Lecture 2: Function Interpolation

Fatih Guvenen
University of Minnesota

November 2023

Interpolation

Introduction

- A value function with continuous state variables—e.g., $V(k, z)$—is an infinite-dimensional object.

- How do we represent it on a computer? How do we solve for it?

Introduction

- A value function with continuous state variables—e.g., $V(k, z)$—is an infinite-dimensional object.

- How do we represent it on a computer? How do we solve for it?

- One idea: Discretize k and z very finely and save values of V at all grid points.

- Problem: Sacrifice accuracy (if grid is coarse) or run into feasibility problems (if it's too fine).

Introduction

- A value function with continuous state variables—e.g., $V(k, z)$—is an infinite-dimensional object.
- How do we represent it on a computer? How do we solve for it?
- One idea: Discretize k and z very finely and save values of V at all grid points.
- Problem: Sacrifice accuracy (if grid is coarse) or run into feasibility problems (if it's too fine).
- Example: Median wealth < \$10,000. Mean of top 0.01% group: ~\$250M. How many grid points to take?
- Limits number of continuous state variables you can use.

Introduction

- A value function with continuous state variables—e.g., $V(k, z)$—is an infinite-dimensional object.
- How do we represent it on a computer? How do we solve for it?
- One idea: Discretize k and z very finely and save values of V at all grid points.
- Problem: Sacrifice accuracy (if grid is coarse) or run into feasibility problems (if it's too fine).
- Example: Median wealth < \$10,000. Mean of top 0.01% group: ~\$250M. How many grid points to take?
- Limits number of continuous state variables you can use.
- Better idea: Define $V(k, z) := V(k_i, z_j)$ for $i = 1, 2, ..., I$ and $j = 1, 2, ..., J$ + an interpolation method for all off-grid points.

First: Function Approximation vs. Interpolation

- Suppose you are given a grid $(x_1, x_{2,} ..., x_n)$ and the function values $(y_1, y_2, ..., y_n)$ at corresponding points generated by function $f(x)$.

First: Function Approximation vs. Interpolation

- Suppose you are given a grid $(x_1, x_{2,} ..., x_n)$ and the function values $(y_1, y_2, ..., y_n)$ at corresponding points generated by function $f(x)$.
- **Q:** How to find values off the grid points that provide a "good approximation" to $f(x)$?

First: Function Approximation vs. Interpolation

- Suppose you are given a grid $(x_1, x_{2,}..., x_n)$ and the function values $(y_1, y_2, ..., y_n)$ at corresponding points generated by function $f(x)$.
- **Q:** How to find values off the grid points that provide a "good **approximation**" to $f(x)$?
- A good approximation is often taken to mean to minimize $\left\|f(x) - \widehat{f}(x)\right\|$ according to some norm (L^p, sup-, etc).

First: Function Approximation vs. Interpolation

- Suppose you are given a grid $(x_1, x_{2,}..., x_n)$ and the function values $(y_1, y_2, ..., y_n)$ at corresponding points generated by function $f(x)$.

- **Q:** How to find values off the grid points that provide a "good **approximation**" to $f(x)$?

- A good approximation is often taken to mean to minimize $\left\|f(x) - \widehat{f}(x)\right\|$ according to some norm (L^p, sup-, etc).

- In economics, often another important concern is to preserve the shape—i.e., concavity or convexity—of the approximated (e.g, utility or value) function

First: Function Approximation vs. Interpolation

- Suppose you are given a grid $(x_1, x_{2,} ..., x_n)$ and the function values $(y_1, y_2, ..., y_n)$ at corresponding points generated by function $f(x)$.
- **Q:** How to find values off the grid points that provide a "good **approximation**" to $f(x)$?
- A good approximation is often taken to mean to minimize $\left\|f(x) - \widehat{f}(x)\right\|$ according to some norm (L^p, sup-, etc).
- In economics, often another important concern is to preserve the shape—i.e., concavity or convexity—of the approximated (e.g, utility or value) function
- Interpolation: Further require that $\widehat{f}(x) = f(x_j) = y_j$ for all $j = 1, 2, ..., n$; e.g., the interpolant must coincide with actual function values at all grid points.

Polynomial Approximation

- **Weierstrass Approximation Theorem.** Suppose f is a continuous real-valued function defined on the real interval [a, b]. For a given $\varepsilon > 0$, there exists a polynomial of order n, $P_n(x)$, such that for all x in [a, b], we have $\|f(x) - P_n(x)\|_\infty < \varepsilon$. In the limit, $\lim_{n\to\infty} \|f(x) - P_n(x)\|_\infty = 0$.

Polynomial Approximation

- **Weierstrass Approximation Theorem.** Suppose f is a continuous real-valued function defined on the real interval [a, b]. For a given $\varepsilon > 0$, there exists a polynomial of order n, $P_n(x)$, such that for all x in [a, b], we have $\|f(x) - P_n(x)\|_\infty < \varepsilon$. In the limit, $\lim_{n\to\infty} \|f(x) - P_n(x)\|_\infty = 0$.

- **Runge Example.** Let $f(x) = 1/(1 + x^2)$ on $[-5, 5]$, and let $L_m f$ be the unique polynomial of order m that interpolates f at m equally-spaced points. Then:

$$\limsup_{m\to\infty} |f(x) - L_m f| = \begin{cases} 0 & \text{if } |x| < 3.633..., \\ \infty & \text{if } |x| > 3.633... \end{cases}$$

Polynomial Approximation

- **Weierstrass Approximation Theorem.** Suppose f is a continuous real-valued function defined on the real interval [a, b]. For a given $\varepsilon > 0$, there exists a polynomial of order n, $P_n(x)$, such that for all x in [a, b], we have $\|f(x) - P_n(x)\|_\infty < \varepsilon$. In the limit, $\lim_{n\to\infty} \|f(x) - P_n(x)\|_\infty = 0$.

