Numerical Methods in Economics

MIT Press, 1998

Chapter 12 Notes
Numerical Dynamic Programming

Kenneth L. Judd

Hoover Institution

November 5, 2007

Discrete-Time Dynamic Programming

- Objective:

$$E\left\{\sum_{t=1}^{T}\pi(x_t,\,u_t,\,t)+W(x_{T+1})\right\},\tag{12.1.1}$$

  - X: set of states
  - $\mathcal{D}$: the set of controls
  - $\pi(x,\,u,\,t)$ payoffs in period t, for $x\in X$ at the beginning of period t, and control $u\in\mathcal{D}$ is applied in period t.
  - $D(x,t)\subseteq\mathcal{D}$: controls which are feasible in state x at time t.
  - $F(A;x,u,t)$: probability that $x_{t+1}\in A\subset X$ conditional on time t control and state

- Value function

$$V(x,t)\equiv\sup_{\mathcal{U}(x,t)}E\left\{\sum_{s=t}^{T}\pi(x_s,\,u_s,\,s)+W(x_{T+1})|x_t=x\right\}.\tag{12.1.2}$$

- Bellman equation

$$V(x,t)=\sup_{u\in D(x,t)}\pi(x,\,u,\,t)+E\left\{V(x_{t+1},\,t+1)|x_t=x,u_t=u\right\}\tag{12.1.3}$$

- Existence: boundedness of π is sufficient

Autonomous, Infinite-Horizon Problem:

- Objective:

$$\max_{u_t} E\left\{\sum_{t=1}^{\infty} \beta^t \pi(x_t,\, u_t)\right\} \tag{12.1.1}$$

  - X: set of states
  - $\mathcal{D}$: the set of controls
  - $D(x) \subseteq \mathcal{D}$: controls which are feasible in state x.
  - $\pi(x,\, u)$ payoff in period t if $x \in X$ at the beginning of period t, and control $u \in \mathcal{D}$ is applied in period t.
  - $F(A; x, u)$: probability that $x^+ \in A \subset X$ conditional on current control u and current state x.

- Value function definition: if $\mathcal{U}(x)$ is set of all feasible strategies starting at x.

$$V(x) \equiv \sup_{\mathcal{U}(x)} E\left\{\sum_{t=0}^{\infty} \beta^t \pi(x_t,\, u_t) \middle| \, x_0 = x\right\}, \tag{12.1.8}$$

- Bellman equation for $V(x)$

$$V(x) = \sup_{u \in D(x)} \; \pi(x,\, u) + \beta\, E\, \left\{ V(x^{+}) | x, u \right\} \equiv (TV)(x), \tag{12.1.9}$$

- Optimal policy function, $U(x)$, if it exists, is defined by

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

- Standard existence theorem:

Theorem 1 *If X is compact, $\beta < 1$, and π is bounded above and below, then the map*

$$TV = \sup_{u \in D(x)} \; \pi(x, u) + \beta E \left\{ V(x^{+}) \mid x, u \right\} \tag{12.1.10}$$

is monotone in V, is a contraction mapping with modulus β in the space of bounded functions, and has a unique fixed point.

Applications

- Economics
  - Business investment
  - Life-cycle decisions on labor, consumption, education
  - Portfolio problems
  - Economic policy
- Operations Research
  - Scheduling, queueing
  - Blood bank
  - See new book by Powell - "Approximate Dynamic Programming"
- Climate change
  - Business response to climate policies
  - Optimal policy response to global warming problems

Deterministic Growth Example

- Problem:

$$
\begin{aligned}
V(k_0) = \max_{c_t} \textstyle\sum_{t=0}^{\infty} \beta^t u(c_t), \\
k_{t+1} = F(k_t) - c_t \\
k_0 \text{ given}
\end{aligned}
\tag{12.1.12}
$$

  - Euler equation:

$$u'(c_t) = \beta u'(c_{t+1}) F'(k_{t+1})$$

  - Bellman equation

$$V(k) = \max_c \; u(c) + \beta V(F(k) - c). \tag{12.1.13}$$

  - Solution to (12.1.12) is a policy function $C(k)$ and a value function $V(k)$ satisfying

$$0 = u'(C(k))F'(k) - V'(k) \tag{12.1.15}$$

$$V(k) = u(C(k)) + \beta V(F(k) - C(k)) \tag{12.1.16}$$

- (12.1.16) defines the value of an arbitrary policy function $C(k)$, not just for the optimal $C(k)$.
- The pair (12.1.15) and (12.1.16)
  - expresses the value function given a policy, and
  - a first-order condition for optimality.

