Intro to Projection Methods in Economics

Frantisek Masek

Sapienza University of Rome

May 2022

Introduction

Content

- What does it mean to solve a model?
  - Distinction of state and control variables. Policy function and its approximation. Outline of function approximation.

- Projection Methods. Intro and the very first intuition.
  - Intro to logic behind the projection methods based on simple generic equations.

- Tools needed to be able to use projection methods.
  - Some concepts from function approximation and numerical integration. Specifically, Chebyshev polynomials, splines, Gauss–Hermite quadrature, collocation...

- Let's set it up within reasoning of economics.
  - Using projection to solve well-known models.

Original sources

- Let's be honest from the very beginning
  - There is not many innovative thoughts of mine in these slides
  - More or less I just retell awesome presentations and notes from Wouter den Haan, Jesús Fernández-Villaverde, and Ken Judd in a way I understand the topic.
  - On top of these slides, I also use textbooks of Judd (1998), Miranda and Fackler (2004), Heer and Maussner (2009) and papers of Fernández-Villaverde, Rubio Ramírez, and Schorfheide (2016) or Miftakhova, Schmedders, and Schumacher (2020).
  - There is a plenty of material out there nowadays, so do not be shy to do your own search for what suits you best.

Solving a model

Solving a model

- We have derived a system of non-linear equations. What's next?
  - We do not know exact shapes of the resulting functions (indeed, analytical closed-form solutions are ultra-rare). It is necessary to approximate them.
  - Put it differently, we have to solve for so-called policy functions.

- It might be possibly quite a tricky problem.
  - Remember about the general name of this sort of models.
  - Dynamic Stochastic General Equilibrium

What does it mean to solve a model?

We have to find policy rules; i.e., relationships defining how choice variables depend on state variables.

Solving a model

- What are control and state variables? And what are the policy functions?
  - Let me bring up some example. Consider the simplest version of RBC model with the exogenous process for technology and without labour.
  - The model boils down to the following system of non-linear equations.

$$C_t^{-\eta} = \beta \, \mathbb{E}_t \, C_{t+1}^{-\eta} \alpha Z_{t+1} k_{t+1}^{\alpha-1}$$

$$C_t + k_{t+1} = Z_t k_t^{\alpha}$$

$$Z_t = (1-\rho)Z + \rho Z_{t-1} + \sigma \epsilon_t,$$

where σ controls the degree of uncertainty.

Examples

Sort out this simple model. Denoting the state variables s, we have: $s_t = [k_t, Z_t]$. When control variables are denoted c, we get: $c_t = [c_t]$.

Solving a model

- Have a look at the system of equations above. What do we know?
  - We surely know Z_t. We also know the initial given value of k. However, we need to choose C_t -> the policy function! Once we obtain C_t, k_{t+1} comes out straight from the budget constraint equation.
- We need to solve out (approximate) for C_t.

Function approximation

Function approximation

- Function approximation, econometrics, and machine learning
  - What do we see if we focus on underlying logic instead of buzzwords?
  - There is actually a lot in common