- **Runge Example.** Let $f(x) = 1/(1 + x^2)$ on $[-5, 5]$, and let $L_m f$ be the unique polynomial of order m that interpolates f at m equally-spaced points. Then:

$$\limsup_{m\to\infty} |f(x) - L_m f| = \begin{cases} 0 & \text{if } |x| < 3.633..., \\ \infty & \text{if } |x| > 3.633... \end{cases}$$

- How can Runge example not contradict the Weierstrass Thm?

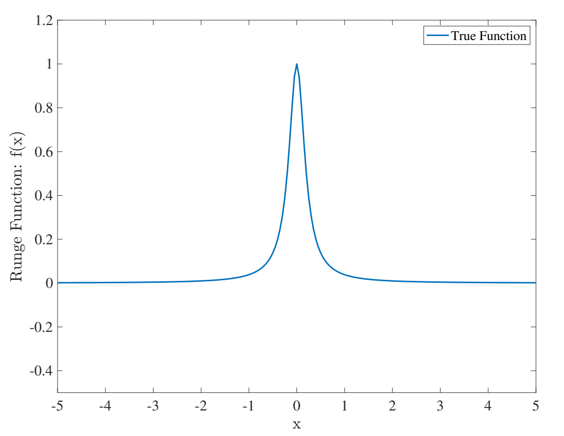
Polynomial Approximation

- **Weierstrass Approximation Theorem.** Suppose f is a continuous real-valued function defined on the real interval [a, b]. For a given $\varepsilon > 0$, there exists a polynomial of order n, $P_n(x)$, such that for all x in [a, b], we have $\|f(x) - P_n(x)\|_\infty < \varepsilon$. In the limit, $\lim_{n\to\infty} \|f(x) - P_n(x)\|_\infty = 0$.
- **Runge Example.** Let $f(x) = 1/(1 + x^2)$ on $[-5, 5]$, and let $L_m f$ be the unique polynomial of order m that interpolates f at m equally-spaced points. Then:

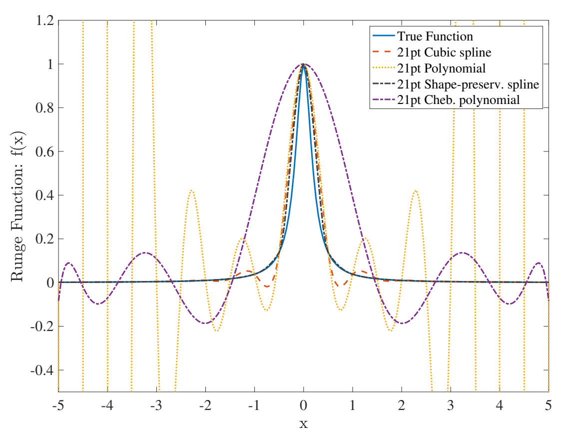
$$\limsup_{m\to\infty} |f(x) - L_m f| = \begin{cases} 0 & \text{if } |x| < 3.633..., \\ \infty & \text{if } |x| > 3.633... \end{cases}$$

- How can Runge example not contradict the Weierstrass Thm?
- Weierstrass does not provide a way of finding the right $P_n(x)$ and Runge example shows a naive approach can fail spectacularly. (Who said equally-spaced points, right?)

Runge Example



Runge Example



- We will learn more about the different interpolation schemes seen in this example.

Spline Interpolation: Three Objectives

- As in the Runge example, higher-order polynomials are flexible but can easily display wild oscillations $\rightarrow$ not ideal for interpolation.
- Idea behind cubic splines: Easier to approximate functions over smaller intervals. So, rather than one global polynomial of high degree, use piecewise polynomials of lower degree.

Spline Interpolation: Three Objectives

- As in the Runge example, higher-order polynomials are flexible but can easily display wild oscillations $\to$ not ideal for interpolation.
- Idea behind cubic splines: Easier to approximate functions over smaller intervals. So, rather than one global polynomial of high degree, use piecewise polynomials of lower degree.
- **Cubic Splines:**

Spline Interpolation: Three Objectives

- As in the Runge example, higher-order polynomials are flexible but can easily display wild oscillations $\rightarrow$ not ideal for interpolation.
- Idea behind cubic splines: Easier to approximate functions over smaller intervals. So, rather than one global polynomial of high degree, use piecewise polynomials of lower degree.
- **Cubic Splines:**
  1. Match the function values at grid points $(y_1, y_2, ..., y_n)$ exactly.

Spline Interpolation: Three Objectives

- As in the Runge example, higher-order polynomials are flexible but can easily display wild oscillations $\rightarrow$ not ideal for interpolation.
- Idea behind cubic splines: Easier to approximate functions over smaller intervals. So, rather than one global polynomial of high degree, use piecewise polynomials of lower degree.
- **Cubic Splines:**
  1. Match the function values at grid points $(y_1, y_2, ..., y_n)$ exactly.
  2. Generate first derivatives that are continuous and differentiable for all $x \in [x_1, x_n]$.

Spline Interpolation: Three Objectives

- As in the Runge example, higher-order polynomials are flexible but can easily display wild oscillations $\rightarrow$ not ideal for interpolation.
- Idea behind cubic splines: Easier to approximate functions over smaller intervals. So, rather than one global polynomial of high degree, use piecewise polynomials of lower degree.
- **Cubic Splines:**
  1. Match the function values at grid points $(y_1, y_2, ..., y_n)$ exactly.
  2. Generate first derivatives that are continuous and differentiable for all $x \in [x_1, x_n]$.
  3. Generate second derivatives that are continuous for all $x \in [x_1, x_n]$

Splines: An Optimality Result

- Consider the following minimization problem:

$$\min_f RSS(f, \lambda) = \sum_i^N \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt,$$

where λ is a fixed positive *smoothing parameter*.

Splines: An Optimality Result

- Consider the following minimization problem:

$$\min_f RSS(f, \lambda) = \sum_i^N \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt,$$

where λ is a fixed positive *smoothing parameter*.

- $\lambda = 0$: Any interpolating function is a solution
- $\lambda = \infty$: linear least squares fit, since curvature has infinite cost.

