An Introduction to Interpolation Methods©

Dean Corbae

November, 2020

Interpolation as an Approximation Tool

- We sometimes know the numerical value of a function $f(x)$ at a set of points $\{x_i\}_{i=1}^{N}$ where $x_1 < x_2 < ... < x_N$, but we don't have an analytic expression for $f(x)$ that lets us calculate its value at any arbitrary point.
- Discrete grid on a : Let D be the grid points. Problem is

$$\begin{aligned} V_D(a) &= \max_{a'_D \in D} u(y + a - qa'_D) + \beta V_D(a'_D) \\ s.t. a &\in D. \end{aligned}$$

- Interpolation. Problem is

$$\begin{aligned} V_I(a) &= \max_{a' \geq \underline{a}} u(y + a - qa'_I) + \beta V_I(a'_I) \\ s.t. a &\in D. \end{aligned}$$

- Even though $V_I(a'_I)$ is piecewise linear, the objective will be concave since $u(y + a - qa'_I)$ induces strict concavity.

Approximation of Functions

Some References:

1. Judd, K. Numerical Methods in Economics, Cambridge: MIT Press. Chapter 6.
2. Press, W. et. al. Numerical Recipes in Fortran 77, Cambridge: Cambridge University Press. Chapter 3 and 5

Theorem

Weierstrass Approximation: If $f \in C([a,b])$,then for all $\varepsilon > 0$,there exists a polynomial $p(x)$ such that $\|f - p\|_\infty < \varepsilon$ where $\|g\|_\infty = \sup_{x \in [a,b]} |g(x)|$ is the sup norm. Hence there is a sequence of polynomials p_n such that $p_n \to f$ uniformly on $[a,b]$.

Piecewise Linear Interpolation

- Suppose we know $\{y_i = f(x_i)\}_{i=1}^N$ at some discrete set of points $\{x_i\}_{i=1}^N$. For instance, this could be the value function $v^j(k_i)$ at iteration j at capital grid points.

- We construct a function $\ell(x)$ such that:
  - $\ell(x_i) = y_i,\ i = 1, ..., N$ and
  - on each interval $[x_i, x_{i+1}]$ for $i = 1, ..., N-1$,

  the function is linear

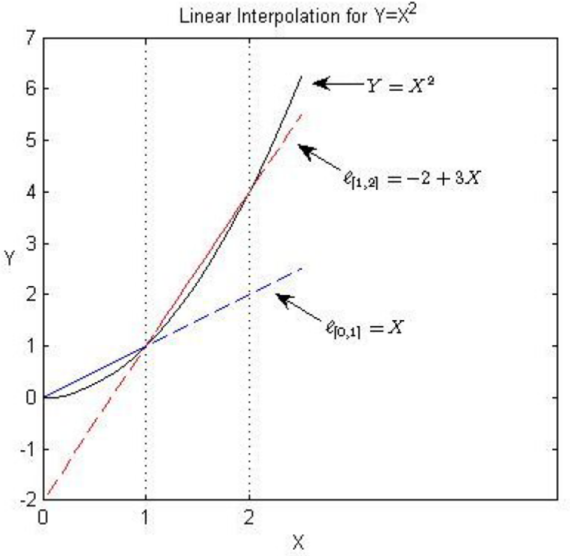
$$\ell_{[x_i, x_{i+1}]}(x) = A_i(x) y_i + (1 - A_i(x))\, y_{i+1}, \tag{1}$$

where the weights A_i measure the relative distance between the point x and the grid points x_i and x_{i+1} given by $A_i(x) = \left(\frac{x_{i+1} - x}{h}\right)$ where $h = x_{i+1} - x_i$.

Piecewise Linear Interpolation Example

- Suppose $f(x) = x^2$ and $\{x_i\}_{i=1}^{N} = \{0, 1, 2\}$. Then $y_1 = 0$, $y_2 = 1$, and $y_3 = 4$.
  - For $x \in [0, 1]$,
$$\ell_{[0,1]}(x) = \left(\frac{1-x}{1}\right) \cdot 0 + \left(\frac{x-0}{1}\right) \cdot 1 = x.$$
  - For $x \in [1, 2]$,
$$\begin{aligned} \ell_{[1,2]}(x) &= \left(\frac{2-x}{1}\right) \cdot 1 + \left(\frac{x-1}{1}\right) \cdot 4 \\ &= 2 - x + 4x - 4 \\ &= -2 + 3x \end{aligned}$$

