for any V_1, V_2.
   - Guarantees **existence**, **uniqueness**, and **convergence** of VFI.
3. **Blackwell's Sufficient Conditions** Blackwell (1965), *Ann. Math. Stat.*:
   - **Monotonicity:** $V_1 \leq V_2 \Rightarrow TV_1 \leq TV_2$.
   - **Discounting:** $T(V + c) \leq TV + \gamma c$ for $c \geq 0$.
   - Easy-to-check conditions that imply contraction.

Economists **prize uniqueness***: it makes "the model predicts X" well-posed — no ambiguity over which solution to report, which is what lends structural counterfactuals their credibility.*

Full Expectation Backup — and DP's Quiet Requirements

The heart of Value Function Iteration (VFI):

$$(TV)(k,z) = \max_{k'} \left\{ u(c) + \beta \sum_{z' \in \mathcal{Z}} P(z'|z)\, V(k',z') \right\}$$

a **full expectation backup**: to update V at *one* state, sum over **all** possible next states, weighted by their probabilities.

The fine print — three requirements so routine we forget they are assumptions:

- **R1** **A known model:** the law of motion and $P(z'|z)$ must be *writable* — else nothing to sum over.
- **R2** **A sweepable state space:** V is a table on a grid, updated state by state — feasible only if the states can be enumerated.
- **R3** **No exploration needed:** with the model in hand, the $\max$ *inspects* every action's consequence — nothing is ever tried.

Each requirement fails somewhere economically important. The next stop shows where — and each failure gets its own RL fix.

Where DP Breaks — and How RL Answers

Two walls, three departures — the plan for the rest of the lecture

Wall 1: The Curse of Dimensionality — R2 Falls

The wall is not exotic. Go's rules fit on one page, yet the game has $\approx 10^{170}$ legal positions (vs. $\sim 10^{80}$ atoms in the universe) — a *fully known* model, and still no table, no enumeration, no sweep.

Economics hits the same wall sooner:

- VFI requires a **grid** over the state space. 2D model (k, z): $100 \times 100 = 10{,}000$ points — easy.
- Multi-sector model with d state variables, m grid points each: m^d points.
  - $d = 5,\ m = 100$: 10^{10} points. $d = 8,\ m = 100$: 10^{16} points — **completely infeasible**.
- Sparse grids (Smolyak) and projection methods (Chebyshev) partially mitigate this, but break down beyond ∼5–8 dimensions.
- **Heterogeneous agents** make it worse: the wealth **distribution** Φ is itself a state variable — an infinite-dimensional object. Krusell–Smith (1998) approximates Φ by its mean K, but accuracy degrades in complex models.

Wall 2: Unknown or Intractable Environments — R1 Falls

Some environments cannot be written down — only interacted with: robot dynamics, city traffic, a strategic opponent. Engineering built RL for exactly this. Economics has its own version:

- In a representative-agent model: agent knows own state $\Rightarrow$ knows prices $\Rightarrow$ environment is **transparent**.
- In a heterogeneous-agent model:
  - Prices depend on aggregate capital $K = \int a \, d\Phi(a, z)$.
  - Individual agent does **not observe** Φ directly.
  - The distribution's law of motion $\Phi' = \mathcal{T}(\Phi)$ has no closed form in general.
- The "environment" (price dynamics) that an agent faces is **endogenous** and **unknown**.

The core issue for economists: prices *are* the environment. When prices depend on distributions you cannot track, the environment becomes something you must *learn* rather than *specify*.

Why (Deep) Reinforcement Learning Helps

1. **Flexible Function Approximation (Neural Networks):**
   - Traditional DP uses structured grids (Chebyshev, splines) that scale exponentially.
   - Neural networks approximate $V(s)$ or $\pi(s)$ **without a grid**; parameters scale gracefully with dimension.

2. **Sample-Based Learning (No Full Expectation Backup):**
   - RL uses **simulated trajectories** instead of summing over all s'.
   - Targets **relevant regions** of the state space via experience.
   - Especially valuable when Φ's law of motion lacks a closed form.