Splines: An Optimality Result

- Consider the following minimization problem:

$$\min_f RSS(f, \lambda) = \sum_i^N \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt,$$

where λ is a fixed positive *smoothing parameter*.

- $\lambda = 0$: Any interpolating function is a solution
- $\lambda = \infty$: linear least squares fit, since curvature has infinite cost.

- **Question:** When $\lambda \in (0, \infty)$, is there a solution? is it unique? can we find it?

Splines: An Optimality Result

- Consider the following minimization problem:

$$\min_f RSS(f,\lambda) = \sum_i^N \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt,$$

where λ is a fixed positive *smoothing parameter*.

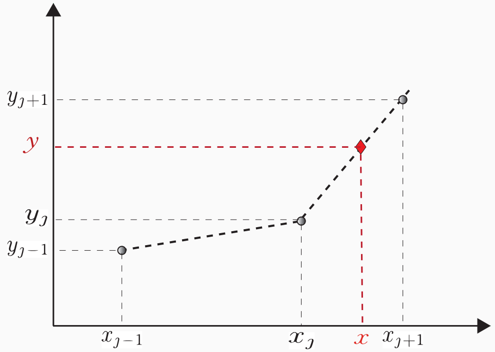
- $\lambda = 0$: Any interpolating function is a solution
- $\lambda = \infty$: linear least squares fit, since curvature has infinite cost.
- **Question:** When $\lambda \in (0,\infty)$, is there a solution? is it unique? can we find it?
- **Answer:** In the (infinite-dimensional) Sobolev space of functions (finite f''), there is a unique solution, which is the natural cubic spline interpolation with knots at $\{x_1, x_2, ..., x_N\}$!

Splines: Building from Ground Up

- Begin with the interval between two generic knots, x_i and x_{i+1}. If we were to construct a linear interpolant:

$$y = Ay_j + By_{j+1}$$

$$A(x) \equiv \frac{x_{j+1} - x}{x_{j+1} - x_j} \qquad B(x) \equiv 1 - A = \frac{x - x_j}{x_{j+1} - x_j} \tag{1}$$



Splines: Building from Ground Up

- Begin with the interval between two generic knots, x_i and x_{i+1}. If we were to construct a linear interpolant:

$$y = Ay_j + By_{j+1}$$
$$A(x) \equiv \frac{x_{j+1} - x}{x_{j+1} - x_j} \qquad B(x) \equiv 1 - A = \frac{x - x_j}{x_{j+1} - x_j} \tag{1}$$

- Although linear interpolation is sometimes useful, it has important shortcomings:
  - First derivative changes abruptly at knot points, i.e., interpolants have as many kinks as the knot points.
  - Second derivative does not exist at knot points.
  - $\rightarrow$ Can create many problems, e.g., with derivative-based algorithms, etc.

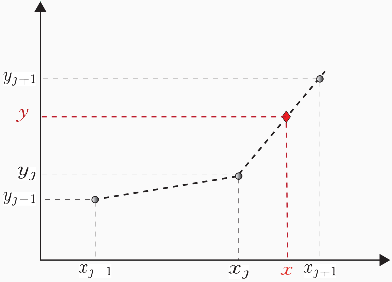
Splines: Building from Ground Up

- Begin with the interval between two generic knots, x_i and x_{i+1}. If we were to construct a linear interpolant:

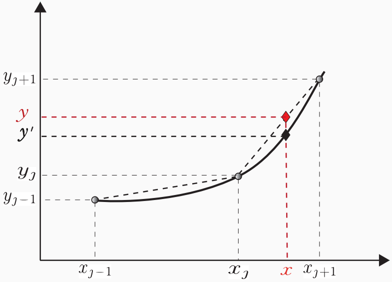
$$y = Ay_j + By_{j+1}$$
$$A(x) \equiv \frac{x_{j+1} - x}{x_{j+1} - x_j} \qquad B(x) \equiv 1 - A = \frac{x - x_j}{x_{j+1} - x_j} \tag{1}$$

- Although linear interpolation is sometimes useful, it has important shortcomings:
  - First derivative changes abruptly at knot points, i.e., interpolants have as many kinks as the knot points.
  - Second derivative does not exist at knot points.
  - $\rightarrow$ Can create many problems, e.g., with derivative-based algorithms, etc.
- Question: How to modify (1) to fix these problems?

Visualizing Linear vs Spline Interpolation



(a) Linear



(b) Bicubic

Figure 1: Linear and Bi-cubic Interpolation

- Compare the behavior of the two interpolants at x_j. Both are continuous, but spline only also has a derivative that is continuous and smooth (visually).

Splines: Building from Ground Up

- Generalize (1) using second derivative values at knot points:

$$y = A(x)y_j + B(x)y_{j+1} + C(x)y_j'' + D(x)y_{j+1}'' \tag{2}$$

where $C(x) = \frac{1}{6}(A^3(x) - A(x))(x_{j+1} - x_j)^2$ and
$D(x) = \frac{1}{6}(B^3(x) - B(x))(x_{j+1} - x_j)^2$

Splines: Building from Ground Up

- Generalize (1) using second derivative values at knot points:

$$y = A(x)y_j + B(x)y_{j+1} + C(x)y_j'' + D(x)y_{j+1}'' \tag{2}$$

  where $C(x) = \frac{1}{6}(A^3(x) - A(x))(x_{j+1} - x_j)^2$ and
  $D(x) = \frac{1}{6}(B^3(x) - B(x))(x_{j+1} - x_j)^2$

- Note that you only need to know A and B to calculate everything.

