# Numerical Methods in Economics and Finance
# Lecture IV&V: Function Approximation and Application

Xi Wang
*Department of Finance, School of Economics*
*Peking University*

May 4, 2019

# OVERVIEW OF HEREAFTER LECTURES

- To be Covered
  - Function Approximation
  - Equation Solving
  - Optimization
  - Numerical Differentiate and Integration
  - Continuous Time
  - Projection Method
- No-Plan topics will be covered, if time allows.
  - Dynamic Programming
  - Approximate DP
  - Projection Method

# A FUNNY STORY OF FEYMAN

- Purterbation; I want to know $\int_0^1 \frac{x^5-1}{\log(x)}dx$, Emm do you know how to integral it?
- Set $f(\alpha) = \int_0^1 \frac{x^\alpha-1}{\log(x)}dx$, I will evaluate $f(5)$

$$f'(\alpha) = \int_0^1 \frac{d}{d\alpha}\left(\frac{x^\alpha - 1}{\log(x)}\right)dx$$

$$= \int_0^1 \frac{x^\alpha \log(x)}{\log(x)}dx = \int_0^1 x^\alpha dx = \left.\frac{x^{\alpha+1}}{\alpha+1}\right|_0^1 = \frac{1}{\alpha+1}$$

- Surely then $f(\alpha) = \int \frac{1}{\alpha+1}d\alpha = \log(\alpha+1) + c$
  - Need to discover an appropriate way to introduce a parameter into the integral.
  - The result was, when guys at MIT or Princeton had trouble doing a certain integral, it was because they couldn't do it with the standard methods they had learned in school. If it was contour integration, they would have found it; if it was a simple series expansion, they would have found it. Then I come along and try differentiating under the integral sign, and often it worked. So I got a great reputation for doing integrals.
    —Surely You're Joking Mr. Feynman, p.86–87

# TAUCHEN (1986)

- Still remember

$$z = (1-\rho)\mu + \rho z_{-1} + \sigma\varepsilon, \quad \varepsilon \sim N(0,1) \tag{1}$$

- We have policy function depends on $k, z, \sigma$. Easy to discretize $k$, Easy to get start at least; How about $z$?
- Tauchen (1986)
  - Get n points set, say $\mathcal{Z} = \{z_1, \ldots, z_n\}$, linearly spaced points on $[\mu - \kappa\frac{\sigma}{\sqrt{1-\rho^2}},\ \mu + \kappa\frac{\sigma}{\sqrt{1-\rho^2}}]$
  - Get midpoint $m_{i,i+1} = \frac{z_i + z_{i+1}}{2}$
  - Determines the transition probabilities Matrix $P_{j|i}$ as

$$F\left(z_i|z_j\right) = \begin{cases} \Phi\left(m_{1,2}; a_j, \sigma^2\right) & \text{if } i = 1 \\ \Phi\left(m_{i,i+1}; a_j, \sigma^2\right) - \Phi\left(m_{i-1,i}; a_j, \sigma^2\right) & \text{if } i \in [2, n-1] \\ 1 - \Phi\left(m_{n-1,n}; a_j, \sigma^2\right) & \text{if } i = n \end{cases} \tag{2}$$

where $a_j = (1-\rho)\mu + \rho z_j$

# TAUCHEN (1986) COMMENT

- Widely used and understood
- Hand code the $\mathcal{Z}$
- Performs fairly well under high autocorrelation
- How many points to choose $n = 1 + 2\kappa$, $\kappa = 4, 3$
- Rouwenhorst Method, Kopecky and Suen (2010)
- Dis $k$? $[.7k^*, 1.3k^*]$

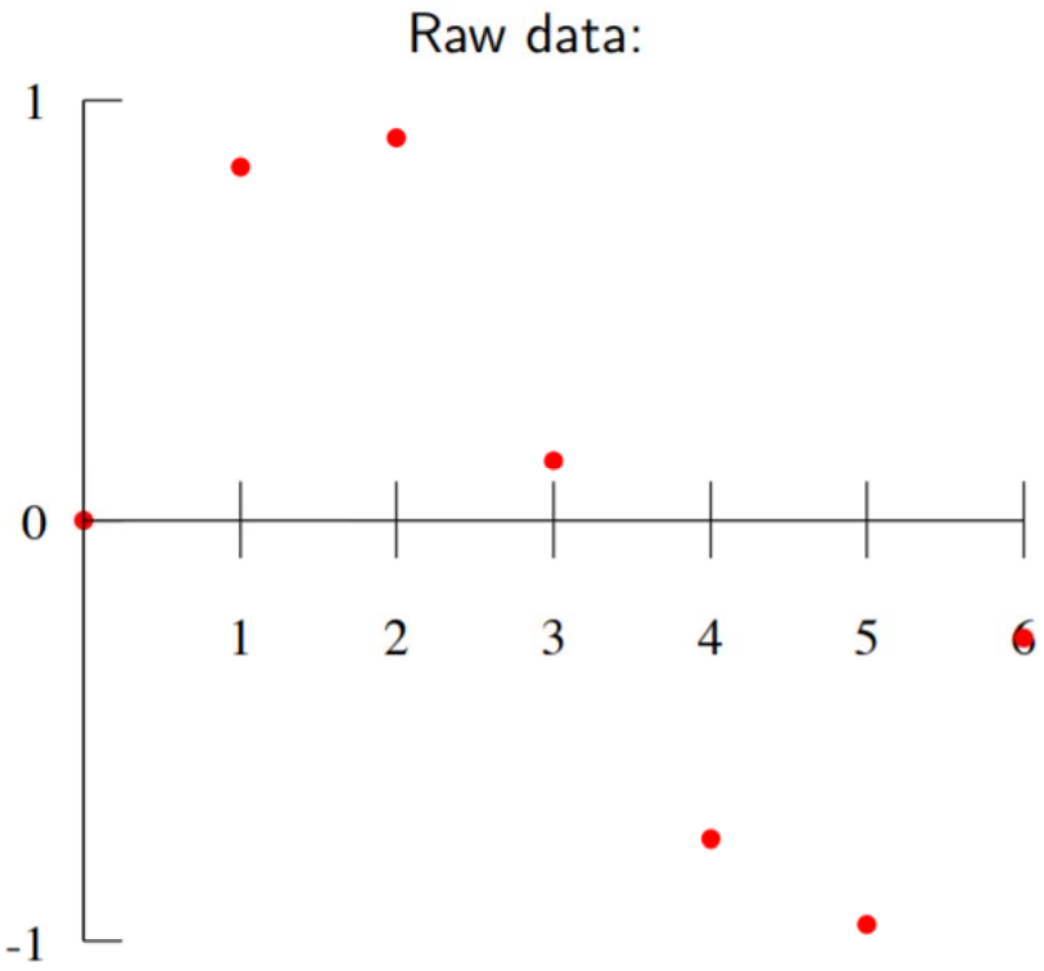
# INTERPOLATION

- Take some data $\{x_i, y_i\}$ and get values of $y$ for independent variable $x$ not in the range of $\{x_i, y_i\}$. Did I remind u anything?
  - All supervised learning method belongs to interpolation
- What we want is a prediction function $\hat{f}$