Stochastic Growth Accumulation

- Problem:

$$
\begin{aligned}
V(k,\theta) &= \max_{c_t,\ell_t} \; E\left\{\sum_{t=0}^{\infty} \beta^t \; u(c_t)\right\} \\
k_{t+1} &= F(k_t,\theta_t) - c_t \\
\theta_{t+1} &= g(\theta_t,\varepsilon_t) \\
\varepsilon_t &: \text{ i.i.d. random variable} \\
k_0 &= k, \;\; \theta_0 = \theta.
\end{aligned}
$$

- State variables:
  - k: productive capital stock, endogenous
  - θ: productivity state, exogenous

- The dynamic programming formulation is

$$
\begin{aligned}
V(k,\theta) &= \max_{c} \;\; u(c) + \beta E\{V(F(k,\theta)-c,\theta^+)|\theta\} \qquad (12.1.21)\\
\theta^+ &= g(\theta,\varepsilon)
\end{aligned}
$$

- The control law $c = C(k,\theta)$ satisfies the first-order conditions

$$
0 = u_c\left(C(k,\theta)\right) - \beta\, E\left\{u_c(C(k^+,\theta^+))F_k(k^+,\theta^+) \mid \theta\right\}, \qquad (12.1.23)
$$

where

$$
k^+ \equiv F(k,L(k,\theta),\theta) - C(k,\theta),
$$

General Stochastic Accumulation

- Problem:

$$V(k,\theta) = \max_{c_t,\ell_t} \; E\left\{\sum_{t=0}^{\infty} \beta^t \; u(c_t,\ell_t)\right\}$$
$$k_{t+1} = F(k_t,\ell_t,\theta_t) - c_t$$
$$\theta_{t+1} = g(\theta_t,\varepsilon_t)$$
$$k_0 = k, \;\; \theta_0 = \theta.$$

- State variables:
  - k: productive capital stock, endogenous
  - θ: productivity state, exogenous

- The dynamic programming formulation is

$$V(k,\theta) = \max_{c,\ell} \; u(c,\ell) + \beta E\{V(F(k,\ell,\theta) - c, \theta^+)|\theta\}, \tag{12.1.21}$$

where θ^+ is next period's θ realization.

- Control laws $c = C(k,\theta)$ and $\ell = L(k,\theta)$ satisfy foc's

$$\begin{aligned} 0 &= u_c(C(k,\theta), L(k,\theta))F_k(k, L(k,\theta),\theta) - V_k(k,\theta), \\ 0 &= u_\ell(C(k,\theta), L(k,\theta)) + F_\ell(k,\theta)u_c(C(k,\theta), L(k,\theta)). \end{aligned}$$

- Euler equation implies

$$0 = u_c\left(C(k,\theta), L(k,\theta)\right) - \beta\, E\left\{u_c(C(k^+,\theta^+),\ell^+)F_k(k^+,\ell^+,\theta^+) \mid \theta\right\}, \tag{12.1.23}$$

where next period's capital stock and labor supply are

$$\begin{aligned} k^+ &\equiv F(k, L(k,\theta),\theta) - C(k,\theta), \\ \ell^+ &\equiv L(k^+,\theta^+), \end{aligned}$$

Discrete State Space Problems

- State space $X = \{x_i,\ i = 1, \cdots, n\}$
- Controls $\mathcal{D} = \{u_i | i = 1, ..., m\}$
- $q_{ij}^t(u) = \Pr\left(x_{t+1} = x_j | x_t = x_i, u_t = u\right)$
- $Q^t(u) = \left(q_{ij}^t(u)\right)_{i,j}$: Markov transition matrix at t if $u_t = u$.

Value Function iteration

- Terminal value:

$$V_i^{T+1} = W(x_i), \; i = 1, \cdots, n.$$

- Bellman equation: time t value function is

$$V_i^t = \max_u \, [\pi(x_i, u, t) + \beta \sum_{j=1}^n q_{ij}^t(u) \, V_j^{t+1}], \; i = 1, \cdots, n$$

- Bellman equation can be directly implemented.
  - Called *value function iteration*
  - It is only choice for finite-horizon problems because each period has a different value function.

