Solving Dynamic Models Numerically

Lecture 1

Stephen J. Terry
BU Macro & NBER

June 2023

Course Basics & Introductions

- Syllabus and logistics
- MATLAB? C++? Fortran? Julia? Python? R? Stata?
- Stokey-Lucas? Ljungqvist-Sargent? Miranda-Fackler? Judd?
- Experience *doing* applied dynamic programming?
- Experience *teaching* applied dynamic programming?

Roadmap

The basic dynamic programming setup

A firm's dynamic price adjustment problem

Value function iteration

Discretization

Simulation

Fun with MATLAB!

A Dynamic Optimization Problem

$$\max_{\{s_{t+k+1}\}_{k=0}^{\infty}} \mathbb{E}_t \sum_{k=0}^{\infty} \beta^k R(s_{t+k}, x_{t+k}, s_{t+k+1})$$

$$\text{s.t. } s_{t+k+1} \in \Gamma(s_{t+k}, x_{t+k}) \quad \forall k$$

$$x_{t+k}|x_{t+k-1}, ... \sim F(x_{t+k}|x_{t+k-1})$$

- Decisionmaker maximizes an expected present discounted value with discount rate $0 < \beta < 1$ by choosing a contingent plan.
- Current payoff $R(\cdot)$ in t depends on potentially vector-valued:
  - *State* s_t
  - Stationary Markov *shock* x_t
  - *Policy* s_{t+1}, i.e., future state
- Policy choice is subject to constraints $\Gamma(\cdot)$, which usually embed a "today vs tomorrow" dynamic tradeoff

Note: Lots of stuff fits into this structure. Household savings for welfare maximization, firm investment, financing, etc...

The Key Recursive Insight of Dynamic Programming

Sequence Formulation

$$\max_{\{s_{t+k+1}\}_{k=0}^{\infty}} \mathbb{E}_t \sum_{k=0}^{\infty} \beta^k R(s_{t+k}, x_{t+k}, s_{t+k+1}) =$$

$$\underbrace{\max_{\{s_{t+k+1}\}_{k=0}^{\infty}} R(s_t, x_t, s_{t+1}) + \beta \mathbb{E}_t \left[\underbrace{R(s_{t+1}, x_{t+1}, s_{t+2}) + \beta \mathbb{E}_{t+1} R(s_{t+2}, x_{t+2}, s_{t+3}) + \ldots}_{V(s_{t+1}, x_{t+1})} \right]}_{V(s_t, x_t)}$$

Stationary Bellman Equation, Fixed Point, or Recursive Formulation

$$V(s, x) = \max_{s' \in \Gamma(s,x)} \left[R(s, x, s') + \beta \mathbb{E}_{x'|x} V(s', x') \right]$$

$$V = \mathcal{T}(V), \quad \mathcal{T}(f) = \max[R + \beta \mathbb{E}(f)]$$

The Goal

Starting with a Bellman equation formulation

$$V(s,x) = \max_{s' \in \Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V(s',x') \right],$$

the goal is to introduce computational methods allowing us to

- **solve** the model for the optimal policy function $s'(s,x)$, which usually involves solving for the value function $V(s,x)$, and
- **simulate** the model by drawing exogenous Markov shock processes $x_1, x_2, ...$ which lead to endogenous sequences of states $s_1, s_2, ...$

Note: The goals aren't picked randomly. Structural estimation embeds them within each evaluation of an objective function.

Roadmap

The basic dynamic programming setup

A firm's dynamic price adjustment problem

Value function iteration

Discretization

Simulation

Fun with MATLAB!

A Concrete Example

A firm facing downward-sloping demand and marginal cost shocks chooses prices to maximize the expected PDV of its payouts, which are variable profits net of price adjustment costs.

$$V(p_{-1}, m) = \max_{p} \left[\Pi(p, m) - AC(p_{-1}, p) + \beta \mathbb{E}_{m'|m} V(p, m') \right]$$

$$\log m' = \rho \log m + \sigma \eta', \quad \eta' \sim N(0, 1)$$

$$\Pi(p, m) = (p - m) p^{-\varepsilon}, \quad AC(p_{-1}, p) = \frac{c}{2}(p - p_{-1})^2$$

$$0 < \beta < 1, \quad 0 < \rho < 1, \quad \sigma > 0, \quad \varepsilon > 1, \quad c > 0$$

Stepping Back

- Why is this a dynamic problem rather than a series of identically structured static monopolistic pricing decisions?
- What do we expect intuitively from the shape and slopes of the solutions $p(p_{-1}, m)$ and $V(p_{-1}, m)$?

