Shape-Preserving Approximation Methods

Cecilia Silvina Diaz Campo

Based on Kenneth Judd's lectures at the nICE 2018

Department of Economics
University of Western Ontario

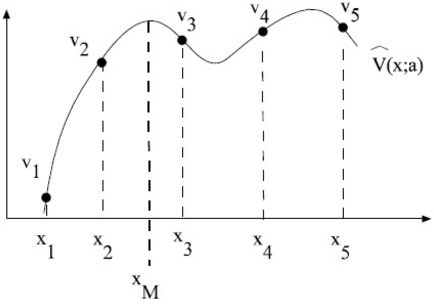
Western
UNIVERSITY · CANADA

Introduction

- Dynamic programming is the essential tool in dynamic economic analysis.
- Numerical methods typically approximate the value function and use value function iteration to compute the value function for the optimal policy.
- Polynomial approximations are natural choices for approximating value functions when we know that the true value function is smooth.
- This approach is unstable because standard methods such as interpolation and least squares fitting do not preserve shape.
- Introduce shape-preserving approximation methods that stabilize value function iteration.

Why do we care about shape preservation?

- **Interpolation** is any procedure that finds a “nice” function that goes through a collection of prescribed points.
- We refer to the curvature and monotonicity properties of a function as aspects of its **shape**.
- Concave (monotone) data may lead to nonconcave (nonmonotone) approximations.



Example: Neoclassical Growth Model

The Dynamic Programming version of the discrete-time finite-horizon optimal growth problem is the Bellman equation:

$$\begin{aligned} V_t(k) = \quad & \max_{c} \quad u(c) + \beta V_{t+1}(k^+) \\ & \text{s.t.} \quad k^+ = F(k) - c \\ & \qquad 0 \leq k^+ \leq \bar{K} \\ & \qquad c \geq \epsilon \\ & \qquad k_0, \ V_T(k) \text{ given} \end{aligned}$$

- The value function is defined on a continuum $k \in [0, \bar{K}]$.
- But computers cannot deal with this directly.

Strategies to solve the model

1. **Discretize the State Space:**
   - discretize continuous state space $k \in [0, \bar{K}]$ so that $k \in \{k_1, \ldots, k_{n_K}\}$.
   - the values of c must be such that they keep k in the grid.
   - **no shape issues.**
2. **Linear-Quadratic Models:**
   - assume that value function is quadratic and policy function is linear.
   - solve for the unknown coefficients of these functions.
   - **no shape issues.**
3. **Parametric Dynamic Programming:**
   - parameterize the critical functions and find some parameter choice which generates a good approximation.
   - **shape issues.**

Generalization of the Example

Deterministic Finite-Horizon Dynamic Programming Problem:

$$V_t(x) = \max_{a \in \mathcal{D}(x,t)} u_t(x,a) + \beta V_{t+1}(x^+)$$
$$\text{s.t.} \quad x^+ = g_t(x,a)$$
$$V_T(x) \text{ given}$$

- x: vector of continuous state variables in $\mathbb{R}^d$
- $V_t(x)$: value function at time $t \leq T$
- $a \in \mathcal{D}(x,t)$: vector of choice variables in the feasible set
- x^+: value of the continuous state variables in the next period
- $g_t(x,a)$: time-specific law of motion
- $u_t(x,a)$: payoff flow at time t
- β: discount factor

Parametric Dynamic Programming

- Useful when we don't want to discretize the state space nor assume a specific functional form for the value function.
- Approximate the value function with a continuous parametric function:

  $$V(x) \approx \hat{V}(x; c)$$

  where $\hat{V}(x; c)$ is an approximation with parameters c.
- Solving the approximate problem means finding a set of parameters c such that $\hat{V}(x; c)$ "approximately" satisfies the Bellman equation.

General Algorithm

Parametric DP with value function iteration for-finite horizon problems:

- **Initialization:**
  - choose approximation nodes $X_t = \{x_{it} : i = 1, \ldots, m_t\}$ for every $t < T$.
  - choose a functional form for $\hat{V}(x; c)$.
  - let $\hat{V}(x; c^T) \equiv V_T(x)$ given.
- for $t = T-1, T-2, \ldots, 0$, iterate through steps 1 and 2.
  1. **Maximization step:** Compute

     $$v_i = \max_{a_i \in \mathcal{D}(x_i, t)} u_t(x_i, a_i) + \beta \hat{V}(x_i^+; c^{t+1})$$
     $$\text{s.t.} \quad x_i^+ = g(x_i, a_i)$$

     for each $x_{it} \in X_t$, $i = 1, \ldots, m_t$.
  2. **Fitting step:** Compute the c^t such that $\hat{V}(x; c^t)$ approximates (x_i, v_i) data.