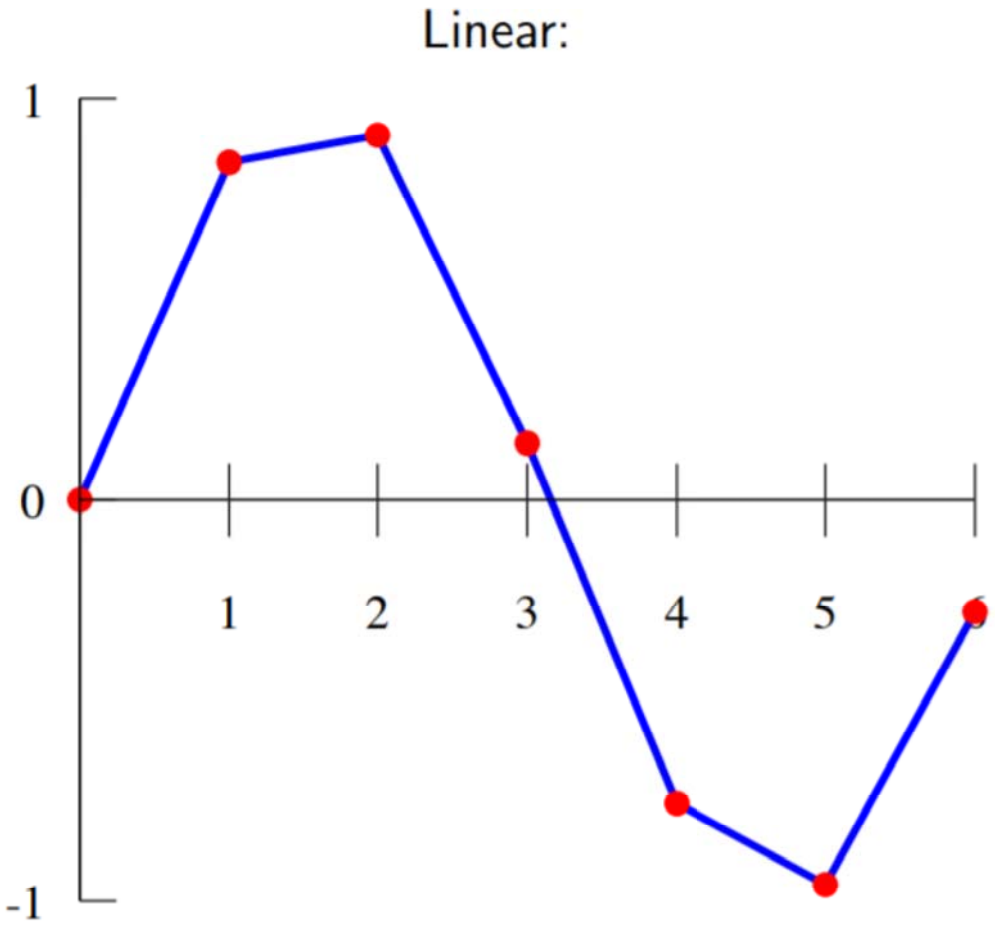
# INTERPOLATION

- Linear interpolation
- Polynomial interpolation (including Chebyshev collocation)
- Spline interpolation
- KNN
- Neural Net