- Infinite-horizon problems
  - Bellman equation is now a simultaneous set of equations for V_i values:

$$V_i = \max_u \left[\pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u)\, V_j \right], \;\; i = 1, \cdots, n$$

  - Value function iteration is now

$$V_i^{k+1} = \max_u \left[\pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u)\, V_j^k \right], \;\; i = 1, \cdots, n$$

  - Can use value function iteration with arbitrary V_i^0 and iterate $k \to \infty$.
  - Error is given by contraction mapping property:

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

Algorithm 12.1: Value Function Iteration Algorithm

Objective: Solve the Bellman equation, (12.3.4).

Step 0: Make initial guess V^0; choose stopping criterion $\epsilon > 0$.

Step 1: For $i = 1, ..., n$, compute

$V_i^{\ell+1} = \max_{u \in D} \ \pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u) V_j^{\ell}$.

Step 2: If $\| V^{\ell+1} - V^{\ell} \| < \epsilon$, then go to step 3; else go to step 1.

Step 3: Compute the final solution, setting

$U^* = \mathcal{U} V^{\ell+1}$,

$P_i^* = \pi(x_i, U_i^*), \qquad i = 1, \cdots, n$,

$V^* = (I - \beta Q^{U^*})^{-1} P^*$,

and STOP.

Output:

Policy Iteration (a.k.a. Howard improvement)

- Value function iteration is a slow process
  - Linear convergence at rate β
  - Convergence is particularly slow if β is close to 1.
- Policy iteration is faster
  - Current guess:
$$V_i^k, \ i = 1, \cdots, n.$$
  - Iteration: compute optimal policy today if V^k is value tomorrow:
$$U_i^{k+1} = \arg\max_u \left[\pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u)\, V_j^k \right], \ i = 1, \cdots, n,$$
  - Compute the value function if the policy U^{k+1} is used forever, which is solution to the linear system
$$V_i^{k+1} = \pi\left(x_i, U_i^{k+1}\right) + \beta \sum_{j=1}^{n} q_{ij}(U_i^{k+1})\, V_j^{k+1}, \ i = 1, \cdots, n,$$

- Comments:
  - Policy iteration depends on only monotonicity
  - Policy iteration is faster than value function iteration
    - If initial guess is above or below solution then policy iteration is between truth and value function iterate
    - Works well even for β close to 1.

Algorithm 12.2: Policy Function Algorithm

Objective:	Solve the Bellman equation, (12.3.4).
Step 0:	Choose stopping criterion $\epsilon > 0$.
	EITHER make initial guess, V^0, for the value function and go to step 1,
	OR make initial guess, U^1, for the policy function and go to step 2.
Step 1:	$U^{\ell+1} = \mathcal{U} V^{\ell}$
Step 2:	$P_i^{\ell+1} = \pi\left(x_i, U_i^{\ell+1}\right), \qquad i = 1, \cdots, n$
Step 3:	$V^{\ell+1} = \left(I - \beta Q^{U^{\ell+1}}\right)^{-1} P^{\ell+1}$
Step 4:	If $\parallel V^{\ell+1} - V^{\ell} \parallel < \epsilon$, STOP; else go to step 1.

- Modified policy iteration
  - If n is large, difficult to solve policy iteration step
  - Alternative approximation: Assume policy $U^{\ell+1}$ is used for k periods:

$$V^{\ell+1} = \sum_{t=0}^{k} \beta^t \left(Q^{U^{\ell+1}}\right)^t P^{\ell+1} + \beta^{k+1} \left(Q^{U^{\ell+1}}\right)^{k+1} V^\ell. \tag{12.4.1}$$

  - Theorem 4.1 points out that as the policy function gets close to U^*, the linear rate of convergence approaches β^{k+1}. Hence convergence accelerates as the iterates converge.