Approximation Method

An approximation method consists of two parts:

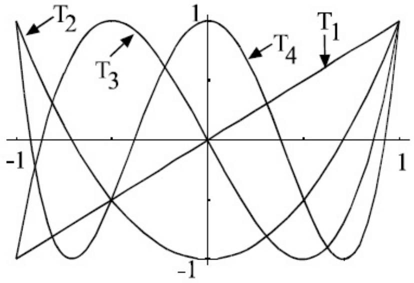
1. **Basis functions:** $\phi_j(x)$, $j = 0, 1, \ldots, \infty$.
    - a set of basic functional building blocks that can be stacked on top of one another so as to have the features that we need.
    - Every continuous function in the function space can be represented as a linear combination of basis functions.
    - $\hat{V}(x; c) = \sum_{j=0}^{n} c_j \phi_j(x)$ is a degree-n approximation.
    - Example: $\Phi = \{\phi_j(x)\}_{j=0}^{\infty} = \{1, x, x^2, \ldots, x^n, \ldots\}$.
2. **Approximation nodes/Collocation points:** can be chosen as
    - uniformly spaced nodes.
    - Chebyshev nodes.
    - some other specified nodes.

Chebyshev orthogonal polynomials approximation

- Chebyshev polynomials are a special class of polynomials that are good for smooth nonperiodic functions.
- For every $t < T$:
  1. Set the degree of the polynomial approximation we will use: n.
  2. Choose $m \geq n+1$ approximation nodes/collocation points: these must be m points in the period-t state space.
     - if $m = n+1$, it is interpolation.
     - if $m > n+1$, it is regression.
  3. Chebyshev orthogonal polynomials (basis functions $\phi_j(x) = T_j(x)$):
     - Domain: $[-1, 1]$
     - $T_0(x) = 1$
     - $T_1(x) = x$
     - $T_{j+1}(x) = 2x\, T_j(x) - T_{j-1}(x)$

Chebyshev polynomial approximation

- Chebyshev orthogonal polynomials graphs for $j = 1, 2, 3, 4$:



- General intervals: change of variable from $x \in [a, b]$ to $y \in [-1, 1]$.
- $y = -1 + 2\,\frac{x-a}{b-a}$ and $T_j^*(x) \equiv T_j(y) = T_j\left(-1 + 2\,\frac{x-a}{b-a}\right)$

Chebyshev Approximation Algorithm in $\mathbb{R}^1$

- **Initialization:** for every $t < T$
  - compute the set of Chebyshev approximation nodes (the zeros of the m^{th} degree polynomial), $X_t = \{x_{it} \colon i = 1 \ldots, m_t\}$.
  - $\hat{V}_t(x_{it}; c^t) = \sum_{j=0}^{n_t} c_j^t \, T_j^*(x_{it})$
  - let $\hat{V}_T(x; c^T) \equiv V_T(x)$, given.
- for $t = T-1, T-2, \ldots, 0$:
  - **Maximization step:** Compute

$$v_{it} = \max_{a_{it} \in \mathcal{D}(x_{it}, t)} u_t(x_{it}, a_{it}) + \beta \hat{V}_{t+1}(x_{it}^+; c^{t+1})$$
$$\text{s.t.} \quad x_{it}^+ = g(x_{it}, a_{it})$$

    for each $x_{it} \in X_t$, $i = 1, \ldots, m_t$.
  - Now we have a *Lagrange data set*: $\{(x_{it}, v_{it}) \colon i = 1, \ldots, m_t\}$.

Chebyshev Approximation Algorithm in $\mathbb{R}^1$ (cont.)

