Dynamic Programming

(Lectures on Solution Methods for Economists I)

Jesús Fernández-Villaverde[1] and Pablo Guerrón[2]

May 14, 2022

[1]University of Pennsylvania

[2]Boston College

Theoretical Background

Introduction

- Introduce numerical methods to solve dynamic programming (DP) models.
- DP models with sequential decision making:
  - Arrow, Harris, and Marschak (1951) $\rightarrow$ optimal inventory model.
  - Lucas and Prescott (1971) $\rightarrow$ optimal investment model.
  - Brock and Mirman (1972) $\rightarrow$ optimal growth model under uncertainty.
  - Lucas (1978) and Brock (1980) $\rightarrow$ asset pricing models.
  - Kydland and Prescott (1982) $\rightarrow$ business cycle model.

The basic framework

- Almost any DP can be formulated as Markov decision process (MDP).
- An agent, given state $s_t \in S$ takes an optimal action $a_t \in A(s)$ that determines current utility $u(s_t, a_t)$ and affects the distribution of next period's state s_{t+1} via a Markov chain $p(s_{t+1}|s_t, a_t)$.
- The problem is to choose $\alpha = \{\alpha_1, \dots, \alpha_T\}$, where $a_t = \alpha_t(s_t)$, that solves

$$V(s) = \max_{\alpha} \mathbb{E}_{\alpha} \left\{ \sum_{t=0}^{T} \beta^t u(s_t, a_t) \,|s_0 = s \right\}$$

- The difficulty is that we are not looking for a set of numbers $a = \{a_1, \dots, a_T\}$ but for a set of functions $\alpha = \{\alpha_1, \dots, \alpha_T\}$.

The DP problem

- DP simplifies the MDP problem, allowing us to find $\alpha = \{\alpha_1, \dots, \alpha_T\}$ using a recursive procedure.
- Basically, it uses V as a shadow price to map a stochastic/multiperiod problem into a deterministic/static optimization problem.
- We are going to focus on infinite horizon problems, where V is the unique solution for the Bellman equation $V = \Gamma(V)$.
- Where Γ is called the Bellman operator, that is defined as:

$$\Gamma(V)(s) = \max_a \left[u(s, a) + \beta \int V(s') \, p(s'|s, a) \right]$$

- $\alpha(s)$ is equal to the solution to the Bellman equation for each s.

The Bellman operator and the Bellman equation

- We will revise the mathematical foundations for the Bellman equation.
- It has a very nice property: Γ is a contraction mapping.
- This will allow us to use some numerical procedures to find the solution to the Bellman equation recursively.

Discrete vs. continuous MDPs

- Difference between **Discrete** MDPs –whose state and control variables can only take a finite number of points– and **continuous** MDPs –whose state and control variables can take a continuum of values.
- Value functions for discrete MDPs belong to a subset of the finite-dimensional Euclidean space $R^{\#S}$.
- Value functions for continuous MDPs belong to a subset of the infinite-dimensional Banach space $B(S)$ of bounded, measurable real-valued functions on S.
- Therefore, we can solve **discrete** MDPs exactly (rounding errors) while we can only approximate the solution to **continuous** MDPs.
- **Discrete** MDPs arise naturally in IO/labor type of applications while **continuous** MDPs arise naturally in Macro.

Computation: speed vs. accuracy

- The approximating error ϵ introduces a trade-off: better accuracy (lower ϵ) versus shorter time to find the solution (higher ϵ).
- The time needed to find the solution also depends on the dimension of the problem: d.
- We want the fastest method given a pair (ϵ, d).
- Why do we want the fastest method?
- Normally, this algorithms are nested into a bigger optimization algorithm.
- Hence, we will have to solve the Bellman equation for various values of the "structural" parameters defining β, u, and p.

Approximation to continuous DPs

- There are two ways to approximate **continuous** DPs.
  - Discrete.
  - Smooth.
- Discrete solves an equivalent discrete problem that approximates the original **continuous** DPs.
- Smooth treats the value function V and the decision rule α are smooth functions of s and a finite set of coefficients θ.

Smooth approximation to continuous DPs

- Then we will try to find $\widehat{\theta}$ such that the approximations the approximated value function $V_{\widehat{\theta}}$ and decision rule $\alpha_{\widehat{\theta}}$ are close to V and α using some metric.
- In general, we will use a sequence of parametrization that is dense on $B(S)$.
- That means that for each $V \in B(S)$, $\exists \{\theta_k\}_{k=1}^{\infty}$ such that

$$\lim_{k \to \infty} \inf_{\theta_k} \sup_{s \in S} |V_\theta(s) - V(s)| = 0$$

- Example:
  1. Let $S = [-1, 1]$.
  2. Consider $V_\theta(s) = \sum_{i=1}^{k} \theta_i p_i(s)$ and let $p_i(s) = s^i$.
- Another example is $p_i(s) = \cos\left(i \cos^{-1}(s)\right)$. These are called the Chebyshev polynomials of the first kind.

The Stone-Weierstrass approximation theorem

- Let $\varepsilon > 0$ and V be a continuous function in $[-1, 1]$, then there exists a polynomial V_θ such that

$$\|V - V_\theta\| < \varepsilon$$

- Therefore, the problem is to find θ such that minimizes

$$\left(\sum_{i=1}^{N} \left| V_\theta (s_i) - \widehat{\Gamma} (V_\theta) (s_i) \right|^2 \right)^{1/2}$$

  where $\widehat{\Gamma} (V_\theta)$ is an approximation to the Bellman operator. Why is an approximation?

