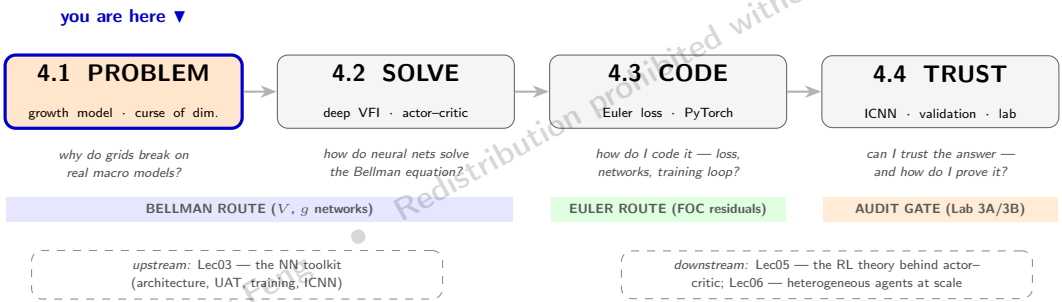
AI for Economic Research: Dynamic Models, Language, and Agents
Lecture 4.1: Motivation & the Optimal Growth Running Example

Zhigang Feng

2026-07-07 20:02:31

Lecture 4 in One Map

One lecture, four decks, one running example: solve the optimal growth model with deep learning — pose the problem, solve it two ways, code it, and prove the answer right.



Act 4.1 (here) *poses the problem: the workhorse growth model and its Bellman equation, why grid methods hit the N^d wall, and the closed-form benchmark plus grid VFI (Lab 3A) that will serve as ground truth for every neural method that follows.*

Motivation: Why Deep Learning for Macro Models?

The workhorse growth model — and the N^d wall every grid method hits

The Workhorse: The Optimal Growth Model

Nearly every quantitative macro model is a variation on one core structure — the optimal (neoclassical) growth model.

▸ New to dynamic macro? Appendix: Solow → Bellman

A planner maximizes lifetime welfare:

$$\max_{\{c_t,\, k_{t+1}\}} \sum_{t=0}^{\infty} \beta^t \, u(c_t)$$

subject to the economy's resource constraint:

$$c_t + k_{t+1} = \underbrace{f(k_t) + (1-\delta)k_t}_{\text{resources } y_t}, \quad k_0 \text{ given}$$

Cobb–Douglas output $f(k) = Ak^{\alpha}$; resources = output + undepreciated capital, split between consumption c_t and next-period capital k_{t+1}.

Anatomy of the model:

Preferences	concave u, discount $\beta \in (0,1)$
Technology	$f(k) = Ak^{\alpha}$, deprec. δ
State	capital k (inherited)
Control	saving $k' = g(k)$
Solve for	value $V(k)$, policy $g(k)$

Two primitives — tastes (u, β) and technology (f, δ) — plus one accounting identity generate the entire dynamic problem.

The Workhorse: Recursive (Bellman) Formulation

The infinite-horizon plan collapses to a one-step problem in a single state.

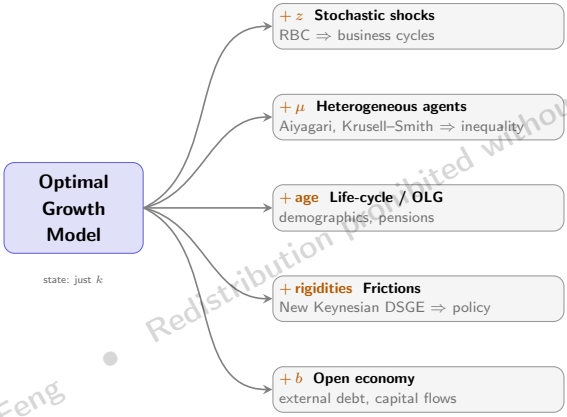
- **From sequence to recursion**: rather than choose the entire path $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ at once, ask only — given capital k today, how much should we save?
- **Bellman equation** — today's value = current utility + discounted continuation value:

$$V(k) = \max_{0 \le k' \le y} \left\{ u(y - k') + \beta V(k') \right\}, \qquad y = f(k) + (1 - \delta)k$$

- The solution is a *pair*: the value function $V(k)$ and the policy $k' = g(k)$. Consumption — and, in market versions, prices — follow from g.
- **A single state variable** k makes this cheap to solve on a grid — and that very tractability is what breaks once the model grows richer.