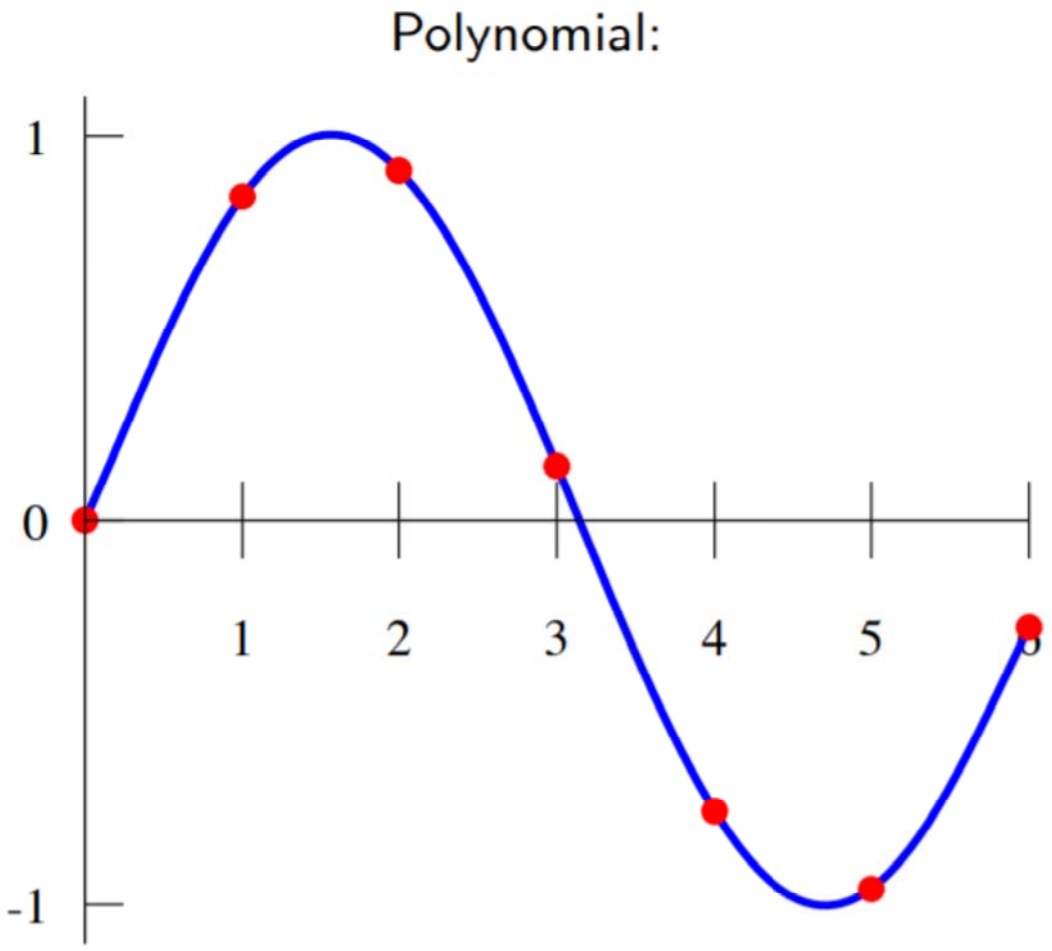
# INTERPOLATION, WIKI



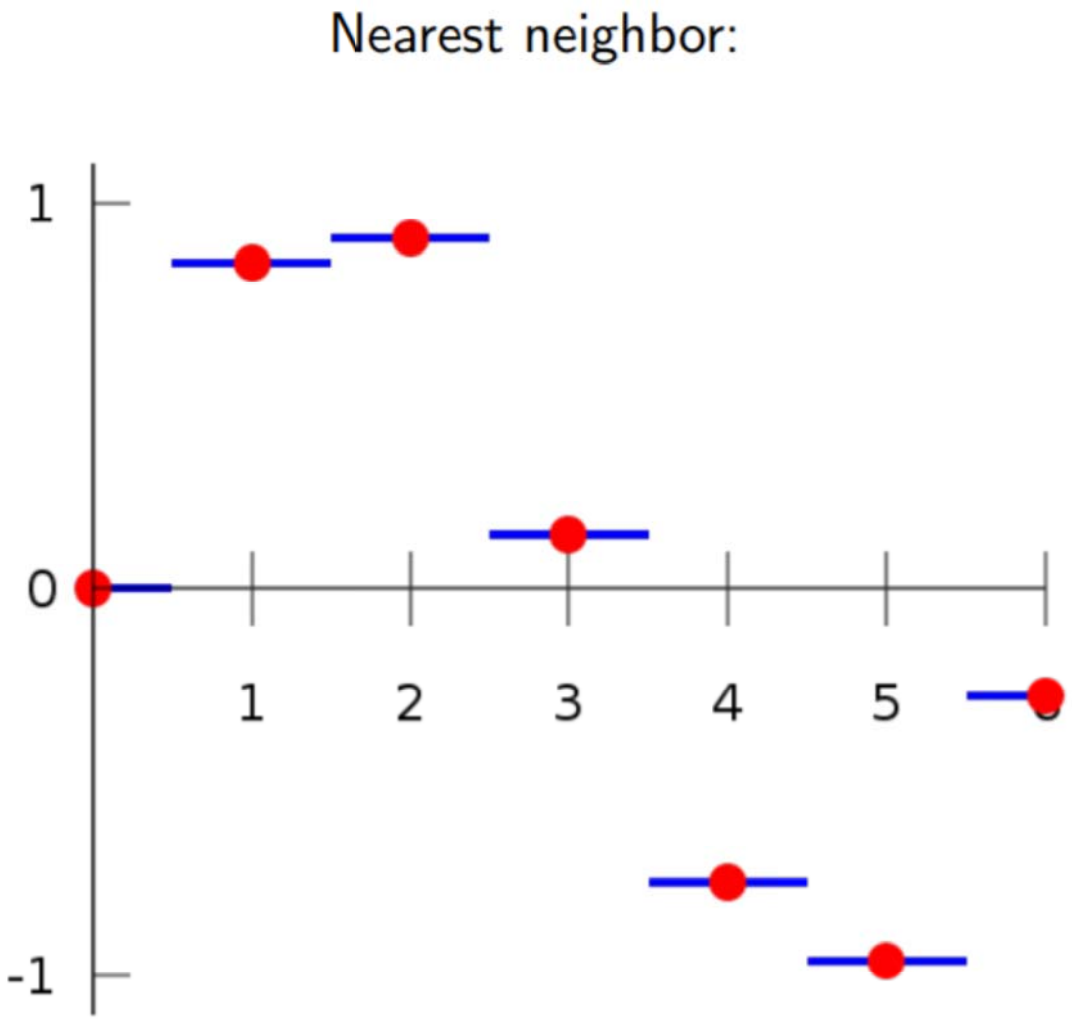
# INTERPOLATION, WIKI



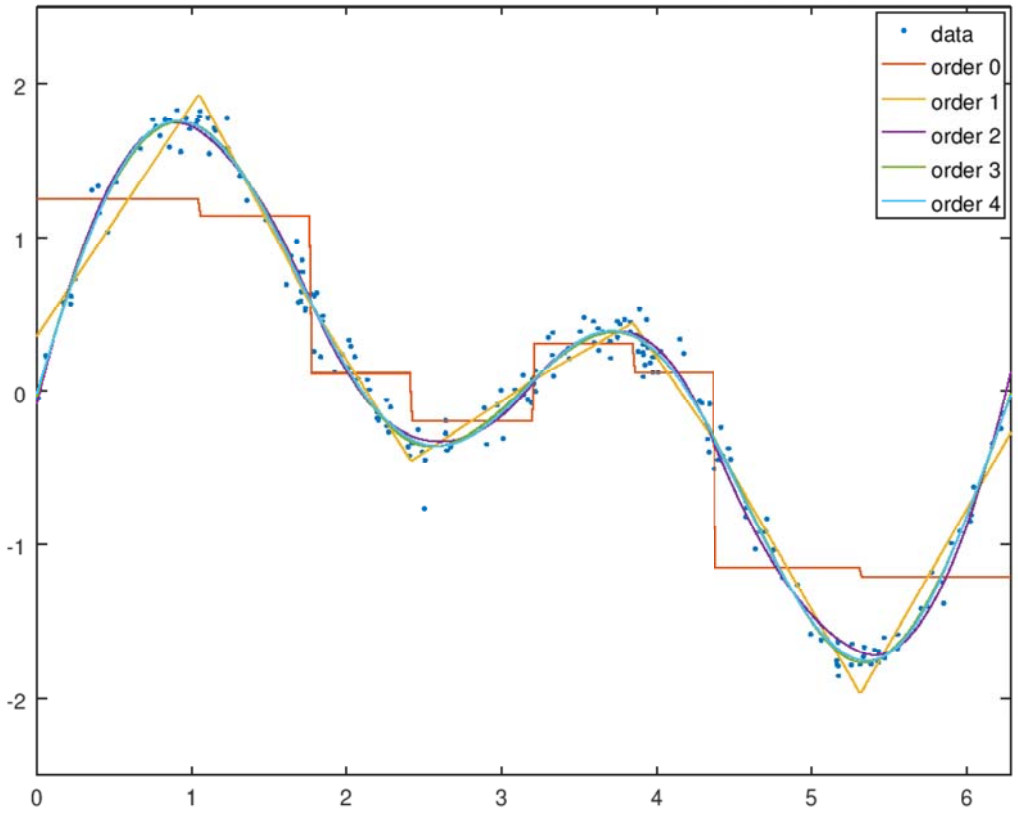
# INTERPOLATION, WIKI



# INTERPOLATION, WIKI



# INTERPOLATION V.S STATISTICS




- Error

# LINEAR INTERPOLATION

- Similar to Local regression:

$$\hat{f}(x) = \begin{cases} y_1 + (x - x_1)\, m_1 & \text{if } x < x_1 \\ y_i + (x - x_i)\, m_i & \text{if } x \in [x_i, x_{i+1}] \\ y_{n-1} + (x - x_{n-1})\, m_n & \text{if } x > x_n \end{cases} \tag{3}$$

where slope $m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$

# LINEAR INTERPOLATION: BINARY SEARCH

- For Prediction, you have to know where does your $x$ locates

$$\hat{f}(x) = \begin{cases} y_1 + (x - x_1) m_1 & \text{if } x < x_1 \\ y_i + (x - x_i) m_i & \text{if } x \in [x_i, x_{i+1}] \\ y_{n-1} + (x - x_{n-1}) m_n & \text{if } x > x_n \end{cases} \tag{4}$$

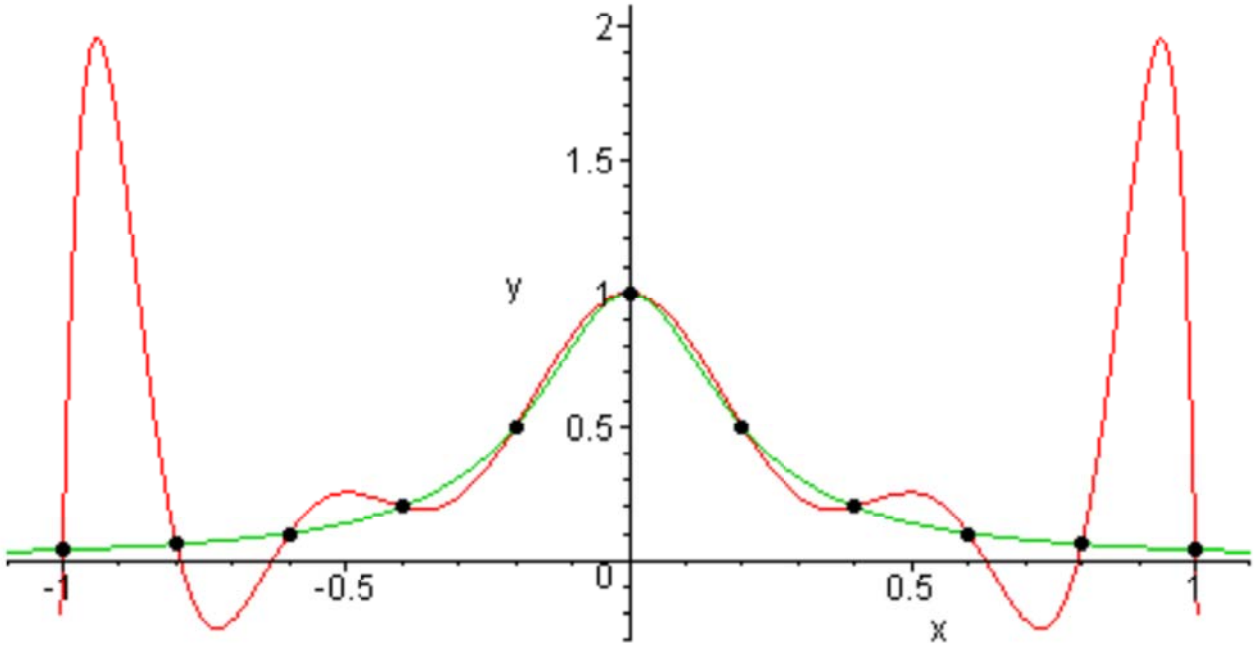
- Here is a question from Gary Gordon's Course: Suppose that the interpolation data $\{x_i\}_{i=1...100000}$, How many times you have to check to find the $x$?
- $\leq \log_2(100000) + 1$

# BINARY SEARCH

- Objection: given $x$, find $i,\ s.t., x \in [x_i, x_{i+1})$
- Algorithm
  - Step 0: set $b = U,\ a = L$
  - Step 1: If $b = a + 1$, stop $i^* = a$, Else go to Step 2
  - Step 2: Set $i = \frac{a+b}{2}$ or the nearest integer from below
  - Step 3: If $x < x_i$, $b = i$, Else, $a = i$. Go to Step 1
- Linear Tech non-differentiable interpolant.

