Introduction to Projection Methods and step #1: Function Approximation

Wouter J. Den Haan London School of Economics

© by Wouter J. Den Haan

Dynamic Stochastic Model

$$\max_{\{c_t, k_{t+1}\}_{t=1}^{\infty}} E_t \left[\sum_{t=1}^{\infty} \frac{c_t^{1-\nu} - 1}{1-\nu} \right]$$

s.t.
$$c_t + k_{t+1} = z_t k_t^{\alpha} + (1-\delta)k_t$$

$$\ln(z_{t+1}) = \rho \ln(z_t) + \varepsilon_{t+1}$$

$$\varepsilon_{t+1} \sim N(0, \sigma^2)$$

$$k_1, z_1 \text{ given}$$

Set $\delta = 1$ to simplify notation.



First-Order Conditions

$$c_t^{-\nu} = \mathsf{E}_t \left[\beta c_{t+1}^{-\nu} \alpha z_{t+1} k_{t+1}^{\alpha-1} \right]$$

$$c_t + k_{t+1} = z_t k_t^{\alpha}$$

$$\ln(z_{t+1}) = \rho \ln(z_t) + \varepsilon_{t+1}$$

$$\varepsilon_{t+1} \sim N(0, \sigma^2)$$

$$k_1, z_1 \text{ given}$$

<ロト < 回 ト < 目 ト < 目 ト 目 の Q () 3/34

Splines

Extra

Solution of the First-Order Conditions

True rational expectations solution:

$$egin{array}{rcl} c_t &=& c(k_t,z_t)\ k_{t+1} &=& k(k_t,z_t) \end{array}$$

Why a difficult problem to find these?

Three steps

- 1. Function Approximation
- 2. Numerical Integration
- 3. Solving DSGE models with projection methods

Step #3 is made difficult because the functions we solve for are only *implicitly* defined by the first-order conditions.

Goal

Obtain an approximation for

f(x)

when

f(x) is unknown, but we have some information, or
f(x) is known, but too complex to work with

Information available



- usually at one point
- **or** finite set of function values
 - f_1, \cdots, f_m at *m* nodes, x_1, \cdots, x_m

Classes of approximating functions

$1. \ polynomials$

- this still gives lots of flexibility
- examples of second-order polynomials

$$a_0 + a_1 x + a_2 x^2$$

•
$$a_0 + a_1 \ln(x) + a_2 (\ln(x))^2$$

•
$$\exp\left(a_0 + a_1 \ln(x) + a_2 (\ln(x))^2\right)$$

2. splines, e.g., linear interpolation

Splines

Extra

Classes of approximating functions

Polynomials and splines can be expressed as

$$f(x) \approx \sum_{i=0}^{n} \alpha_i T_i(x)$$

 T_i(x): the basis functions that define the class of functions used, e.g., for regular polynomials:

$$T_i(x)=x^i.$$

 α_i: the coefficients that pin down the particular approximation

Reducing the dimensionality

unknown f(x): infinite dimensional object $\sum_{i=0}^{n} \alpha_i T_i(x)$: n+1 elements

General procedure

- Fix the order of the approximation *n*
- Find the coefficients $\alpha_0, \cdots, \alpha_n$
- Evaluate the approximation
- ▶ If necessary, increase *n* to get a better approximation

Weierstrass (sloppy definition but true)

Let $f : [a, b] \longrightarrow \mathbb{R}$ be any real-valued function. For large enough *n*, it is approximated arbitrarily well with the polynomial



Thus, we can get an accurate approximation if

- f is not a polynomial
- f is discontinuous

How can this be true?

How to find the coefficients of the approximating polynomial?

With derivatives:

use the Taylor expansion

- ▶ With a set of points (nodes), x₀, · · · , x_m, and function values, f₀, · · · , f_m?
 - use projection
 - Lagrange way of writing the polynomial (see last part of slides)

Function fitting as a projection

Let

$$Y = \begin{bmatrix} f_0 \\ \vdots \\ f_m \end{bmatrix}, X = \begin{bmatrix} T_0(x_0) & T_1(x_0) & \cdots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \cdots & T_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ T_0(x_m) & T_1(x_m) & \cdots & T_n(x_m) \end{bmatrix}$$

then

 $Y \approx X \alpha$

• We need $m \ge n+1$. Is m = n+1 as bad as it is in empirical work?

▶ What problem do you run into if *n* increases?

Orthogonal polynomials

 Construct basis functions so that they are orthogonal to each other, i.e.,

$$\int_{a}^{b} T_{i}(x) T_{j}(x) w(x) dx = 0 \quad \forall i, j \in i \neq j$$

This requires a particular weighting function (density), w(x), and range on which variables are defined, [a, b]

Splines

Extra

Chebyshev orthogonal polynomials

$$[a,b] = [-1,1]$$
 and $w(x) = \frac{1}{(1-x^2)^{1/2}}$

▶ What if function of interest is not defined on [-1,1]?