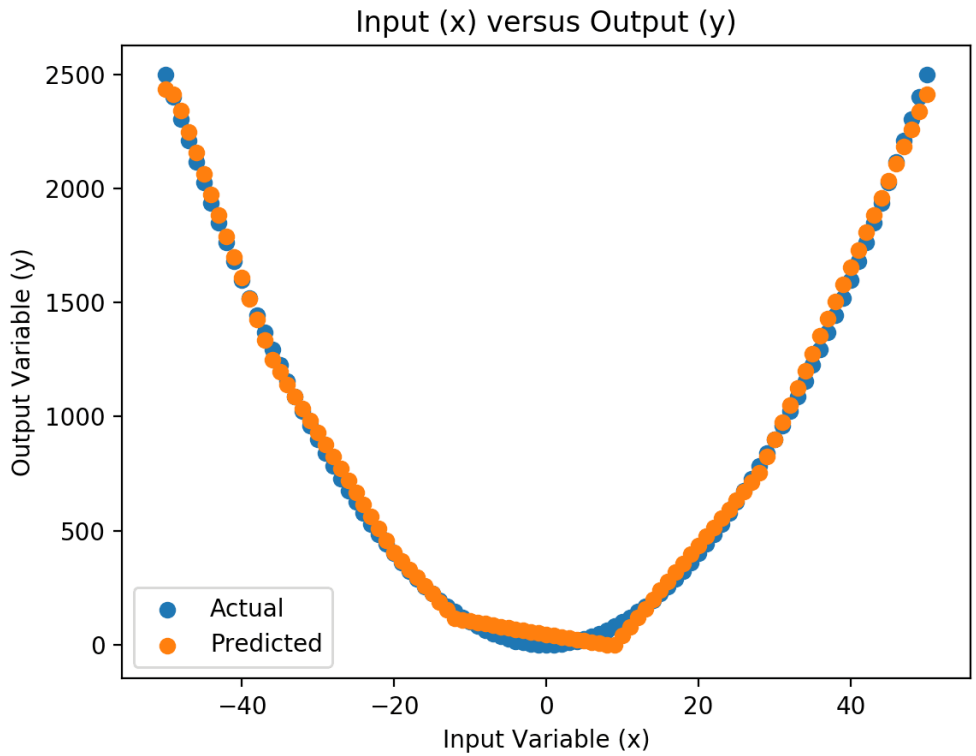

Figure 1: A neural network fitting function $y = x^2$

Function approximation

- Imagine you have a function $y = f(x)$.
  - You need to work with this function, but either you do not know how the function looks like (even though you may have some clue) or you know it, but it is overly complex and unbearable to work with.

- You may know finite set of derivatives of the function at some specific point or possibly finite set of function values.

Function approximation

- With one of the aforementioned info we can approximate our function of interest.
  - Think about polynomials.

$$P_k(x) = \alpha_0 + \alpha_1 * x + \alpha_2 * x^2 + ... + \alpha_k * x^k$$

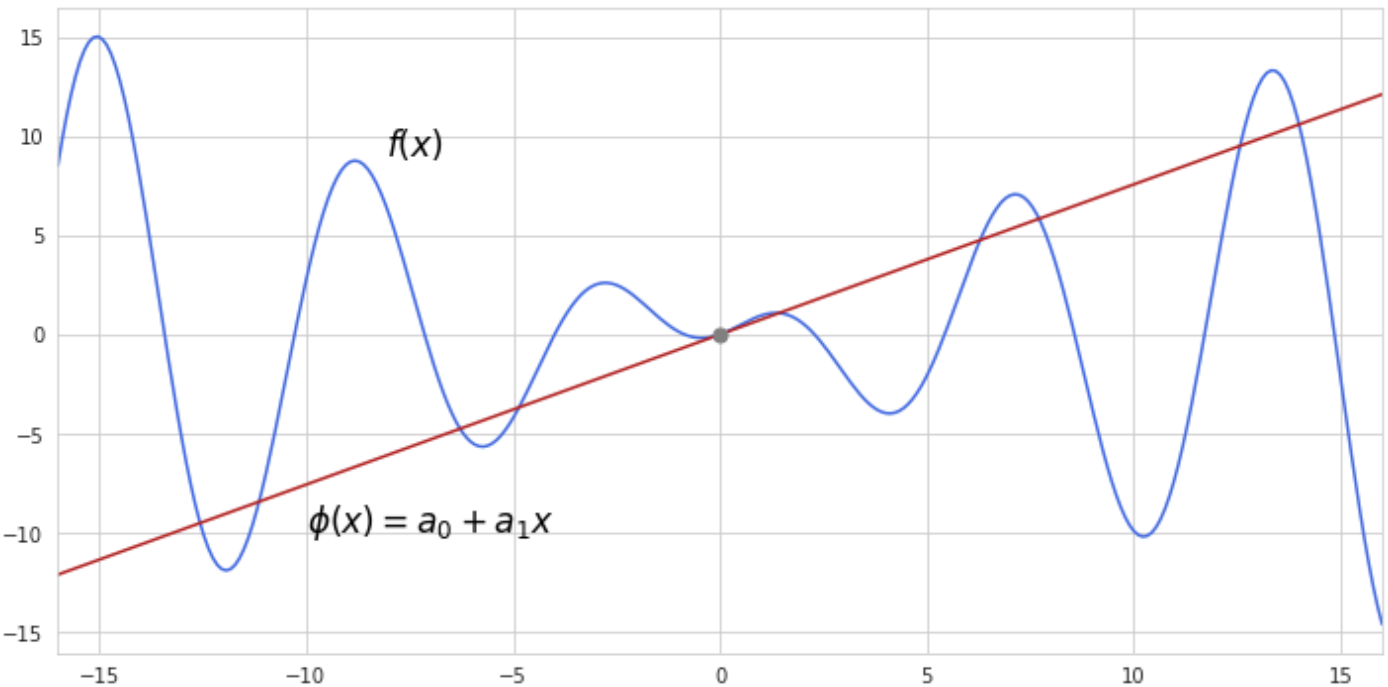

Figure 2: Polynomial approximation of the first degree (linear)