3. **Hardware Acceleration (GPUs/TPUs):**
   - NN training = large-scale matrix multiplication $\Rightarrow$ GPU-friendly.
   - Traditional grid-based DP does not map as naturally onto GPU architectures.

The ideas are old; the takeoff is recent. Core RL dates to the 1980s–90s, but it only "**flew**" once **deep learning** supplied efficient **function approximation in high dimensions**: **DQN** (Atari from raw pixels, 2015) and **AlphaGo** (2016). We return to AlphaGo in T3.

The Main Departures at a Glance

Same Bellman backbone — three departures from classical DP:

Classical DP pillar	The wall	RL's departure	Where
Full expectation backup — needs P, R (R1)	environment unknown or endogenous	**#1 Sample the backup:** learn from trajectories — Monte Carlo, TD, Q-learning	Act 5.2
Value stored in a table / on a grid (R2)	curse of dimensionality	**#2 Approximate the function:** neural V, Q, π — DQN, actor–critic	Act 5.3
The $\max$ inspects every action (R3)	samples only show what you *try*	**#3 Explore deliberately:** ϵ-greedy, Boltzmann, UCB — the price of #1	Act 5.2

The whole lecture in one sentence: the Bellman backbone never changes — only *how the expectation is filled*: full backup (5.1) → sample backup (5.2) → sampled + approximated (5.3).

When DP, When RL? A 2×2 Map

	Model known (P available)	**Model unknown / intractable**
Low-dim	Textbook DP (VFI / PI) — *RL buys little*	Tabular RL (Q-learning, SARSA)
High-dim	"Deep DP": NN approximation *inside* VFI	Deep RL's home turf (HA models, market design)

DP is RL's structural special case: when P is known and $|\mathcal{S}|$ is small, sample backup $\rightarrow$ full backup and function approximation $\rightarrow$ table. (A statement about the *equations*, not a free convergence guarantee.) So RL **extends** DP — it does not replace it. And RL is *not merely a last resort*: even with P known, sampling + approximation can win in high dimensions.

RL: Beyond Classical DP — Summary

- RL algorithms (Q-learning, SARSA, actor-critic) extend DP to:
  - Handle **unknown** transition models (model-free).
  - Address **exploration–exploitation** directly.
  - **Scale up** through function approximation.
- Two pathways:
  - **Model-free RL:** learn policies purely from observed data without explicit transition models.
  - **Model-based RL:** combine simulation from approximate structural models with learning.

The two RL pathways differ only in *where the samples come from* — a real/observed environment (model-free) or a simulated structural model (model-based). Either way, the Bellman backbone is unchanged; only the *backup* changes.

Bridge: From a Model You Specify to Experience You Sample

Act 5.1 framed the problem. Every sequential economic decision is an MDP; the Bellman equation economists already use is RL's backbone; and classical DP's three quiet requirements — a known model (R1), a sweepable state space (R2), no need to explore (R3) — each fail somewhere economics cares about.

- That leaves three departures on the table: **#1 sample the backup**, **#2 approximate the function**, **#3 explore deliberately**.
- **Next (5.2) executes #1 — and pays its price, #3.** The question changes from "how do we *specify* the model?" to "how do we *learn a good policy without one?*" — Monte Carlo, temporal-difference, and Q-learning replace the full expectation backup with a **sample** backup. Act 5.3 then delivers #2 at scale.
- *If the DP side moved fast: Appendix A1–A11 rebuilds it from Solow, self-contained — T2 needs only the Bellman equation and the full-vs-sample backup distinction.*

Arc: **5.1 MODEL** *(framework + DP)* $\rightarrow$ *5.2 SAMPLE (MC / TD / Q-learning)* $\rightarrow$ *5.3 SCALE (actor–critic + deep RL + applications).*