- Faster to solve the previous problem than by brute force discretizations.

MDP definitions

- A MDP is defined by the following objects:
  - A state space S.
  - An action space A.
  - A family of constraints $A(s)$ for $s \in S$.
  - A transition probability $p(ds'|s,a) = \Pr(s_{t+1} = ds'|s_t = s, a_t = a)$.
  - A single period utility $u(s,a)$.
- The agent problem is to choose $\alpha = \{\alpha_1, \ldots, \alpha_T\}$ such that:

$$\max_{\alpha} \int_{s_0} \ldots \int_{s_T} \left[u\left(s_t, \alpha_t\left(s_t\right)\right)\right] p\left(ds_t | s_{t-1}, \alpha_{t-1}\left(s_{t-1}\right)\right) p_0\left(ds_0\right)$$

- $p_0(ds_0)$ is the probability distribution over the initial state.
- This problem is very complicated: search over a set of functions $\{\alpha_1, \ldots, \alpha_T\}$ and make a $T+1$-dimension integral.

The Bellman equation in the finite horizon problem

- If $T < \infty$ (the problem has a finite horizon), DP is equivalent to backward induction. In the terminal period α_T is:

$$\alpha_T(s_T) = \arg\max_{a_T \in A(s_T)} u(s_T, a_T)$$

- And $V_T(s_T) = u(s_T, \alpha_T(s_T))$.
- For periods $t = 1, \ldots, T-1$, we can find V_t and α_t by recursion:

$$\alpha_t(s_t) = \arg\max_{a_t \in A(s_t)} \left[u(s_t, a_t) + \beta \int V_{t+1}(s_{t+1})\, p(ds_{t+1} | s_t, a_t) \right]$$

$$V_t(s_t) = u(s_t, \alpha_t(s_t)) + \beta \int V_{t+1}(s_{t+1})\, p(ds_{t+1} | s_t, \alpha_t(s_t))$$

- It could be the case that $a_t = \alpha_t(s_t, a_{t-1}, s_{t-1}, \ldots)$ depend on the whole history, but it can be shown that separability and the Markovian property of p imply that $a_t = \alpha_t(s_t)$.

The Bellman equation in the infinite horizon problem I

- If $T = \infty$, we do not have a finite state.
- On the other hand, the separability and the Markovian property of p imply that $a_t = \alpha(s_t)$, that is, the problem has a stationary Markovian structure.
- The optimal policy only depend on s, it does not depend on t.
- Thus, the optimal stationary markovian rule is characterized by:

$$\alpha(s) = \arg \max_{a \in A(s)} \left[u(s, a) + \beta \int V(s') p(ds'|s, a) \right]$$

$$V(s) = u(s, \alpha(s)) + \beta \int V(s) p(ds'|s, \alpha(s))$$

- This equation is known as the Bellman equation.
- It is a functional equation (mapping from functions to functions).
- The function V is the fixed point to this functional equation.

The Bellman equation in the infinite horizon problem II

- To determine existence and uniqueness, we need to impose:

    1. S and A are compact metric spaces.
    2. $u(s,a)$ is jointly continuous and bounded.
    3. $s \longrightarrow A(s)$ is a continuous correspondence.

- Let $B(S)$ the Banach space of bounded, measurable real-valued functions on S.
- Let $\|f\| = \sup_{s\in S} |f(s)|$ for $f \in B(S)$ be the sup norm.
- The Bellman operator is:

$$\Gamma(W)(s) = \max_{a\in A(s)} \left[u(s,a) + \beta \int W(s') \, p(ds'|s,a) \right]$$

- The Bellman equation is then a fixed point to the operator:

$$V = \Gamma(V)$$

The Bellman equation in the infinite horizon problem II

- Blackwell (1965) and Denardo (1967) show that the Bellman operator is a contraction mapping: for W, V in $B(S)$,

$$\|\Gamma(V) - \Gamma(W)\| \leq \beta \|V - W\|$$

- **Contraction mapping theorem**: if Γ is a contractor operator mapping on a Banach Space B, then Γ has an unique fixed point.
- **Blackwell's theorem**: the Stationary Markovian α defined by:

$$\alpha(s) = \arg\max_{a \in A(s)} \left[u(s,a) + \beta \int V(s') \, p(ds'|s,a) \right]$$

$$V(s) = u(s, \alpha(s)) + \beta \int V(s) \, p(ds'|s, \alpha(s))$$

solves the associated MDP problem.

A trivial example

- Consider $u(s,a) = 1$.
- Given that u is constant, let us assume that V is also constant.
- If we substitute this result into the Bellman equation, we get:

$$V = \max_{a \in A(s)} \left[1 + \beta \int V p(ds' | s, a) \right]$$

- And the unique solution is $V = \frac{1}{1-\beta}$.
- Clearly, the MDP problem implies that $V = 1 + \beta + \beta^2 + \dots$
- So, they are equivalent.