Function approximation

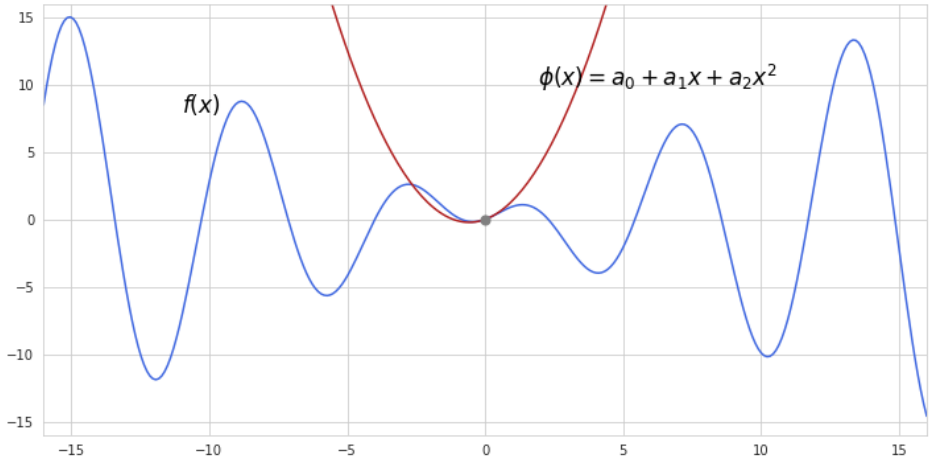

Figure 3: Polynomial approximation of the second degree (quadratic)

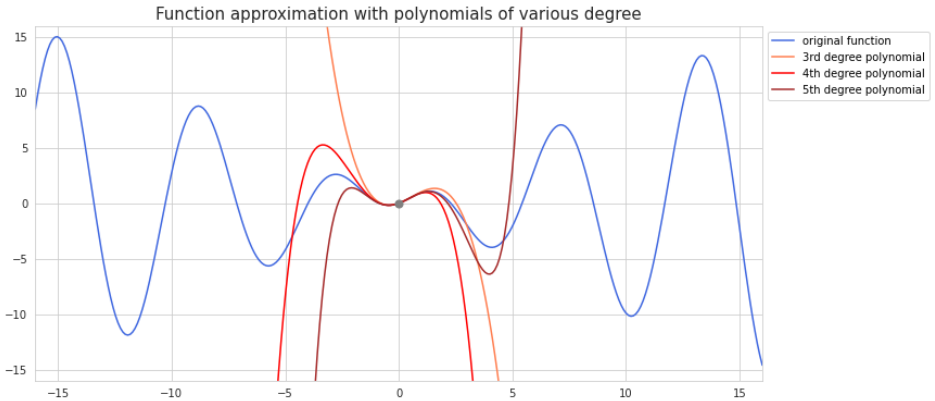

Figure 4: Polynomial approximation of various degrees

Function approximation

- With one of the aforementioned info we can approximate our function of interest.
  - Sometimes it is more suitable to use splines (linear interpolation is the most common example).

$$p_x = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$$

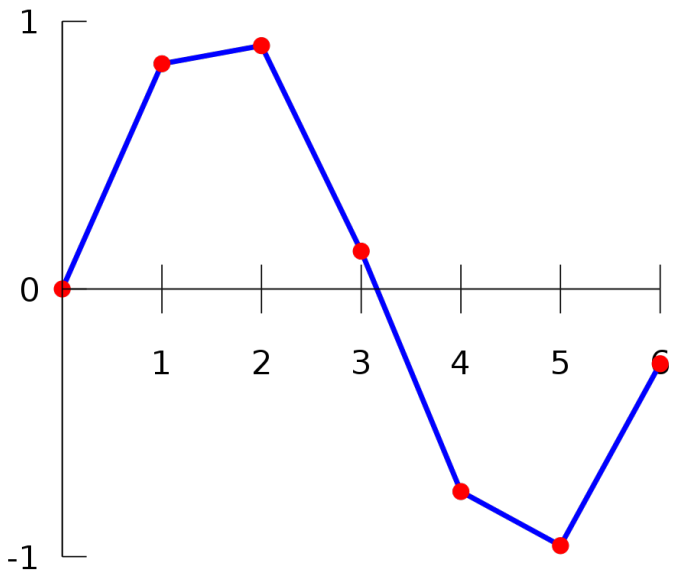

Figure 5: Linear interpolation

Function approximation

- With one of the aforementioned info we can approximate our function of interest.
  - Sometimes it is more suitable to use splines (linear or cubic splines for example).