Roadmap

The basic dynamic programming setup

A firm's dynamic price adjustment problem

Value function iteration

Discretization

Simulation

Fun with MATLAB!

Solving the Model with VFI

Solving the Bellman equation

$$V(s,x) = \max_{s' \in \Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V(s',x') \right]$$

is equivalent to finding a fixed point of the operator $\mathcal{T}$

$$\iff V = \mathcal{T}(V), \quad \mathcal{T}(f) = \max \left[R + \mathbb{E}(f) \right].$$

Under regularity conditions, a famous *contraction mapping theorem* guarantees that **value function iteration (VFI)** will converge, i.e., repeatedly iterating on $\mathcal{T}(\cdot)$ converges

$$V_{(1)}, \quad V_{(2)} = \mathcal{T}(V_{(1)}), \quad ..., \quad V_{(n)} = \mathcal{T}(V_{(n-1)}), \quad ... \to_{n \to \infty} V$$

Some Notes

- I'm skipping over formalism. What regularity conditions? What is convergence in a function space? Does V exist uniquely? Why exactly are the sequence and Bellman formulations equivalent?...
- In each VFI step you compute optimal policies $s'(s,x)$. So both values $V(s,x)$ and optimal policies are obtained, as desired.

A Bit More Detail on VFI

Start with $R(s, x, s')$, $F(x'|x)$, $\Gamma(s, x)$, and $0 < \beta < 1$, as well as a guess $V_{(1)}$ and a specified solution tolerance $\varepsilon_{tol} > 0$. For each step of VFI $n = 1, 2, ...,$ do

1. Compute $V_{(n+1)}$ by solving for optimal s' at each (s, x) via

$$V_{(n+1)}(s, x) = \max_{s' \in \Gamma(s,x)} \left[R(s, x, s') + \beta \mathbb{E}_{x'|x} V_{(n)}(s', x') \right].$$

2. Compute the error

$$||V_{(n+1)} - V_{(n)}|| = \max_{s,x} |V_{(n+1)}(s, x) - V_{(n)}(s, x)|$$

3. If $||V_{(n+1)} - V_{(n)}|| < \varepsilon_{tol}$, exit. If $||V_{(n+1)} - V_{(n)}|| \geq \varepsilon_{tol}$, go to next iteration by jumping back to Step 1 with $n = n + 1$.

Roadmap

The basic dynamic programming setup

A firm's dynamic price adjustment problem

Value function iteration

Discretization

Simulation

Fun with MATLAB!

Some of This Stuff is Tricky

Solving the general version of the Bellman equation

$$V(s,x) = \max_{s' \in \Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V(s',x') \right]$$

involves some stuff that can be computationally tricky:

- **Functions** of continuous inputs like $V(\cdot)$ take some care to store, requiring interpolation methods, attention to smoothness, etc...
- **Expectations** of continuous Markov chains $x'|x$ involve numerical integration or quadrature which can be tricky or time-consuming.
- **Optimization** over continuous choices s' requires application of potentially finicky and time-consuming numerical algorithms.

But if we **discretize** everything by putting s, x, and s' on grids, then

- **functions** are easy vectors, **expectations** are easy weighted sums, and **optimization** is just picking the biggest number from a vector.

