Lecture I: Dynamic Programming

Richard Audoly

ECS506: PhD Macroeconomics I

Fall 2021

Course Overview (first half)

- The main goal for this part of the course is to introduce you to dynamic programming and some applications:
  1. Savings problem
  2. Search models of the labor market
- Emphasis on actually solving these models with the computer
- All feedback very much appreciated: this part of the course is given for the first time
- Prof. Doppelhofer will take over for the second part of the course

Syllabus redux

- Lectures (Friday) and "feedback sessions" (Tuesday)
- Participation is highly encouraged
- Weekly problem sets: Submit 3 out of 5 to sit the exam
- First problem set due on Monday
- Final exam: open book but will require you to code

More details can be found in the Syllabus on Canvas

This week: Dynamic Programming

Intro to Recursive Formulation

Markov Processes and Markov Chains

Recursive Formulation with Uncertainty

This week: Dynamic Programming

Intro to Recursive Formulation

Markov Processes and Markov Chains

Recursive Formulation with Uncertainty

Dynamic programming?

What? "simplifying a complicated problem by breaking it down into simpler sub-problems in a **recursive** manner" (Wikipedia)

Why?

- Most inter-temporal problems can be expressed this way
- Clarifies what is given (the state variables, more on this in a minute) and what is a choice
- Formulation that's most amenable to numerical solution (ie, "programming")
- Essential to read almost any paper with a structural model in macro, IO, labor, etc.

Why called "Dynamic Programming"?

R. Bellman (who coined the term in the 1940s-1950s):

It ["dynamic"] *also has a very interesting property as an adjective, and that is it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.*

Full quote `here`.

A simple problem

- An agent chooses sequences $\{c_t, k_{t+1}\}_{t=0}^{\infty}$

$$\max \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$c_t + k_{t+1} \leq f(k_t), \quad t = 0, 1, 2 \ldots \qquad \text{(Budget constraint)}$$
$$k_0 \qquad \text{(Initial conditions)}$$

- This is the problem in **sequential** form.

A minimal environment

- $\beta \in (0,1)$ is a discount factor
- $f(.)$ is some production technology
- Implicitly, full depreciation of capital in each period. More generally, we would have

$$c_t + k_{t+1} \leq f(k_t) + (1-\delta)k_t, \quad t = 0, 1, 2, \ldots$$

with depreciation rate $\delta \in (0,1]$

Remark: Doesn't really matter whether we are talking about a representative household for now

We know how to solve this one

The Lagrangean is given by

$$\mathcal{L} = \sum_{t=0}^{\infty} \beta^t u(c_t) + \lambda_t \Big[f(k_t) - c_t - k_{t+1} \Big].$$

The optimal choice $\left(\tilde{c}_t, \tilde{k}_{t+1}\right)$ must satisfy the necessary FOCs:

$$\begin{aligned} [c_t]: \quad & 0 = \beta^t u'(\tilde{c}_t) - \lambda_t \\ [k_t]: \quad & 0 = -\lambda_t + \lambda_{t+1} f'(\tilde{k}_{t+1}). \end{aligned}$$

The Euler equation follows directly from eliminating λ_t:

$$u'(\tilde{c}_t) = \beta f'(\tilde{k}_{t+1}) u'(\tilde{c}_{t+1}).$$

Equilibrium conditions

For all t $= 0, 1, \ldots$

1. Euler equation (sufficient): $u'(c_t) = \beta f'(k_{t+1})u'(c_{t+1})$
2. Resource constraint: $c_t + k_{t+1} \leq f(k_t)$
3. Transversality condition (terminal condition):

$$\lim_{T \to \infty} u'(c_T)k_{T+1} = 0$$

OK. But, given a k_0, how should we go about simulating the agent's choice of $\{c_t, k_{t+1}\}_{t=0}^{\infty}$?

We need to put this problem in a more useful format.

Definition: State variable

- A **state variable** s_t (or "state" for short) is a finite vector of variables **pre-determined** at time t.
- Pre-determined means that the variables are given/realized before the agent makes choices. In our example, it means the variables are given at time t before the agent chooses c_t and k_{t+1}.
- Everything we need to know to solve for the agent's problem today and fully determine s_{t+1} so they can solve it again tomorrow

What's the state? The recursive formulation we're going to introduce will make this precise.

Definition: Value function

The value function $V_t(s_t)$ is the expected present discounted value of the agent's utility under the optimal policy for consumption and capital, where this value is computed after the realization of the state s_t, but before the choice of c_t and k_{t+1}.