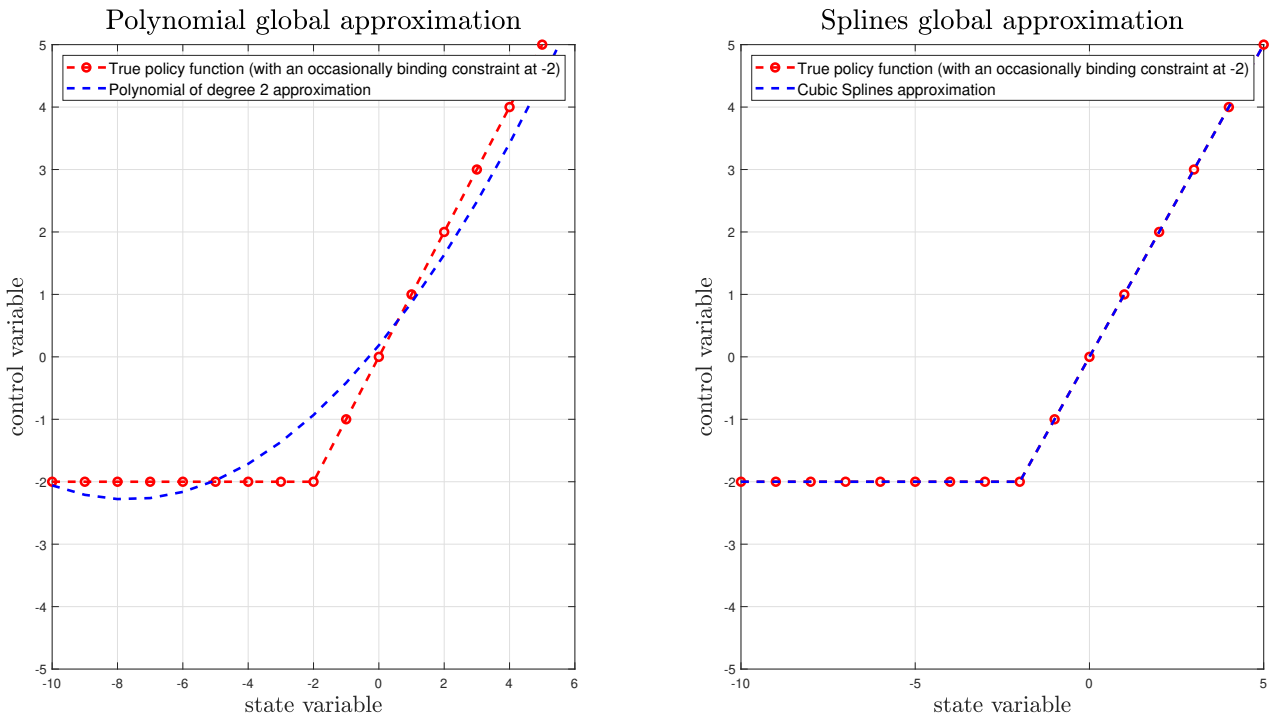

Figure 6: Graphical comparison of global approximation methods.

Function approximation

- Thus, we know that we can try to approximate unknown functions by polynomials or splines.
  - Formally, we have:

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

where $T_i(x)$ is a basis function. In our example above, we use regular polynomials $T_i(x) = x^i$.

Function approximation

- Does that sound like kinda *Wingardium Leviosa* stuff to you?
  - It is a sort of magic, isn't it?

- Why did we need to come up with all this?
  - Remember that we need to work with $y = f(x)$, but we do not know its shape. In other words, y is infinite dimensional object.
  - However, once we use $\sum_{i=0}^{k} \alpha_i T_i(x)$ we have only $k+1$ dimensional object.

Reducing dimensionality

By this procedure, we transferred an infinite dimensional problem into something that can be solved out.

Function approximation

Weierstrass Approximation Theorem

States that any continuous real function defined on a bounded closed interval of real numbers can be approximated uniformly as closely as desired by polynomials.

- A bit of math...
  - Suppose f is a continuous real-valued function defined on the real interval $[a, b]$. For every $\epsilon > 0$ and large enough k, there exists a polynomial $P(x)$ such that for all x in $[a, b]$, we have:

$$|f(x) - P(x)| < \epsilon$$

  - Or equivalently, the supremum norm in case of $\mathbb{R}^2$

$$||f - P|| < \epsilon,$$

where ϵ is discretionary chosen metric.