- **Fitting step:** solve

$$\min_{c_j^t} \sum_{i=1}^{m_t} \left(\sum_{j=0}^{n} c_j^t \; T_j^*(x_{it}) - v_{it} \right)^2$$

which leads to

$$c_j^{t*} = \frac{\sum_{i=1}^{m_t} v_{it} \; T_j^*(x_{it})}{\sum_{i=1}^{m_t} T_j^*(x_{it})^2}$$

- Arrive at the approximation for $V_t(x)$:

$$\hat{V}_t(x; c^t) = \sum_{j=0}^{n} c_j^{t*} \; T_j^*(x)$$

Shape-Preserving Approximation Methods

- In some cases, it is important to construct approximations that preserve shape as well as are accurate.
- One problem for Chebyshev interpolation is the absence of shape-preservation in the algorithm.
- Some basic methods for preserving shape:
  1. **Discretization of the State Space**
  2. **Piecewise-Linear Interpolation:** preserves positivity, monotonicity, and concavity. However, it is not differentiable.
  3. **Shape-preserving quadratic spline interpolation:** Schumaker (1983) algorithm produces a smooth function which both interpolates data and preserves some shape.
  4. **Shape-preserving Chebyshev polynomial interpolation**

Schumaker Shape-Preserving Interpolation

1. Take level v_i (and maybe slope) data at nodes x_i, $i = 1, \ldots, m$.
2. Add intermediate nodes $\xi_i \in [x_i, x_{i+1}]$.
3. Run quadratic spline with nodes at the x and ξ nodes which interpolate data and preserves shape.
4. Schumaker formulas tell one how to choose the ξ and spline coefficients.
5. **Reference:** *"On Shape Preserving Quadratic Spline Interpolation"*, Larry L. Schumaker, SIAM Journal on Numerical Analysis, 1983.
6. A revised version of Schumaker interpolation is given in Cai (2009).

Shape-Preserving Approximation Methods: comments

- Discretization and piecewise linear interpolation preserve monotonicity and concavity. However, they make the objective function in the maximization step nondifferentiable, which forces one to use slow methods in the optimization step.
- The Schumaker method is only a C^1 approximation of the value function, thereby slowing convergence of Newton-style optimization methods.
- The problem with ordinary methods —such as polynomials and splines combined with interpolation or regression— in the fitting step is that the result may be nonconcave or nonmonotone even if the data are consistent with monotonicity and concavity.

Shape-Preserving Chebyshev Polynomial Interpolation

- If theory tells us that the true value function is strictly increasing and concave, then add constraints to the fitting criterion that will impose shape restrictions.
- Create an optimization problem that modifies the Chebyshev coefficients so that concavity and monotonicity of the value function will be preserved.
- We use Chebyshev polynomials and least-squares approximation with shape constraints to guarantee shape preservation.
- **Reference:** *"Shape-preserving dynamic programming"*, Yongyang Cai and Kenneth L. Judd, Math Meth Oper Res (2013).

Shape-Preserving Chebyshev Interpolation

- We begin with the Lagrange data $\{(x_i, v_i) \colon i = 1 \ldots, m\}$ generated by the maximization step of the Algorithm.
- Next, choose some points $z_{i'}$, $i' = 1, \ldots, m'$, called **shape checking nodes**.
- Impose the requirement that $\hat{V}(x; c)$ satisfies the shape conditions at the shape checking nodes.
- Choose the parameters c to minimize approximation errors but also satisfy the shape conditions.

Shape-Preserving Chebyshev Interpolation

- Use an undetermined method where the number of unknown coefficients may be greater than the number of approximation nodes.
- This allows the curve to match the data while also satisfying shape constraints.
- Solve the underidentification problem by penalizing high-order terms.
- Summary: for each $t < T$
  1. Lagrange data: (x_i, v_i), $i = 1, \ldots, m$
  2. Number of unknown coefficients: $n + 1 \geq m$
  3. Function for fitting: $\sum_{j=0}^{n} c_j \, T_j^*(x)$
  4. Shape checking nodes: $z_{i'}$, $i' = 1, \ldots, m'$
  5. Solve a linear programming (LP) problem