So at time $t = 0$ we have

$$V_0(s_0) := \max_{\{c_t, k_{t+1}\}} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$c_t + k_{t+1} \leq f(k_t), \quad t = 0, 1, 2, \ldots$$

Note: ":=" (sometimes "≡") means "by definition"

Recursive formulation

Let's split the sum at $t = 1$ in the expression for $V_0(s_0)$:

$$V_0(s_0) = \max_{\{c_0, k_1\}} u(c_0) + \max_{\{c_t, k_{t+1}\}} \sum_{t=1}^{\infty} \beta^t u(c_t)$$

$$= \max_{\{c_0, k_1\}} u(c_0) + \beta \max_{\{c_t, k_{t+1}\}} \sum_{t'=0}^{\infty} \beta^{t'} u(c_{t'+1}) \quad (t = t' + 1)$$

still subject to

$$c_0 + k_1 \leq f(k_0)$$

$$c_t + k_{t+1} \leq f(k_t), \quad t = 1, 2, \ldots$$

The terms in red are exactly $V_1(s_1)$!

Recursive formulation

Substituting the definition for $V_1(s_1)$, we get

$$V_0(s_0) = \max_{\{c_0, k_1\}} u(c_0) + \beta V_1(s_1) \quad \text{s.t.} \quad c_0 + k_1 \leq f(k_0)$$

Remarks:

1. It's now clear that the pre-determined variable is capital, which is enough info to solve the problem "today" and determine the state "tomorrow", so the state is $s_t = k_t$

2. Given some capital level, the problem doesn't depend on calendar time t, so the value function $V(.)$ is time independent (because it's an infinite horizon problem)

Sequential vs Recursive formulation

So we now have two formulations for the same problem:

1. **Sequential formulation**:

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t.} \quad c_t + k_{t+1} \leq f(k_t), \ t = 0, 1, \ldots$$

2. **Recursive formulation aka the Bellman equation**:

$$V(k) = \max_{\{c, k'\}} u(c) + \beta V(k') \quad \text{s.t.} \quad c + k' \leq f(k)$$

Again note we don't need t subscripts in the recursive formulation.

Principle of optimality

- Take a plan $\{c_t\}_{t=0}^{\infty}$ solving the sequence problem. Then it must also solve the Bellman equation
- *"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"* – Bellman (1957) Dynamic Programming
- Similar to sub-game perfection!
- Makes clear the idea of sunk-cost fallacy

They're equivalent indeed

We get the same Euler equation from the recursive formulation

$$V(k) = \max_{k'} u(f(k) - k') + \beta V(k') \quad \Big(c = f(k) - k'\Big).$$

Optimal choices $\Big(c(k), k'(k)\Big)$ must satisfy the necessary FOC

$$0 = -u'\Big(c(k)\Big) + \beta V'\Big(k'(k)\Big).$$

Envelope condition

$$\frac{dV}{dk} = \frac{\partial V}{\partial k} = u'\Big(c(k)\Big) \cdot f'(k)$$

FOC and envelope condition give the same Euler equation:

$$u'\Big(c(k)\Big) = \beta V'\Big(k'(k)\Big) = \beta f'\Big(k'(k)\Big) u'\Big(c(k')\Big)$$

Reminder envelope condition

- Given a function

$$g(x) = \max_y f(x,y) = f(x, y^*(x)) \quad \Big(y^*(x) \text{ maximizes } f(x,.)\Big)$$

we have

$$\begin{aligned}
\frac{dg}{dx}(x) &= \frac{df}{dx}\Big(x, y^*(x)\Big) \\
&= \frac{\partial f}{\partial x}\Big(x, y^*(x)\Big) + \frac{\partial f}{\partial y}\Big(x, y^*(x)\Big) \cdot (y^*)'(x) \\
&= \frac{\partial f}{\partial x}\Big(x, y^*(x)\Big) \quad \text{[why?]} \\
&= \frac{\partial g}{\partial x}(x)
\end{aligned}$$

- In our model: $g = V$ and f is given by

$$(k, k') \longmapsto u(f(k) - k') + \beta V(k')$$

So how is all this progress?

Our problem is now given by the following **Bellman equation**

$$V(k) = \max_{\{c,k'\}} u(c) + \beta V(k') \quad \text{s.t.} \quad c + k' \leq f(k).$$

This is better because:

1. The solution to this equation is a pair of **policy functions** $(c(.), k'(.))$ of k. So given any k_0, it's easy to simulate the optimal path.