Linear Interpolation of $f(x) = x^2$



Cubic Splines

- We construct a function $s(x)$ such that:
  - $s(x_i) = y_i,\ i = 1, \ldots, N$ and
  - on each interval $[x_i, x_{i+1}]$ for $i = 1, ..., N-1$,

  the function is a cubic

$$s_{[x_i, x_{i+1}]}(x) = a_i + b_i x + c_i x^2 + d_i x^3. \qquad (2)$$

- So we have $N-1$ intervals and N data points.
- Since we have 4 coefficients for each of the $N-1$ intervals, we have $4(N-1)$ unknowns a_i, b_i, c_i, d_i for $i = 1, \ldots, N-1$.

Cubic Splines - cont.

How do we find those coefficients?

- The interpolating conditions at the endpoints and continuity at the interior nodes will provide us with $2(N-1)$ equations
  - $y_1 = s_{[x_1,x_2]}(x_1)$ for $i = 1$,
  - $s_{[x_{i-1},x_i]}(x_i) = y_i = s_{[x_i,x_{i+1}]}(x_i)$ for $i = 2, ..., N-1$,
  - and $y_N = s_{[x_{N-1},x_N]}(x_N)$ for $i = N$
- or:

$$
\begin{aligned}
y_i &= a_{i-1} + b_{i-1}x_i + c_{i-1}x_i^2 + d_{i-1}x_i^3, \quad i = 2, \ldots, N \qquad (3)\\
y_i &= a_i + b_i x_i + c_i x_i^2 + d_i x_i^3, \quad i = 1, \ldots, N-1. \qquad (4)
\end{aligned}
$$

Cubic Splines - cont.

- Continuity of the first derivative on the interior nodes gives $(N-2)$ (i.e. the interior nodes is why it is -2) equations $s'_{[x_{i-1},x_i]}(x_i) = s'_{[x_i,x_{i+1}]}(x_i)$ or:

$$b_{i-1}+2c_{i-1}x_i+3d_{i-1}x_i^2 = b_i+2c_ix_i+3d_ix_i^2, \quad i=2,\ldots,N-1 \tag{5}$$

- Continuity of the second derivative on the interior nodes (i.e. the interior nodes is why it is -2) gives $(N-2)$ equations $s''_{[x_{i-1},x_i]}(x_i) = s''_{[x_i,x_{i+1}]}(x_i)$ or:

$$2c_{i-1}+6d_{i-1}x_i = 2c_i+6d_ix_i, \quad i=2,\ldots,N-1 \tag{6}$$

Cubic Splines - cont.

Since we have $2(N-1)+2(N-2)$ equations but $4(N-1)$ unknowns, we still need two more conditions.

- If we knew the true function, then we could use first derivatives $s'_{[x_1,x_2]}(x_1) = f'(x_1)$ and $s'_{[x_{N-1},x_N]}(x_N) = f'(x_N)$. But in general we don't know $f(x)$, so can't calculate $f'(x)$.
- Hermite splines approximate that derivative. In particular, we use the slopes of the secant lines over $[x_1, x_2]$ and $[x_{N-1}, x_N]$;that is, we choose $s(x)$ to make

$$\begin{aligned} s'_{[x_1,x_2]}(x_1) &= \frac{s_{[x_1,x_2]}(x_2) - s_{[x_1,x_2]}(x_1)}{x_2 - x_1} \\ s'_{[x_{N-1},x_N]}(x_N) &= \frac{s_{[x_{N-1},x_N]}(x_N) - s_{[x_{N-1},x_N]}(x_{N-1})}{x_N - x_{N-1}}. \end{aligned} \tag{7}$$

- Now that we have $4(N-1)$ equations in $4(N-1)$ unknowns, you can just solve a system of equations in matlab.

Cubic Splines Example

Suppose $f(x) = x^2$ and $\{x_i\}_{i=1}^{N} = \{0, 1, 2\}$ (i.e. $N = 3$). Then $y_1 = 0$, $y_2 = 1$, and $y_3 = 4$ as before. For the cubic spline, we need $4(3-1) = 8$ coefficients $\{a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2\}$.