LP model for Chebyshev shape-preserving approximation

$$\min_{c_j, c_j^+, c_j^-} \quad \sum_{j=0}^{m-1} (c_j^+ + c_j^-) + \sum_{j=m}^{n} (j+1-m)^2 (c_j^+ + c_j^-)$$

$$\text{s.t.} \quad \sum_{j=0}^{n} c_j \, T_j^*(x_i) = v_i, \quad i = 1, \ldots, m, \quad \text{m interpolation constraints}$$

$$\sum_{j=0}^{n} c_j \, T_j'^*(z_{i'}) > 0, \quad i' = 1, \ldots, m', \quad \text{m' monotonicity constraints}$$

$$\sum_{j=0}^{n} c_j \, T_j''^*(z_{i'}) < 0, \quad i' = 1, \ldots, m', \quad \text{m' concavity constraints}$$

$$c_j = c_j^+ - c_j^-, \quad j = 0, \ldots, n$$

$$c_j^+ \geq 0, \quad c_j^- \geq 0, \quad j = 1, \ldots, n$$

LP model for Chebyshev shape-preserving approximation

- Usually we need $m' > m$ so that the shape preservation on the shape checking nodes implies that shape is preserved everywhere.
- Increase the set of shape checking nodes if shape has not been preserved.
- Let $n + 1 \geq m$ so that both m interpolation equality constraints and $2m'$ shape-preserving constraints could hold in the model.

Application: discrete-time optimal growth

Find the optimal consumption function and the optimal labor supply function such that the total utility over the T-horizon time is maximal:

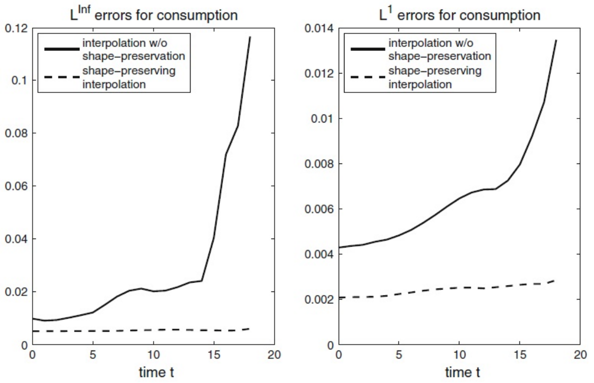
$$\begin{aligned} V_t(k) = \max_{c,l} \quad & u(c,l) + \beta V_{t+1}(k^+) \\ \text{s.t.} \quad & k^+ = F(k,l) - c \\ & \underline{k} \le k^+ \le \bar{k}. \quad c, l \ge \epsilon \\ & V_T(k) = \frac{u(f(k,1),1)}{1-\beta} \end{aligned}$$

where

$$\begin{aligned} u(c,l) &= \frac{(c/A)^{1-\gamma} - 1}{1-\gamma} - (1-\alpha)\frac{l^{1+\eta} - 1}{1+\eta} \\ A &= (1-\beta)/(\alpha\beta) \\ F(k,l) &= k + f(k,l) = k + Ak^{\alpha} l^{1-\alpha} \end{aligned}$$

Application: discrete-time optimal growth

- Cai & Judd (2013), *"Shape-preserving dynamic programming"*.
- $\alpha = 0.25$, $\beta = 0.99$, $\gamma = 8$, $\eta = 1$, $T = 20$, $k \in [0.1, 1.9]$, $\epsilon = 10^{-6}$.
- $m = 10$ *(Chebyshev approx. nodes)*, $n = 9$ *(degree of Chebyshev pol.)*, and $m' = 20$ *(equally-spaced shape checking nodes in $[-1, 1]$)*.



Multidimensional Approximation Methods

- Fix a period t, drop subscript t for clarity.
- Lagrange data: $\{(x_i, v_i)\}_{i=1}^{M} \subset \mathbb{R}^{p+q}$, where $x_i \in \mathbb{R}^p$ and $v_i \in \mathbb{R}^q$.
- Objective: find $f : \mathbb{R}^p \mapsto \mathbb{R}^q$ such that $v_i = f(x_i)$.
- Need to choose interpolation nodes carefully.
- Task: find combinations of interpolation nodes and basis functions to produce a nonsigular (well-conditioned) interpolation matrix.
- I will first quickly introduce interpolation in $p > 1$ dimensions and then shape-preserving methods.