2. The Bellman equation gives rise to a fairly intuitive solution algorithm.

Policy functions

- Given current state k, we denote $c(k)$ and $k'(k)$ the optimal choices

$$\begin{aligned} V(k) &= \max_{\{c,k'\}} u(c) + \beta V(k') \quad \text{s.t.} \quad c + k' \leq f(k) \\ &= u(c(k)) + \beta V(k'(k)) \quad \text{with} \quad c(k) + k'(k) = f(k) \end{aligned}$$

- $c(.)$ and $k'(.)$ are the **policy functions**

- It's now easy to simulate the optimal path $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ starting from some k_0:

$$\begin{aligned} t = 0, \quad & k_1 = k(k_0), \quad c_0 = c(k_0) = f(k_0) - k_1, \\ t = 1, \quad & k_2 = k(k_1), \quad c_1 = c(k_1) = f(k_1) - k_2, \\ t = 2, \quad & \ldots \end{aligned}$$

Value Function Iteration

Here's how to numerically solve

$$V(k) = \max_{\{c,k'\}} u(c) + \beta V(k') \quad \text{s.t.} \quad c + k' \leq f(k)$$

1. Build a grid $\{k_1, \ldots, k_n\}$ for the state k and guess an initial value for $V(.)$ on the grid

2. Solve the agent problem in the equation shown above. This gives an updated value for $V(.)$

3. Go back to the first step with new guess for $V(.)$ and iterate until convergence

How do we know it converges? We'll make the conditions precise in the next lecture.

This week: Dynamic Programming

Intro to Recursive Formulation

Markov Processes and Markov Chains

Recursive Formulation with Uncertainty

Introducing uncertainty

- Up to now, the agent's has made choices in a deterministic environment—no uncertainty about tomorrow
- In many economic contexts, both macro and micro, uncertainty about the future is key:
  1. An individual uncertain about her future income
  2. A business uncertain about its future sales
  3. Agents (firms, households) uncertain about the economy as a whole, say, during a pandemic
- Because we now like the recursive formulation a lot, we want to think about introducing shocks that play well in there

Some definitions

Let z_t be a shock (a random variable) affecting the agent's problem, eg, production is now given by $z_t f(k_t)$

- A **history** z^t is the sequence of all past realizations up to t

$$z^t := \left(z_0, z_1, \ldots, z_t\right) = \left(z^{t-1}, z_t\right)$$

- In general,

$$\Pr(z^t) = \Pr(z_t, z^{t-1}) = \Pr(z_t|z^{t-1})\Pr(z^{t-1})$$

- We say that z_t is a **Markov process** if

$$\Pr(z_t|z^{t-1}) = \Pr(z_t|z_0, \ldots, z_{t-1}) = \Pr(z_t|z_{t-1})$$

Markov processes

Process is Markov if the probability of today only depends on yesterday, and not the day before

$$\Pr(z_t|z^{t-1}) = \Pr(z_t|z_{t-1})$$

This fits well into the recursive formulation: z_t summarizes everything the agent needs to know to forecast z_{t+1}

In our basic example with a z_t shock to production, there will be another state z

Some examples of Markov processes

1. Standard AR(1) (auto-regressive of order 1)

$$\ln z_t = \rho_z \ln z_{t-1} + u_t, \quad |\rho_z| < 1, \quad u_t \sim \mathcal{N}(0, \sigma_z)$$

2. A two-state process $z_t \in \{l, h\}$ (l = low, h = high) with the Markov property. So, for example,

$$\Pr(z_t = l | z_{t-1} = h, z_{t-2} = h, \dots) = \Pr(z_t = l | z_{t-1} = h)$$

When a Markov process only takes discrete values, we say it's a **Markov Chain**

How restrictive is the Markov Property?

Less than it seems, actually.

It's possible to redefine the shock so that it's a Markov process, provided it depends on a finite number of past realizations

Imagine that instead the two last periods matter in our high/low example, then we can let

$$\tilde{z}_t := \left(z_t, z_{t-1}\right)$$

so $\tilde{z}_t$ is a Markov Chain with four possible values

$$\tilde{z}_t \in \Big\{(l,l),\ (l,h),\ (h,l),\ (h,h)\Big\}.$$

Markov Chain and transition matrix

A compact representation of our high/low Markov chain is

$$P = \begin{bmatrix} p_{ll} & p_{lh} \\ p_{hl} & p_{hh} \end{bmatrix}$$

where, for example, $0 \leq p_{lh} \leq 1$ denotes the probability to move to "destination" state h (column) given "origin" state l (row)

$$p_{lh} = \Pr(z_{t+1} = h | z_t = l).$$

By definition, rows must sum to one

$$p_{il} + p_{ih} = \sum_{j \in \{l,h\}} \Pr(z_{t+1} = j | z_t = i) = 1, \quad \forall i \in \{l, h\}.$$