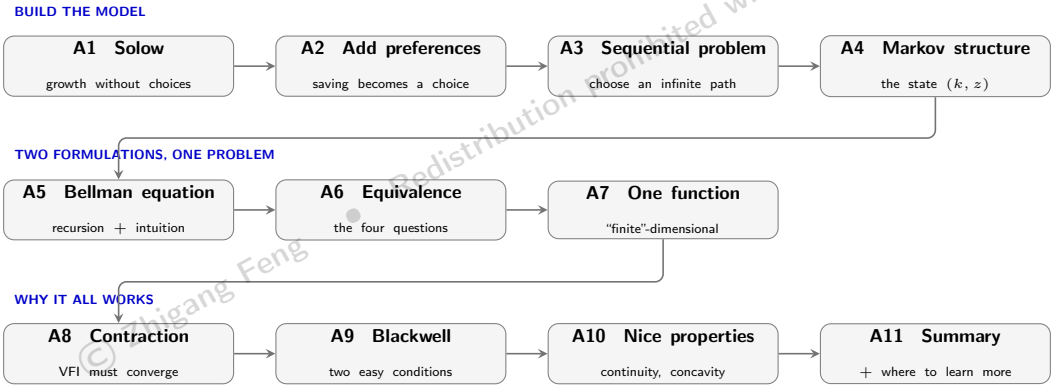
Appendix: Dynamic Macro Foundations

From Solow to Bellman — a self-contained refresher, no dynamic macro assumed
(statements and intuition only; proofs live in your macro-theory course)

Appendix Roadmap: From Solow to Bellman

Everything the lecture takes for granted, built in eleven short steps — click any step to jump; the arrow button on every page returns to the lecture.

Return to lecture



Appendix A1 — Where It All Starts: The Solow Growth Model

The one dynamic model every undergraduate meets — capital accumulation with a rule-of-thumb saver.

◂ Return to lecture

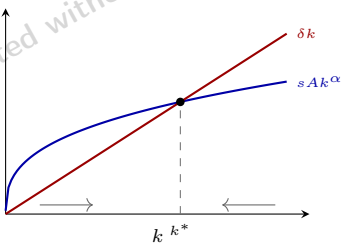
Technology (per-worker terms): output from capital, $y_t = f(k_t) = Ak_t^{\alpha}$, $\alpha \in (0,1)$ (diminishing returns).

Behavior — one assumed number: save a *fixed* fraction s of output: $i_t = s\, y_t$, consume the rest, $c_t = (1-s)\, y_t$.

The engine — capital accumulation:

$$k_{t+1} = (1-\delta)\, k_t + sAk_t^{\alpha}$$

Steady state (saving = replacement):
$sAk^{*\alpha} = \delta k^* \Rightarrow k^* = (sA/\delta)^{\frac{1}{1-\alpha}}$; concavity makes it unique, with monotone convergence from any $k_0 > 0$.



Below k^* saving exceeds replacement, so k grows; above, it shrinks — the arrows on the axis.

Growth without choices. Given s, the dynamics are fully mechanical — nobody in the model ever decides anything. Solow (1956, *QJE*) explains capital accumulation but stays silent on welfare: is s a *good* saving rate?

Appendix A2 — From Solow to Optimal Growth: Make Saving a Choice

Same technology, new question: instead of assuming how much society saves, derive it from preferences.

◂ Return to lecture

- **What Solow cannot answer:** Is s too high or too low for welfare? Would it respond to a capital-income tax? A "golden-rule" s exists — but nothing makes households pick it.
- **The upgrade — keep the technology, add preferences:** technology stays Solow's ($f(k) = Ak^{\alpha}$, depreciation δ); households now rank consumption streams $\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$, with $u' > 0$, $u'' < 0$ (the smoothing motive) and $\beta \in (0, 1)$ (impatience).
- Each period, resources are *chosen* between c_t (utility now) and k_{t+1} (resources tomorrow) — the intertemporal trade-off.
- **The saving rate becomes an outcome** — endogenous, generally time-varying — and the model becomes the *optimal growth model* — this lecture's MDP running example. Planner = competitive equilibrium here (welfare theorems), so the planner's problem is without loss.

Solow, vindicated as a special case: with $u = \log$, $f(k) = k^{\alpha}$, $\delta = 1$, the *optimal* policy is $c^*(y) = (1 - \alpha\beta)\, y$ — a constant saving rate $\alpha\beta$. Solow's assumption re-emerges as the *answer* — the course's closed-form benchmark (Lab 3A).