Splines: Building from Ground Up

- Generalize (1) using second derivative values at knot points:

$$y = A(x)y_j + B(x)y_{j+1} + C(x)y_j'' + D(x)y_{j+1}'' \tag{2}$$

where $C(x) = \frac{1}{6}(A^3(x) - A(x))(x_{j+1} - x_j)^2$ and
$D(x) = \frac{1}{6}(B^3(x) - B(x))(x_{j+1} - x_j)^2$

- Note that you only need to know A and B to calculate everything.
- Verify that $\frac{d^2y}{dx^2} = A(x)y_j'' + B(x)y_{j+1}''$

Splines: Building from Ground Up

- Generalize (1) using second derivative values at knot points:

$$y = A(x)y_j + B(x)y_{j+1} + C(x)y_j'' + D(x)y_{j+1}'' \tag{2}$$

where $C(x) = \frac{1}{6}(A^3(x) - A(x))(x_{j+1} - x_j)^2$ and
$D(x) = \frac{1}{6}(B^3(x) - B(x))(x_{j+1} - x_j)^2$

- Note that you only need to know A and B to calculate everything.
- Verify that $\frac{d^2y}{dx^2} = A(x)y_j'' + B(x)y_{j+1}''$
- Since $A(x_j) = 1$ and $B(x_{j+1}) = 1 - A(x_{j+1}) = 1$, the second derivative agrees with y''at end points.

Splines: Building from Ground Up

- Generalize (1) using second derivative values at knot points:

$$y = A(x)y_j + B(x)y_{j+1} + C(x)y_j'' + D(x)y_{j+1}'' \tag{2}$$

  where $C(x) = \frac{1}{6}(A^3(x) - A(x))(x_{j+1} - x_j)^2$ and
  $D(x) = \frac{1}{6}(B^3(x) - B(x))(x_{j+1} - x_j)^2$

- Note that you only need to know A and B to calculate everything.
- Verify that $\frac{d^2y}{dx^2} = A(x)y_j'' + B(x)y_{j+1}''$
- Since $A(x_j) = 1$ and $B(x_{j+1}) = 1 - A(x_{j+1}) = 1$, the second derivative agrees with y''at end points.
- But how to find y_j'' and y_{j+1}'' ?

Splines: Building from Ground Up

- Differentiate (2) to obtain expression for $\frac{dy}{dx}$ that involves y_j'' and y_{j+1}''.

Splines: Building from Ground Up

- Differentiate (2) to obtain expression for $\frac{dy}{dx}$ that involves y_j'' and y_{j+1}''.
- Then impose condition that $\frac{dy}{dx}$ calculated using (x_j, x_{j+1}) or (x_{j+1}, x_{j+2}) equal each other at x_{j+1}. We get:

$$\underbrace{\frac{x_j - x_{j-1}}{6}}_{c_{j-1}} y_{j-1}'' + \underbrace{\frac{x_{j+1} - x_{j-1}}{3}}_{d_{j-1}} y_j'' + \underbrace{\frac{x_{j+1} - x_j}{6}}_{c_j} y_{j+1}'' = \underbrace{\frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}}_{s_j - s_{j-1}} . \quad (3)$$

Splines: Building from Ground Up

- Differentiate (2) to obtain expression for $\frac{dy}{dx}$ that involves y_j'' and y_{j+1}''.
- Then impose condition that $\frac{dy}{dx}$ calculated using (x_j, x_{j+1}) or (x_{j+1}, x_{j+2}) equal each other at x_{j+1}. We get:

$$\underbrace{\frac{x_j - x_{j-1}}{6}}_{c_{j-1}} y_{j-1}'' + \underbrace{\frac{x_{j+1} - x_{j-1}}{3}}_{d_{j-1}} y_j'' + \underbrace{\frac{x_{j+1} - x_j}{6}}_{c_j} y_{j+1}'' = \underbrace{\frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}}_{s_j - s_{j-1}}. \quad (3)$$

- For interior knot points, $j = 2, ..., N-1$, we have an equation like this. But we have N unknowns (y_j'' for $j = 1, ..., N$).

Splines: Building from Ground Up

- Differentiate (2) to obtain expression for $\frac{dy}{dx}$ that involves y_j'' and y_{j+1}''.
- Then impose condition that $\frac{dy}{dx}$ calculated using (x_j, x_{j+1}) or (x_{j+1}, x_{j+2}) equal each other at x_{j+1}. We get:

$$\underbrace{\frac{x_j - x_{j-1}}{6}}_{c_{j-1}} y_{j-1}'' + \underbrace{\frac{x_{j+1} - x_{j-1}}{3}}_{d_{j-1}} y_j'' + \underbrace{\frac{x_{j+1} - x_j}{6}}_{c_j} y_{j+1}'' = \underbrace{\frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}}_{s_j - s_{j-1}}. \quad (3)$$

- For interior knot points, $j = 2, ..., N-1$, we have an equation like this. But we have N unknowns (y_j'' for $j = 1, ..., N$).
- Impose two boundary conditions. Some common choices:
  - Set y_1' and y_N' to specified values, or
  - Set y_1'' and y_N'' to zero (natural spline) (can help with extrapolation)

Splines: Building from Ground Up

- Differentiate (2) to obtain expression for $\frac{dy}{dx}$ that involves y_j'' and y_{j+1}''.
- Then impose condition that $\frac{dy}{dx}$ calculated using (x_j, x_{j+1}) or (x_{j+1}, x_{j+2}) equal each other at x_{j+1}. We get:

$$\underbrace{\frac{x_j - x_{j-1}}{6}}_{c_{j-1}} y_{j-1}'' + \underbrace{\frac{x_{j+1} - x_{j-1}}{3}}_{d_{j-1}} y_j'' + \underbrace{\frac{x_{j+1} - x_j}{6}}_{c_j} y_{j+1}'' = \underbrace{\frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}}_{s_j - s_{j-1}}. \quad (3)$$