Markov transition matrix

Such a matrix is called a **transition matrix**:

- Obviously all these definitions generalize to shocks taking any finite number of values
- Properties of the transition matrix tell you something about the underlying process
- What about state 1 (first row) for the matrix below?

$$P = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.1 & 0.7 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

We'll look more into this in PS1.

This week: Dynamic Programming

Intro to Recursive Formulation

Markov Processes and Markov Chains

Recursive Formulation with Uncertainty

Adding a productivity shock in our framework

Let's go back to our simple example

$$\max_{\{c_t, k_{t+1}\}} \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t.} \quad c_t + k_{t+1} \leq f(k_t), \ t = 0, 1, 2, \ldots$$

But now introduce a productivity shock z_t

$$\max_{\{c_t, k_{t+1}\}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t.} \quad c_t + k_{t+1} \leq z_t f(k_t), \ t = 0, 1, 2, \ldots$$

Can we still write this problem recursively?

Contingent choices

With shocks, the problem is now one of choosing $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ *contingent* on the realization of the states:

$$\max_{\{c_t, k_{t+1}\}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t.} \quad c_t + k_{t+1} \leq z_t f(k_t), \ t = 0, 1, 2, \ldots$$

Let's make the dependence on the history of the states s^t clear. We're now choosing $\{c(s^t), k_{+1}(s^t)\}_{t=0}^{\infty}, \forall s^t$ to

$$\begin{aligned} &\max \sum_{t=0}^{\infty} \beta^t \sum_{s^t | s_0} \pi(s^t | s_0) u(c(s^t)) \\ \text{s.t.} \quad &c(s^t) + k_{+1}(s^t) \leq z(s^t) f(k(s^t)), \ t = 0, 1, \ldots \forall s^t \\ &z(s_0), \ k(s_0) \text{ given} \end{aligned}$$

Recursive formulation again

$$
\begin{aligned}
V(s_0) &:= \max_{c(s^t),k_{+1}(s^t)} \sum_{t=0}^{\infty} \beta^t \sum_{s^t|s_0} \pi(s^t|s_0) u(c(s^t)) \\
&= \max_{c(s_0),k_{+1}(s_0)} u(c(s_0)) \\
&+ \left\{ \max_{c(s^t),k_{+1}(s^t)} \sum_{t=1}^{\infty} \beta^t \sum_{s^t|s_0} \pi(s^t|s^1)\pi(s_1|s_0) u(c(s^t)) \right\}
\end{aligned}
$$

subject to the constraints

$$
\begin{aligned}
c(s_0) + k_{+1}(s_0) &\leq z(s_0) f(k(s_0)), \\
c(s^t) + k_{+1}(s^t) &\leq z(s^t) f(k(s^t)), \ \forall t > 0, \ \forall s^t|s_0, s_1
\end{aligned}
$$

Recursive formulation again

$$
\begin{aligned}
V(s_0) &:= \max_{c(s^t),k_{+1}(s^t)} \sum_{t=0}^{\infty} \beta^t \sum_{s^t|s_0} \pi(s^t|s_0) u(c(s^t)) \\
&= \max_{c(s_0),k_{+1}(s_0)} u(c(s_0)) \\
&+ \sum_{s_1|s_0} \pi(s_1|s_0) \left\{ \max_{c(s^t),k_{+1}(s^t)} \sum_{t=1}^{\infty} \beta^t \sum_{s^t|s_1} \pi(s^t|s_1) u(c(s^t)) \right\}
\end{aligned}
$$

subject to the constraints

$$
\begin{aligned}
&c(s_0) + k_{+1}(s_0) \leq z(s_0) f(k(s_0)), \\
&c(s^t) + k_{+1}(s^t) \leq z(s^t) f(k(s^t)), \ \forall t > 0, \ \forall s^t|s_0, s_1
\end{aligned}
$$