Sources: Ramsey (1928, Econ. J.); Cass (1965, REStud); Koopmans (1965) — the Ramsey–Cass–Koopmans model.

Appendix A3 — The Sequential Problem: Choosing an Entire Future

The problem exactly as the lecture's opening slide states it — now taken apart.

Return to lecture

The sequential problem (SP), deterministic version:

$$\max_{\{c_t,\, k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t.} \quad c_t + k_{t+1} = A k_t^{\alpha} + (1-\delta) k_t, \quad c_t,\, k_{t+1} \geq 0, \quad k_0 \text{ given.}$$

- **What "a solution" is:** an *infinite list* $(k_1, k_2, k_3, \dots)$ — one number for every future date — chosen all at once at date 0.

- **Why that is hard:**
  - *Infinitely many unknowns* — no computer can store an infinite vector.
  - The calculus route gives a first-order condition at *every* date — the **Euler equation**

$$u'(c_t) \;=\; \beta\, u'(c_{t+1}) \left[\, f'(k_{t+1}) + 1 - \delta \,\right] \qquad \text{for all } t = 0, 1, 2, \dots$$

  — infinitely many equations, plus a terminal (*transversality*) condition $\lim_{t\to\infty} \beta^t u'(c_t)\, k_{t+1} = 0$.

Keep the Euler equation in mind: Lecture 4 turned it into a neural network's **loss function**; this lecture's sample backups (TD, Q-learning) enforce the same one-step optimality from experience.

Appendix A4 — Adding Uncertainty: Shocks and the Markov Structure

Real economies are hit by shocks — and shocks are what make the sequential problem truly explode. Return to lecture

- **Add productivity risk:** output becomes $y_t = z_t f(k_t)$ with z_t random. A plan can no longer be a fixed list of numbers — the right choice at date t depends on the shocks seen so far, so a plan is a *history-contingent function* $k_{t+1}(z_0, z_1, \ldots, z_t)$. The unknown object explodes combinatorially.

- **Two facts tame it — one per state variable:**
  - **The exogenous shock is Markov:** $\Pr(z_{t+1} \mid z_t, z_{t-1}, \ldots) = P(z_{t+1} \mid z_t)$ — tomorrow's distribution depends on the past only through today (e.g. AR(1) log-TFP, or a finite chain).
  - **The endogenous state has a law of motion:** tomorrow's capital is whatever we choose today from the feasible set $\Gamma(k_t, z_t) = \big[\,0,\ z_t f(k_t) + (1-\delta)k_t\,\big]$ — no memory beyond today.

- **Punchline:** the pair (k_t, z_t) is a *sufficient statistic* for the future — given today's state, history adds nothing. Decisions can be functions of the current state alone: $k_{t+1} = g(k_t, z_t)$.

State variable = the minimal information that makes the future conditionally independent of the past. Exogenous shocks contribute z (by the Markov property); accumulation decisions contribute k (by the law of motion). This *Markov structure* is exactly what recursion exploits next.

Appendix A5 — The Recursive Formulation: The Bellman Equation

One equation in one unknown function — the infinite future compressed into "the value of arriving at a state."

Return to lecture

Let $V(k,z)$ = the best achievable expected lifetime utility *starting from* state (k,z). Then, with resources $y = zf(k) + (1-\delta)k$,

$$V(k,z) \;=\; \max_{0 \le k' \le y} \Big\{ \underbrace{u(y-k')}_{\text{utility today}} + \underbrace{\beta}_{\text{impatience}} \underbrace{\textstyle\sum_{z'} P(z' \mid z)\, V(k',z')}_{\text{expected value of tomorrow's state}} \Big\}.$$

- V turns the whole infinite future into *one number per state* — so optimality becomes a **one-step trade-off**: utility now vs. the value of the state you wake up in tomorrow.
- The solution is a *pair of functions*: the value $V(k,z)$ and the policy $k' = g(k,z)$ attaining the max. Solve them once and you have the answer for *every* initial condition — optimal paths are recovered by iterating g along realized shocks.