- From (3) we have

$$\begin{aligned} y_2 &= a_1 + b_1 x_2 + c_1 x_2^2 + d_1 x_2^3 \iff 1 = a_1 + b_1 + c_1 + d_1 . \\ y_3 &= a_2 + b_2 x_3 + c_2 x_3^2 + d_2 x_3^3 \iff 4 = a_2 + 2b_2 + 4c_2 + 8d_2 . \end{aligned}$$

- From (4) we have

$$\begin{aligned} y_1 &= a_1 + b_1 x_1 + c_1 x_1^2 + d_1 x_1^3 \iff 0 = a_1 \\ y_2 &= a_2 + b_2 x_2 + c_2 x_2^2 + d_2 x_2^3 \iff 1 = a_2 + b_2 + c_2 + d_2 . \end{aligned}$$

- From (5) we have

$$b_1 + 2c_1 x_2 + 3d_1 x_2^2 = b_2 + 2c_2 x_2 + 3d_2 x_2^2 \iff b_1 + 2c_1 + 3d_1 = b_2 + 2c_2 + 3d_2 .$$

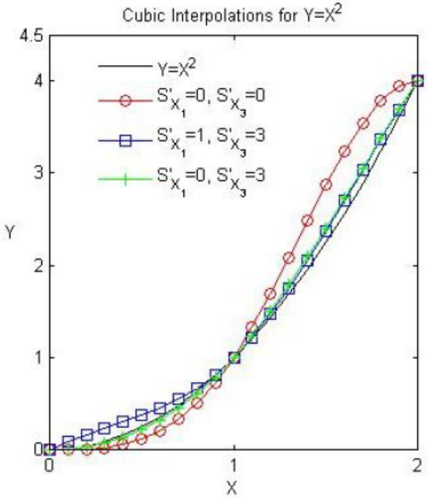
- From (6) we have

$$2c_1 + 6d_1 x_2 = 2c_2 + 6d_2 x_2 \iff 2c_1 + 6d_1 = 2c_2 + 6d_2 .$$

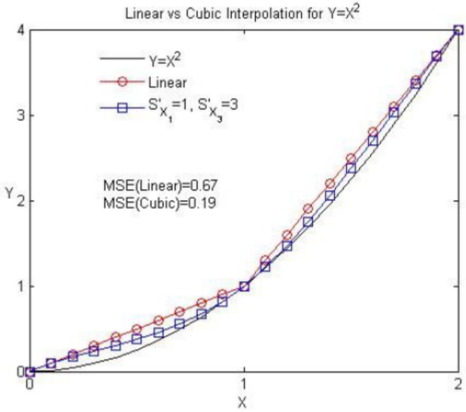
- From (7), using $s'_{[x_1, x_2]}(x) = b_1 + 2c_1 x + 3d_1 x^2$, we have

$$\begin{aligned} s'_{[0,1]}(0) &= \frac{y_2 - y_1}{1} \iff b_1 = 1 \\ \text{and } s'_{[1,2]}(2) &= \frac{y_3 - y_2}{1} \iff b_2 + 4c_2 + 12d_2 = 3 \end{aligned}$$

Cubic Spline Interpolation of $f(x) = x^2$



Linear vs. Cubic Spline Interpolation of $f(x) = x^2$

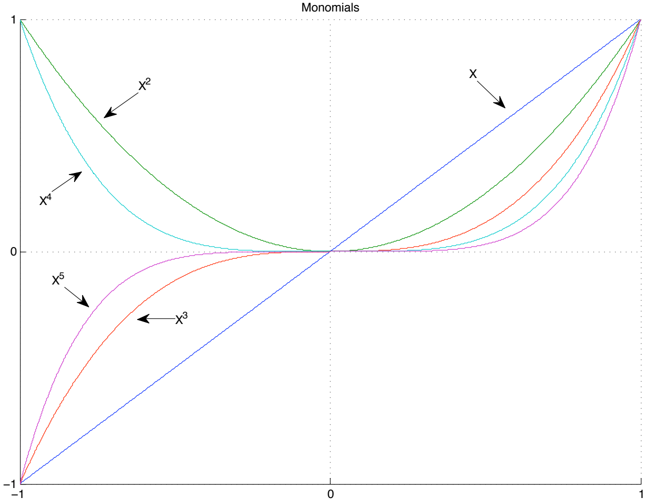


Since linear interpolation is a special case of the Cubic hermite spline, its mean square error must be weakly higher.