Running Example (Section 2) specializes this to the analytically solvable case ($A = 1,\ \delta = 1,\ u = \log$), giving the closed-form policy $c^*(y) = (1 - \alpha\beta)\, y$ — our ground truth for every neural method.

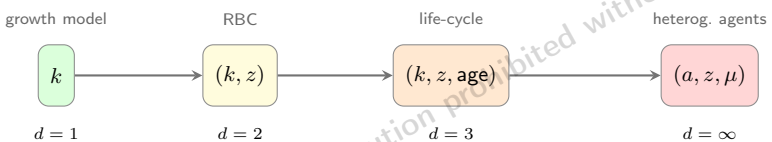
Why a Workhorse? One Model, Many Directions



Same skeleton, richer flesh — the trunk is the optimal growth model, each realistic direction a branch (cf. the extensions map in Lecture 1). Master the trunk once and every branch is within reach — but *each branch adds state variables*, and more state means "harder to compute."

The Catch: Every Direction Adds State Variables

- Each realistic ingredient enlarges the **state vector** the solution must carry:



- A grid needs N^d points — cost **explodes** with every added dimension (Bellman's *curse of dimensionality*).
- The workhorse is trivial at $d = 1$; its research-frontier descendants quickly reach $d \geq 10$ — or carry a whole *distribution* μ (infinite-dimensional).
- **This is the wall.** The rest of Topic 4.1 asks: can *deep learning* scale where grids cannot?

The Computational Frontier in Macro

- Quantitative macroeconomics solves dynamic optimization problems:

$$V(k) = \max_{k'} \Big\{ u\big(f(k) - k'\big) + \beta\, V(k') \Big\}$$

- **Standard toolkit**: Value Function Iteration (VFI), Euler equation methods, perturbation

- These methods have served the profession well for decades
- But they face fundamental limitations as models grow richer:
  - More state variables (capital, bonds, housing, human capital...)
  - Heterogeneous agents (wealth distributions, idiosyncratic risk)
  - Aggregate uncertainty (business cycles in HA models)

- **This lecture**: How deep learning and reinforcement learning offer a fundamentally different approach

Course arc: Lec02 T2 framed AI as **cognitive capital** *(the why); Lec03 delivered NN architectures and training algorithms (the what); Lec04 closes the loop — turning those tools on the optimal-growth Bellman equation (the how).*

The Naive Approach: Grid-Based Methods

- **Value Function Iteration on a grid**:
  1. Discretize the state space: $\{k_1, k_2, \ldots, k_N\}$
  2. For each grid point, solve $\max_{k'}\{u(f(k_i) - k') + \beta V(k')\}$
  3. Iterate until convergence

- **This works well** when:
  - The state space is low-dimensional (1–3 variables)
  - The functions are smooth
  - We can afford to evaluate V at every grid point

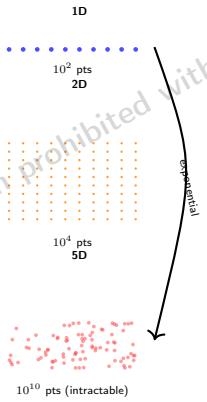
- Alternatives like Chebyshev polynomials and finite elements improve efficiency but share the same fundamental scaling problem

The Curse of Dimensionality: The N^d Explosion

A grid places N points *per dimension*, so covering d dimensions costs N^d points. With $N = 100$:

- $d = 1$: 10^2 pts (trivial)
- $d = 2$: 10^4 pts (manageable)
- $d = 5$: 10^{10} pts (infeasible)
- $d = 10$: 10^{20} pts (absurd)

Linear in dimension, but exponential in resolution — every extra state variable multiplies the cost by another factor of N.