Function approximation

- What do we need to do practically?
- Choose suitable basis functions $T(x)$.
- Define the order of the approximation k
- Find the values of the coefficients α_i
- Adjust k if the approximation is not good enough

- Remember that we have info about x. These are our state variables.

Function approximation

- Find the values of the coefficients α_i
  - By using info that we know about the function.

- Local approximation
  - Knowing a finite set of derivatives (due to the Implicit function theorem), we can apply the Taylor expansion.

- Global approximation
  - Knowing a finite set of points (nodes) x_i and thus their function values $f(x_i)$, one can use projection methods.

Global approximation

- In this presentation, we are interested mostly in the global approximation.
- Let's dig into projection methods. About perturbation techniques look at my other presentations.

General intro into projection methods

A general introductory example

- Consider a simple ordinary differential equation of the form:

$$f'(x) + x = 0, \quad f(0) = 1$$

- Easy for us to find an exact solution:

$$f(x) = e^{-x}$$

- But let's pretend we are not capable of finding it. Hence, we have to solve it at least approximately.
  - We use monomials $(1, x, x^2, x^3)$ and approximate the equation within an interval $x \in [0, 3]$.
- Specifically, our approximate function (approximant) is now:

$$\hat{f}_x = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3,$$

where $\alpha_0 = 1$ to satisfy the boundary conditions $f(0) = 1$.

A general introductory toy example

- The whole problem boils down to choosing the values of α_i for $i = 1, 2, 3$.
  - Does it resemble something? Econometricians know.
- In the following step, we define a residual function:

$$R(\alpha, x) := \underbrace{\alpha_1 + 2\alpha_2 x + 3\alpha_3 x^2}_{f'(x)} + \underbrace{1 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3}_{f(x)},$$

  - which defines the error of the approximated solution relative towards the true one.
- The problem is thus to find α_i s.t. $R(\alpha, x) \approx 0$. Formally:

$$\min_{\alpha_1, \alpha_2, \alpha_3} \int_0^3 R(\alpha, x)^2 \, dx$$

A general introductory toy example

- To be able to well-enough approximate $f(x)$ by $\hat{f}(x)$, one need to choose the projection conditions on the residual function (hence the name of the method) that must be satisfied. We can pick one of the following methods:
  - Least-square projection
  - Galerkin method
  - Collocation method
- In this presentation, I profoundly describe the method of collocation. For the remaining two, see my other presentations or different sources.

Collocation method shown on the toy example

- Let's start from our simple case to depict the idea. Coming back to our residual function:

$$R(\alpha, x) := \alpha_1 + 2\alpha_2 x + 3\alpha_3 x^2 + 1 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3$$

- The collocation method sets the residual function equal to zero at a given set of points. In our case, we will assume X = {1,2,3}. This leads to the following linear system:

$$R(1, x_1) := 0 = 1 + 2\alpha_1 + 3\alpha_2 + 4\alpha_3$$

$$R(2, x_2) := 0 = 1 + 3\alpha_1 + 8\alpha_2 + 20\alpha_3$$

$$R(3, x_3) := 0 = 1 + 4\alpha_1 + 15\alpha_2 + 54\alpha_3$$

- which is simple to solve and thus obtain α_i.

Collocation method shown on the toy example

- Better to try to solve the system on your laptop.