# POLYNOMINAL INTERPOLATION

- Objection: solve function $y = b_0 + b_1 x + b_2 x^2 + \cdots + b_N x^N$
- How many points do you have? Say, 5 points, then you fit a 4-order poly.
- Global Fitting: I draw several points from p.d.f of N(0,1)



# SPLINE INTERPOLATION

- Local Poly Fitting: I only use poly to fit a region, if I use $n$-order poly on each region, I will call this spline with order $n+1$, See Judd(1998)
  - Linear spline/interpolant
  - Quadratic spline
  - Cube spline
- Good convergence properties for smooth functions as the number of knots increase. But how about extreme data?

# SPLINE INTERPOLATION NOT SPLINE REGRESSION

- Linear spline/interpolant
  - Linear Fitting within interval
- Quadratic spline
  - Quadratic Function within interval, $y = a + bx + cx^2$.
  - N points, N-1 Interval, 3(N-1) Paremeters to Estimate.
  - Points Fitting: N Equation $f_i(x_i) = y_i$
  - Value Matching: N-2 Equation $f_i(x_{i+1}) = f_{i+1}(x_{i+1})$
  - Slop Fitting on the middle point: N-2
  - Need one more condition, usually I impose slops at two ending points aggree
- Cube spline
  - Cub Function within interval, $y = a + bx + cx^2 + ex^3$.
  - Points Fitting: N Equation
  - Value Matching: N-2
  - Slop Fitting on the middle point: N-2
  - Curve Fitting on the middle point: N-2
  - Need two more conditions: slops at two ending points are 0s.

# SPLINE REGRESSION

- A kind of Nonlinear Regression
  - Local Regression
- Local Regression
  - Say cubic regression. Two cubic regression for two region. K+1 fittings for K knots.
  - 4(K+1) paremters to fit if I have K knots
- Continuous Piecewise Spline Regression
  - I impose the value matching conditions on K knots.
  - 4(K+1) - K
- Spline Regression
  - I impose the value matching conditions on K knots, and first, second Derivatives Agree
  - 4(K+1) - 3K

# SPLINE REGRESSION ESTIMATION

- Estimation

$$
\begin{aligned}
y_i = \beta_0 + \alpha_1 x_i + \alpha_2 x_i^2 + \alpha_3 x_i^2 + \ldots \\
+ \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_K b_K(x_i) + \epsilon_i
\end{aligned}
$$

$$
h(x, \xi) = (x - \xi)_+^3 = \begin{cases} (x - \xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise} \end{cases}
$$

$$
b_i(x) = h(x, \xi_i)
$$

where $\xi_1, \ldots, \xi_K$ are knots

- See Hastie et al.(2017): The Elements of Statistical Learning: Data Mining, Inference, and Prediction
- There are Free PDF on Hastie's Website.

# B-SPLINE

- Thereom: Any spline of degree k + 1 can be represented as a linear combination of basis splines (B-splines) of the same degree.
- I learn this from Gary' Course.

$$S_k(x) = \sum_{i=1}^{n-k} \alpha_i B_{i,k}(x) \tag{5}$$

Where B will be defined recursively

$$B_{i,1}(x) = \begin{cases} 1 & \text{if } x \in [x_i, x_{i+1}) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

And

$$B_{i,k}(x) = \frac{x - x_i}{x_{i+k-1} - x_i} B_{i,k-1}(x) + \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} B_{i+1,k-1}(x) \tag{7}$$

# CHEBYSHEV POLYNOMIALS

- Polynomial interpolation at evenly spaced nodes often does not produce an accurate approximant. Say

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n \tag{8}$$

And you have n+1 Points to fit So solve

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \dots x_1^n \\ 1 & x_2 & x_2^2 & \dots x_2^n \\ \dots & & & \\ 1 & x_n & x_1^2 & \dots x_{n+1}^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \dots \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_{n+1} \end{bmatrix} \tag{9}$$

- Lots of disadvantages
- But you can change the knots, not even spaced, say Chebychev nodes:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{n-i+0.5}{n}\pi\right), \quad \forall i = 1, 2, \dots, n \tag{10}$$