Dots are a schematic sample, not the literal count ($N = 100$ per axis).

Why Grids Break: Real Macro Models Are High-Dimensional

- **Real macro models** routinely have $d \geq 10$ — and often far more:
  - Krusell & Smith (1998), JPE: individual k + aggregate K + shocks
  - Life-cycle: age + assets + earnings + health + $\cdots$
  - HA + aggregate risk: the *whole* wealth distribution (infinite-dimensional)
- *Canonical HA benchmarks:* Aiyagari (1994), QJE; Krusell & Smith (1998), JPE.
- These are the curse-of-dimensionality test cases **Lec06** attacks with DL/RL methods; **Lec09** reuses Aiyagari as its running example.
- **Grid-based methods cannot scale to these problems** — at 10^{20} points, no amount of hardware helps.

Beyond Grids: Other Classical Limitations

- **Perturbation methods** (linearization around steady state):
  - Fast and accurate locally
  - But may lose accuracy far from steady state
  - Struggle with non-linearities, occasionally binding constraints, regime switches
- **Projection methods** (Chebyshev, finite elements):
  - Globally accurate for smooth problems
  - Require structured grids or node sets
  - Still suffer from the curse of dimensionality
- **Kinks and non-convexities**:
  - Borrowing constraints, irreversible investment, discrete choices
  - First-order conditions fail or require complex root-finding
- We need function approximators that scale gracefully with dimension and handle irregular structures

The Deep Learning Opportunity: Approximation That Scales

- **Neural networks are universal function approximators**:
  - With enough hidden units, a network can represent *any* continuous function on a compact set (Cybenko 1989; Hornik et al. 1989).
- **And they scale with dimension**:
  - Parameter count grows *polynomially* with the input dimension, not exponentially (Barron 1993).
  - A grid needs N^d points; a network needs a weight count that grows gently with d.
- **So represent the value or policy function as a network** — $V_\theta(k)$ or $g_\theta(k)$ — and *learn* the weights θ; there is no grid to discretize.
- This is the escape from the curse of dimensionality: replace "store a value at every grid point" with "fit one smooth function that generalizes."

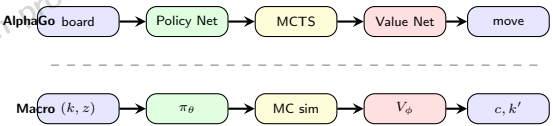
The Deep Learning Opportunity: Differentiation & Hardware

- **Automatic differentiation gives gradients for free**: Euler equations, FOCs, and Jacobians are differentiated straight from the code (PyTorch) — no hand-derived, hand-coded derivatives.
- **Why GPUs accelerate deep learning**:
  - A network's forward and backward passes are *dense matrix multiplications* (Wx); training repeats billions of multiply–adds.
  - A GPU packs *thousands* of arithmetic cores (a CPU has a handful) that apply one operation to many numbers at once (SIMD), fed by very high memory bandwidth — exactly what matrix multiplication needs.
  - The work is *parallel across the batch*: thousands of sampled states (k, z), or a whole panel of simulated agents, run at once — so one gradient step over thousands of states costs about what a CPU spends on a handful.
- **The key insight**: recast the macro model as a *learning problem* — the policy or value function is the object to be learned, trained by stochastic gradient descent on sampled states.

The Inspiration: AlphaGo and Economic Models

Silver et al. (2016, Nature), "Mastering the Game of Go with Deep Neural Networks and Tree Search."