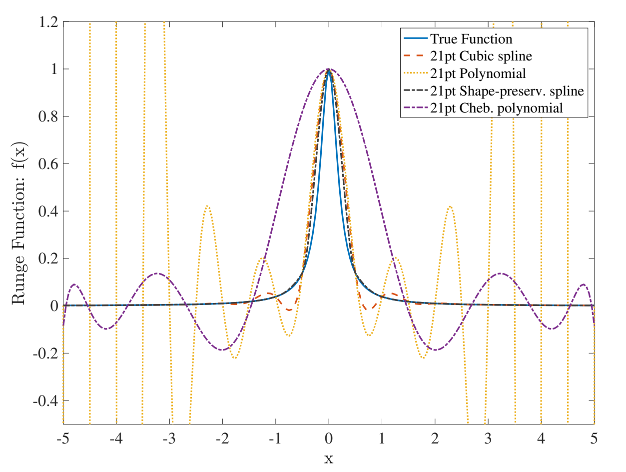
- For interior knot points, $j = 2, ..., N-1$, we have an equation like this. But we have N unknowns (y_j'' for $j = 1, ..., N$).
- Impose two boundary conditions. Some common choices:
  - Set y_1' and y_N' to specified values, or
  - Set y_1'' and y_N'' to zero (natural spline) (can help with extrapolation)
- Caution: no condition is imposed for y' or y'' to agree with f' and f'', since these are unknown. This can create problems as we will see.

A Tridiagonal System of Equations

First and last lines are dy/dx at end points:

$$\begin{bmatrix} 2c_1 & -c_1 & & & & & \\ c_1 & 2d_1 & c_2 & & & & \\ & & \ddots & & & & \\ & & c_{j-1} & 2d_{j-1} & c_j & & \\ & & & & \ddots & & \\ & & & & c_{n-2} & 2d_{n-2} & c_{n-1} \\ & & & & & -c_{n-1} & 2c_{n-1} \end{bmatrix} \cdot \begin{bmatrix} y_1'' \\ y_2'' \\ \vdots \\ y_j'' \\ \vdots \\ y_{n-1}'' \\ y_n'' \end{bmatrix} = \begin{bmatrix} s_1 - a_1^* \\ s_2 - s_1 \\ \vdots \\ s_j - s_{j-1} \\ \vdots \\ s_{n-1} - s_{n-2} \\ s_n - a_n^* \end{bmatrix}. \qquad (4)$$

Runge Example, Second Try

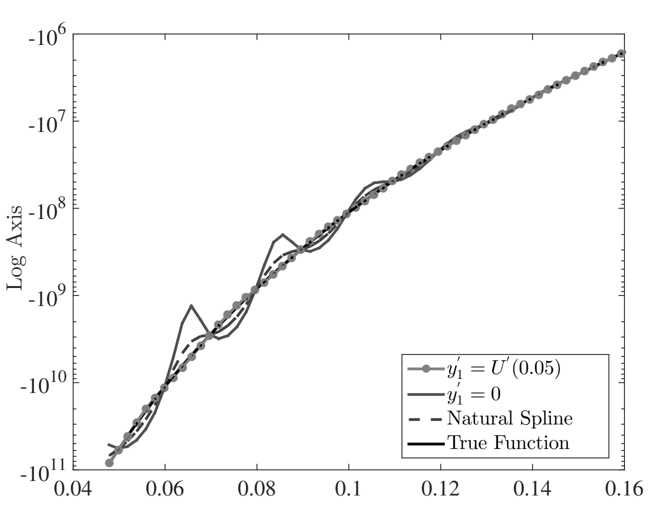


- Notice how well (Schumaker's) Shape preserving spline does. Cubic spline also does very well everywhere except the small ripples between –2 and 2.

Comparing Interpolation Methods for U(C)