Theorem 2 *(Putterman and Shin) The successive iterates of modified policy iteration with k steps, (12.4.1), satisfy the error bound*

$$\frac{\|V^* - V^{\ell+1}\|}{\|V^* - V^\ell\|} \leq \min\left[\beta,\ \frac{\beta(1-\beta^k)}{1-\beta} \parallel U^\ell - U^* \parallel + \beta^{k+1}\right] \tag{12.4.3}$$

Gaussian acceleration methods for infinite-horizon models

- Key observation: Bellman equation is a simultaneous set of equations

$$V_i = \max_u \left[\pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u)\, V_j \right], \; i = 1, \cdots, n$$

- Idea: Treat problem as a large system of nonlinear equations
- Value function iteration is the *pre-Gauss-Jacobi* iteration

$$V_i^{k+1} = \max_u \left[\pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u)\, V_j^k \right], \; i = 1, \cdots, n$$

- True Gauss-Jacobi is

$$V_i^{k+1} = \max_u \left[\frac{\pi(x_i, u) + \beta \sum_{j \neq i} q_{ij}(u)\, V_j^k}{1 - \beta q_{ii}(u)} \right], \; i = 1, \cdots, n$$

- pre-Gauss-Seidel iteration
  - Value function iteration is a pre-Gauss-Jacobi scheme.
  - Gauss-Seidel alternatives use new information immediately
    * Suppose we have V_i^ℓ
    * At each x_i, given $V_j^{\ell+1}$ for $j < i$, compute $V_i^{\ell+1}$ in a pre-Gauss-Seidel fashion

$$V_i^{\ell+1} = \max_u \; \pi(x_i, u) + \beta \sum_{j<i} q_{ij}(u) V_j^{\ell+1} + \beta \sum_{j \geq i} q_{ij}(u) V_j^\ell \tag{12.4.7}$$

    * Iterate (12.4.7) for $i = 1, .., n$

- Gauss-Seidel iteration

  - Suppose we have V_i^ℓ
  - If optimal control at state i is u, then Gauss-Seidel iterate would be

$$V_i^{\ell+1} = \pi(x_i, u) + \beta \frac{\sum_{j<i} q_{ij}(u) V_j^{\ell+1} + \sum_{j>i} q_{ij}(u) V_j^\ell}{1 - \beta q_{ii}(u)}$$

  - Gauss-Seidel: At each x_i, given $V_j^{\ell+1}$ for $j < i$, compute $V_i^{\ell+1}$

$$V_i^{\ell+1} = \max_u \frac{\pi(x_i, u) + \beta \sum_{j<i} q_{ij}(u) V_j^{\ell+1} + \beta \sum_{j>i} q_{ij}(u) V_j^\ell}{1 - \beta q_{ii}(u)}$$

  - Iterate this for $i = 1, .., n$

- Gauss-Seidel iteration: better notation

  - No reason to keep track of ℓ, number of iterations
  - At each x_i,

$$V_i \longleftarrow \max_u \frac{\pi(x_i, u) + \beta \sum_{j<i} q_{ij}(u) V_j + \beta \sum_{j>i} q_{ij}(u) V_j}{1 - \beta q_{ij}(u)}$$

  - Iterate this for $i = 1, .., n, 1, ..., $ etc.

Upwind Gauss-Seidel

- Gauss-Seidel methods in (12.4.7) and (12.4.8)
  - Sensitive to ordering of the states.
  - Need to find good ordering schemes to enhance convergence.
- Example:
  - Two states, x_1 and x_2, and two controls, u_1 and u_2
    * u_i causes state to move to x_i, $i = 1, 2$
    * Payoffs:

$$\begin{aligned} \pi(x_1, u_1) &= -1,\ \pi(x_1, u_2) = 0, \\ \pi(x_2, u_1) &= 0,\ \pi(x_2, u_2) = 1. \end{aligned} \tag{12.4.9}$$

    * $\beta = 0.9$.
  - Solution:
    * Optimal policy: always choose u_2, moving to x_2
    * Value function:

$$V(x_1) = 9,\ V(x_2) = 10.$$

    * x_2 is the unique steady state, and is stable

- Value iteration with $V^0(x_1) = V^0(x_2) = 0$ converges linearly:

$$V^1(x_1) = 0,\ V^1(x_2) = 1,\ U^1(x_1) = 2,\ U^1(x_2) = 2,$$
$$V^2(x_1) = 0.9,\ V^2(x_2) = 1.9,\ U^2(x_1) = 2,\ U^2(x_2) = 2,$$
$$V^3(x_1) = 1.71,\ V^3(x_2) = 2.71,\ U^3(x_1) = 2,\ U^3(x_2) = 2,$$