- **AlphaGo** (DeepMind, 2016) solved a problem with $\sim 10^{170}$ board states.
- It did *not* enumerate the state space or solve the full Bellman equation on a grid.
- **It used two neural networks**:
  - **Policy Net (Actor)**: "Intuition" — which move to play.
  - **Value Net (Critic)**: "Judgment" — who is winning.
- **Lesson for economics**: when the state space is too vast (e.g., wealth distribution in a HA model), use the same two-network approach.
- **This lecture**: apply the actor-critic idea to the optimal growth model as a proof of concept, then discuss scaling.



Same architectural pattern; different state space.

Reframing Macro as a Machine Learning Problem

- The optimal growth model can be recast as a **learning problem**:
  - The decision rule $g(k)$ is approximated by a neural network
  - The first-order conditions provide a natural **loss function**
  - Modern optimization (Adam, SGD) and automatic differentiation drive training

- **Two complementary approaches**:
  1. **Deep Value Function Iteration**: Train networks to satisfy the Bellman equation (reinforcement learning flavor)
  2. **Euler Equation Minimization**: Train a network to satisfy the FOCs (supervised learning flavor)

- Both use the same building blocks:
  - Neural network function approximation
  - Stochastic gradient descent
  - Monte Carlo sampling of state space

Running Example: The Optimal Growth Model

A closed-form benchmark and grid VFI: the ground truth the networks must beat

Optimal Growth Model: Setup

Recall the workhorse from Section 1 — here we pin down the analytically solvable special case.

- **Objective**: Maximize discounted lifetime utility:

$$\max_{\{k_{t+1}\}} \sum_{t=0}^{\infty} \beta^t u\Big(f(k_t) - k_{t+1}\Big)$$

- **Production function**: $f(k_t) = k_t^{\alpha}$
- **Resource constraint**: Consumption $c_t = f(k_t) - k_{t+1}$
- **Depreciation**: $\delta = 1$ (full depreciation) in our benchmark; adjustable for general cases
- **Steady-state capital**:

$$k_{ss} = \left(\frac{1}{\alpha\beta}\right)^{\frac{1}{\alpha-1}} = (\alpha\beta)^{\frac{1}{1-\alpha}}$$

- **Why this model?** Simple enough for an analytical solution (ground truth), yet contains all the essential structure of richer models

Bellman Equation and VFI

- **Recursive formulation** (Bellman equation):

$$V(k) = \max_{k'} \Big\{ u\big(f(k) - k'\big) + \beta\, V(k') \Big\}$$

- **Value Function Iteration**:
  1. Start with initial guess $V^0(k)$
  2. Update: $V^{n+1}(k) = \max_{k'} \{u(f(k) - k') + \beta V^n(k')\}$
  3. Repeat until $\|V^{n+1} - V^n\| <$ tolerance

- **Contraction Mapping Theorem (Banach)**: Under standard assumptions, the Bellman operator T is a contraction with modulus β
  - Guarantees convergence to a unique fixed point V^*
  - VFI converges *geometrically* at rate β
  - This is the theoretical foundation we build on

▸ Full statements: CMT + Blackwell (Appendix)

Stochastic Extension

- Add a productivity shock z:

$$v(y) = \max_{0 \le c \le y} \Big\{ u(c) + \beta \, \mathbb{E}\big[v(y')\big] \Big\}$$

- Transition: $y' = f(y - c) \cdot z$, where z is i.i.d. log-normal
- **Analytical solution** for log utility $u(c) = \ln(c)$ and Cobb-Douglas $f(k) = k^{\alpha}$:

$$V^*(y) = A + B \ln y \quad \text{(value function)}$$
$$c^*(y) = (1 - \alpha\beta)\, y \quad \text{(optimal policy: consume fixed fraction)}$$

- This closed-form solution serves as our **ground truth** for validating all neural network methods
- Lab 3A builds this benchmark via classical VFI with interpolation

VFI with Interpolation: The Classical Algorithm

- Since y is continuous, we use interpolation on a grid:

