# Introduction to Projection Methods and step \#1: Function Approximation 

Wouter J. Den Haan<br>London School of Economics

(C) by Wouter J. Den Haan

## Dynamic Stochastic Model

$$
\begin{array}{cc}
\max _{\left\{c_{t}, k_{t+1}\right\}_{t=1}^{\infty}} & \mathrm{E}_{t}\left[\sum_{t=1}^{\infty} \frac{c_{t}^{1-v}-1}{1-v}\right] \\
& \text { s.t. } \\
c_{t}+k_{t+1} & =z_{t} k_{t}^{\alpha}+(1-\delta) k_{t} \\
\ln \left(z_{t+1}\right) & =\rho \ln \left(z_{t}\right)+\varepsilon_{t+1} \\
& \varepsilon_{t+1} \sim N\left(0, \sigma^{2}\right) \\
& k_{1}, z_{1} \text { given }
\end{array}
$$

Set $\delta=1$ to simplify notation.

## First-Order Conditions

$$
\begin{aligned}
c_{t}^{-v}= & \mathrm{E}_{t}\left[\beta c_{t+1}^{-v} \alpha z_{t+1} k_{t+1}^{\alpha-1}\right] \\
c_{t}+k_{t+1}= & z_{t} k_{t}^{\alpha} \\
\ln \left(z_{t+1}\right)= & \rho \ln \left(z_{t}\right)+\varepsilon_{t+1} \\
& \varepsilon_{t+1} \sim N\left(0, \sigma^{2}\right) \\
& k_{1}, z_{1} \text { given }
\end{aligned}
$$

## Solution of the First-Order Conditions

## True rational expectations solution:

$$
\begin{aligned}
c_{t} & =c\left(k_{t}, z_{t}\right) \\
k_{t+1} & =k\left(k_{t}, z_{t}\right)
\end{aligned}
$$

- 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

- Either finite set of derivatives
- 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

## 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:

$$
T_{i}(x)=x^{i}
$$

- $\alpha_{i}$ : the coefficients that pin down the particular approximation


## Reducing the dimensionality

unknown $f(x)$ : infinite dimensional object

$$
\sum_{i=0}^{n} \alpha_{i} T_{i}(x): \quad n+1 \text { elements }
$$

## 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

$$
\sum_{i=0}^{n} \alpha_{i} x^{i}
$$

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), $x_{0}, \cdots, x_{m}$, and function values, $f_{0}, \cdots, f_{m}$ ?
- use projection
- Lagrange way of writing the polynomial (see last part of slides)


## Function fitting as a projection

Let

$$
Y=\left[\begin{array}{c}
f_{0} \\
\vdots \\
f_{m}
\end{array}\right], X=\left[\begin{array}{cccc}
T_{0}\left(x_{0}\right) & T_{1}\left(x_{0}\right) & \cdots & T_{n}\left(x_{0}\right) \\
T_{0}\left(x_{1}\right) & T_{1}\left(x_{1}\right) & \cdots & T_{n}\left(x_{1}\right) \\
\vdots & \vdots & \ddots & \vdots \\
T_{0}\left(x_{m}\right) & T_{1}\left(x_{m}\right) & \cdots & T_{n}\left(x_{m}\right)
\end{array}\right]
$$

then

$$
Y \approx X \alpha
$$

- We need $m \geq n+1$. 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.,

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

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


## Chebyshev orthogonal polynomials

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

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


## Constructing Chebyshev polynomials

- The basis functions of the Chebyshev polynomials are given by

$$
\begin{aligned}
T_{0}^{c}(x) & =1 \\
T_{1}^{c}(x) & =x \\
T_{i+1}^{c}(x) & =2 x T_{i}^{c}(x)-T_{i-1}^{c}(x) \quad i>1
\end{aligned}
$$

## 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.,

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

## Chebyshev nodes

- The $n^{\text {th }}$-order Chebyshev basis function has $n$ solutions to

$$
T_{n}^{c}(x)=0
$$

- These are the $n$ Chebyshev nodes


## Discrete orthogonality property

- Evaluated at the Chebyshev nodes, the Chebyshev polynomials satisfy:

$$
\sum_{i=1}^{n} T_{j}^{c}\left(x_{i}\right) T_{k}^{c}\left(x_{i}\right)=0 \text { for } j \neq k
$$