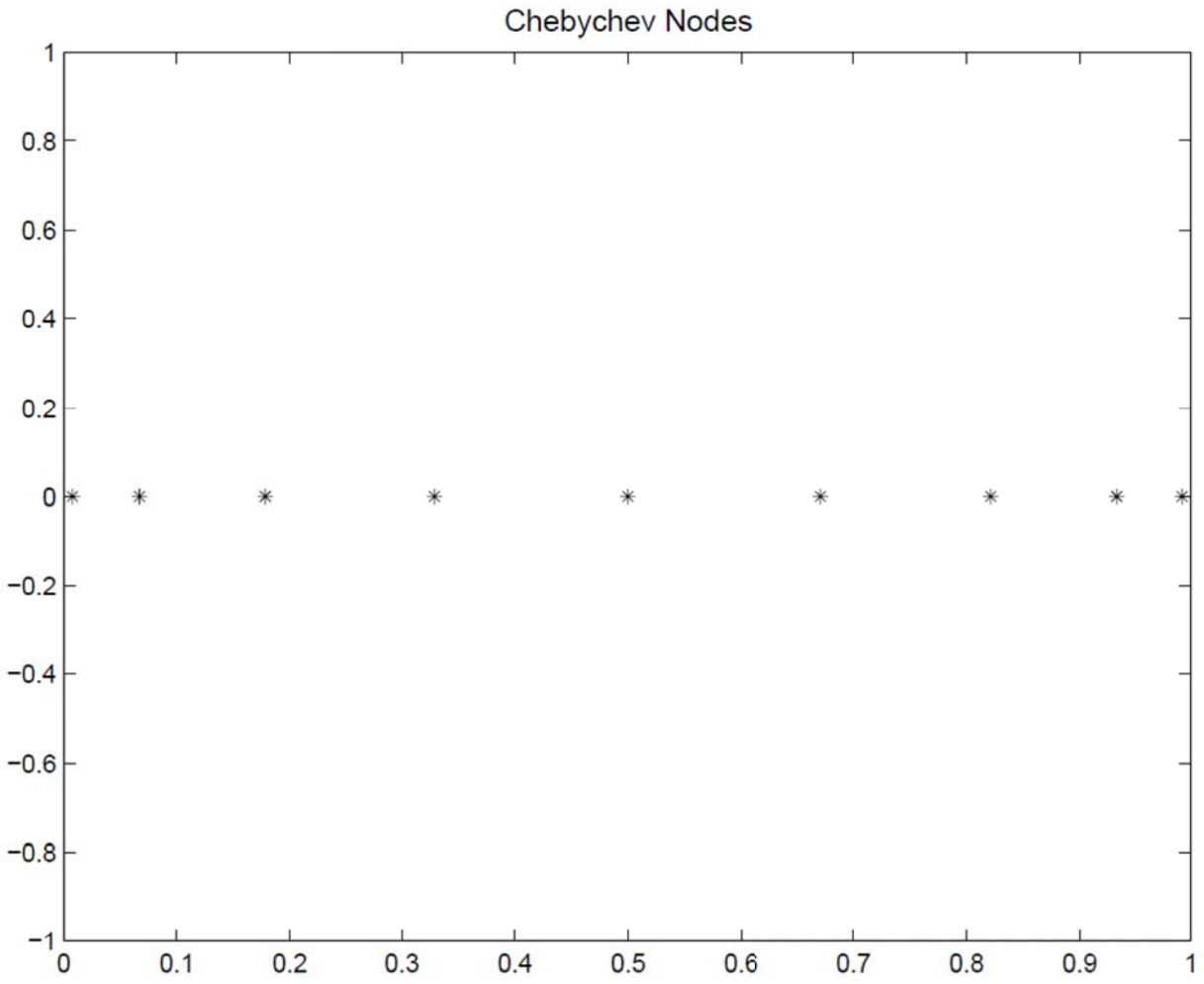
# CHEBYSHEV POLYNOMIALS II

- They are roots for Chebyshev polynomials, say for $x \in [a, b]$, $z \equiv 2\frac{x-a}{b-a} \in [-1, 1]$, C-Poly can be defined recursively as:

$$
\begin{aligned}
T_0(z) &= 1 \\
T_1(z) &= z \\
T_2(z) &= 2z^2 - 1 \\
T_3(z) &= 4z^3 - 3z \\
&\vdots \\
T_j(z) &= 2zT_{j-1}(z) - T_{j-2}(z)
\end{aligned}
\tag{11}
$$

# CHEBYSHEV POLYNOMIALS III

- Che-notes



# CHEBYSHEV POLYNOMIALS IV

- Chebyshev functions

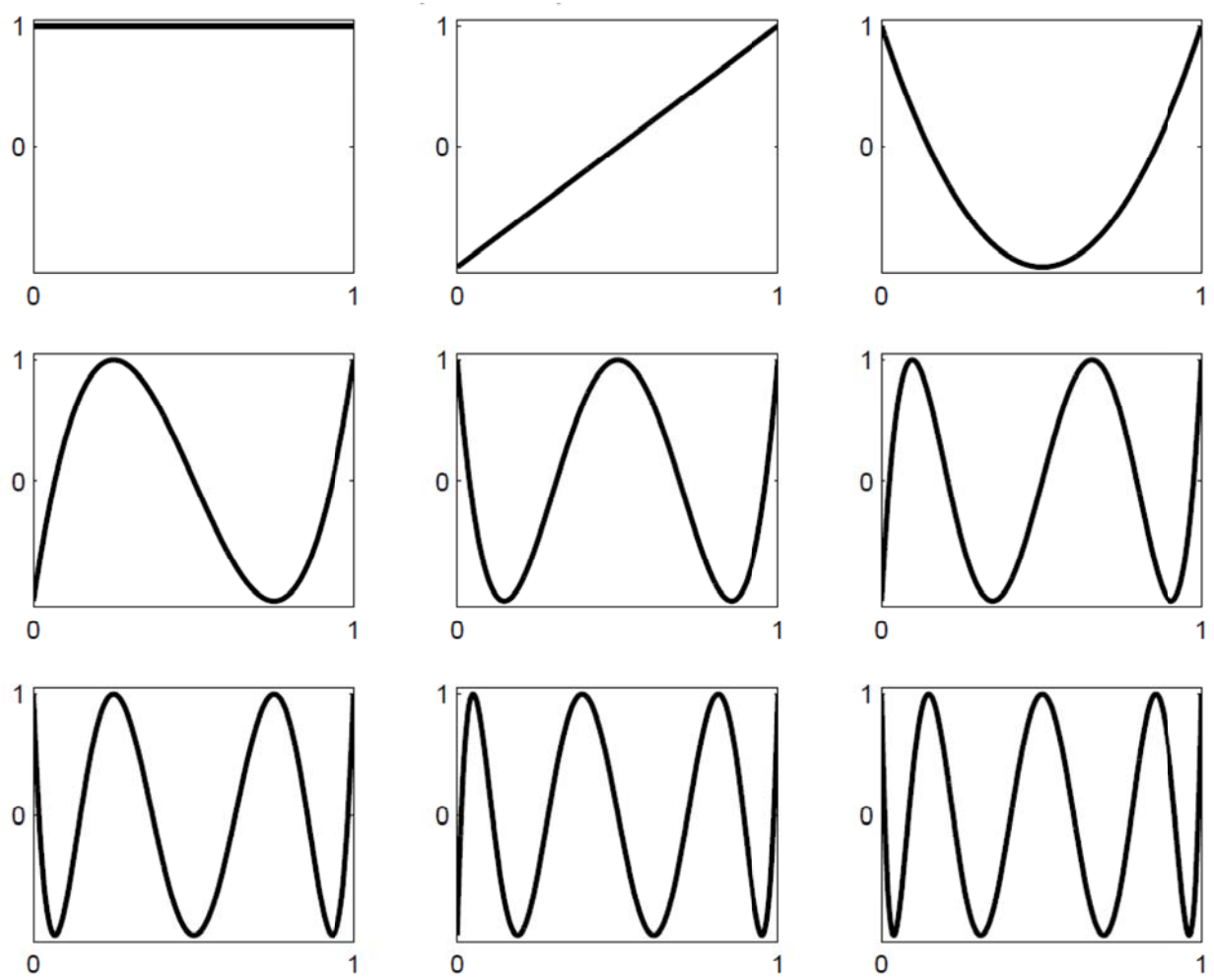



Figure: From Miranda and Fackler(2004)

# CHEBYSHEV POLYNOMIALS V.S POLUNOMIALS

- Chebyshev functions


Figure: From Miranda and Fackler(2004)

# CHEBYSHEV POLYNOMIALS V

- Fiting target funtion with Chebyshev polynomials pn Chebyshev Notes. Solving

$$[\Phi_{ij} = \cos((n - i + 0.5)(j - 1)\pi/n)]_{ij} \begin{bmatrix} c_0 \\ c_1 \\ \dots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix} \quad (12)$$

# INTERPOLATION

- Take some data $\{x_i, y_i\}$ and get values of $y$ for independent variable $x$ not in the range of $\{x_i, y_i\}$. And $x_i \in R^d$
- We can represent our data as

$$\left\{ \left( x_{i_1}^1, x_{i_2}^2, \ldots, x_{i_d}^d \right), y \right\}_{\{(i_1,\ldots,i_d)|1 \leq i \leq n_j\}} \tag{13}$$

# BILINEAR

- Think about Linear interpolation, it is a weighting average of two points

$$y(x) = (1 - \alpha_i(x))\, y_{i+1} - \alpha_i(x) y_i$$
$$\text{if } x \in [x_i, x_{i+1}]$$
$$\alpha_i(x) = \frac{x_{i+1} - x_i}{x_{i+1} - x_i}$$

- However, we have two dimensions, or two weights now

$$\alpha_i(x) = \left(x^1_{i+1} - x^1\right) / \left(x^1_{i+1} - x^1_i\right) \tag{14}$$
$$\beta_i(x) = \left(x^2_{i+1} - x^2\right) / \left(x^2_{i+1} - x^2_i\right) \tag{15}$$
$$\text{and } x = (x^1, x^2) \tag{16}$$

- Cross weighting

# BILINEAR

- Cross Weighting

$$y(x) = \begin{array}{rr} (1-\alpha_i(x))\left(1-\beta_j(x)\right) & y_{i+1,j+1} \\ +\alpha_i(x)\left(1-\beta_j(x)\right) & y_{i,j+1} \\ +(1-\alpha_i(x))\beta_j(x) & y_{i+1,j} \\ +\alpha_i(x)\beta_j(x) & y_{i,j} \end{array} \tag{17}$$

- $\alpha_i(x)$ measures how close $x$ is to $x_i^1$
- $\beta_i(x)$ measures how close $x$ is to $x_i^2$
  - Widely used / understood and easy to extend to higher dimensions
  - No Curvature Preserving
- Photo zooming or Shrinkage

# INTUITION

- What am I doing with Cross-weighting?
- Kind of Creating a distribution. I have $(x_i^1, x_j^2, \ldots, x_k^N, y_{i,j\ldots,k})$
- Extended KNN method.
- But this is still a linear interpolation.
- What if you really want an nonlinear interploation?
- You may want to try Simplical Interpolation, which you can google.