Phelps' (1972) example I

- The agent has to decide between consume and save.
- The state variable, w, is the wealth of the agent and the decision variable, c, is how much to consume.
- The agent cannot borrow, so the choice set $A(w) = \{c | 0 \leq c \leq w\}$.
- The saving are invested in a single risky asset with iid return R_t with distribution F.
- The Bellman Equation is:

$$V(w) = \max_{c \in A(w)} \log(c) + \beta \int_0^{\infty} V(R(w-c)) F(dR)$$

Phelps' (1972) example II

- Since it operator Γ is a contraction, we can start $V = 0$.
- If that is the case, $V_t = \Gamma^t(0) = f_t \log(w) + g_t$ for f_t and g_t constant.
- So, $V_\infty = \Gamma^\infty(0) = f_\infty \log(w) + g_\infty$.
- If we substitute V_∞ into the Bellman equation and we look for f_∞ and g_∞, we get:

$$f_\infty = \frac{1}{1-\beta}$$

$$g_\infty = \frac{\log(1-\beta)}{1-\beta} + \frac{\beta \log(\beta)}{(1-\beta)^2} + \frac{\beta E\{\log(R)\}}{(1-\beta)^2}$$

  and $\alpha(w) = (1-\beta)w$.
- Therefore, permanent income hypothesis still holds in this environment.

Numerical Implementation

Motivation

- Before, we reviewed some theoretical background on dynamic programming
- Now, we will discuss its numerical implementation
- Perhaps the most important solution algorithm to learn:
  1. Wide applicability
  2. Many known results
  3. Template for other algorithms
- Importance of keeping the "curse of dimensionality" under control
- Two issues to discuss:
  1. Finite versus infinite time
  2. Discrete versus continuous state space.

Finite time

- Problems where there is a terminal condition.
- Examples:
    1. Life cycle.
    2. Investment with expiration date.
    3. Finite games.
- Why are finite time problems nicer? Backward induction.
- You can think about them as a particular case of multivariate optimization.

Infinite time

- Problems where there is no terminal condition.
- Examples:
  1. Industry dynamics.
  2. Business cycles.
  3. Infinite games.
- However, we will need the equivalent of a terminal condition: transversality condition.

Discrete state space

- We can solve problems up to floating point accuracy.
- Why is this important?
  1. ε-equilibria.
  2. Estimation.
- However, how realistic are models with a discrete state space?

Infinite state space

- More common cases in economics.
- Problem: we have to rely on a numerical approximation.
- Interaction of different approximation errors (computation, estimation, simulation).
- Bounds?
- Interaction of bounds?

Different strategies

- Four main strategies:
    1. Value function iteration.
    2. Policy function iteration.
    3. Projection.
    4. Perturbation.
- Many other strategies are actually particular cases of the previous ones.

Value function iteration

- Well-known, basic algorithm of dynamic programming. Aka as value improvement.
- We have tight convergence properties and bounds on errors.
- Well suited for parallelization.
- It will always (perhaps quite slowly) work.
- How do we implement the operator?
  1. We come back to our two distinctions: finite versus infinite time and discrete versus continuous state space.
  2. Then we need to talk about:
     - Initialization.
     - Discretization.

Value function iteration in finite time

- We begin with the Bellman operator:

$$\Gamma\left(V^{t}\right)(s)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V^{t'}\left(s'\right)p\left(ds'|s,a\right)\right]$$

- Specify V^{T} and apply Bellman operator:

$$V^{T-1}\left(s\right)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V^{T}\left(s'\right)p\left(ds'|s,a\right)\right]$$

- Iterate until first period:

$$V^{1}\left(s\right)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V^{2}\left(s'\right)p\left(ds'|s,a\right)\right]$$

Value function iteration in infinite time

- We begin with the Bellman operator:

$$\Gamma\left(V\right)\left(s\right)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V\left(s'\right)p\left(ds'|s,a\right)\right]$$

- Specify V^0 and apply Bellman operator:

$$V^1\left(s\right)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V^0\left(s'\right)p\left(ds'|s,a\right)\right]$$

- Iterate until convergence:

$$V^T\left(s\right)=\max_{a\in A(s)}\left[u\left(s,a\right)+\beta\int V^{T-1}\left(s'\right)p\left(ds'|s,a\right)\right]$$

Policy function iteration

- With infinite time, we can also apply policy function iteration (aka as Howard improvement algorithm):

  1. We guess a policy function a^0.
  2. We compute the V^0 associated to it (by matrix operations or iteration).
  3. We compute the new policy function a^1 implied by V^0.
  4. We iterate until convergence.

- Under some conditions, if can be faster than value function iteration (more on this later).

- Most of the next slides applies to policy function iteration without any (material) change.

Normalization

- Before initializing the algorithm, it is usually a good idea to normalize problem:

$$V(s) = \max_{a \in A(s)} \left[(1-\beta)\, u(s,a) + \beta \int V(s')\, p(ds'|s,a) \right]$$

- Three advantages:

  1. We save one iteration.
  2. Stability properties.
  3. Convergence bounds are interpretable.

- More general case: reformulation of the problem.

Initial value in finite time problems

- Usually, economics of the problem provides natural choices.
- Example: final value of an optimal expenditure problem is zero.
- However, some times there are subtle issues.
- Example: what is the value of dying? And of bequests? OLG.

Initial guesses for infinite time problems

- Theorems tell us we will converge from any initial guess.
- That does not mean we should not be smart picking our initial guess.
- Several good ideas:
  1. Steady state of the problem (if one exists). Usually saves at least one iteration.
  2. Perturbation approximation.
  3. Collapsing one or more dimensions of the problem. Which one?