Orthogonal Polynomials

- The Weierstrass theorem is conceptually valuable, but not important from a computational point of view because it uses Bernstein polynomials which converge slowly.
- Since the space of continuous functions is spanned by the monomials x^n, $n = 0, 1, \ldots$,it is natural to think of the monomials as a basis for the space of continuous functions.
- But these monomials are very nearly linearly dependent, so higher n aren't very efficient at helping (like multicollinearity in regression where we don't get a good fit (i.e. high standard errors)).

Spanning: Monomials



Orthogonal Polynomials

- To avoid this problem, can use an alternative basis that is orthogonal in $C([a,b])$. The Gram-Schmidt algorithm can construct an orthonormal basis.
- A special case of orthogonal polynomials is the Chebyshev polynomials on $x \in [-1,1]$ given by the recursive formula

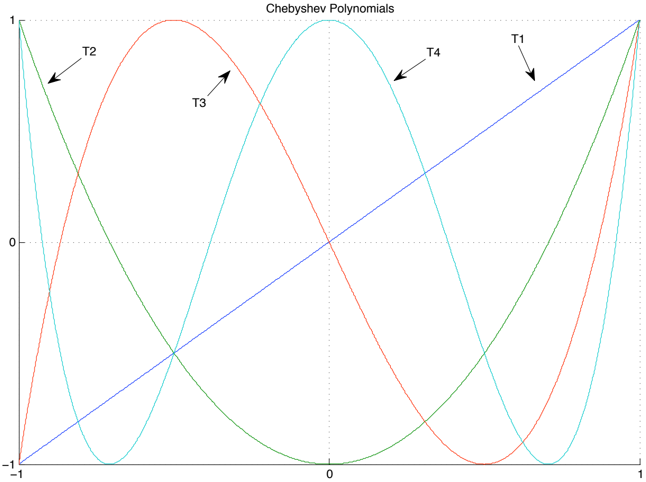
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), n \geq 1 \tag{8}$$

where $T_0(x) = 1$ and $T_1(x) = x$. Notice

$$\begin{aligned} T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 2x\left(2x^2 - 1\right) - x = 4x^3 - 3x \\ &\ldots. \end{aligned}$$

- The different spanning capabilities of monomials and Chebyshev polynomials on $[-1,1]$ is evident by comparing the two figures.

Spanning: Chebeyshev Polynomials



Chebeyshev Approximation Algorithm

- Chebyshev approximation chooses grid points to get a good fit.
- Judd (p. 223) presents the following algorithm (6.2) where you choose m nodes and use them to construct a degree $n < m$ polynomial approximation of $f(x)$ on $[a, b]$ via the following steps:

1. Compute the $m \geq n + 1$ Chebyshev interpolation nodes on $[-1, 1]$:

$$z_k = -\cos\left(\frac{2k-1}{2m} \cdot \pi\right), k = 1, ..., m.$$

2. Adjust the nodes to the $[a, b]$ interval:

$$x_k = (z_k + 1)\left(\frac{b-a}{2}\right) + a, k = 1, ..., m.$$

Chebeyshev Approximation Algorithm - cont.

3. Evaluate f at the approximation nodes:

$$y_k = f(x_k), k = 1, ..., m.$$

4. Compute Chebyshev coefficients, $c_i, i = 0, ...,$ where T_i is given by (8):

$$c_i = \frac{\sum_{k=1}^{m} y_k T_i(z_k)}{\sum_{k=1}^{m} T_i(z_k)^2}$$

to arrive at the approximation for $f(x), x \in [a, b]$ given by

$$\widehat{f}(x) = \sum_{i=0}^{n} c_i T_i \left(2 \cdot \frac{x-a}{b-a} - 1\right).$$

Bilinear Interpolation

- Now consider two dimensions. In this case, you can think of one of the dimensions being along the grid $\{x_i\}_{i=1}^{N}$ and another being along the grid $\{z_j\}_{j=1}^{M}$, forming an $N \times M$ matrix of grid points (x_i, z_j) in a "Cartesian Mesh"(language from the Press, et. al. Numerical Recipes book).
- For any arbitrary point (x, z), let the functional values be given by $y = f(x, z)$ and let the associated matrix of functional values on the grid be given by the $N \times M$ matrix $y_{i,j} = f(x_i, z_j)$.
- We want to approximate, by interpolation, the function f at some untabulated point (x, z).

