



- Dynamic Programming Recursive Formulations
- Investment Choice
- Coding Structure and Basics

## Dynamic Programming and Investment

### Essential Reading:

1. **Adda and Cooper:** Dynamic Economics: Quantitative Methods and Applications - Chapter 1.

## DP - Revisiting Investment

We will apply **Dynamic Programming** to the **adjustment cost model**.

The firm's optimization problem is as follows:

$$\max_{K_{t+1}} E_0 \sum_{t=0}^{\infty} \beta^t \left[ \pi(\theta_t, K_t) - (K_{t+1} - (1 - \delta)K_t) - \frac{\phi}{2}(K_{t+1} - (1 - \delta)K_t)^2 \right]$$

where  $\frac{\phi}{2}\mathcal{I}_t^2$  is the convex adjustment cost function, and we have substituted the capital accumulation constraint for investments

$$\mathcal{I}_t = K_{t+1} - (1 - \delta)K_t$$

where  $K_0$  and  $\theta_0$  is given.

This is a sequential constrained optimization problem. Your task is to find a sequence of  $\{K_{t+1}\}_{t=0}^{\infty}$  that maximizes firm value subject to the capital accumulation constraint.

# Recursive Investment Problem

The **recursive functional equation** of the adjustment cost model is given by the Bellman equation as:

$$V(\theta, K) = \max_{K'} \left[ \pi(\theta, K) - (K' - (1 - \delta)K) - \frac{\phi}{2}(K' - (1 - \delta)K)^2 + \beta \mathbb{E} V(\theta', K') \right] \quad (1)$$

Transforms the problem to finding a **function** rather than a sequence.

- Value function -  $V(\theta, K)$
- Policy functions -  $K'(\theta, K)$  and  $I(\theta, K)$ .
- Expectation here is conditional on current state -  $\mathbb{E}[\cdot] = \mathbb{E}_{\theta'|\theta}$ .

## Productivity as a random variable

The variable  $\theta$  is actually a **time varying stochastic process**, even though we do not state it explicitly.

For example,  $\theta$  could follow an **autoregressive process** of order one AR(1):

$$\theta_{t+1} = \rho\theta_t + \epsilon_{t+1}, \quad \text{where } \epsilon \sim N(0, 1)$$

Here the next period value,  $\theta'$ , depends on **its previous period value**,  $\theta$ , and a **random error** normally distributed with mean zero and variance 1.

This is one way to model the unpredictable (random) evolution of technological productivity/innovation.

# Recursive Lagrangian with 1 Constraint

**Recursive constrained optimisation:**

$$\begin{aligned} V(\theta, K) = \max_{K', \mathcal{I}} & \left\{ \pi(\theta, K) - \mathcal{I} - \frac{\phi}{2}(\mathcal{I})^2 + \beta \mathbb{E} V(\theta', K') \right\} \\ \text{s.t. } & K' = (1 - \delta)K + \mathcal{I} \end{aligned} \quad (2)$$

**Recursive Lagrangian with single constraint:**

$$\mathcal{L} = \pi(\theta, K) - \mathcal{I} - \frac{\phi}{2}(\mathcal{I})^2 + \beta \mathbb{E} V(\theta', K') - q(K' - (1 - \delta)K - \mathcal{I}) \quad (3)$$



# Recursive Lagrangian with 1 Constraint

**First order conditions (FOCs)**  $(\mathcal{I}, K')$  should be familiar!:

$$\mathcal{L} = \pi(\theta, K) - \mathcal{I} - \frac{\phi}{2}(\mathcal{I})^2 + \beta \mathbb{E} V(\theta', K') - q(K' - (1 - \delta)K - \mathcal{I}) \quad (4)$$

$$[\mathcal{I}] : \quad -1 - \phi \mathcal{I} = q = 0 \quad (5)$$

$$[K'] \quad \beta \mathbb{E} \left[ \frac{\partial V(\theta', K')}{\partial K'} \right] - q = 0 \quad (6)$$

$$[EC :] V_K = \pi_K(\theta, K) + q(1 - \delta) \quad (7)$$

$$[ST :] V_{K'} = \pi_{K'}(\theta', K') + q'(1 - \delta) \quad (8)$$

Remember, by stationarity (eg)  $V(1, 10)$  is  $V(1, 10)$  no matter when in time, so I can simply roll the EC equation forward by one period

# Recursive Lagrangian with 1 Constraint

**First order conditions (FOCs)**  $(\mathcal{I}, K')$  should be familiar!:

Marginal Benefit = Marginal Cost (price and adjustment cost)

$$q = (1 + \phi\mathcal{I})$$

Marginal Benefit = Expected Marginal Value from profits over useful lifetime

$$q = \beta \mathbb{E} \left[ \frac{\partial V(\theta', K')}{\partial K'} \right] = \beta \mathbb{E} [\pi_{K'}(\theta', K') + q'(1 - \delta)]$$

The firm manager solves the DP in eq. (1) by choosing the next period level of capital  $K'$ .