Discretization

- In the case where we have a continuous state space, we need to discretize it into a grid.
- How do we do that?
- Dealing with curse of dimensionality.
- Do we let future states lie outside the grid?

New approximated problem

- Exact problem:

$$V(s) = \max_{a \in A(s)} \left[(1-\beta)\, u(s,a) + \beta \int V(s')\, p(ds' | s,a) \right]$$

- Approximated problem:

$$\widehat{V}(s) = \max_{a \in \widehat{A}(s)} \left[(1-\beta)\, u(s,a) + \beta \sum_{k=1}^{N} \widehat{V}(s'_k)\, p_N(s'_k | s,a) \right]$$

Grid generation

- Huge literature on numerical analysis on how to efficiently generate grids.
- Two main issues:
  1. How to select points s_k.
  2. How to approximate p by p_N.
- Answer to second issue follows from answer to first problem.
- We can (and we will) combine strategies to generate grids.

Uniform grid

- Decide how many points in the grid.
- Distribute them uniformly in the state space.
- What is the state space is not bounded?
- Advantages and disadvantages.

Non-uniform grid

- Use economic theory or error analysis to evaluate where to accumulate points.
- Standard argument: close to curvatures of the value function.
- Problem: this an heuristic argument.
- Self-confirming equilibria in computations.

Discretizing stochastic process

- Important case: discretizing exogenous stochastic processes.

- Consider a general AR(1) process:

$$z' = (1-\rho)\mu_z + \rho z + \varepsilon', \ \varepsilon' \overset{iid}{\sim} \mathcal{N}(0, \sigma_\varepsilon^2)$$

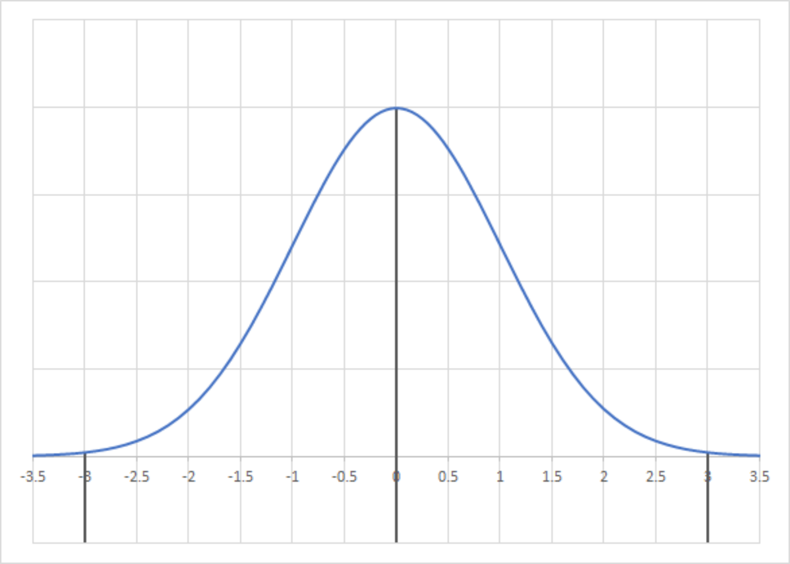
- Recall that $\mathbb{E}[z] = \mu_z$ and $Var[z] = \sigma_z^2 = \frac{\sigma_\varepsilon^2}{(1-\rho^2)}$.

- First step is to choose m (e.g., $m = 3$) and N, and define:

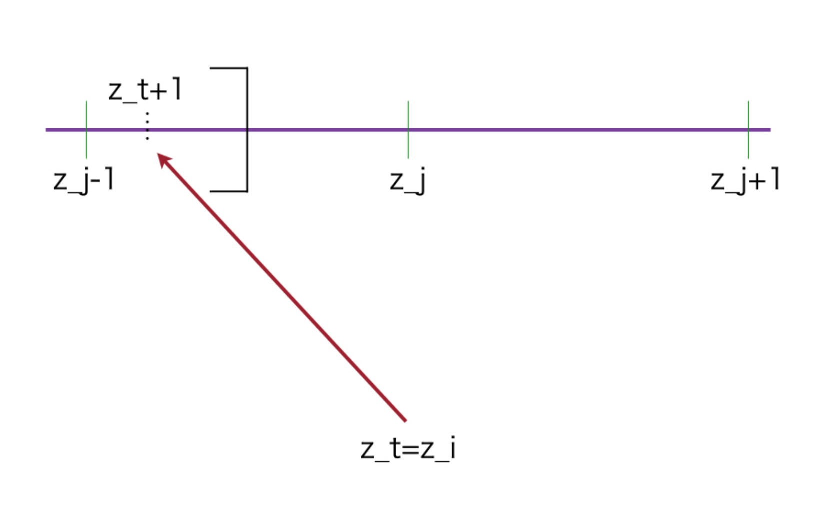
$$z_N = \mu_z + m\sigma_z \ z_1 = \mu_z - m\sigma_z$$

- $z_2, z_3, ..., z_{N-1}$ are equispaced over the interval $[z_1, z_N]$ with $z_k < z_{k+1}$ for any $k \in \{1, 2, ..., N-1\}$