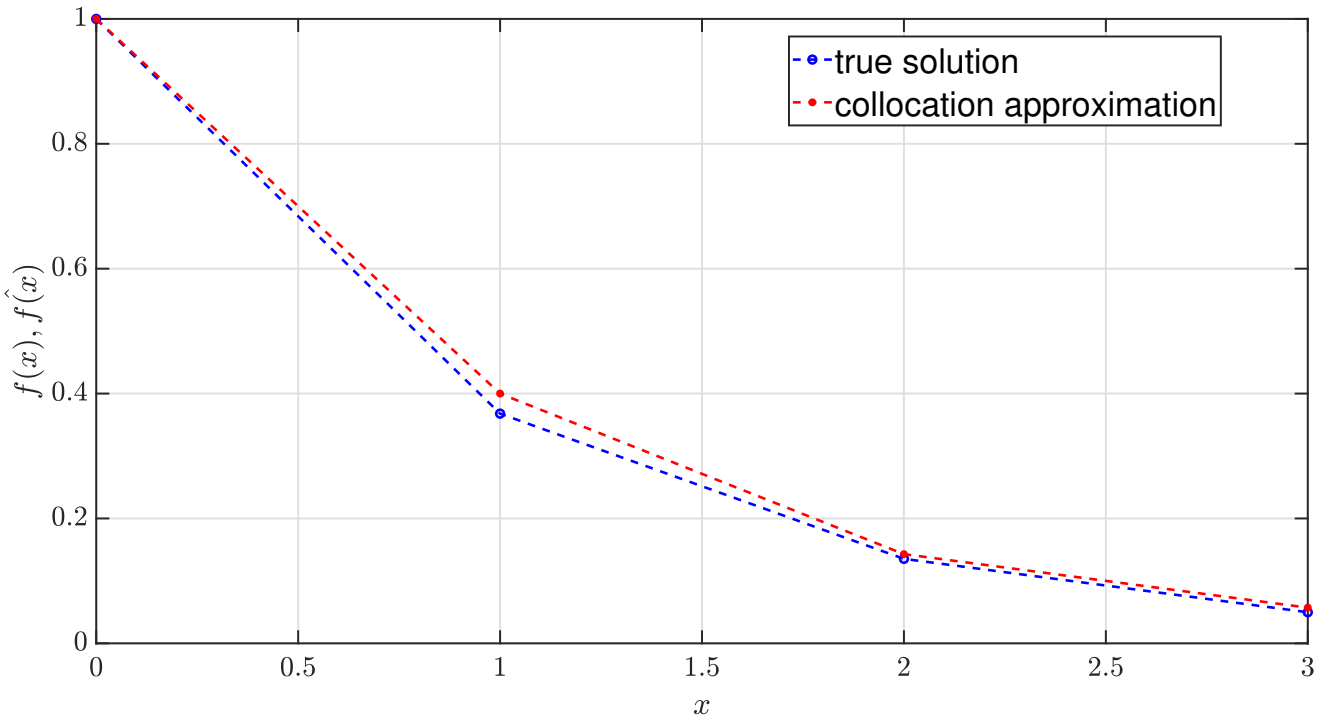

Figure 7: Collocation approximation of an ODE
$f'(x) + x = 0, \quad f(0) = 1$

- You have just solved the very first projection problem on your own.

Collocation method generally

- Expressing the collocation problem generally, let me consider uniform-collocation projection method. Imagine we have a grid of points D. Then we want to satisfy R(α,x_i)=0 at all $x_j \in D$.
- Denote δ Dirac delta function, one can write:

$$\langle R(\alpha, x), \delta(x - x_j) \rangle = 0$$

Thus, generic description of the collocation projections is following:

- Express an approximant as a linear combination of known basis functions.
- Fix the basis function coefficients to satisfy R(α,x) = 0 at the collocation nodes.
- Now we have a system of n nonlinear equations in n unknowns.
- This problem may be solved using any nonlinear equation solution method. For example, one can deploy function iteration or rewrite it as a root finding problem and use Newton's method.

General procedure

- Switching into a general cookbook, we use the following procedure when using the projection methods to solve a model.
- Imagine we approximate an unknown function $f : X \longrightarrow Y$, where $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$.
- The function is defined implicitly as a functional equation $F(f) = 0$ where F denotes an operator mapping between function spaces. Thus, $F : C_1 \longrightarrow C_2$.
  - What about allowing for some econ intuition?
  - F can be a system of necessary first-order conditions and market-clearing equations or a Bellman equation of a dynamic programming problem.
  - f is the vector of equilibrium policy and price functions defined on a state space or the value function of the dynamic programming problem.

General procedure

- Our goal is to solve for an unknown decision rule f. Projection method solution is about approximating f by a finite linear combination of the first p+1 members of some family of polynomials $\mathcal{P}$:

$$\hat{f}(\mathrm{x}) = \sum_{i=0}^{p} \alpha_i \varphi_i(\mathrm{x}), \quad \mathrm{x} \in X \subset \mathbb{R}^n$$

- Then the residual function is as follows:

$$R(\alpha, \mathrm{x}) := F\left(\hat{f}(\alpha, \mathrm{x})\right), \quad \alpha := (\alpha_0, ..., \alpha_p)$$

- Our problem is to minimize the residual function on a given state space (aka find the value of R small enough to satisfy our discretionary chosen metric).