Principle of Optimality (Bellman, 1957): "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." — why the one-step view loses nothing.

Appendix A6 — Are the Two Problems the Same? The Four Questions

Before working recursively one must know the rewrite is legitimate. Four questions settle it — stated here, proved in macro theory.

◂ Return to lecture

Write v^* for the value of the sequential problem (SP) and FE for the Bellman functional equation.

1. **SP ⇒ FE (values).** Does v^* satisfy the Bellman equation?
2. **FE ⇒ SP (values).** If some v satisfies the Bellman equation, is $v = v^*$? *Needs a boundedness condition* $\lim_{t\to\infty} \beta^t v(k_t) = 0$ (rules out "bubble" solutions).
3. **SP ⇒ FE (plans).** Does an optimal plan $\{k^*_{t+1}\}$ for the SP attain the maximum in the Bellman equation at every date?
4. **FE ⇒ SP (plans).** Does a plan generated by following the Bellman policy g period after period solve the original SP?

License to work recursively — all four answers are yes under standard assumptions (nonempty feasible set, well-defined discounted sums; the limit condition for questions 2 and 4). Values and plans translate in both directions: solve the Bellman equation and you have solved the original problem. From here on, work with V and g.

Where it is proved: Stokey, Lucas & Prescott (1989, Harvard UP), Ch. 4, §4.1 — four theorems, one per question (Ch. 9 for the stochastic case); also Acemoglu (2009), Ch. 6.

Appendix A7 — What Did We Gain? From Infinite Sequences to One Function

The honest accounting behind "the recursive problem is finite-dimensional" — and why the quotation marks matter.

◂ Return to lecture

	Sequential problem	**Recursive problem**
Unknown	infinite contingent sequence $\{k_{t+1}(z_0, \ldots, z_t)\}_{t \geq 0}$	**one function** $V(k, z)$ (or g) on the state space
Optimality	infinitely many Euler equations + transversality	one functional equation (Bellman)
What you get	one optimal path from one k_0	a decision rule for *every* state

The catch — "finite" deserves its quotation marks. We traded infinitely many numbers for *one* unknown, but that unknown is a *function* — itself an infinite-dimensional object. Exact formulas exist only in special cases (the log benchmark). Computing means choosing a **finite representation** of V or g: a table on a grid (Section 2, Lab 3A) · polynomial coefficients · **neural-network weights θ — Lecture 4 and deep RL (Topic 5.3)**.

This slide is why Lectures 4–5 exist: which finite representation survives a large state space (Lecture 4) — and what replaces the full-expectation backup when P cannot even be written down (this lecture)?

Appendix A8 — Why Iteration Works: The Contraction Mapping Theorem

The Bellman equation is solved by a fixed point — and one theorem guarantees simple iteration always finds it.

◂ Back to DP foundations ◂ Return to lecture

- **The Bellman operator** T: feed in *any* candidate value function W, get back the value of behaving optimally for one period against it,

$$(TW)(k,z) = \max_{k' \in \Gamma(k,z)} \Big\{ u\big(zf(k) + (1-\delta)k - k'\big) + \beta \textstyle\sum_{z'} P(z' \mid z)\, W(k', z') \Big\}.$$

  Solving the Bellman equation = finding the **fixed point** $V^* = TV^*$ — the self-consistent guess.
- **Contraction:** T shrinks distances between candidate functions, $\|TV - TW\|_\infty \le \beta \|V - W\|_\infty$.
- **Contraction Mapping Theorem** (Banach, 1922): on a complete metric space, a contraction has
  - a **unique** fixed point V^*;
  - **global convergence**: $V_{n+1} = TV_n \to V^*$ from *any* starting guess V_0;
  - a **geometric rate with an error bound**: $\|V_n - V^*\|_\infty \le \frac{\beta^n}{1-\beta} \|V_1 - V_0\|_\infty$.

Practical reading — VFI cannot fail: start from $V \equiv 0$, apply T repeatedly, and the error shrinks by a factor β every sweep; the bound even says when to stop. Uniqueness makes "the model predicts X" well-posed — one solution to report.

