# Continuous-State Dynamic Programming

Kenneth L. Judd, Hoover Institution

April 27, 2020

# Continuous Methods for Continuous-State Problems

- Basic Bellman equation:

$$V_{new}(x) = \max_{u \in D(x)} \pi(u,x) + \beta \, E\{V_{old}(x^+)|x,u)\} \equiv (TV_{old})(x).$$

- Notation for value functions
  - $V_{new}$ is the current value function if $V_{old}$ is the "next period's" value function
  - In finite horizon problems, $V_{old}$ is $V_{t+1}$ and $V_{new}$ is $V_t$
  - In infinite horizon problems, $V_{old}$ is the old guess and $V_{new}$ is the new guess
- Discretization essentially approximates $V$ with a step function
  - Value functions are typically continuous
  - Approximation theory provides better methods to approximate continuous functions.
- General Task
  - Find good approximation for $V_{new}$ given $V_{old}$
  - In nonstationary models, we want to find good approximation for $V_t$ for all times $t$
  - Identify parameters of approximation

# General Parametric Approach: Approximating $V(x)$

- Choose a finite-dimensional parameterization

$$V(x) \doteq \hat{V}(x; a), \ a \in R^m$$

  and a finite number of states

$$X = \{x_1, x_2, \cdots, x_n\}$$

- polynomials with coefficients $a$ and collocation points $X$
  - splines with coefficients $a$ with uniform nodes $X$
  - rational function with parameters $a$ and nodes $X$
  - neural network
  - specially designed functional forms
- Objective: find coefficients $a \in R^m$ such that $\hat{V}(x; a)$ "approximately" satisfies the Bellman equation.

# General Parametric Approach: Approximating $T$

The key element is the $T$ operator that takes the old value function approximation to the new one.

- $T$ maps functions to functions, not vectors to vectors
- For each $x$, the value of $(TV)(x)$ is defined by

$$(TV)(x) = \max_{u \in D(x)} \pi(u, x) + \beta \int \hat{V}(x^+; a) dF(x^+ | x, u)$$

- Computers cannot map functions to functions
- We instead must map approximations of $V$ to approximations of $V$

# Definition of $\hat{T}$

- For each $x_j$, $(TV)(x_j)$ is defined by

$$v_j = (TV)(x_j) = \max_{u \in D(x_j)} \pi(u, x_j) + \beta \int \hat{V}(x^+; a) dF(x^+ | x_j, u)$$

- In practice, we compute the approximation $\hat{T}$

$$v_j = (\hat{T}V)(x_j) \doteq (TV)(x_j)$$

  - Integration step: for $\omega_j$ and $x_j$ for some numerical quadrature formula

$$\begin{aligned} E\{V(x^+; a) | x_j, u)\} &= \int \hat{V}(x^+; a) dF(x^+ | x_j, u) \\ &= \int \hat{V}(g(x_j, u, \varepsilon); a) dF(\varepsilon) \\ &\doteq \sum_{\ell} \omega_\ell \hat{V}(g(x_j, u, \varepsilon_\ell); a) \end{aligned}$$

  - Maximization step: for $x_i \in X$, evaluate

$$v_i = (T\hat{V})(x_i)$$

  - Fitting step:
    - Data: $(v_i, x_i)$, $i = 1, \cdots, n$
    - Objective: find an $a \in R^m$ such that $\hat{V}(x; a)$ best fits the data

# General Parametric Approach: Value Function Iteration

$$\begin{aligned} a &\longrightarrow \hat{V}(x;a) \\ &\longrightarrow (v_i, x_i),\ i = 1, \cdots, n \\ &\longrightarrow V_{new}(x) = \hat{V}(x; a_{new}) \end{aligned}$$

- Comparison with discretization
  - This procedure examines only a finite number of states, $x_i$:
    - But does *not* assume that the state is always in this finite set.
    - Choices for the $x_i$ are guided by approximation methods
  - Procedure examines only a finite number of $\varepsilon$ values for the stochastic shocks
    - But does *not* assume that they are the only ones realized
    - Choices for the $\varepsilon_i$ come from quadrature methods

- Synergies
  - Smooth interpolation helps Newton's method for max step.
  - Smooth interpolation allows more efficient quadrature in (12.7.5).
  - Efficient quadrature reduces cost of computing objective in max problem
- Finite-horizon problems
  - Must use value function iteration since $V(x,t)$ depends on time $t$.
  - Begin with terminal value function, $V(x,T)$
  - Compute approximations for each $V(x,t)$, $t = T-1, T-2$, etc.

Algorithm 12.5: Parametric Dynamic Programming with Value Function Iteration

| | |
|---|---|
| Objective: | Solve the Bellman equation, (12.7.1). |
| Step 0: | Choose functional form $\hat{V}(x; a)$, and choose the approximation grid, $X = \{x_1, ..., x_n\}$. Make initial guess $\hat{V}(x; a^0)$, and choose stopping criterion $\epsilon > 0$. |
| Step 1: | Maximization step: Compute $v_j = (T\hat{V}(\cdot; a^i))(x_j)$ for all $x_j \in X$. |
| Step 2: | Fitting step: Using the appropriate approximation method, compute the $a^{i+1} \in R^m$ such that $\hat{V}(x; a^{i+1})$ approximates the $(v_i, x_i)$ data. |
| Step 3: | If $\parallel \hat{V}(x; a^i) - \hat{V}(x; a^{i+1}) \parallel < \epsilon$, STOP; else go to step 1. |

# Convergence

- $T$ is a contraction mapping
- $\hat{T}$ may be neither monotonic nor a contraction
- Shape problems
  - Standard approximation methods do not preserve shape
    - monotone data may not result in a monotone approximation
    - concave data may not result in a concave approximation
  - Shape problems may become worse with value function iteration

# General Parametric Approach: Policy Iteration

- Basic Bellman equation:

$$V(x) = \max_{u \in D(x)} \pi(u,x) + \beta\, E\{V(x^+)|x,u)\} \equiv (TV)(x).$$

- Policy iteration:
  - Current guess: $V(x) \doteq \hat{V}(x;a)$ for some $a \in R^m$
  - Iteration: compute optimal policy today if $\hat{V}(x;a)$ is value tomorrow

$$U(x) = \arg\max_{u \in D(x)} \pi(x_i,u,t) + \beta E\left\{\hat{V}\left(x^+;a\right)|x,u)\right\}$$

  - If solution is interior, then $U(x_i)$ solves

$$0 = \pi_u(x_i, U(x_i), t) + \beta \frac{d}{du}\left(E\left\{\hat{V}\left(x^+;a\right)|x_i, U(x_i))\right\}\right)$$

  - Take $u_i = U(x_i)$ data for $x_i$ nodes, and approximate $U(x)$ with some method $\hat{U}(x;b)$ with parameters $b$
  - Compute the value function if the policy $\hat{U}(x;b)$ is used forever. This is defined by the linear integral equation

$$\hat{V}(x;a') = \pi(\hat{U}(x;b),x) + \beta\, E\{\hat{V}(x^+;a')|x,\hat{U}(x;b))\}$$

    that can be solved by a projection method

# Summary

- Discretization methods
  - Easy to implement
  - Numerically stable
  - Amenable to many accelerations
  - Inefficient approximation to continuous problems
- Continuous approximation methods
  - Can exploit smoothness in problems
  - Must work to avoid numerical instabilities