Bilinear Approximation

- A grid rectangle is just the 2 dimensional analogue of an interval in the 1 dimensional case we saw before.
  - the four tabulated points that surround the desired interior point starting with the lower left and moving counterclockwise numbering the points y^1, y^2, y^3, y^4.
  - If $x \in [x_i, x_{i+1}]$ and $z \in [z_j, z_{j+1}]$ implicitly defines i and j, then
    - $y^1 = y_{i,j}$, $y^2 = y_{i+1,j}$, $y^3 = y_{i+1,j+1}$, and $y^4 = y_{i,j+1}$
- Then the equation of the surface over the grid rectangle using bilinear interpolation is given by

$$b_{y^1,y^2,y^3,y^4}(x,z) = (1-t)\cdot(1-u)\cdot y^1 + t\cdot(1-u)\cdot y^2 + t\cdot u\cdot y^3 + (1-t)\cdot u\cdot y^4 \tag{9}$$

where

$$t \equiv \frac{x - x_i}{x_{i+1} - x_i}$$
$$u \equiv \frac{z - z_j}{z_{j+1} - z_j}$$

so that t and u lie between 0 and 1.

Bilinear Mesh

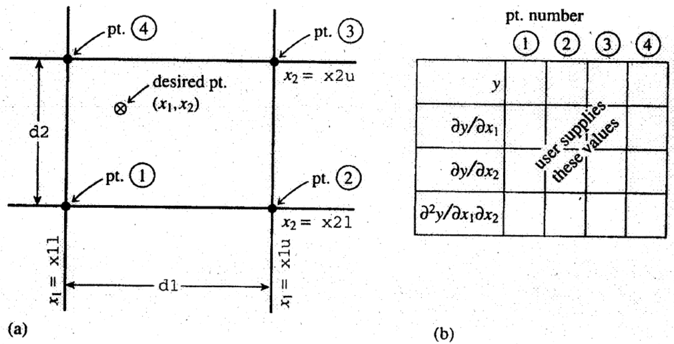


Figure 3.6.1. (a) Labeling of points used in the two-dimensional interpolation routines bcuint and bcucof. (b) For each of the four points in (a), the user supplies one function value, two first derivatives, and one cross-derivative, a total of 16 numbers.

Bilinear Approximation

One way to think of this is that for a given $z \in \{z_j, z_{j+1}\}$, b is a linear interpolation of the type in section 2.

- Specifically, suppose $z = z_j$, then $u = 0$,

$$b_{y^1,y^2,y^3,y^4}(x, z_j) = \left[1 - \left(\frac{x - x_i}{x_{i+1} - x_i}\right)\right] \cdot y^1 + \left(\frac{x - x_i}{x_{i+1} - x_i}\right) \cdot y^2 \quad (10)$$

  where

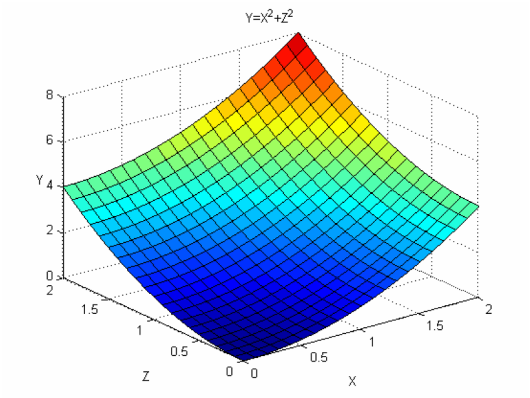
$$y^1 = f(x_i, z_j), y^2 = f(x_{i+1}, z_j).$$

- Notice that (10) is just like (1) where the first term is like $1 - A$ and the second like A.