The solution is given by the FOC w.r.t  $K'$ :

$$\frac{\partial V(\theta, K)}{\partial K'} = 0$$

$$\Rightarrow -(1 + \phi\mathcal{I}) + \beta \mathbb{E} V_{K'}(\theta', K') = 0$$

## Investment Decision: MC = MB

Thus, the optimal investment decision is based on the following condition:

$$\underbrace{1 + \phi\mathcal{I}}_{\text{Marginal Cost}} = \underbrace{\beta\mathbb{E}[V_{K'}(\theta', K')]}_{\text{Expected discounted marginal benefit}}$$

- The left side of this condition is the marginal cost of capital accumulation and includes the direct cost of buying capital and the marginal adjustment cost.
- The right side indicates the expected and discounted marginal gains given by the derivative (change) in the value of the firm.

The expected discounted marginal value of the firm is also the Marginal Q.

$$q = \beta\mathbb{E}[V_{K'}(\theta', K')] = 1 + \phi\mathcal{I}$$

## Using Stationarity and Envelope Condition

We can take the **derivative of today's** value function

$$\frac{\partial V(\theta, K)}{\partial K} = \pi_K(\theta, K) + (1 - \delta)(1 + \phi \mathcal{I}) \quad (9)$$

By **stationarity**, we can roll one period forward (the function itself doesn't move around)

$$\frac{\partial V(\theta', K')}{\partial K'} = \pi_{K'}(\theta', K') + (1 - \delta)(1 + \phi \mathcal{I}') \quad (10)$$

Now we have the RHS as well

## Recursive Problem with Substitution of the Constraint

We can follow a second way to solve the same problem. Plug the constraint in directly:

$$V(\theta, K) = \max_{K'} \left\{ \pi(\theta, K) - (K' - (1 - \delta)K) - \frac{\phi}{2}(K' - (1 - \delta)K)^2 + \beta \mathbb{E} V(\theta', K') \right\} \quad (11)$$

$$[K'] : \quad -1 - \phi(K' - (1 - \delta)K) + \beta \mathbb{E} \left[ \frac{\partial V(\theta', K')}{\partial K'} \right] = 0 \quad (12)$$

$$1 + \phi \mathcal{I} = \beta \mathbb{E} \left[ \frac{\partial V(\theta', K')}{\partial K'} \right]$$

We get the part in square-brackets by envelope condition plus stationarity:

$$[EC :] V_K = \pi_K(\theta, K) + (1 - \delta)(1 + \phi \mathcal{I}) \quad (13)$$

$$[ST :] V_{K'} = \pi_{K'}(\theta', K') + (1 - \delta)(1 + \phi \mathcal{I}') \quad (14)$$

Define

$$\mathbf{q} = (1 + \phi \mathcal{I})$$

$$[FOC K'] : \quad (1 + \phi \mathcal{I}) = \beta \mathbb{E} [\pi_{K'}(\theta', K') + (1 - \delta)(1 + \phi \mathcal{I}')] \quad (15)$$

$$\mathbf{q} = \beta \mathbb{E} [\pi_{K'}(\theta', K') + (1 - \delta)\mathbf{q}'] \quad (16)$$

## Investment Decision - Marginal Q

- Taking the derivative one period forward and substituting back in the FOC we get the investment decision condition,

$$1 + \phi \mathcal{I} = \beta \mathbb{E}[\pi_{K'}(\theta', K') + (1 - \delta)(1 + \phi \mathcal{I}')] ]$$

where subscripts denote partial derivatives and primes denote next period values.

- In terms of marginal Q,

$$q = \beta \mathbb{E}[\pi_{K'}(\theta', K') + (1 - \delta)q']$$

- **Marginal Cost today** equals expected discounted **additional profits** tomorrow and the value of **non-depreciated capital** priced at  $q'$ .  
See why  $q$  can also be interpreted as the "shadow price" or "shadow value" of capital.

# Investment Decision - Expectation, Probabilities

- The manager needs to take into account **productivity shocks**,  $\theta$ , when deciding how much to invest, and **they become an argument of the value function**.
- We use **conditional expectations** using a **transition matrix** of probabilities:

$$Prob(\theta_{t+1} = \theta_j | \theta_t = \theta_i) = P_{ij}; i^{th} row, j^{th} col \text{ e.g. } P(\theta_H | \theta_L) = P_{LH}$$

- The manager thus **weights different possible scenarios** in the future using their **associated probabilities** and takes an average value of these.

$$\mathbb{E}_{\theta'|\theta} V(\theta', K') = \sum_{\theta'} P(\theta'|\theta) \times V(\theta', K')$$

- The implication here is that its not just **current productivity shocks** that impact firm value but **also future uncertainty**.
- See Mathematical Toolkit.pdf for more on transition matrices



# Solving Recursive Models Numerically

# Value Function Iteration: The Big Picture

## Typical Functional Equation:

$$V(\text{state}) = \max_{\text{choices}} \{ \text{payoff} + \beta \mathbb{E} V(\text{state}') \} = \mathcal{T}[V(\text{state})] \quad (17)$$