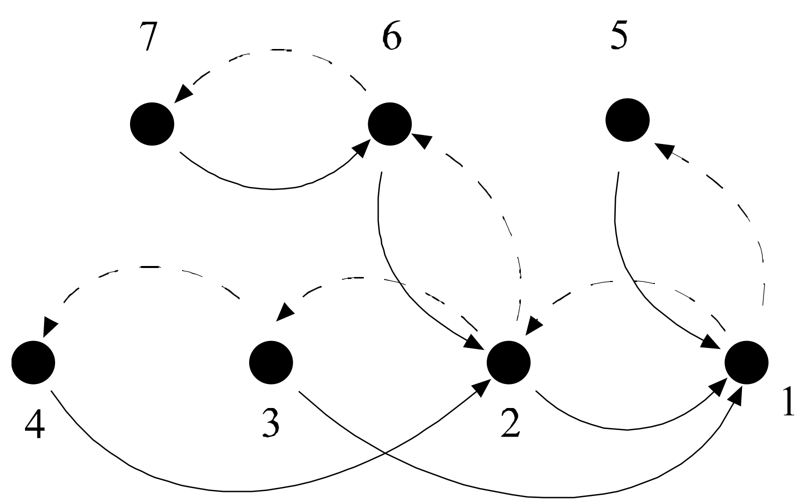
- Policy iteration converges after two iterations

$$V^1(x_1) = 0,\ V^1(x_2) = 1,\ U^1(x_1) = 2,\ U^1(x_2) = 2,$$
$$V^2(x_1) = 9,\ V^2(x_2) = 10,\ U^2(x_1) = 2,\ U^2(x_2) = 2,$$

- Upwind Gauss-Seidel
  - Value function at absorbing states is trivial to compute
    - Suppose s is absorbing state with control u
    - $V(s) = \pi(s,u)/(1-\beta)$.
  - With absorbing state $V(s)$ we compute $V(s')$ of any s' that sends system to s.

$$V(s') = \pi(s', u) + \beta V(s)$$

  - With $V(s')$, we can compute values of states s'' that send system to s'; etc.



- Alternating Sweep
  - It may be difficult to find proper order.
  - Idea: alternate between two approaches with different directions.

$$\begin{aligned} W &= V^k, \\ W_i &= \max_u \ \pi(x_i,u) + \beta \textstyle\sum_{j=1}^n q_{ij}(u) W_j, \ \ i = 1,2,3,...,n \\ W_i &= \max_u \ \pi(x_i,u) + \beta \textstyle\sum_{j=1}^n q_{ij}(u) W_j, \ \ i = n, n-1, ..., 1 \\ V^{k+1} &= W \end{aligned}$$

  - Will always work well in one-dimensional problems since state moves either right or left, and alternating sweep will exploit this half of the time.
  - In two dimensions, there may still be a natural ordering to be exploited.
- Simulated Upwind Gauss-Seidel
  - It may be difficult to find proper order in higher dimensions
  - Idea: simulate using latest policy function to find downwind direction
    - Simulate to get an example path, $x_1, x_2, x_3, x_4, ..., x_m$
    - Execute Gauss-Seidel with states $x_m, x_{m-1}, x_{m-2}, ..., x_1$

Linear Programming Approach

- If $\mathcal{D}$ is finite, we can reformulate dynamic programming as a linear programming problem.
- (12.3.4) is equivalent to the linear program

$$\begin{array}{ll} \min_{V_i} \sum_{i=1}^{n} V_i & \\ s.t. \quad V_i \geq \pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u) V_j, \ \forall i, u \in \mathcal{D}, \end{array} \tag{12.4.10}$$

- Computational considerations
  - (12.4.10) may be a large problem
  - Trick and Zin (1997) pursued an acceleration approach with success.
  - OR literature did not favor this approach, but recent work by Daniela Pucci de Farias and Ben van Roy has revived interest.

Continuous states: discretization

- Method:
  - "Replace" continuous X with a finite

$$X^* = \{x_i,\, i = 1, \cdots, n\} \subset X$$

  - Proceed with a finite-state method.
- Problems:
  - Sometimes need to alter space of controls to assure landing on an x in X.
  - A fine discretization often necessary to get accurate approximations

Continuous States: Linear-Quadratic Dynamic Programming

- Problem:

$$\max_{u_t} \sum_{t=0}^{T} \beta^t \left(\frac{1}{2} x_t^\top Q_t x_t + u_t^\top R_t x_t + \frac{1}{2} u_t^\top S_t u_t \right) + \frac{1}{2} x_{T+1}^\top W_{T+1} x_{T+1} \tag{12.6.1}$$

$$x_{t+1} = A_t x_t + B_t u_t,$$

- Bellman equation:

$$V(x,t) = \max_{u_t} \frac{1}{2} x^\top Q_t x + u_t^\top R_t x + \frac{1}{2} u_t^\top S_t u_t + \beta V(A_t x + B_t u_t, t+1). \tag{12.6.2}$$