Appendix A9 — Proving a Contraction the Easy Way: Blackwell's Conditions

The CMT needs T to be a contraction — and checking that from the definition is painful. Two simple properties do it instead.

◂ Return to lecture

Blackwell's sufficient conditions (Blackwell 1965, *Ann. Math. Stat.*): let T map bounded functions to bounded functions (sup norm). If T satisfies

1. **Monotonicity:** $V \leq W$ pointwise $\implies TV \leq TW$, and
2. **Discounting:** $T(V + a) \leq TV + \beta\, a$ for every constant $a \geq 0$, some $\beta \in (0, 1)$,

then T is a contraction with modulus β.

- **The Bellman operator passes in two lines:** *monotonicity* — a better continuation value can only raise today's maximized value; *discounting* — adding a constant a to tomorrow's value adds exactly βa today, since $\beta \sum_{z'} P\,(W + a) = \beta \sum_{z'} P\,W + \beta a$.
- So the growth model's T *is* a contraction — the CMT applies and VFI converges. $\beta < 1$ is not just taste: discounting is what makes the whole machine converge.

Fine print: Blackwell assumes bounded functions and $\log c$ is unbounded — standard fixes (compact state space, weighted norms) preserve all conclusions (Stokey, Lucas & Prescott 1989, Ch. 4).

Appendix A10 — Properties That Make Computation Work

Existence is not enough — computation leans on the value function being continuous, concave, and differentiable. Each property has a theorem.

◂ Return to lecture

- **Continuity — Theorem of the Maximum** (Berge, 1963; French original 1959): if the period objective is continuous and Γ is a continuous, compact-valued correspondence, then V is **continuous** and the policy correspondence is well-behaved (nonempty, compact-valued, upper hemi-continuous).
- **Concavity:** add strictly concave u and a convex feasible set $\Longrightarrow$ V is **strictly concave** and the policy is a **single-valued, continuous function** g.
- **Differentiability** (Benveniste & Scheinkman 1979, *Econometrica*): at interior solutions V is differentiable, with envelope condition $V_k(k,z) = u'(c)\left[\, zf'(k) + 1 - \delta \,\right]$ — combine it with the first-order condition and the **Euler equation** of A3 drops out.

Property	What it buys the computer
V continuous	interpolating between grid points is legitimate (Lab 3A)
V strictly concave	the max is unique; policies are monotone; hill-climbing is safe
V differentiable	FOC/Euler-residual methods are well-posed — the losses Lecture 4 autodiffs

Appendix A11 — The Complete Logical Chain, and Where to Learn More

Every link, one line each — this is the entire foundation the lecture stands on.

Return to lecture

Result	What it guarantees
Markov state (k, z)	future depends on the past only through today $\Rightarrow$ recursion possible
Principle of Optimality + SLP §4.1	sequential $\equiv$ recursive — values and plans, both directions
Contraction Mapping Thm. (Banach)	V^* exists, is unique; VFI converges geometrically
Blackwell's conditions	the contraction property verified in two lines
Theorem of the Maximum (Berge)	V, g continuous $\Rightarrow$ grids and interpolation sound
Concavity + envelope (B–S 1979)	unique policy + Euler equation $\Rightarrow$ FOC losses, autodiff

Where to learn it properly:

- Stokey, Lucas & Prescott (1989, Harvard UP), *Recursive Methods in Economic Dynamics*, Ch. 3–4 (deterministic), Ch. 9 (stochastic); Ljungqvist & Sargent, *Recursive Macroeconomic Theory* (MIT Press), Ch. 3–5.
- Acemoglu (2009), *Introduction to Modern Economic Growth* (Princeton UP), Ch. 6 and 16; QuantEcon (Sargent & Stachurski), dynamic programming lectures at `quantecon.org` — free, and all code runs locally in Python (no Colab needed).

Hand-back. In low dimension this toolkit is airtight: recursion is licensed, iteration converges, the solution is smooth. The *only* open issues: representing one infinite-dimensional function when the state space is large, and knowing the transition model P at all — exactly the two walls this lecture's DP-limits act runs into.

Return to the lecture