**Bellman operator:** everything we do on the right side <sup>1</sup>: function in  $\rightarrow$  function out

- Blackwell's sufficient conditions power VFI on the computer:

(A1) **Discounting:** With  $\beta < 1$ , Bellman operator **shrinks distance between functions**

(A2) **Monotonicity:** **Preserves rankings** across functions  $V > W \rightarrow \mathcal{T}[V] > \mathcal{T}[W]$

(A1+2) guarantee unique **fixed point** in function-space exists:  $V^* = \mathcal{T}[V^*]$

- Iteration  $V_{n+1} = \mathcal{T}[V_n]$  **converges to**  $V^*$  from **any starting point**
- VFI is just:  $V_0 \rightarrow \mathcal{T}(V_0) \rightarrow \mathcal{T}(\mathcal{T}(V_0)) \rightarrow \mathcal{T}(\mathcal{T}(\mathcal{T}(V_0))) \rightarrow \dots$

---

<sup>1</sup>you know some other **operators** already (eg) the derivative maps function  $3x^2 \rightarrow 6x$

## The Algorithm:

1. **Grids:** Discretize state/action spaces, exogeneous state process  $P(s'|s)$  matrix
2. **Guess** initial  $V_0$ . Some guesses are better than others!
3. **Iterate:**  $V_{n+1}(\text{state}) = \max_{\text{choice}} \{\text{payoff} + \beta E[V(\text{state}')]\} = \mathcal{T}(V_n)$
4. **Stop** when  $\|V_{n+1} - V_n\| < \text{tolerance}$

Set your grids and parameters first:

- Set scalar **parameters**,  $\alpha, \beta, \dots$ , etc.
- Productivity lives on a discrete grid  $z \in \mathcal{Z}$ , with  $N_z$  points
- Capital will be chosen from a grid:  $(k, k') \in \mathcal{K} = \text{kgrid}$
- value function will have dimensions  $(N_z, N_k)$
- You will need a matrix  $V$  and a matrix  $V_{new}$
- Set bounds on capital grid:  $k_{min}, k_{max}$
- `kgrid = linspace( $k_{min}, k_{max}, N_k$ ), eg linspace(0.01, 50, 101)`
- set convergence tolerances, eg `tol = 1e-6` and `maxiter = 500`
- prefill you polices: `policy_kp = zeros(1, N_k)`, `policy_inv = zeros(1, N_k)`
- Assume  $z'$  follows AR(1) process with transition probability matrix  $P$

Search for optimal policy. The grid is your menu of options.

- for each  $k \in \mathcal{K}$ , need to find optimal  $k'$  (which also lives in `kgrid`)
- let **i** denote loop index,  $ik$ ,  $iz$  etc; let **ikp** be prime

```
for iz = 1:Nz
    for ik = 1:Nk ...
```
- need to evaluate different choices of  $k'$ , and account for probability of productivity process in  $\beta E_{z'|z} V(z', k' \in \text{kgrid})$
- max can be done with loops to brute force a solution and ignore max functions

## Solution Method - Finding the optimal choice

For example, an investment model:

for all  $(z, k) \in \mathcal{Z} \times \mathcal{K}$  :

$$V(z, k) = \max_{k' \in \mathcal{K}} \left\{ zk^{\alpha} - (k' - (1 - \delta)k) + \beta \sum_{z'} P(z'|z) V(z', k') \right\}$$

- state var  $(k)$  lives on the kgrid
  - state var  $(z)$  lives on the zgrid
  - choice var  $(k')$  must be chosen from the same kgrid
- ⇒ Just a question of looping over all state space  $(z, k)$  and checking  $k'$  to find  $k'^*(z, k)$ , and weighting with probabilities for  $z'|z$

## Skeleton Code

```

while diff > tol && iter < max_iter
for iz = 1:Nz
    for ik = 1:Nk
        Vbest = -1e9 % initialize very low value
        for ikp = 1:Nk
            profit = zgrid(iz) * kgrid(ik)^alpha
                    - kgrid(ikp) + (1-delta)*kgrid(ik)
            %>> can add profit restriction here: if xyz
            cont_value = beta * sum(P(iz,:),) .* V(:,ikp))
            Vtemp = profit + cont_value
            if Vtemp > Vbest
                Vbest = Vtemp
                policy_kp(iz,ik) = kgrid(ikp)
            end
            %>> end condition
        end
        Vnew(iz,ik) = Vbest
    end
end
diff = max(abs(V(:) - Vnew(:)));
V = Vnew;
iter = iter + 1;
end

```



for all  
 $\underbrace{(z,k) \in \mathcal{Z} \times \mathcal{K} :}_{\text{theloops}}$

$$V(z, k) = \max_{k' \in \mathcal{K}} \left\{ \underbrace{zk^\alpha - (k' - (1 - \delta)k)}_{\text{profit}} + \beta \underbrace{\sum_{z'} P(z'|z) V(z', k')}_{\text{sun}(P(iz,:), *) * V(:, ikp))} \right\}$$