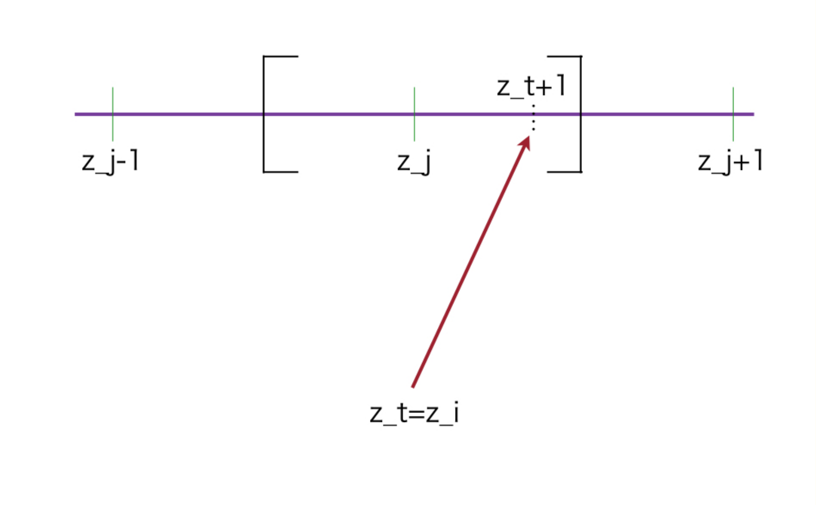
Example



Transition I



Transition II



Transition probability

- Let $d = z_{k+1} - z_k$. Then

$$\begin{aligned}
\pi_{i,j} &= \Pr\{z' = z_j | z = z_i\} \\
&= \Pr\{z_j - d/2 < z' \leq z_j + d/2 | z = z_i\} \\
&= \Pr\{z_j - d/2 < (1-\rho)\mu_z + \rho z_i + \varepsilon \leq z_j + d/2\} \\
&= \Pr\left\{\frac{z_j + d/2 - (1-\rho)\mu_z - \rho z_i}{\sigma_\varepsilon} < \frac{\varepsilon}{\sigma_\varepsilon} \leq \frac{z_j - d/2 - (1-\rho)\mu_z - \rho z_i}{\sigma_\varepsilon}\right\} \\
&= \Phi\left(\frac{z_j + d/2 - (1-\rho)\mu_z - \rho z_i}{\sigma_\varepsilon}\right) - \Phi\left(\frac{z_j - d/2 - (1-\rho)\mu_z - \rho z_i}{\sigma_\varepsilon}\right)
\end{aligned}$$

- Adjust for tails:

$$\pi_{i,j} = \begin{cases} 1 - \Phi\left(\frac{z_N - d/2 - (1-\rho)\mu_z - \rho z_i}{\sigma_\varepsilon}\right) & \text{if } j = N \\ \Phi\left(\frac{z_j + d/2 - (1-\rho)\mu_z - \rho z_i}{\sigma_\varepsilon}\right) - \Phi\left(\frac{z_j - d/2 - (1-\rho)\mu_z - \rho z_i}{\sigma_\varepsilon}\right) & \text{otherwise} \\ \Phi\left(\frac{z_1 + d/2 - (1-\rho)\mu_z - \rho z_i}{\sigma_\varepsilon}\right) & \text{if } j = 1 \end{cases}$$

VAR(1) case: state space

- We can apply Tauchen's method to VAR(1) case with $z \in \mathbb{R}^K$.

$$z' = Az + \varepsilon' \text{ where } \varepsilon' \overset{iid}{\sim} \mathcal{N}(0, \Sigma_\varepsilon)$$

- Pick N_k's for $k = 1, ..., K$. We now have $N = N_1 \times N_2 \times \cdots N_K$ possible states.
- For each $k = 1, ..., K$, we can define

$$z^k_{N_k} = m\sigma_{z_k} \qquad z^k_1 = -z^k_{N_k}$$

and remaining points are equally spaced.

- $\sigma^2_{z_k}$ can be obtained from $vec(\Sigma_z) = (I - A \otimes A)^{-1} vec(\Sigma_\varepsilon)$.

VAR(1) case: transition probability

- Consider a transition from $z_i = (z^1_{i_1}, z^2_{i_2}, ..., z^K_{i_K})$ to $z_j = (z^1_{j_1}, z^2_{j_2}, ..., z^K_{j_K})$.
- Associated probability for each state variable k given state i_k to $j_k$is now:

$$\pi^k_{i_k,j_k} = \begin{cases} 1 - \Phi\left(\frac{z^k_{N_k} - d_k/2 - A_{kk} z^k_{i_k}}{\sigma_{\varepsilon_k}}\right) \\ \Phi\left(\frac{z^k_{j_k} + d_k/2 - A_{kk} z^k_{i_k}}{\sigma_{\varepsilon_k}}\right) - \Phi\left(\frac{z^k_{j_k} - d/2 - A_{kk} z^k_{i_k}}{\sigma_{\varepsilon_k}}\right) & j \neq 1, N_k \\ \Phi\left(\frac{z^k_1 + d_k/2 - A_{kk} z^k_{i_k}}{\sigma_{\varepsilon_k}}\right) \end{cases}$$

- Therefore, $\pi_{i,j} = \prod_{k=1}^K \pi^k_{i_k,j_k}$.
- We can use this method for discretizing higher order AR processes.