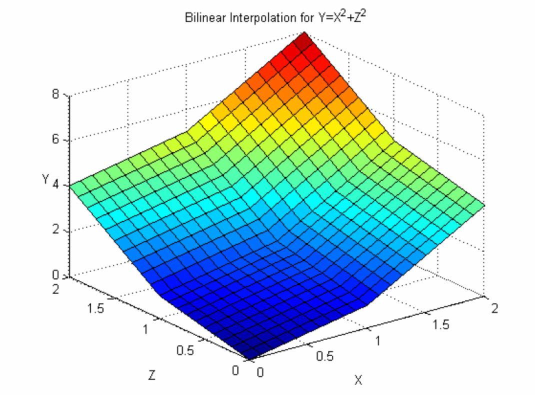
- We can always rearrange (10) to yield

$$\begin{aligned} b_{y^1,y^2,y^3,y^4}(x, z_j) &= \left[y^1 - \left(\frac{x_i}{x_{i+1} - x_i}\right) \cdot \left[y^2 - y^1\right]\right] + \left(\frac{y^2 - y^1}{x_{i+1} - x_i}\right) \cdot x \\ &= b + mx \end{aligned}$$

$f(x, z) = x^2 + z^2$



Bilinear Interpolation of $f(x, z) = x^2 + z^2$



One dimensional linear interpolation algorithm

- Suppose we have a set of grid points $X = \{x_i\}_{i=1}^N$ where $x_1 < \ldots < x_i < \ldots < x_N$ and we have a piecewise linear function $v^j(x)$ at iteration j.
- Specifically, the function is defined by $N-1$ intervals for which we have an intercept and a slope:

interval	value $v^j(x)$
$x \in [x_1, x_2]$	$a^j_{1,2} + b^j_{1,2}x$
...	...
$x \in [x_n, x_{n+1}]$	$a^j_{n,n+1} + b^j_{n,n+1}x$
...	...
$x \in [x_{N-1}, x_N]$	$a^j_{N-1,N} + b^j_{N-1,N}x$

- For each x_i, we are trying to solve for v^{j+1} given by

$$v^{j+1}(x_i) = \max_{x' \in [x_1, x_N]} u(x_i - x') + \beta v^j(x'). \tag{11}$$

- The fact that $x' \in [x_1, x_N]$ rather than $x' \in X$ is the gain to doing interpolation.

Value Function Interpolation

An algorithm to solve this problem (not necessarily the fastest way) is:

- While $\sup_x |v^j(x) - v^{j-1}(x)| > \varepsilon$, do
  - for $i = 1, ..., N$,
    - use fminsearch to minimize $-V_i^{j+1}$ where:

$$V_i^{j+1} = \begin{bmatrix} u(x_i - x') + \beta \left[a_{1,2}^j + b_{1,2}^j x\right], x' \in [x_1, x_2] \\ \dots \\ u(x_i - x') + \beta \left[a_{k,k+1}^j + b_{k,k+1}^j x\right], x' \in [x_k, x_{k+1}] \\ \dots \\ u(x_i - x') + \beta \left[a_{N-1,N}^j + b_{N-1,N}^j x\right], x' \in [x_{N-1}, x_N] \end{bmatrix}$$

      call the minimized value $v^{j+1}(x_i)$. To be clear, this is just a vector of functions to be minimized by choice of $x' \in [x_1, x_N]$.
    - with the new $v^{j+1}(x_i)$, recompute $\{a_{i,i+1}^{j+1}, b_{i,i+1}^{j+1}\}_{i=1}^{N-1}$.

Value Function Interpolation

- How do we use the matlab function fminsearch? fminsearch(W,initial x_0', [],parameters(i.e. $\{a^j_{i,i+1}, b^j_{i,i+1}\}_{i=1}^{N-1}$)) where you create a matlab function in order to define the function $W = -V_i^{j+1}(x')$.
- For each x_i, we can create the following function $Wfun$ of the choice variable x' (in matlab) as:
  - function $W = Wfun(x')$
  - if $x' \in [x_1, x_2]$, then $W = -\left\{u(x_i - x') + \beta[a^j_{1,2} + b^j_{1,2}x']\right\}$
  - else if ....$x' \in [x_k, x_{k+1}]$, then $W = -\left\{u(x_i - x') + \beta[a^j_{k,k+1} + b^j_{k,k+1}x'] \right\}$
  - else if $x' \in [x_{N-1}, x_N]$, then $W = -\left\{u(x_i - x') + \beta[a^j_{N-1,N} + b^j_{N-1,N}x']\right\}$
  - else $W = 1e + 25$
- The fminsearch routine will vary $x' \in [x_1, x_N]$ among the different intervals $\{[x_1, x_2], ..., [x_{N-1}, x_N]\}$ until it finds the value of x' which minimizes $Wfun(x')$.