# MULTI-DIMENSIONAL GENERAL INTERPOLATION

- WE CAN Construct a general multi-dimensional interpolation procedure from one-dimensional interpolation methods.
- Interpolate in each dimension sequentially, so interpolation order will affect your result.
  - VAR short-run Decomposition has the same property
- Non-local methods will become infeasible on higher dimension space. It is hard to find neighborhood on high dimensional space.
- For dimension $k$, I have interpolation fiting function $f^k(.)$, given data points $\{x_i^k, y_i\}_{i=1}^{N_k}$. Say I want the value on $(x_*^1, x_*^2, \ldots, x_*^d)$
  - Hence for every possible combination point of $(x^1, x^2, \ldots, x^{d-1})$, I can evaluate the $f^d(x_*^d | \{x_i^k, y_i\}_{i=1}^{N_k})$
- Every possible combination point of $(x^1, x^2, \ldots, x^{d-1})$ gives you a set of $y$s, from which you will interpolate and evaluate at $x_*^d$ so you get a series of $y^{-d}$
- Then we reduce the problem into d-1 dimension if you replace the original $y$ by $y^{-d}$
- Iterate

# MULTI-DIMENSIONAL GENERAL INTERPOLATION: EXAMPLE

- **Take Notes**

# MULTI-DIMENSIONAL GENERAL INTERPOLATION: NOTE

- Rank the dimension by amount of grid points
- Interpolate the dimension with most points first, ...
- There is no nice way currently to preserve shape.

# SCHUMAKER SPLINE: THE MOTIVATION

- Shape reserving fit by quadratic spline.
  - Concavity, Monotonicity
- Actually, this is almost the same with Quatradic spline as we dicussed in previous slides. For more infomation, see Judd(1998); Except that it does not impose any condition on boundary
- As before, I want a quadratic function on $[x_1, x_2]$ to satisfy

$$f(x_i) = y_i, f'(x_i) = z_i \text{ for } i = 1, 2 \tag{18}$$

  where $z_i,\ i = 1,\ 2$ is the predetermined first derivaties of "true" function at $x_i$
- Different with the process above? This kind of question used to show up in path-smoothing condition, which is common if you do stochastic calculus.

# SCHUMAKER SPLINE I

- Hence, we have four equations but only three paramters. Hence, you can solve the equations iff

$$\frac{z_1 + z_2}{2} = \frac{y_2 - y_1}{x_2 - x_1} \tag{19}$$

Furthermore, the fitting function is

$$f(x) = y_1 + z_1 (x - x_1) + \frac{(z_2 - z_1)}{2 (x_2 - x_1)} (x - x_1)^2 \tag{20}$$

- What if I add a knot, say $\xi \in [x_1, x_2]$? and still satisfy the four condition? what is the fitting then?

$$f(x) = \begin{cases} a_1 + b_1 (x - x_1) + c_1 (x - x_1)^2 & \text{if } x \in [x_1, \xi] \\ a_2 + b_2 (x - \xi) + c_2 (x - \xi)^2 & \text{if } x \in [\xi, x_2] \end{cases} \tag{21}$$

# SCHUMAKER SPLINE II

- The fitting function is

$$f(x) = \begin{cases} a_1 + b_1 (x - x_1) + c_1 (x - x_1)^2 & \text{if } x \in [x_1, \xi] \\ a_2 + b_2 (x - \xi) + c_2 (x - \xi)^2 & \text{if } x \in [\xi, x_2] \end{cases} \tag{22}$$

Where the parameters are as following

$$\begin{gathered} a_1 = y_1, \quad b_1 = z_1, \quad c_1 = (\overline{y}' - z_1) / (2\alpha) \\ a_2 = a_1 + b_1 \alpha + c_1 \alpha^2, \quad b_2 = \overline{y}', \quad c_2 = (z_2 - \overline{y}') / (2\beta) \\ \overline{y}' = \frac{2(y_2 - y_1) - (\alpha z_1 + \beta z_2)}{x_2 - x_1} \\ \alpha = \xi - x_1, \beta = x_2 - \xi \end{gathered} \tag{23}$$

- Idea behind Schumaker
  - 4 equations from the value and slop matching condition
  - But you have 6 parameters to go
  - Value matching and Slop Smoothing condition on $\xi$, which is the knot.
  - Under some condition, Schumaker preserves shape.

# SCHUMAKER SPLINE SHAPE CONDITION

- Let $s = (y_2 - y_1)/(x_2 - x_1)$, Then $(z_2 - s)(z_1 - s) \geq 0$ implies fitting cannot be concave or convex. So Let me suppose $(z_2 - s)(z_1 - s) < 0$, Let $\xi = (x_1 + x_2)/2$,
  - Intuition? What if say, we can have a concave or convex fitting? Emm, *Slope* $\in [z_1, z_2]$ *or* $[z_2, z_1]$, hence at every point the slope is larger(smaller) than $s$, average slope then should be larger(smaller) than s.
- Only when $(z_2 - s)(z_1 - s) < 0$, we can move on to choose a knot $\xi$:
  - If $|z_2 - s| < |z_1 - s|$, For all $x_1 < \xi < \bar{\xi}$, $\bar{\xi} = x_1 + \frac{2(x_2 - x_1)(z_2 - s)}{z_2 - z_1}$.
  - If $|z_2 - s| > |z_1 - s|$, For all $\underline{\xi} < \xi < x_2$ $\underline{\xi} = x_2 + \frac{2(x_2 - x_1)(z_1 - \delta)}{z_2 - z_1}$
  - Then fitting spline can keep convex/concave and monotone if the data are.
- Either you fit the data by chance, or we fit it by adding a middle point $\xi$, which is

$$\xi = \begin{cases} (x_1 + x_2)/2 & \text{if } (z_2 - s)(z_1 - s) \geq 0 \\ (x_1 + \bar{\xi})/2 & \text{if } (z_2 - s)(z_1 - s) < 0 \text{ and } |z_2 - s| < |z_1 - s| \\ (\underline{\xi} + x_2)/2 & \text{if } (z_2 - s)(z_1 - s) < 0 \text{ and } |z_2 - s| > |z_1 - s| \end{cases} \tag{24}$$

# DYNAMIC PROGRAMMING

- In Dynamic Programming, Reinforcement Learning and Econoimics, Bellman equation is widely used.
- Let me go back to our objective: solve a model say

$$\max_{C_1, C_2, \ldots, C_t} \sum_{t=1}^{\infty} \beta^{t-1} u(C_t)$$

$$C_t + K_{t+1} \leq F(N_t, K_t) + (1-\delta)K_t$$

$$C_t \geq 0$$

$$K_{t+1} \geq 0$$

# DYNAMIC PROGRAMMING BREAK DOWN

- Let me use Value function following the lecture 2.

$$
\begin{aligned}
V(K_t) &= U(C_t) + \beta V(K_{t+1}) \\
C_t + K_{t+1} &\leq F(N_t, K_t) + (1-\delta)K_t \\
C_t &\geq 0 \\
K_{t+1} &\geq 0
\end{aligned}
$$