  1. **Initialize**: Grid $\{y_1, \ldots, y_N\}$, initial guess v^0
  2. **Interpolate**: Create function $\hat{v}(y)$ connecting (y_i, v_i^0) pairs
  3. **Bellman update**: For each y_i:

$$v_i^{\text{new}} = \max_c \left\{ \ln(c) + \beta \sum_j \hat{v}\big(f(y_i - c)\, z_j\big)\, \phi(z_j) \right\}$$

  4. **Check convergence**: If $\|v^{\text{new}} - v^0\| < \text{tol}$, stop; else repeat

- This is the **benchmark** algorithm implemented in Lab 3A
- Works beautifully for 1D... but the grid makes it impractical for higher dimensions

Bridge: The Problem Is Set — Now Replace the Grid

Act 4.1 posed the problem. The optimal growth model is the workhorse of quantitative macro, its Bellman equation is the object every solver must satisfy — and grid-based VFI, exact in one dimension, collapses under the N^d explosion.

- You now hold the benchmark: a closed-form special case and grid VFI (Lab 3A) — the ground truth every neural solution in this lecture must reproduce.
- **Next (4.2):** swap the grid for parameters — approximate V and g with neural networks and train them on the Bellman equation itself, actor–critic style.

Arc: **4.1 PROBLEM** *(growth model + curse)* $\rightarrow$ *4.2 SOLVE (deep VFI + actor–critic)* $\rightarrow$ *4.3 CODE (Euler + PyTorch)* $\rightarrow$ *4.4 TRUST (structure + validation).*

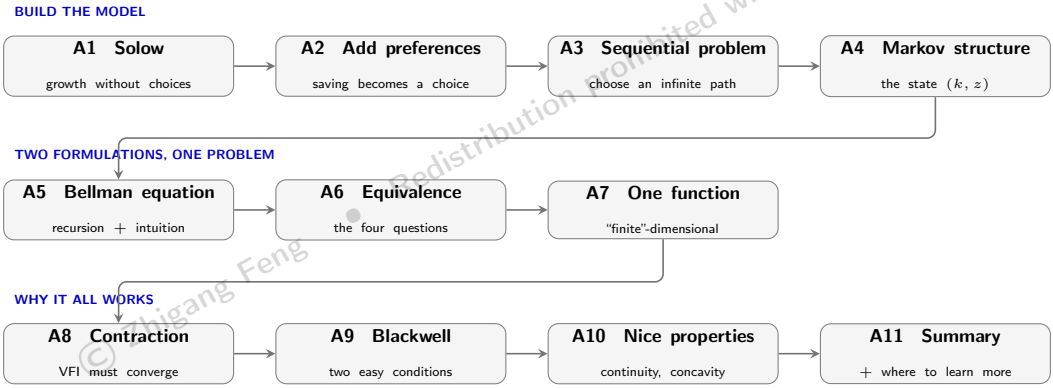
Appendix: Dynamic Macro Foundations

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

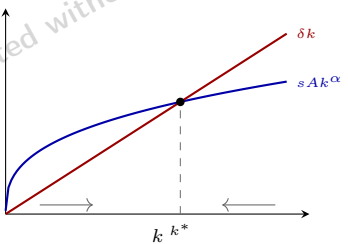
Technology (per-worker terms): output from capital, $y_t = f(k_t) = Ak_t^{\alpha}$, $\alpha \in (0,1)$ (diminishing returns).

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* of this deck's opening slides. 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* — and is the lecture's ground truth.

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: it returns as a **loss function** in Topic 4.3, where a neural network is trained to drive its residual to zero.

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 — **this lecture**.

This slide is why Lecture 4 exists: which finite representation survives a large state space? The four acts answer it.

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 VFI ◂ 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 Topic 4.3 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 issue: representing one infinite-dimensional function when the state space is large — exactly where the lecture picks up.

Return to the lecture