Example

- For simplicity, $\Sigma_\varepsilon = I$, and

$$\begin{pmatrix} z^1_{t+1} \\ z^2_{t+1} \end{pmatrix} = \begin{pmatrix} 0.72 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} z^1_t \\ z^2_t \end{pmatrix} + \begin{pmatrix} \varepsilon^1_{t+1} \\ \varepsilon^2_{t+1} \end{pmatrix}$$

- Let $m = 3$, $N_1 = 3$, $N_2 = 5$. Thus, $N = 3 \times 5$ states in total.

- In this case, $d_1 = 4.3229$, $d_2 = 1.7321$.

- Transition from (z^1_2, z^2_3) to (z^1_3, z^2_4) is given by $\pi^1_{2,3} \times \pi^2_{3,4}$ where

$$\begin{aligned} \pi^1_{2,3} &= 1 - \Phi\left(z^1_3 - d_1/2 - 0.72 z^1_2\right) \\ &= 0.0153 \\ \pi^2_{3,4} &= \Phi\left(z^2_4 + d_2/2 - 0.5 z^2_3\right) - \Phi\left(z^2_4 - d_2/2 - 0.5 z^2_3\right) \\ &= 0.1886 \end{aligned}$$

Quadrature grid

- Tauchen and Hussey (1991).
- Motivation: quadrature points in integrals

$$\int f(s)\, p(s)\, ds \simeq \sum_{k=1}^{N} f(s_k)\, w_k$$

- Gaussian quadrature: we require previous equation to be exact for all polynomials of degree less than or equal to $2N-1$.

Rouwenhorst (1995) Method

- Consider again $z' = \rho z + \varepsilon'$ with $\varepsilon' \overset{iid}{\sim} \mathcal{N}(0, \sigma_\varepsilon^2)$.
- Again, we want to approximate it by N-state Markov chain process with
  - $\{z_1, ..., z_N\}$ state space.
  - Transition probability Θ_N.
- Set endpoints as $z_N = \sigma_z \sqrt{N-1} \equiv \psi$, and $z_1 = -\psi$.
- $z_2, z_3, ..., z_{N-1}$ are equispaced.
- We will derive transition matrix with size n recursively until $n = N$:
  1. For $n = 2$, define Θ_2.
  2. For $2 < n \leq N$, derive Θ_n from Θ_{n-1}.

State and transition probability

- Define $p = q = \frac{1+\rho}{2}$ (under the assumption of symmetric distribution) and

$$\Theta_2 \quad = \quad \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}$$

- Compute Θ_n by:

$$\begin{aligned} \Theta_n \quad = \quad & p \begin{bmatrix} \Theta_{n-1} & \mathbf{0} \\ \mathbf{0}' & 0 \end{bmatrix} + (1-p) \begin{bmatrix} \mathbf{0} & \Theta_{n-1} \\ 0 & \mathbf{0}' \end{bmatrix} \\ & +(1-q) \begin{bmatrix} \mathbf{0}' & 0 \\ \Theta_{n-1} & \mathbf{0} \end{bmatrix} + q \begin{bmatrix} 0 & \mathbf{0}' \\ \mathbf{0} & \Theta_{n-1} \end{bmatrix} \end{aligned}$$

  where $\mathbf{0}$ is a $(n-1)$ column vector.
- Divide all but the top and bottom rows in Θ_n by 2 after each iteration.

Why divide by two?

- For $n = 3$ case, we have

$$\begin{aligned}\Theta_3 &= p\begin{bmatrix} p & 1-p & 0 \\ 1-q & q & 0 \\ 0 & 0 & 0 \end{bmatrix} + (1-p)\begin{bmatrix} 0 & p & 1-p \\ 0 & 1-q & q \\ 0 & 0 & 0 \end{bmatrix} \\ &\quad +(1-q)\begin{bmatrix} 0 & 0 & 0 \\ p & 1-p & 0 \\ 1-q & q & 0 \end{bmatrix} + q\begin{bmatrix} 0 & 0 & 0 \\ 0 & p & 1-p \\ 0 & 1-q & q \end{bmatrix}\end{aligned}$$

- We can see that the 2nd row sums up to 2!

Invariant distribution

- Distribution generated by Θ_N converges to the invariant distribution $\lambda^{(N)} = (\lambda_1^{(N)}, ..., \lambda_N^{(N)})$ with

$$\lambda_i^{(N)} = \begin{pmatrix} N-1 \\ i-1 \end{pmatrix} s^{i-1}(1-s)^{N-1}$$

where

$$s = \frac{1-p}{2-(p+q)}$$

- From this invariant distribution, we can compute moments associate with Θ_N analytically.

Which method is better?

- Kopecky and Suen (2010) argue that Rouwenhorst method is the best approx., especially for high persistence ($\rho \to 1$).
- Test bed:

$$V(k,a) = \max_{c,k' \geq 0} \left\{ log(c) + \beta \int V(k',a') dF(a'|a) \right\}$$

$$\text{s.t. } c + k' = \exp(a) k^{\alpha} + (1-\delta) k$$

$$a' = \rho a + \varepsilon'$$

$$\varepsilon' \overset{iid}{\sim} \mathcal{N}(0, \sigma_{\varepsilon}^2)$$

- Compare statistics under approximated stationary distribution to quasi-exact solution using Chebyshev parameterized expectation algorithm.
- Comparison also with Adda and Cooper (2003).

Results

Table 2
Business cycle moments for the stochastic growth model.