Constructing Chebyshev polynomials

The basis functions of the Chebyshev polynomials are given by

$$T_0^c(x) = 1$$

$$T_1^c(x) = x$$

$$T_{i+1}^c(x) = 2xT_i^c(x) - T_{i-1}^c(x) \quad i > 1$$

Splines

Extra

Chebyshev versus regular polynomials

Chebyshev polynomials, i.e.,

$$f(x) \approx \sum_{j=0}^{n} a_j T_j^c(x),$$

can be rewritten as regular polynomials, i.e.,

$$f(x)\approx\sum_{j=0}^n b_j x^j,$$

<ロト < 部 > < 言 > < 言 > こ き く こ > こ の Q () 18/34

Chebyshev nodes

• The n^{th} -order Chebyshev basis function has n solutions to

$$T_n^c(x) = 0$$

▶ These are the *n* Chebyshev nodes

Splines

イロト 不得 トイヨト イヨト

Extra

Discrete orthogonality property

Evaluated at the Chebyshev nodes, the Chebyshev polynomials satisfy:

$$\sum_{i=1}^{n} T_j^c(x_i) T_k^c(x_i) = 0 \text{ for } j \neq k$$

Thus, if

$$X = \begin{bmatrix} T_0(x_0) & T_1(x_0) & \cdots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \cdots & T_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ T_0(x_m) & T_1(x_m) & \cdots & T_n(x_m) \end{bmatrix}$$

then X'X is a diagonal matrix

3

Uniform convergence

- Weierstrass \implies there is a good polynomial approximation
- Weierstrass $\Rightarrow f(x) = \lim_{n \to \infty} p_n(x)$ for every sequence $p_n(x)$
- If polynomials are fitted on Chebyshev nodes even uniform convergence is guaranteed

Splines

Inputs:

- 1. n+1 nodes, x_0, \cdots, x_n
- 2. n+1 function values, $f(x_0)\cdots, f(x_n)$
- nodes are fixed =>> the n+1 function values are the coefficients of the spline



Piece-wise linear

For
$$x \in [x_i, x_{i+1}]$$

 $f(x) \approx \left(1 - \frac{x - x_i}{x_{i+1} - x_i}\right) f_i + \left(\frac{x - x_i}{x_{i+1} - x_i}\right) f_{i+1}.$

- That is, a separate linear function is fitted on the n intervals
- Still it is easier/better to think of the coefficients of the approximating function as the n+1 function values

Splines

Extra

Piece-wise linear versus polynomial

- Advantage: Shape preserving
 - in particular monotonicity & concavity (strict?)
- Disadvantage: not differentiable

Extra material

- 1. Lagrange interpolation
- 2. Higher dimensional polynomials
- 3. Higher-order splines

Lagrange interpolation

Let

$$L_i(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} \text{ and }$$

$$f(x) \approx f_0 L_0(x) + \cdots + f_n L_n(x).$$

- ▶ Right-hand side is an *n*th-order polynomial
- By construction perfect fit at the n+1 nodes?
- \blacktriangleright \implies the RHS is the *n*th-order approximation

イロト 不得 トイヨト イヨト 二日

27 / 34

Extra

Higher-dimensional functions

second-order complete polynomial in x and y:

$$\sum_{0 \le i+j \le 2} a_{i,j} x^i y^j$$

second-order tensor product polynomial in x and y:

$$\sum_{i=0}^{2} \sum_{j=0}^{2} a_{i,j} x^{i} y^{j}$$

Complete versus tensor product

tensor product can make programming easier

simple double loop instead of condition on sum

▶ n^{th} tensor has higher order term than $(n+1)^{\text{th}}$ complete

2nd-order tensor has fourth-order power

at least locally, lower-order powers are more important
 ⇒ complete polynomial may be more efficient

Higher-order spline

Cubic (for example)

- III Same inputs as with linear spline, i.e. n+1 function values at n+1 nodes which can still be thought of as the n+1 coefficients that determine approximating function
- > Now fit 3^{rd} -order polynomials on each of the *n* intervals

$$f(x) \approx a_i + b_i x + c_i x^2 + d_i x^3 \text{ for } x \in [x_{i-1}, x_i].$$

What conditions can we use to pin down these coefficients?

Cubic spline conditions: levels

- We have 2+2(n−1) conditions to ensure that the function values correspond to the given function values at the nodes.
 - For the intermediate nodes we need that the cubic approximations of both adjacent segments give the correct answer. For example, we need that

$$f_1 = a_1 + b_1 x_1 + c_1 x_1^2 + d_1 x_1^3 \text{ and} f_1 = a_2 + b_2 x_1 + c_2 x_1^2 + d_2 x_1^3$$

For the two endpoints, x₀ and x_{n+1}, we only have one cubic that has to fit it correctly.

Cubic spline conditions: 1st-order derivatives

 To ensure differentiability at the intermediate nodes we need

 $b_i + 2c_ix_i + 3d_ix_i^2 = b_{i+1} + 2c_{i+1}x_i + 3d_{i+1}x_i^2$ for $x_i \in \{x_1, \cdots, x_{n-1}\}$

which gives us n-1 conditions.

Cubic spline conditions: 2nd-order derivatives

▶ To ensure that second derivatives are equal we need

$$2c_i + 6d_ix_i = 2c_{i+1} + 6d_{i+1}x_i$$
 for $x_i \in \{x_1, \cdots, x_{n-1}\}$.

- We now have 2+4(n-1) = 4n-2 conditions to find 4n unknowns.
- We need two additional conditions; e.g. that 2nd-order derivatives at end points are zero.

Splines - additional issues

- (standard) higher-order splines do not preserve shape
- higher-order difficult for multi-dimensional problems
- first-order trivial for multi-dimensional problems
 - if interval is small then nondifferentiability often doesn't matter

References

- Den Haan, W.J., Numerical Integration, online lecture notes.
- Heer, B., and A. Maussner, 2009, Dynamic General Equilibrium Modeling.
- ▶ Judd, K. L., 1998, Numerical Methods in Economics.
- Miranda, M.J, and P.L. Fackler, 2002, Applied Computational Economics and Finance.