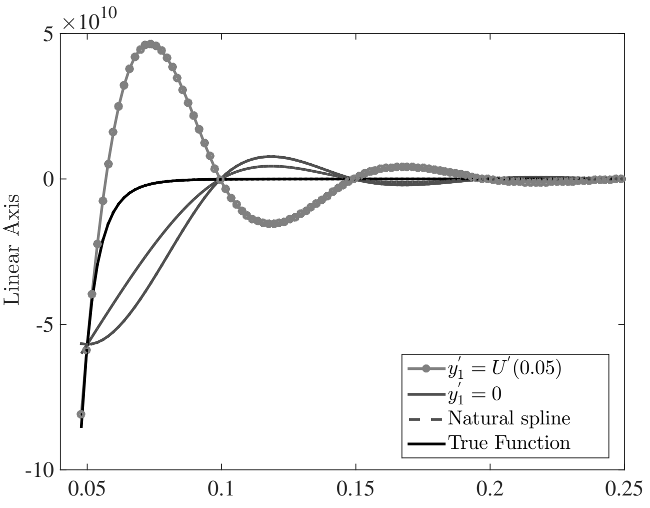
Comparing Boundary Conditions

Figure 2: $N = 500$ pts

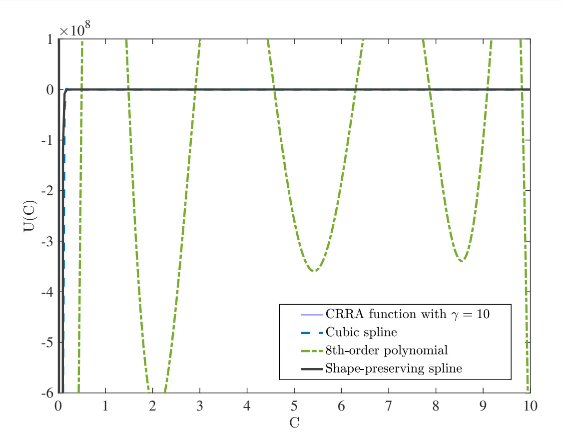


Comparing Boundary Conditions

Figure 3: $N = 100$ pts

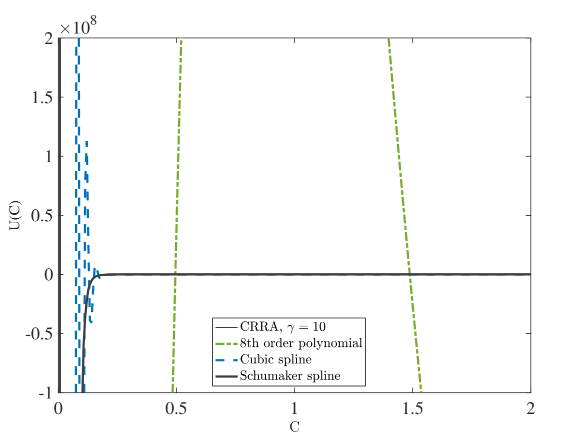


Interpolation: What Can Go Wrong



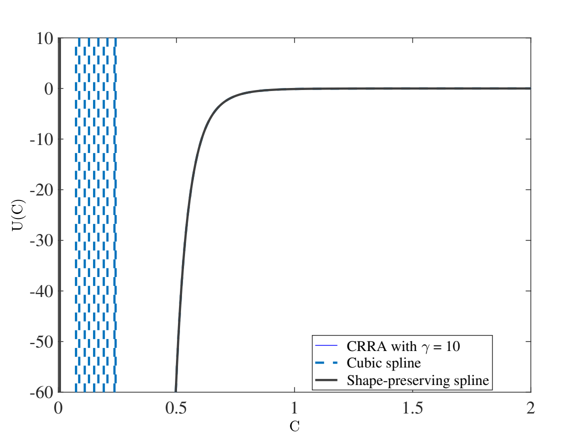
- Interpolate U(C) at 100 equally spaced points from 0.05 to 10.
- Notice the enormous fluctuations of polynomial interpolation.
- Shape-preserving & cubic spline seem to fit well. **Or do they?**

Interpolation: Change x-axis scale, Zoom in



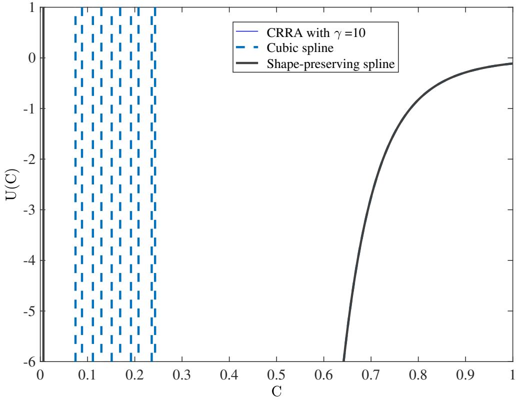
- Notice the wild fluctuations in cubic spline at low end!
- Shape-preserving also fluctuates but $C < 0.05$, so that's fair.

Interpolation: Zoom into the y-axis



- Zoom in more, you see even more fluctuations (because your screen can now actually plot them!)

Interpolation: Zoom further

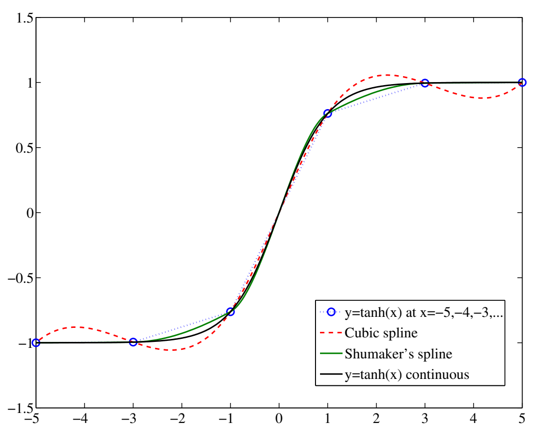


- And even more fluctuations!

Taking Stock

- So, what's the point of all of this?
- Oftentimes, a beginner will hear about splines or another interpolation method, will give it a try, and get wild oscillations as you see here.
- Sometimes, they won't even plot the functions, all they will know is that their algorithm keeps crashing, and they will give up and settle for linear interpolation or something simple like that.
- The truth is, most utility functions are very, very difficult to interpolate at the low end, because they have a "pole" at zero. That is, they diverge to (minus) infinity.
- Despite this, they can be interpolated extremely accurately but we need to learn a few important tricks.

Digression: Standard Spline vs Shape-Preserving



- Shape preserving splines can be very useful to ensure concavity or convexity.

First Trick: Spacing of Grid Points is Crucial

- **One heuristic**: put more grid points where f has more curvature.

First Trick: Spacing of Grid Points is Crucial

- **One heuristic**: put more grid points where f has more curvature.
- **Another one:** put more points near the parts of the function that are more relevant.

First Trick: Spacing of Grid Points is Crucial

- **One heuristic**: put more grid points where f has more curvature.
- **Another one:** put more points near the parts of the function that are more relevant.
- In incomp. mkts models, $V''(\omega)$ is largest for very low ω. So should put more points there.

First Trick: Spacing of Grid Points is Crucial

- **One heuristic**: put more grid points where f has more curvature.
- **Another one:** put more points near the parts of the function that are more relevant.
- In incomp. mkts models, $V''(\omega)$ is largest for very low ω. So should put more points there.
- Is this true if there are not many individuals near the constraint? (Answer: Typically, Yes. But why?)

First Trick: Spacing of Grid Points is Crucial

- **One heuristic**: put more grid points where f has more curvature.
- **Another one:** put more points near the parts of the function that are more relevant.
- In incomp. mkts models, $V''(\omega)$ is largest for very low ω. So should put more points there.
- Is this true if there are not many individuals near the constraint? (Answer: Typically, Yes. But why?)
- In some DP problems, with max operator on the RHS, the value function may have a kink or significant curvature somewhere in the middle of the state space.
  - Linear interpolation maybe your best choice.

Expanding Grid

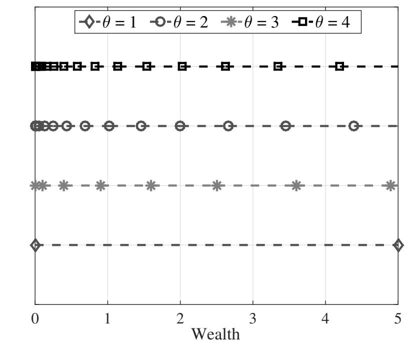
Algorithmus 1 : CREATING A POLYNOMIALLY-EXPANDING GRID

Step 1. First, create an equally-spaced [0,1] grid:
$\{z_j : z_j = \frac{j-1}{N-1},\ j = 1, ..., N\}$.