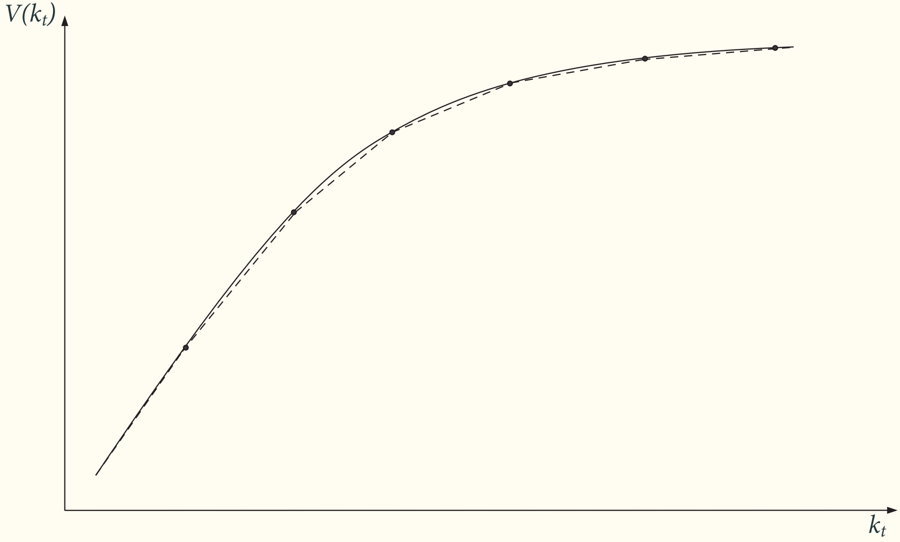
	$N=5$				
	Generated values relative to true values				
	Tau*	T–H	F	A–C	R
ρ	1.0097	0.9453	1.0096	0.9993	1.0000
σ_ε	0.8167	0.8905	0.5019	1.5599	1.0000
σ_a	1.0000	0.4006	0.7742	0.9471	1.0000
σ_k	1.0060	0.3332	0.7485	0.8880	0.9980
σ_{ka}	1.0733	0.0810	0.6528	0.6629	0.9981
σ_y	1.0150	0.3515	0.7847	0.8904	0.9995
σ_c	1.0523	0.2905	0.8423	0.7949	1.0055
σ_i	0.9321	0.6555	0.6549	1.2853	1.0253
ρ_y	1.0037	0.9412	1.0061	0.9779	1.0000

Stochastic grid

- Randomly chosen grids.
- Rust (1995): it breaks the curse of dimensionality.
- Why?
- How do we generate random numbers in the best way?

Interpolation

- Discretization also generates the need for interpolation.
- Simpler approach: linear interpolation.
- Problem: in one than more dimension, linear interpolation may not preserve concavity.
- Shape-preserving splines: Schumaker scheme.
- Trade-off between speed and accuracy interpolation.


$V(k_t)$
k_t

Multigrid algorithms

- Old tradition in numerical analysis.
- Basic idea: solve first a problem in a coarser grid and use it as a guess for more refined solution.
- Examples:
  1. Differential equations.
  2. Projection methods.
  3. Dynamic programming (Chow and Tsitsiklis, 1991).
- Great advantage: extremely easy to code.

Applying the algorithm

- After deciding initialization and discretization, we still need to implement each step:

$$V^{T}(s) = \max_{a \in A(s)} \left[u(s,a) + \beta \int V^{T-1}(s')\, p(ds'|s,a) \right]$$

- Two numerical operations:

  1. Maximization.

  2. Integral.

Maximization

- We need to apply the max operator.
- Most costly step of value function iteration.
- Brute force (always works): check all the possible choices in the grid.
- Sensibility: using a Newton or quasi-Newton algorithm.
- Fancier alternatives: simulated annealing, genetic algorithms,...

Brute force

- Some times we do not have any other alternative. Examples: problems with discrete choices, non-differentiabilities, non-convex constraints, etc.

- Even if brute force is expensive, we can speed things up quite a bit:

  1. Previous solution.

  2. Monotonicity of choices.

  3. Concavity (or quasi-concavity) of value and policy functions.

Newton or Quasi-Newton

- Much quicker.
- However:
  1. Problem of global convergence.
  2. We need to compute derivatives.
- We can mix brute force and Newton-type algorithms.

Generalized policy iteration

- Maximization is the most expensive part of value function iteration.
- Often, while we update the value function, optimal choices are not.
- This suggests a simple strategy: apply the max operator only from time to time.
- This should remind you of an incomplete policy function iteration.
- Often known as generalized policy iteration.
- How do we choose the optimal timing of the max operator (i.e., the relative sweeps of value and policy)?
- Related: asynchronous implementations of value and policy function iterations.

How do we integrate?

- Exact integration.
- Approximations: Laplace's method.
- Quadrature.
- Monte Carlo.

Convergence assessment

- How do we assess convergence?
- By the contraction mapping property:

$$\left\| V - V^k \right\|_\infty \leq \frac{1}{1-\beta} \left\| V^{k+1} - V^k \right\|_\infty$$

- Relation of value function iteration error with Euler equation error.

Non-local accuracy test

- Proposed by Judd (1992) and Judd and Guu (1997).