- Value function is a function of Capital
- Rewrite above as

$$
\begin{aligned}
V(K_t) &= \max_{K_{t+1}} U(F(N, K_t) + (1-\delta)K_t - K_{t+1}) + \beta V(K_{t+1}) \\
0 &\leq K_{t+1} \leq F(N, K_t) + (1-\delta)K_t
\end{aligned}
$$

- Value-Base Algorithm would be similar

$$
\begin{aligned}
V^{M+1}(K_t) &= \max_{K_{t+1}} U(F(N, K_t) + (1-\delta)K_t - K_{t+1}) + \beta V^{M}(K_{t+1}) \\
0 &\leq K_{t+1} \leq F(N, K_t) + (1-\delta)K_t
\end{aligned}
$$

# VALUE FUNCTION METHOD: DISCRETE STATE SPACE

- Smooth Function, by the idea of iterpolation, can be nearly perfectly backed out from enough Points.
- In other words, what I need is a table, which list out several points $\{(k_i, y_i)\}_{i=1^N}$, where $k_i \in [.7k^*,\ 1.3k^*]$
- Hence give a partition $\{K_i\}_{i=1}^N$, I will give the first round $N$ values, which corresponds to $y_i$. WIW is a vector $\mathbf{v}$, where

$$V_i^{M+1} = \max_{K_j} U(F(N, K_i) + (1-\delta)K_i - K_j) + \beta V_j^M$$
$$0 \leq K_j \leq F(N, K_i) + (1-\delta)K_i$$
$$k_j \in \{K_1,\ K_2, \ldots,\ K_N\}$$

# DISCRETE STATE SPACE: ALGORITHM

- Set out $\{K_1,\ K_2, \ldots,\ K_N\}$, Set $\mathbf{v}^0$ as any N-dimensional Vector you like, normally you can set it as a zero vector. Or $\frac{1}{1-\beta}U(F(N,K^*)-\delta K^*)\mathbf{1}$
- Iterative Step

$$V_i^{M+1} = \max_{k_j} U(F(N,K_i) + (1-\delta)K_i - k_j) + \beta V_j^M$$

$$0 \leq k_j \leq F(N,K_i) + (1-\delta)K_i$$

$$k_j \in \{K_1,\ K_2, \ldots,\ K_N\}$$

$$for\ i = 1, 2, \ldots, N$$

- STOP when $\mathbf{v}^M$ converges
- Tryout: Repeat what you did in your HW2. Set N = 10,000
- If you really want approximate your value function well, set N large. However this will bring great computation burden, so you need to speed up your algorithm.

# SPEED UP: EXPLOITINGMONOTONICITY AND CONCAVITY

- Since policy function is increasing, as long as we impose increasing utility function, See Stocky and Lucas with Prescott(1989)

$$K_i \geq K_j \Rightarrow K_i' = h(K_i) \geq K_j' = h(K_j) \tag{25}$$

  Hence you do not need to try out all the $K_j' \in \{K_1, K_2, \ldots, K_N\}$, simply try values above $K_{j-1}'$
- Value function $\phi(K') := u(f(K) - K') + \beta v(K')$ is concave w.r.t $K'$ as long as the utiliy function is. Ok, for a concave function
  - If our function is decreasing (increasing) over the whole grid, maximum is the first (last) point of the grid.
  - If our function is first increasing and then decreasing, then pick the mid-point of the grid and the points next to it, and decide wheter the optimal point is on the right side or left
    - Left if $\phi(K_m) > \phi(K_{m+1})$
    - Right if $\phi(K_m) < \phi(K_{m+1})$

# POLICY FUNCTION ITERATION

- Each time a policy function $\mathbf{h}^M$ is computed, we solve for the value function that would occur, if the policy were followed forever.
- And this updated value function is used to get next policy function $\mathbf{h}^{M+1}$, which will be used to solve the value function again ...
- OpenAI
- You may wonder what if I use a stochastic Policy? Same
- Policy function $\mathbf{h}$ maps current $K_i$ into $K_j$, hence following such policy will result

$$v_i = u\left(f\left(K_i\right) - K_j\right) + \beta v_j, \quad i = 1, 2, \ldots, n \tag{26}$$

These are N equations. In matrix term:

$$\mathbf{v} = \mathbf{u}_\mathbf{h} + \beta Q_\mathbf{h} \mathbf{v} \tag{27}$$

$$\mathbf{v} = \left[I - \beta Q_\mathbf{h}\right]^{-1} \mathbf{u}_\mathbf{h} \tag{28}$$

# POLICY FUNCTION ITERATION: ALGORITHM

- Set out $\{K_1, K_2, \ldots, K_N\}$, Set $\mathbf{v}^0$ as any N-dimensional Vector you like, normally I will set $\mathbf{v}^0 = \frac{1}{1-\beta} U(F(N, K^*) - \delta K^*)\mathbf{1}$
- Get policy function from it hence I have $Q^1$, I do not need the iterated value function, instead I will set $\mathbf{v}^1 = \left[I - \beta Q^1\right] \mathbf{v}^0$, you may wonder Q is a matrix which is hard to store...
- You can update policy funtion after K iteration of value function:

$$\begin{aligned} \mathbf{w}^1 &= \mathbf{v}^0 \\ \mathbf{w}^{l+1} &= \mathbf{u} + \beta Q^1 \mathbf{w}^l, \quad l = 1, \ldots, k \\ \mathbf{v}^1 &= \mathbf{w}^{k+1} \end{aligned}$$

- Stop when policy converges.

# MORE ACCURATE VALUE FUNCTION ITERATION

- Yes I use discrete state space to approximate:
  $V^{M+1}(K_i) = \max_K U(F(N, K_i) + (1-\delta)K_i - K) + \beta V^M(K)$

$$V_i^{M+1} = \max_{K_j} U(F(N, K_i) + (1-\delta)K_i - K_j) + \beta V_j^M$$

$$0 \leq K_j \leq F(N, K_i) + (1-\delta)K_i$$

$$k_j \in \{K_1, K_2, \ldots, K_N\}$$

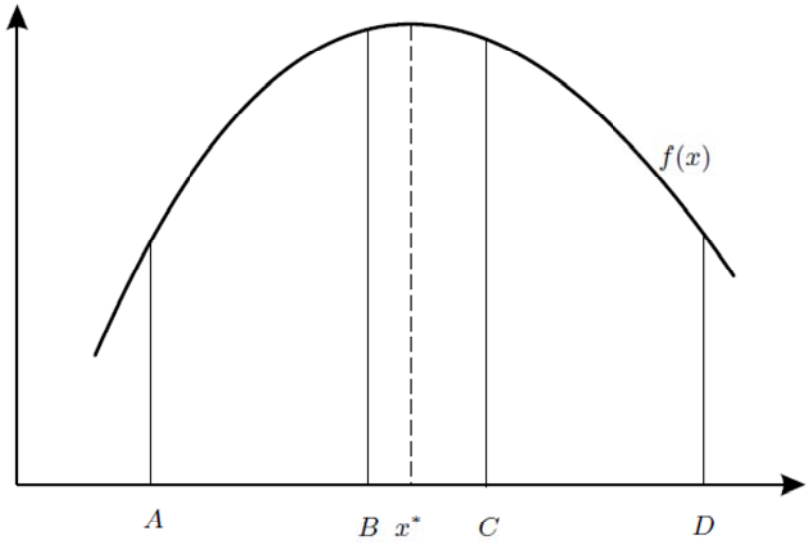
- Say $K_j$ is our current optiomal choice. However, the true optimal may not be $K_j$, it lies around $K_j$, in other words, $\in (K_{j-1}, K_{j+1})$.
- We can use golden search to find the optimum, HOLD ON!
  - But we do not know the value function at point other than $K_j$...
  - Gold Search, Find the maximum of a single peaked function $f(x)$ in Interval $[L, U]$