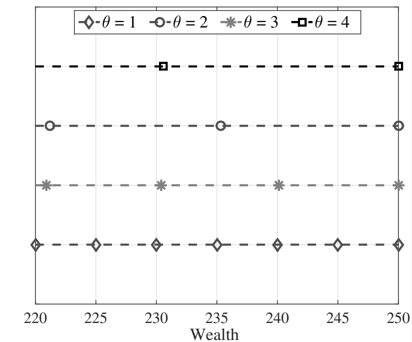
Step 2. Shift and expand the grid: $x = \{x_j : x_j = a + (b-a)z_j^{\theta}\}$, where
$\theta > 1$ is the expansion factor.

Figure 4: Grid Point Locations: 51-Point Expanding Grid From 0 to 250

(a) Low End of Grid

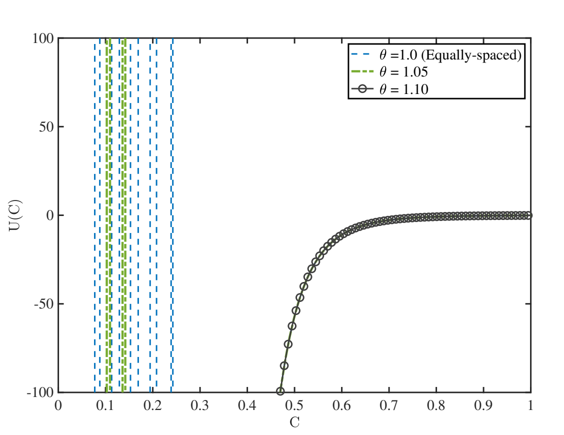


(b) High End of Grid



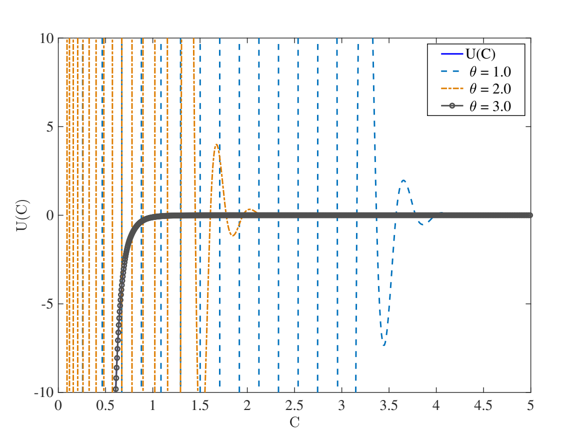
Note: The number of grid points between 0 and 4.99 is 1, 8, 14, and 19 when θ is equal to 1, 2, 3, and 4, respectively.

Spline w/ Expanding Grid (1000 pts)



- When number of grid points is large (1000), even a very small expansion (exponent of $\theta = 1.1$) can deliver perfect spline interpolation.

Spline w/ Expanding Grid (100 pts)



- But the real power of expanding grid is that we can take a larger θ and reduce grid points from 1000 to 100 and still get a perfect interpolation!!

A Trick to Reduce Curvature of $V(w)$

A Trick to Reduce the Curvature of V(w)

- Samuelson (1969) showed that in a standard portfolio choice problem with CRRA preferences and a linear budget set, the value function inherits the curvature of U:

$$U(c_0, c_1, \ldots) = \sum_{t=1}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} \Rightarrow V(\omega, A) = \phi(A) \times \omega^{1-\gamma}$$

A Trick to Reduce the Curvature of V(w)

- Samuelson (1969) showed that in a standard portfolio choice problem with CRRA preferences and a linear budget set, the value function inherits the curvature of U:

$$U(c_0, c_1, \ldots) = \sum_{t=1}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} \Rightarrow V(\omega, A) = \phi(A) \times \omega^{1-\gamma}$$

- The same result holds approximately true in a variety of different problems.

A Trick to Reduce the Curvature of V(w)

- Samuelson (1969) showed that in a standard portfolio choice problem with CRRA preferences and a linear budget set, the value function inherits the curvature of U:

$$U(c_0, c_1, ...) = \sum_{t=1}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} \Rightarrow V(\omega, A) = \phi(A) \times \omega^{1-\gamma}$$

- The same result holds approximately true in a variety of different problems.
- With incomplete markets, $V(w)$ will typically have even more curvature than $U(c)$ especially at low wealth levels.

A Trick to Reduce the Curvature of V(w)

- Samuelson (1969) showed that in a standard portfolio choice problem with CRRA preferences and a linear budget set, the value function inherits the curvature of U:

$$U(c_0, c_1, ...) = \sum_{t=1}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} \Rightarrow V(\omega, A) = \phi(A) \times \omega^{1-\gamma}$$

- The same result holds approximately true in a variety of different problems.
- With incomplete markets, $V(w)$ will typically have even more curvature than $U(c)$ especially at low wealth levels.
- As we have seen so far, this high curvature creates a lot of headache when you try to interpolate the value function.

A Trick to Reduce the Curvature of V(w)

- Samuelson (1969) showed that in a standard portfolio choice problem with CRRA preferences and a linear budget set, the value function inherits the curvature of U:

$$U(c_0, c_1, ...) = \sum_{t=1}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} \Rightarrow V(\omega, A) = \phi(A) \times \omega^{1-\gamma}$$

- The same result holds approximately true in a variety of different problems.
- With incomplete markets, $V(w)$ will typically have even more curvature than $U(c)$ especially at low wealth levels.
- As we have seen so far, this high curvature creates a lot of headache when you try to interpolate the value function.
- Fortunately, there is a way out!