- Example: Euler equation from a stochastic neoclassical growth model

$$\frac{1}{c^i\left(k_t, z_t\right)} = \mathbb{E}_t\left(\frac{\alpha e^{z_{t+1}} k^i(k_t, z_t)^{\alpha-1}}{c^i\left(k^i(k_t, z_t), z_{t+1}\right)}\right)$$

  we can define:

$$EE^i\left(k_t, z_t\right) \equiv 1 - c^i\left(k_t, z_t\right) \mathbb{E}_t\left(\frac{\alpha e^{z_{t+1}} k^i(k_t, z_t)^{\alpha-1}}{c^i\left(k^i(k_t, z_t), z_{t+1}\right)}\right)$$

- Units of reporting.

- Interpretation.

Error analysis

- We can use errors in Euler equation to refine grid.
- How?
- Advantages of procedure.
- Problems.

The endogenous grid method

- Proposed by Carroll (2005) and Barillas and Fernández-Villaverde (2006).
- Links with operations research: pre-action and post-action states.
- It is actually easier to understand with a concrete example: a basic stochastic neoclassical growth model.
- The problem has a Bellman equation representation:

$$\mathbb{V}(k_t, z_t) = \max_{k_{t+1}} \left\{ \frac{\left(e^{z_t} k_t^{\alpha} + (1-\delta) k_t - k_{t+1}\right)^{1-\tau}}{1-\tau} + \beta \mathbb{E}_t \mathbb{V}(k_{t+1}, z_{t+1}) \right\}$$

$$s.t.\ z_{t+1} = \rho z_t + \varepsilon_{t+1}$$

where $\mathbb{V}(\cdot, \cdot)$ is the value function of the problem.

Changing state variables

- We will use a state variable called "market resources" or "cash-on-hand," instead of k_t:

$$Y_t = c_t + k_{t+1} = y_t + (1-\delta)\, k_t = e^{z_t} k_t^{\alpha} + (1-\delta)\, k_t$$

- We use a capital Y_t to denote the total market resources and a lower y_t for the production function.
- More general point: changes of variables are often key in solving our problems.
- As a result, we write the problem recursively with the Bellman equation:

$$V\left(Y_t, z_t\right) = \max_{k_{t+1}} \left\{ \frac{\left(Y_t - k_{t+1}\right)^{1-\tau}}{1-\tau} + \beta \mathbb{E}_t V\left(Y_{t+1}, z_{t+1}\right) \right\}$$

$$s.t.\ z_{t+1} = \rho z_t + \varepsilon_{t+1}$$

- Note difference between $\mathbb{V}\left(k_t, z_t\right)$ and $V\left(Y_t, z_t\right)$.

Optimilaty condition

- Since Y_{t+1} is only a function of k_{t+1} and z_{t+1}, we can write:

$$\tilde{V}(k_{t+1,}z_t) = \beta \mathbb{E}_t V\left(Y_{t+1}, z_{t+1}\right)$$

  to get:

$$V\left(Y_t, z_t\right) = \max_{k_{t+1}} \left\{ \frac{\left(Y_t - k_{t+1}\right)^{1-\tau}}{1-\tau} + \tilde{V}(k_{t+1,}z_t) \right\}$$

- The first-order condition for consumption:

$$\left(c_t^*\right)^{-\tau} = \tilde{V}_{k_{t+1}}(k_{t+1}^*, z_t)$$

  where $c_t^* = Y_t - k_{t+1}^*$.

Backing up consumption

- So, if we know $\tilde{V}(k_{t+1}, z_t)$, consumption:

$$c_t^* = \left(\tilde{V}_{k_{t+1}}(k_{t+1}, z_t)\right)^{-\frac{1}{\tau}}$$

  for each point in a grid for k_{t+1} and z_t.
- It should remind you of Hotz-Miller type estimators.
- Then, given c_t^* and k_{t+1}, we can find $Y_t^* = c_t^* + k_{t+1}$ and obtain

$$V\left(Y_t^*, z_t\right) = \left\{ \frac{\left(c_t^*\right)^{1-\tau}}{1-\tau} + \tilde{V}(k_{t+1}, z_t) \right\}$$

  where we can drop the max operator, since we have already computed the optimal level of consumption.
- Since $Y_t^* = e^{z_t}\left(k_t^*\right)^{\alpha} + (1-\delta)\, k_t^*$, an alternative interpretation of the algorithm is that, during the iterations, the grid on k_{t+1} is fixed, but the values of k_t change endogenously. Hence, the name of Endogenous Grid.

Comparison with standard approach

- In the standard VFI, the optimality condition is:

$$\left(c_t^*\right)^{-\tau} = \beta \mathbb{E}_t \mathbb{V}_k \left(k_{t+1}^*, z_{t+1}\right)$$

- Since $c_t = e^{z_t} k_t^{\alpha} + (1-\delta) k_t - k_{t+1}$, we have to solve

$$\left(e^{z_t} k_t^{\alpha} + (1-\delta) k_t - k_{t+1}^*\right)^{-\tau} = \beta \mathbb{E}_t \mathbb{V}_k \left(k_{t+1}^*, z_{t+1}\right)$$

  a nonlinear equation on k_{t+1}^* for each point in a grid for k_t.
- The key difference is, thus, that the endogenous grid method defines a fixed grid over the values of k_{t+1} instead of over the values of k_t.
- This implies that we already know what values the policy function for next period's capital take and, thus, we can skip the root-finding.