Recursive formulation again

$$V(s_0) := \max_{c(s^t),k_{+1}(s^t)} \sum_{t=0}^{\infty} \beta^t \sum_{s^t|s_0} \pi(s^t|s_0) u(c(s^t))$$

$$= \max_{c(s_0),k_{+1}(s_0)} u(c(s_0))$$

$$+ \beta \sum_{s_1|s_0} \pi(s_1|s_0) \left\{ \max_{c(s^t),k_{+1}(s^t)} \sum_{t'=0}^{\infty} \beta^{t'} \sum_{s^{t'+1}|s_1} \pi(s^{t'+1}|s_1) u(c(s^{t'+1})) \right\}$$

subject to the constraints

$$c(s_0) + k_{+1}(s_0) \leq z(s_0) f(k(s_0)),$$
$$c(s^t) + k_{+1}(s^t) \leq z(s^t) f(k(s^t)), \ \forall t > 0, \ \forall s^t|s_0, s_1$$

Recursive formulation again

$$V(s_0) := \max_{c(s^t),k_{+1}(s^t)} \sum_{t=0}^{\infty} \beta^t \sum_{s^t|s_0} \pi(s^t|s_0) u(c(s^t))$$

$$= \max_{c(s_0),k_{+1}(s_0)} u(c(s_0))$$

$$+ \beta \sum_{s_1|s_0} \pi(s_1|s_0) \left\{ \max_{c(s^t),k_{+1}(s^t)} \sum_{t'=0}^{\infty} \beta^{t'} \sum_{s^{t'+1}|s_1} \pi(s^{t'+1}|s_1) u(c(s^{t'+1})) \right\}$$

subject to the constraints

$$c(s_0) + k_{+1}(s_0) \leq z(s_0) f(k(s_0)),$$
$$c(s^t) + k_{+1}(s^t) \leq z(s^t) f(k(s^t)), \ \forall t > 0, \ \forall s^t|s_0, s_1$$

Again terms are exactly $V(s^1)$

Recursive form with shocks

So we get (shifting to time t)

$$V(s_t) = \max_{c(s^t), k_{+1}(s^t)} u(c(s_t)) + \beta \mathbb{E}_{s_{t+1}|s_t} V(s_{t+1})$$
$$\text{s.t.} \quad c(s_t) + k_{+1}(s_t) \leq z(s_t) f(k(s_t))$$

What's the state? Everything the agent needs to solve the problem tomorrow:

- k_t fully summarizes the production frontier
- In general, the full history of shocks $z^t = \left(z_t, z_{t-1}, \dots\right)$ may be required to take the expectation

So, $s_t = \left(k_t, z^t\right)$.

Recursive form with Markov shocks

$s_t = \left(k_t, z^t\right)$ is not ideal from a computational point of view as the dimension of s_t increases as time goes by

Let's instead assume z_t is Markov. Then,

- $s_t = \left(k_t, z_t\right)$ since z_t is enough to take the expectation
- We no longer need t to express the agent's problem

$$V(k, z) = \max_{c, k'} u(c) + \beta \mathbb{E}_{z'|z}\left[V(k', z')\right] \quad \text{s.t.} \quad c + k' \leq z f(k)$$

VFI with Markov shocks

Value Function Iteration can be used again to numerically solve

$$V(k,z) = \max_{k'} u(zf(k) - k') + \beta \mathbb{E}_{z'|z} V(k',z).$$

1. Construct 2-D grid $\mathcal{S} := \mathcal{K} \times \mathcal{Z}$ for state variables k and s. Guess initial values $V_0(k_i, z_j)$ for each $(k_i, z_j) \in S$. Choose a stopping rule $\varepsilon > 0$

2. Starting at $l = 0$, for each $(k_i, z_j) \in \mathcal{S}$, evaluate

$$V_{l+1}(k_i, z_j) = \max_{k' \in \mathcal{K}} u(z_j f(k_i) - k') + \beta \sum_{m \in \mathcal{Z}} p_{jm} V(k', z_m).$$

3. If $||V_{l+1}(k,z) - V_l(k,z)|| < \varepsilon$ stop. Else set $l = l + 1$, go back to 2.

This week: Wrapping up

Key concepts:

- Sequential form vs Recursive form
- State variables, choice variables
- Value function, policy function, Bellman equation
- History, Markov Process, Transition Matrix, Markov Chain

Next week: Introduction to the theory behind recursive formulation and study application to savings problem.

References

"Recursive Macroeconomic Theory" (RMT) by Lars Ljungqvist and Thomas Sargent, 3rd edition

- Value function, theory and practice: Chapters 3 and 4
- Markov Processes, Markov Chains: Chapter 2

`Simon Mongey's ECON20210 lecture notes` on "Dynamic Programming"