Finite horizon

- Key fact: We know solution is quadratic, solve for the unknown coefficients
- The guess $V(x,t) = \frac{1}{2} x^\top W_{t+1} x$ implies f.o.c.

$$0 = S_t u_t + R_t x + \beta B_t^\top W_{t+1}(A_t x + B_t u_t),$$

  - F.o.c. implies the time t control law

$$\begin{aligned} u_t &= -(S_t + \beta B_t^\top W_{t+1} B_t)^{-1}(R_t + \beta B_t^\top W_{t+1} A_t) x \\ &\equiv U_t x. \end{aligned} \tag{12.6.3}$$

  - Substitution into Bellman implies *Riccati equation* for W_t:

$$W_t = Q_t + \beta A_t^\top W_{t+1} A_t + (\beta B_t^\top W_{t+1} A_t + R_t^\top) U_t. \tag{12.6.4}$$

  - Value function method iterates (12.6.4) beginning with known W_{T+1} matrix of coefficients.

Autonomous, Infinite-horizon case.

- Assume $R_t = R$, $Q_t = Q$, $S_t = S$, $A_t = A$, and $B_t = B$
- The guess $V(x) \equiv \frac{1}{2}x^\top Wx$ implies the *algebraic Riccati equation*

$$\begin{aligned} W =& Q + \beta A^\top WA - (\beta B^\top WA + R^\top) \\ & \times (S + \beta B^\top WB)^{-1}(\beta B^\top WB + R^\top). \end{aligned} \tag{12.6.5}$$

- Two convergent procedures:
  - Value function iteration:

$$\begin{aligned} W_0 &: \text{a negative definite initial guess} \\ W_{k+1} =& Q + \beta A^\top W_k A - (\beta B^\top W_k A + R^\top) \\ & \times (S + \beta B^\top W_k B)^{-1}(\beta B^\top W_k B + R^\top). \end{aligned} \tag{12.6.6}$$

  - Policy function iteration:

$$\begin{aligned} W_0 &: \text{initial guess} \\ U_{i+1} =& -(S + \beta B^\top W_i B)^{-1}(R + \beta B^\top W_i A) : \text{optimal policy for } W_i \\ W_{i+1} =& \frac{\frac{1}{2}Q + \frac{1}{2}U_{i+1}^\top S U_{i+1} + U_{i+1}^\top R}{1-\beta} : \text{value of } U_i \end{aligned}$$

Lessons

- We used a functional form to solve the dynamic programming problem
- We solve for unknown coefficients
- We did not restrict either the state or control set
- Can we do this in general?

Continuous Methods for Continuous-State Problems

- Basic Bellman equation:

$$V(x) = \max_{u \in D(x)} \pi(u, x) + \beta\, E\{V(x^+)|x, u\} \equiv (TV)(x). \tag{12.7.1}$$

  - Discretization essentially approximates V with a step function
  - Approximation theory provides better methods to approximate continuous functions.

- General Task

  - Find good approximation for V
  - Identify parameters

General Parametric Approach: Approximating $V(x)$

- Choose a finite-dimensional parameterization

$$V(x) \doteq \hat{V}(x; a), \ a \in R^m \tag{12.7.2}$$

  and a finite number of states

$$X = \{x_1, x_2, \cdots, x_n\}, \tag{12.7.3}$$

  - 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

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

$$v_j = (TV)(x_j) = \max_{u \in D(x_j)} \pi(u, x_j) + \beta \int \hat{V}(x^+; a) dF(x^+ | x_j, u) \tag{12.7.5}$$

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

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

  – Integration step: for ω_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

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

    * Hot starts
    * Concave stopping rules

  – 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
    * Methods: determined by $\hat{V}(x; a)$

Approximating T with Hermite Data

- Conventional methods just generate data on $V(x_j)$:

$$v_j = \max_{u \in D(x_j)} \pi(u, x_j) + \beta \int \hat{V}(x^+; a) dF(x^+ | x_j, u) \tag{12.7.5}$$