A Trick to Reduce the Curvature of V(w)

- There is an alternative formulation of CRRA preferences:

$$U(c_0, c_1, ...) = \left(\sum_{t=1}^{\infty} \beta^t c_t^{(1-\gamma)}\right)^{1/(1-\gamma)}$$

A Trick to Reduce the Curvature of V(w)

- There is an alternative formulation of CRRA preferences:

$$U(c_0, c_1, \ldots) = \left(\sum_{t=1}^{\infty} \beta^t c_t^{(1-\gamma)} \right)^{1/(1-\gamma)}$$

- This as a special case of Epstein-Zin (1989, E'trica) utility and represents the same preferences as CRRA utility with $RRA = \gamma$.

A Trick to Reduce the Curvature of V(w)

- There is an alternative formulation of CRRA preferences:

$$U(c_0, c_1, \ldots) = \left(\sum_{t=1}^{\infty} \beta^t c_t^{(1-\gamma)}\right)^{1/(1-\gamma)}$$

- This as a special case of Epstein-Zin (1989, E'trica) utility and represents the same preferences as CRRA utility with $RRA = \gamma$.
- Now the value function is linear: $V(\omega, A) = \phi(A) \times \omega$

A Trick to Reduce the Curvature of V(w)

- There is an alternative formulation of CRRA preferences:

$$U(c_0, c_1, \ldots) = \left(\sum_{t=1}^{\infty} \beta^t c_t^{(1-\gamma)} \right)^{1/(1-\gamma)}$$

- This as a special case of Epstein-Zin (1989, E'trica) utility and represents the same preferences as CRRA utility with $RRA = \gamma$.
- Now the value function is linear: $V(\omega, A) = \phi(A) \times \omega$
- Although incomplete markets introduces some curvature, this value function is much easier to interpolate than the one above.

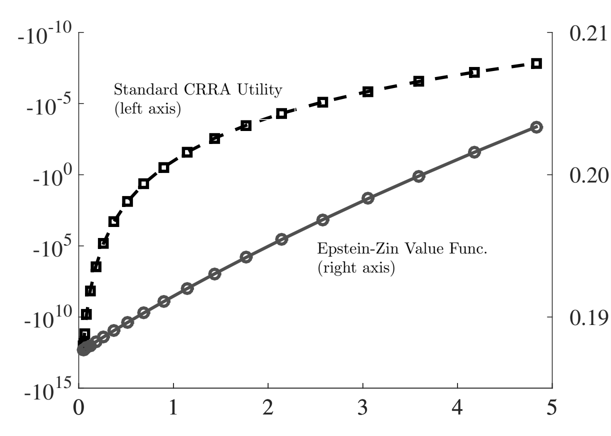
A Trick to Reduce the Curvature of V(w)

- There is an alternative formulation of CRRA preferences:

$$U(c_0, c_1, \ldots) = \left(\sum_{t=1}^{\infty} \beta^t c_t^{(1-\gamma)} \right)^{1/(1-\gamma)}$$

- This as a special case of Epstein-Zin (1989, E'trica) utility and represents the same preferences as CRRA utility with $RRA = \gamma$.
- Now the value function is linear: $V(\omega, A) = \phi(A) \times \omega$
- Although incomplete markets introduces some curvature, this value function is much easier to interpolate than the one above.
- I once solved a GE model for asset pricing and a risk aversion of 6 using only 30 points in the wealth grid and linear interpolation!

Which Function Would You Rather Interpolate?



- Notice the enormous difference in the range of variation on the left scale (from -10^{15} to -10^{-10}!) and right (from 0.19 to 0.21)!

Final Thoughts

- Use the CES (or Epstein-Zin) formulation described here in the wealth direction whenever it is feasible to do so. You will thank me later.
  - It will remove most of the headaches associated with the wild fluctuations discussed above.

Final Thoughts

- Use the CES (or Epstein-Zin) formulation described here in the wealth direction whenever it is feasible to do so. You will thank me later.
  - It will remove most of the headaches associated with the wild fluctuations discussed above.
- In addition, use an expanding grid because with incomplete markets there will still be a little curvature at the bottom end. I often use $\theta \approx 3$.
  - Of course, there will be times when you cannot use the CES trick, so expanding grid is important.

Final Thoughts

- Use the CES (or Epstein-Zin) formulation described here in the wealth direction whenever it is feasible to do so. You will thank me later.
  - It will remove most of the headaches associated with the wild fluctuations discussed above.
- In addition, use an expanding grid because with incomplete markets there will still be a little curvature at the bottom end. I often use $\theta \approx 3$.
  - Of course, there will be times when you cannot use the CES trick, so expanding grid is important.
- Choose the lowest point in the c grid of your interpolation carefully. The lower you go, the more curvature you have to deal with.

Final Thoughts

- For any model you solve, you **must** eventually re-solve it on a much finer grid and confirm that your main results are not changing (much if at all).

Final Thoughts

- For any model you solve, you must eventually re-solve it on a much finer grid and confirm that your main results are not changing (much if at all).
- This is the only realistic way to check if approximation errors coming from interpolations are important.

Final Thoughts

- For any model you solve, you must eventually re-solve it on a much finer grid and confirm that your main results are not changing (much if at all).
- This is the only realistic way to check if approximation errors coming from interpolations are important.
- You will be surprised to find that some bad-looking interpolations actually yield the same results as much more accurate (and more costly to compute) interpolations.

Final Thoughts

- For any model you solve, you must eventually re-solve it on a much finer grid and confirm that your main results are not changing (much if at all).
- This is the only realistic way to check if approximation errors coming from interpolations are important.
- You will be surprised to find that some bad-looking interpolations actually yield the same results as much more accurate (and more costly to compute) interpolations.
- And vice versa..

Final Thoughts

- For any model you solve, you must eventually re-solve it on a much finer grid and confirm that your main results are not changing (much if at all).
- This is the only realistic way to check if approximation errors coming from interpolations are important.
- You will be surprised to find that some bad-looking interpolations actually yield the same results as much more accurate (and more costly to compute) interpolations.
- And vice versa..
- Some problems are especially sensitive to any kind of approximation errors. We will see examples.