General procedure

- Equivalently to our specific toy example, we solve:

$$\min_{\alpha} \int_X R(\alpha, x)^2 \, dx$$

  - To be completely accurate, we first have to choose a projection function g_i, a weighting function w, and compute the inner product below:

$$G_i := \int_X w(\mathrm{x}) R(\alpha, \mathrm{x}) g_i(\mathrm{x}) \, dx \quad \forall i, \tag{1}$$

  - where g_i and $w(\mathrm{x})$ may differ depending if we use the least squares solution, the Galerkin method, or the collocation method.
  - In our case, we deploy the collocation. Therefore, we have $g_i = 1$ and the Dirac delta function as a weighting function ($w(\mathrm{x}) = 1$ if $x = x_i$ and 0 otherwise).

General procedure

- The collocation strategy replaces the functional equation with a system of p nonlinear equations in p unknowns. What comes next, the nonlinear equation system $\sum_{i=0}^{p} \alpha_i \varphi_i(x_j)$ with $x_j \in X$ may be expressed using a vector form as a collocation equation:

$$\mathfrak{p}\alpha = \mathfrak{f}(\alpha),$$

- where $\mathfrak{p}$ is $p \times p$ matrix with ji^{th} element denoting i^{th} basis function evaluated at the j^{th} collocation node. Mathematically:

$$\mathfrak{p}_{ji} = \varphi_i(x_j),$$

- while $\mathfrak{f}$ is the collocation function evaluated at a particular vector of basis coefficients α delivering a vector with j^{th} element denoting the value given by solving the optimization problem embedded in F(f) at each of j^{th} collocation node.

General procedure

- This procedure results in replacing f representing e.g. equilibrium policy with the approximant $\sum_{i=0} \alpha_i \varphi_i$.
- In order to solve for α in the collocation equation, one can use whatever nonlinear equation solution method.
  - For example, the collocation equation may be reshuffled into a fixed-point problem as follows:

    $$\alpha = \mathfrak{p}^{-1}\mathfrak{f}(\alpha)$$

    and solved by function iteration applying the iterative update rule:

    $$\alpha \longleftarrow \mathfrak{p}^{-1}\mathfrak{f}(\alpha)$$

  - Alternatively, it is straightforward to write the collocation equation as a roof-finding problem $\mathfrak{p}\alpha - \mathfrak{f}(\alpha) = 0$ and solve for α using Newton's method:

    $$\alpha \longleftarrow \alpha - \left[\mathfrak{p} - \mathfrak{f}'(\alpha)\right]^{-1}\left[\mathfrak{p}\alpha - \mathfrak{f}(\alpha)\right],$$

    where $\mathfrak{f}'(\alpha)$ is $p \times p$ Jacobian of the collocation function at α.

Basis functions

Choice of basis functions

- We said that the projection method solution is about approximating f by a finite linear combination of the first p+1 members of some family of polynomials $\mathcal{P}$.
- However, so far we have been unforgivably vague about $\mathcal{P}$.
- Firstly, we have to clarify the difference between two approaches often used in function approximation. Meet spectral methods and finite element method.

- Spectral approximation
  - High-degree polynomials are fitted on the entire state space (e.g. Chebyshev polynomials).
- Finite element approximation
  - Low-degree polynomials are fitted over non-overlapping subdomains of the state space (e.g. splines).

Spectral methods

Spectral approximation

When working with an approximant $\hat{f}(\alpha, \mathrm{x})$ using the spectral methods, each parameter in vector α affects $\hat{f}(\alpha, \mathrm{x})$ for all $x \in X$

- So far we have relied on monomial basis which just consists of the simple power function $1, x^2, x^3, ...x^n$ even though it may not be preferable option in most cases. Specifically, they suffer from two serious problems:
  - Monomials are (nearly) multicollinear.
  - Monomials vary considerably in size (accumulation of numerical errors).

Spectral methods

- Monomials are (nearly) multicollinear.