Discretizing Markov Chains

We desire to work with methods converting from a general continuous, stationary Markov chain described by the distribution $F(x'|x)$ to

- A **grid** $\{\bar{x}_1, ..., \bar{x}_{N_x}\}$ of N_x points in which discretized x lives.
- A **transition matrix** Π^x with size $N_x \times N_x$ with entries

$$\Pi^x_{i,j} = \pi^x_{ij} = \mathbb{P}(x' = \bar{x}_j | x = \bar{x}_i)$$

Once we do this, note that integration collapses to a simple weighted sum:

$$\mathbb{E}_F \left[g(x') | x \right] = \int g(x') dF(x'|x) = \int g(x') f(x'|x) dx$$

$$\Longleftrightarrow$$

$$\mathbb{E}_{\Pi^x} \left[g(x') | x = \bar{x}_i \right] \approx \sum_{j=1}^{N_x} \pi^x_{ij} g(\bar{x}_j)$$

Tauchen (1986)

One classic simple method for discretizing the stationary AR(1) processes

$$x' = \rho x + \sigma \eta', \quad \eta' \sim N(0,1), \quad 0 < \rho < 1, \quad \sigma > 0$$

comes from a famous paper Tauchen (1986).

1. Choose an (odd) value for N_x and a multiple M_x of standard deviations $\sigma_x = \frac{\sigma}{\sqrt{1-\rho^2}}$ which you want the grid to span.
2. Linearly space the grids points $\bar{x}_i$ with gaps $\Delta > 0$

$$\{-M_x\sigma_x, -M_x\sigma_x + \Delta, ..., -\Delta, 0, \Delta, ..., M_x\sigma_x - \Delta, M_x\sigma_x\}$$

3. Compute the entries of Π^x by integrating under the error term, i.e.,

$$\Pi^x_{ij} = \mathbb{P}(x' = \bar{x}_j | x = \bar{x}_i) = H\left(\bar{x}_j + \frac{\Delta}{2}; \rho\bar{x}_i, \sigma^2\right) - H\left(\bar{x}_j - \frac{\Delta}{2}; \rho\bar{x}_i, \sigma^2\right)$$

where $H(\cdot; \mu, \sigma^2)$ is the $N(\mu, \sigma^2)$ CDF. Endpoints absorb the tails.

Note: Within Tauchen's method, you can add constants, make this a lognormal process, apply more complicated grids, do this with vector processes, etc... Also, many other fancier non-Tauchen methods now exist.

Discretized Bellman Equation

$$V(s,x) = \max_{s' \in \Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V(s',x') \right].$$

- Discretize the Markov chain $x'|x$ with N_x points.
- Choose a grid of N_s points on which s lives $\{\bar{s}_1, ..., \bar{s}_{N_s}\}$.
- The discretized Bellman equation for each $(\bar{s}_i, \bar{x}_j)$ is

$$V(\bar{s}_i, \bar{x}_j) = \max_{\substack{s' \in \Gamma(\bar{s}_i, \bar{x}_j), \\ s' \in \{\bar{s}_1, ..., \bar{s}_{N_s}\}}} \left[R(\bar{s}_i, \bar{x}_j, s') + \beta \sum_{k=1}^{N_x} \pi^x_{j,k} V(s', \bar{x}_k) \right].$$

- Collecting across all (i,j), we have the convenient vectorized form

$$\underbrace{\mathbf{V}}_{N_s N_x \times 1 \text{vector}} = \underbrace{\mathbf{RHS}}_{\text{row max of specially constructed } N_s N_x \times N_s \text{matrix}}$$

Roadmap

The basic dynamic programming setup

A firm's dynamic price adjustment problem

Value function iteration

Discretization

Simulation

Fun with MATLAB!

Simulating Your Model

After model solution, we often want to examine simulated data.

- Simulate the exogenous Markov chain $\{x_t\}_{t=1}^T$ for a large T.
- Using optimal policies $s'(\cdot)$, compute the implied states $\{s_t\}_{t=1}^T$.
- Process the data $\{s_t, x_t\}_{t=1}^T$ in whichever manner is needed for your question of interest.

Some Notes

In practice, when simulating for application of structural estimation techniques you may need to:

- Compare $U(0,1)$ shocks with thresholds implied by Π^x.
- Simulate data with a panel structure, i.e., (s_{it}, x_{it}) for firms $i = 1, ..., N$ and $t = 1, ..., T$.
- Be careful with initial conditions, which can be overly influential for persistent processes, by discarding "burn-in" periods.
- Be careful to set seeds for replicability of random draws in the simulation of exogenous processes.

Roadmap

The basic dynamic programming setup

A firm's dynamic price adjustment problem

Value function iteration

Discretization

Simulation

Fun with MATLAB!