# GOLDEN SECTION SEARCH

- Set $A = L,\ D = U$ and compute

$$B = pA + (1-p)D$$
$$C = (1-p)A + pD$$
$$p = \frac{\sqrt{5}-1}{2}$$

- If $f(B) > f(C)$, Set $D = C$; Else If $f(C) > f(B)$, Set $A = B$ Iterate until $|B - C| < tol$



# SOLUTION OF STOCHASTIC MODELS

- Let **K** denote the endogenous state variable of the model, **Z** a purely exogenous shock governed by a stationary stochastic process, I can abstractly define

$$v(K,Z) = \max_{K' \in \mathcal{D}_{K,Z}} u\left(Z,K,K'\right) + \beta E\left[v\left(K',Z'\right)|Z\right] \tag{29}$$

- Discretize again, But now we have two dimension, hence I discretize $K,\ Z$, with n and m points. For $(K_i, Z_j)$, value function take value $V_{ij}$, Hence I will get a $n \times m$ matrix.

$$v_{ij} = \max_{K_k \in \mathcal{K}_{ij}} u(Z_j, K_i, K_k) + \beta \sum_{l=1}^{m} p_{jl} v_{kl}$$

$$i = 1, 2, \ldots, n \ and \ j = 1, 2, \ldots, m$$

And

$$p_{jl} = Pr(next\ state\ = Z_l | Current\ = Z_j)$$

# SOLUTION OF STOCHASTIC MODELS: DISCRETIZE

- Easy to discretize $Z$. I talked about it at the very beginning of this slides, Tauchen Method, let me brief it here
- Consider $Z_{t+1} = \varrho Z_t + \epsilon_t, \quad \epsilon_t \sim N\left(0, \sigma_\epsilon^2\right)$, with unconditional mean 0 and variance $\sigma_Z^2 = \sigma_\epsilon^2 / \left(1 - \varrho^2\right)$.
- I will use equidistant points $z_1 < z_2, \ldots, < z_m$, with $z_m = \lambda \sigma_Z$, $z_1 = -z_m$, let me use $dz = \frac{z_2 - z_1}{2}$:

$$\text{prob}\left(z_j - dz \leq z \leq z_j + dz \middle| z_i\right) = \pi\left(z_j + dz\right) - \pi\left(z_j - dz\right)$$

$$\pi \text{ is c.d.f of } N(\rho z_i, \sigma^2)$$

$$p_{ij} = \phi(\frac{z_j - \varrho z_i + dz}{\sigma_\epsilon}) - \phi(\frac{z_j - \varrho z_i - dz}{\sigma_\epsilon}), j \neq 1, m$$

for the boundary points I denote

$$p_{i1} = \phi(\frac{z_1 - \varrho z_i + dz}{\sigma_\epsilon}) \tag{30}$$

$$p_{im} = 1 - \phi(\frac{z_1 - \varrho z_i - dz}{\sigma_\epsilon}) \tag{31}$$

# MARKOVIAN TRANSITION PROBABILITY CONTINUED

- Value function can be viewed as the accumulated value you stand at the finity
- Hence time does not matter, or every thing stays stationary. What is the expected value function at stationary state?
- Denote $\pi_t = [\pi_1^t, \pi_2^t, \ldots, \pi_N^t]^T$, distribution of z at time t Hence

$$\pi'_{t+1} = \pi'_t P \tag{32}$$

Stationary means

$$\left(I - P'\right)\pi = 0 \tag{33}$$

- I want the $\pi$, take $\lim_{k\to\infty} P^k$, take any row would be the $\pi$
- Or solve

$$\pi' \begin{bmatrix} p_{11}-1 & p_{12} & \ldots & p_{1,n-1} & 1 \\ p_{21} & p_{22}-1 & \ldots & p_{1,n-1} & 1 \\ \vdots & & & & \\ p_{n1}-1 & p_{n2} & \ldots & p_{n,n-1}-1 & 1 \end{bmatrix} = (0,0,\ldots,1) \tag{34}$$

Why?

# ALGORITHM FOR STOCHASTIC CASE

- Fundermentally, I will use $v_{ij} = \max_{K_k \in \mathcal{K}_{ij}} u(Z_j, K_i, K_k) + \beta \sum_{l=1}^{m} p_{jl} v_{kl}$
- Discretize $K$ is not that straightforward, In deterministic case, if $K_0 < K^*$, I know it will increase in the future, But here we have $Z$, which is random
- Let's choose a partition $\mathcal{K}$ by guess and verify, start with a small interval, enlarge it if policy function hits the boundaries of this interval.
- First interval? $1 = \beta\left(1 - \delta + Z_j f'\left(K_j^*\right)\right)$, $K_j = \lambda K_j^*$
- OK, How about the first guess about value function? An educated guess would be

$$v_{ij}^0 = u\left(Z_j f\left(K_i^*\right) - \delta K_i^*\right) + \beta \sum_{l=1}^{m} p_{jl} v_{il}^0 \tag{35}$$

In matrix

$$V^0 = \left(I - \beta P'\right)^{-1} U \tag{36}$$

$$U = \left(u_{ij}\right), u_{ij} = u\left(Z_j f\left(K_j^*\right) - \delta K_j^*\right) \tag{37}$$

# BASIC ALGORITHM FOR STOCHASTIC CASE

- Start with a coarse grid on the interval
- Use

$$v_{ij}^0 = u\left(Z_j f\left(K_i^*\right) - \delta K_i^*\right) + \beta \sum_{l=1}^{m} p_{jl} v_{il}^0 \tag{38}$$

  to compute(update) the value function
- Make the grid finer by using more points
- Interpolate column-wise between neighboring points of the old grid and the respective points of Value function

# POLICY-BASED ALGORITHM FOR STOCHASTIC CASE

- Start with a Policy function $H$, where $h_{ij}$ points to which $K_k$ is chosen if current state is $K_i, Z_j$, suppose $H$ will be implemented for ever

$$v_{ij} = u_{ij} + \beta \sum_{l=1}^{m} p_{jl} v_{h_{ij},l} \tag{39}$$

$$u_{ij} = u\left(Z_j, K_i, K_{h_{ij}}\right) \tag{40}$$

- Use the (Several step) updated value function to find (update) Policy function. Continue until converged.
- For interpolation, when I add point K between $K_i, K_{i+1}$ from $\mathcal{K}$:

$$\hat{\phi}(K) = u\left(Z_j, K_i, K\right) + \beta \sum_{l=1}^{m} p_{jl} \hat{v}_l(K) \tag{41}$$

# POLICY-BASED ALGORITHM

- Start with a Partition $\mathcal{K}$, and initial $V^0$. Update using

$$v_{ij}^1 = u_{ij} + \beta \sum_{l=1}^{m} p_{jl} v_{h_{ij},l}^0 \tag{42}$$

$$u_{ij} = u\left(Z_j, K_i, K_{h_{ij}}\right) \tag{43}$$

  and get the policy function $H^1$
- Updating: For each $j = 1, 2, \ldots, m$:
  - Set $k_{0,j}^* = 1$
  - For each $i = 1, ..., n$, and $k_{0,i-1}^*$ find index $k^*$ to maximize

$$\omega_k = u\left(Z_j, K_i, K_k\right) + \beta \sum_{l=1}^{m} p_{jl} v_{kl}^0, \; k \in \left\{k_{i-1j}^*, k_{i-1j}^* + 1, \ldots, n\right\} \tag{44}$$

    - If no need interpolation, then you get the $H^1$ and $V^1$
    - If $k^* = 1$, boundary hitted, extropolate and get $\hat{\phi}(K_1 + \epsilon)$, if it is less then $\hat{\phi}(K_1)$, I know the optimal is $K_1$. No interpolation;
      Else I know the optiomal should be in $[K_1, K_2]$, use golden search get the new point and update the partition and value function. Similar for $k^* = n$
  - Policy-assisted step: use policy to update the value function again
- Continue until converges

# REINFORCEMENT LEARNING ABC, SIMPLY FOR FUN

- Notes