Tensor Products Bases

- Lagrange data: $\{(x_i, v_i)\}_{i=1}^{M} \subset \mathbb{R}^{p+q}$, where $x_i \in \mathbb{R}^p$ and $v_i \in \mathbb{R}^q$.
- We can use *tensor products* of univariate functions to form bases of multivariate functions.
- Given a basis for functions of the single variable x_{ik}, $\Phi^k = \{\phi_j^k(x_{ik})\}_{j=0}^{\infty}$, the *p-fold tensor product* basis for functions of p variable $(x_1, x_2, \ldots, x_p)$ is

$$\Phi = \left\{ \prod_{k=1}^{p} \phi_{j_k}^k(x_{ik}) \,\middle|\, j_k = 0, 1, \ldots, \quad k = 1, \ldots, p \right\}$$

- One problem with tensor product bases is their size.

Tensor Products Bases

- Use finite subsets of the full tensor product basis.
- Take the first n elements of each univariate basis and construct the tensor product of these subbases.
- The p-dimensional tensor product will have n^p elements.
- This exponential growth in dimension makes it quite costly to use the full tensor product subbasis.
- Solution: bases that grow only polynomially as the dimension increases (*complete polynomials*).
- Now I'm going to present an example in two dimensions ($p = 2, q = 1$) using Chebyshev basis functions (without shape preservation).

Chebyshev Approximation Algorithm in $\mathbb{R}^2$ $(p = 2, q = 1)$

- **Objective:** given a function $V(x_1, x_2)$ defined on $[a, b]\, x\, [d, e]$, find its Chebyshev polynomial approximation $\hat{V}(x_1, x_2; c)$.
- Compute the $m \geq n + 1$ Chebyshev approximation nodes on $[-1, 1]$:

$$z_i = -\cos\left(\frac{2i-1}{2m}\pi\right), \quad i = 1, \ldots, m$$

- Adjust the nodes to the $[a, b]$ and $[d, e]$ intervals:

$$x_{1i} = (z_i + 1)\left(\frac{b-a}{2}\right) + a, \quad i = 1, \ldots, m$$

$$x_{2i} = (z_i + 1)\left(\frac{e-d}{2}\right) + d, \quad i = 1, \ldots, m$$

- Evaluate V at the approximation nodes:

$$v_{i,l} = V(x_{1i}, x_{2l}), \quad i = 1, \ldots, m, \quad l = 1, \ldots, m$$

Chebyshev Approximation Algorithm in $\mathbb{R}^2$ (cont.)

1. Compute Chebyshev coefficients, c_{hj}, $\forall h, j = 0, \dots, n$:

$$c^*_{hj} = \frac{\sum_{i=1}^m \sum_{l=1}^m v_{i,l} T_h(z_i) T_j(z_l)}{(\sum_{i=1}^m T_h(z_i)^2)(\sum_{l=1}^m T_j(z_l)^2)}$$

2. Arrive at the approximation for $V(x_1, x_2)$, $x_1 \in [a, b]$, $x_2 \in [d, e]$:

$$\hat{V}(x_1, x_2; c) = \sum_{h=0}^{n} \sum_{j=0}^{n} c^*_{hj} T^*_h(x_1) T^*_j(x_2)$$

Shape-preserving approximations in higher dimensions

- Shape issues are much harder in higher dimensions.
- There is no general method.
- Some basic methods for preserving shape in higher dimensions:
  1. **Discretization of the State Space**
  2. **Bilinear Interpolation:** interpolates the data linearly in both coordinate directions.
     - it preserves positivity and monotonicity, but not concavity.
     - it produces kinks in the value function and discontinuities in the policy function.
  3. **Shape-preserving Chebyshev interpolation**

Bilinear Interpolation

- Bilinear interpolation constructs an approximation that interpolates the data linearly in both coordinate directions.
- Suppose we have the values of $f(x, y)$ at $(x, y) = (\pm 1, \pm 1)$.
- Cardinal interpolation basis on $[-1, 1]^2$:

$$\begin{aligned} \phi_1(x, y) &= \frac{1}{4}(1-x)(1-y), \quad \phi_2(x, y) = \frac{1}{4}(1+x)(1-y) \\ \phi_3(x, y) &= \frac{1}{4}(1+x)(1+y), \quad \phi_4(x, y) = \frac{1}{4}(1-x)(1+y) \end{aligned}$$

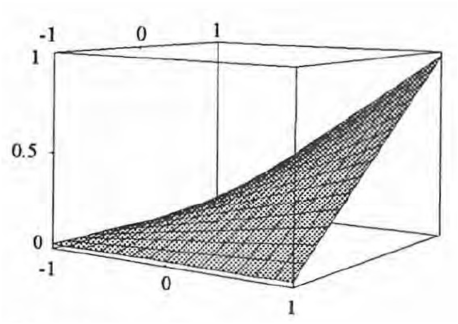
- Each of these functions is zero at all but one of the points $(\pm 1, \pm 1)$.