- Envelope theorem:
  - If solution u is interior,

$$v_j' = \pi_x(u, x_j) + \beta \int \hat{V}(x^+; a) dF_x(x^+ | x_j, u)$$

  - If solution u is on boundary

$$v_j' = \mu + \pi_x(u, x_j) + \beta \int \hat{V}(x^+; a) dF_x(x^+ | x_j, u)$$

  where μ is a Kuhn-Tucker multiplier

- Since computing v_j' is cheap, we should include it in data:
  - Data: $(v_i, v_i', x_i), \ i = 1, \cdots, n$
  - Objective: find an $a \in R^m$ such that $\hat{V}(x; a)$ best fits Hermite data
  - Methods: determined by $\hat{V}(x; a)$

General Parametric Approach: Value Function Iteration

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

- Comparison with discretization
  - This procedure examines only a finite number of points, but does *not* assume that future points lie in same finite set.
  - Our choices for the x_i are guided by systematic numerical considerations.
- Synergies
  - Smooth interpolation schemes allow us to use Newton's method in the maximization step.
  - They also make it easier to evaluate the integral in (12.7.5).
- Finite-horizon problems
  - Value function iteration is only possible procedure 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 for $\hat{V}(x;a)$, 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
  - An instructive example

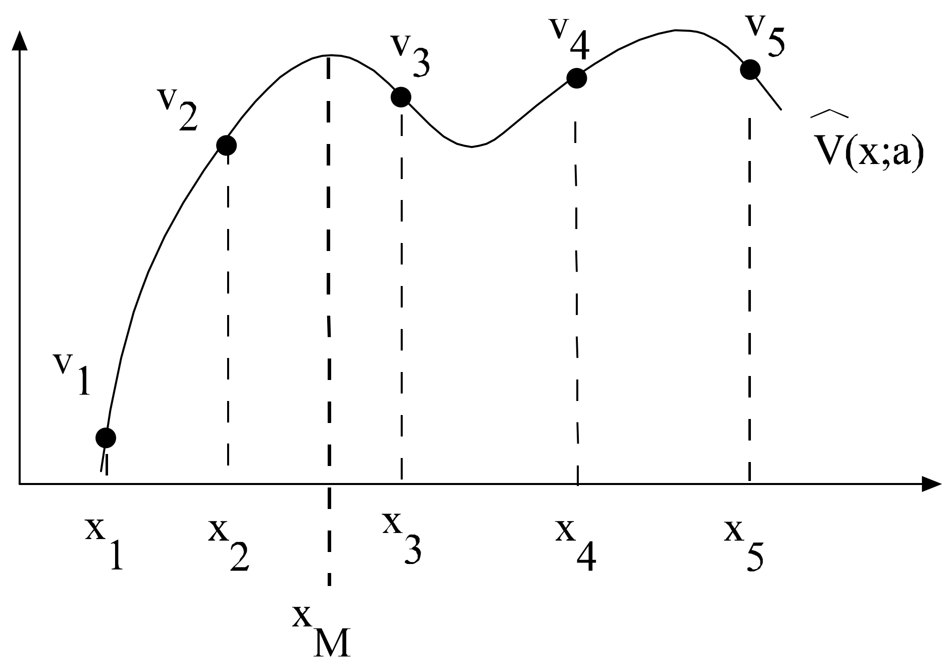


Figure 1:

  - Shape problems may become worse with value function iteration

- Solution to shape problems
  - Use shape-preserving approximations
    - Piecewise linear preserves shape in one dimension.
    - Multilinear approximation does not preserve shape
    - Shape preserving splines are available for dimensions one and two.
  - Impose shape restrictions in fitting
    - Use least squares, not interpolation
    - Add shape constraints to least squares problem
      - Demand correct slopes at some points
      - Demand correct curvature at some points.
    - These methods work well in one dimension, but slow algorithm down considerably for higher dimensions
  - Open research question: What is the best combination of smooth functional form and fitting procedure that preserves shape?

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: a finite-dimensional linear parameterization

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

  - 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

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

  - Take $u_i = U(x_i)$ data for x_i nodes, and use some approximation scheme $\hat{U}(x;b)$ with parameters b to approximate $U(x)$
  - Compute the value function if the policy $\hat{U}(x;b)$ is used forever. This is solution to 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
  - Poor approximation to continuous problems
- Continuous approximation methods
  - Can exploit smoothness in problems
  - Must work to avoid numerical instabilities
  - Acceleration is less possible