- Thus, if

$$
X=\left[\begin{array}{cccc}
T_{0}\left(x_{0}\right) & T_{1}\left(x_{0}\right) & \cdots & T_{n}\left(x_{0}\right) \\
T_{0}\left(x_{1}\right) & T_{1}\left(x_{1}\right) & \cdots & T_{n}\left(x_{1}\right) \\
\vdots & \vdots & \ddots & \vdots \\
T_{0}\left(x_{m}\right) & T_{1}\left(x_{m}\right) & \cdots & T_{n}\left(x_{m}\right)
\end{array}\right]
$$

then $X^{\prime} X$ is a diagonal matrix

## Uniform convergence

- Weierstrass $\Longrightarrow$ there is a good polynomial approximation
- Weierstrass $\nRightarrow f(x)=\lim _{n \rightarrow \infty} p_{n}(x)$ for every sequence $p_{n}(x)$
- If polynomials are fitted on Chebyshev nodes $\Longrightarrow$ even uniform convergence is guaranteed


## Splines

Inputs:

1. $n+1$ nodes, $x_{0}, \cdots, x_{n}$
2. $n+1$ function values, $f\left(x_{0}\right) \cdots, f\left(x_{n}\right)$

- nodes are fixed $\Longrightarrow$ the $n+1$ function values are the coefficients of the spline


## Piece-wise linear

- For $x \in\left[x_{i}, x_{i+1}\right]$

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

- 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


## 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

$$
\begin{gathered}
L_{i}(x)=\frac{\left(x-x_{0}\right) \cdots\left(x-x_{i-1}\right)\left(x-x_{i+1}\right) \cdots\left(x-x_{n}\right)}{\left(x_{i}-x_{0}\right) \cdots\left(x_{i}-x_{i-1}\right)\left(x_{i}-x_{i+1}\right) \cdots\left(x_{i}-x_{n}\right)} \text { and } \\
f(x) \approx f_{0} L_{0}(x)+\cdots+f_{n} L_{n}(x) .
\end{gathered}
$$

- Right-hand side is an $n^{\text {th }}$-order polynomial
- By construction perfect fit at the $n+1$ nodes?
$>\Longrightarrow$ the RHS is the $n^{\text {th }}$-order approximation


## Higher-dimensional functions

- second-order complete polynomial in $x$ and $y$ :

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

- 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
$-n^{\text {th }}$ tensor has higher order term than $(n+1)^{\text {th }}$ complete
- $2^{\text {nd }}$-order tensor has fourth-order power
- at least locally, lower-order powers are more important $\Longrightarrow$ complete polynomial may be more efficient


## Higher-order spline

## Cubic (for example)

- !!! 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 $3^{\text {rd }}$-order polynomials on each of the $n$ intervals

$$
f(x) \approx a_{i}+b_{i} x+c_{i} x^{2}+d_{i} x^{3} \text { for } x \in\left[x_{i-1}, x_{i}\right] .
$$

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

$$
\begin{aligned}
& f_{1}=a_{1}+b_{1} x_{1}+c_{1} x_{1}^{2}+d_{1} x_{1}^{3} \text { and } \\
& f_{1}=a_{2}+b_{2} x_{1}+c_{2} x_{1}^{2}+d_{2} x_{1}^{3}
\end{aligned}
$$

- For the two endpoints, $x_{0}$ and $x_{n+1}$, we only have one cubic that has to fit it correctly.

Cubic spline conditions: $1^{\text {st }}$-order derivatives

- To ensure differentiability at the intermediate nodes we need
$b_{i}+2 c_{i} x_{i}+3 d_{i} x_{i}^{2}=b_{i+1}+2 c_{i+1} x_{i}+3 d_{i+1} x_{i}^{2}$ for $x_{i} \in\left\{x_{1}, \cdots, x_{n}\right.$
which gives us $n-1$ conditions.

Cubic spline conditions: $2^{\text {nd }}$-order derivatives

- To ensure that second derivatives are equal we need

$$
2 c_{i}+6 d_{i} x_{i}=2 c_{i+1}+6 d_{i+1} x_{i} \text { for } x_{i} \in\left\{x_{1}, \cdots, x_{n-1}\right\} .
$$

- We now have $2+4(n-1)=4 n-2$ conditions to find $4 n$ unknowns.
- We need two additional conditions; e.g. that $2^{\text {nd }}$-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.