Bilinear Interpolation

- The approximation to f on the square $[-1,1]\,x\,[-1,1]$ is:

  $$\hat{f}(x,y) = f(-1,1)\phi_1(x,y) + f(1,-1)\phi_2(x,y) + f(1,1)\phi_3(x,y) + f(-1,1)\phi_4(x,y)$$

- Graph of the basis function ϕ_1:



- ϕ_1 is convex in the $(-1,-1)$ to $(1,1)$ direction, and concave in the $(-1,1)$ to $(1,-1)$ direction.

Shape-preserving Chebyshev interpolation

- Begin with Lagrange data $\{(x_i, v_i)\colon x_i \in \mathbb{R}^2,\ v_i \in \mathbb{R},\ i = 1, \dots, M\}$ generated by the maximization step of the Algorithm.
- Choose some points $z_{i'} \in \mathbb{R}^2$, $i' = 1, \dots, M'$, called **shape checking nodes**.
- Choose functional form $\hat{V}(x; c) = \sum_{h=0}^{n} \sum_{j=0}^{n} c_{hj}\, T_h^*(x_1)\, T_j^*(x_2)$.
- Add shape-preserving constraints: restrictions on directional derivatives.
- There will be many constraints, but these will be linear constraints (i.e., linear in the coefficients c).
- Number of unknown coefficients: $(n+1)^2 \geq M$

Shape-Preserving Chebyshev approximation in $\mathbb{R}^2$

$$\min_{c_{hj}, c_{hj}^+, c_{hj}^-} \quad \sum_{h=0}^{M-1}\sum_{j=0}^{M-1}(c_{hj}^+ + c_{hj}^-) + \sum_{h=M}^{n}\sum_{j=M}^{n}(h+j+1-2M)^2(c_{hj}^+ + c_{hj}^-), \quad \text{s.t.}$$

$$\sum_{h=0}^{n}\sum_{j=0}^{n} c_{hj}\, T_h^*(x_{1i})\, T_j^*(x_{2l}) = v_{il}, \quad i, l = 1, \ldots, m,$$ M=m^2 interpolation constraints

$$\sum_{h=0}^{n}\sum_{j=0}^{n} c_{hj}\, T_h'^*(x_{1i'})\, T_j^*(x_{2l'}) > 0, \quad i', l' = 1, \ldots, m',$$ M'=m'^2 monotonicity constraints

$$\sum_{h=0}^{n}\sum_{j=0}^{n} c_{hj}\, T_h^*(x_{1i'})\, T_j'^*(x_{2l'}) > 0, \quad i', l' = 1, \ldots, m',$$ M'=m'^2 monotonicity constraints

$$\sum_{h=0}^{n}\sum_{j=0}^{n} c_{hj}\, T_h''^*(x_{1i'})\, T_j^*(x_{2l'}) < 0, \quad i', l' = 1, \ldots, m',$$ M'=m'^2 concavity constraints

$$\sum_{h=0}^{n}\sum_{j=0}^{n} c_{hj}\, T_h^*(x_{1i'})\, T_j''^*(x_{2l'}) < 0, \quad i', l' = 1, \ldots, m',$$ M'=m'^2 concavity constraints

$+$ cross derivatives, $\quad c_{hj} = c_{hj}^+ - c_{hj}^-, \quad h, j = 0, \ldots, n \quad \ldots$

Summary

- Shape-preserving methods in one dimension:
  1. Discretization of the state space.
  2. Linear-quadratic models.
  3. Linear interpolation.
  4. Schumaker quadratic spline interpolation algorithm.
  5. Chebyshev orthogonal polynomials with shape checking constraints.

- Shape-preserving methods in higher dimensions:
  1. Discretization of the state space.
  2. Bilinear (multilinear) interpolation (only positivity and monotonicity).
  3. Chebyshev approximation in $\mathbb{R}^2$ with shape checking constraints.
  4. Tensor product approximation with only complete polynomials with shape checking constraints.