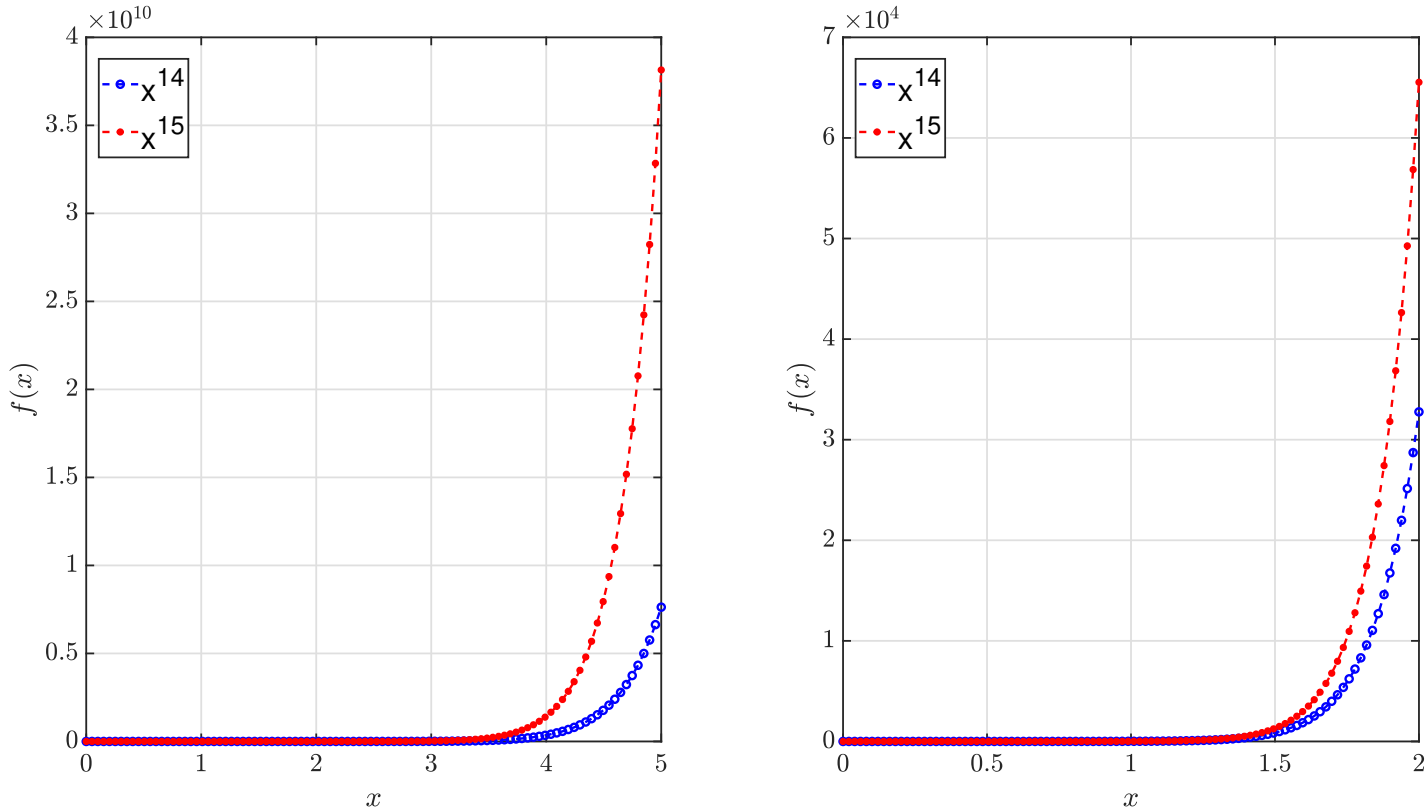

Figure 8: Monomials of x^{14} and x^{15} for $x \in [0, 5]$ and $x \in [0, 2]$

Spectral methods

- Monomials are (nearly) multicollinear.
- We have a motivation to use orthogonal polynomials.
  - The basis functions are by construction orthogonal with respect to a certain measure.
  - Orthogonality means that when we add more elements of the basis, the new ones help us to deliver a more satisfactory shape in order to capture our unknown function.
  - Formally, considering a range [a,b] and a particular weighting function (density) w we can write:

$$\int_a^b T_i(x)\,T_j(x)w(x)dx = 0 \quad \forall i,j \ni i \neq j$$

- Chebyshev polynomials are often used in economics.

Spectral methods

- Chebyshev polynomials are often used in economics.
  - They are defined on interval [-1,1] and the weighting function has the following form:

$$w(x) = \frac{1}{\sqrt{1-x^2}},$$

  - where $x = 2\frac{z-a}{b-a} - 1$ to normalize the domain to the interval [-1,1] as mentioned above for some given arbitrary interval $z \in [a, b]$.
- The basis functions are defined as:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2xT_1(x) - T_0x = 2x^2 - 1$$

$$T_3(x) = 2xT_2(x) - T_1x = 4x^3 - 3x$$

$$\dots$$

$$T_{i+1}(x) = 2xT_i(x) - T_{i-1}x = 4x^3 - 3x,$$

Spectral methods

- Orthogonality of Chebyshev polynomials with respect to $w(x) = \frac{1}{\sqrt{1-x^2}}$ is the reason why they are so useful in various function approximation problems.

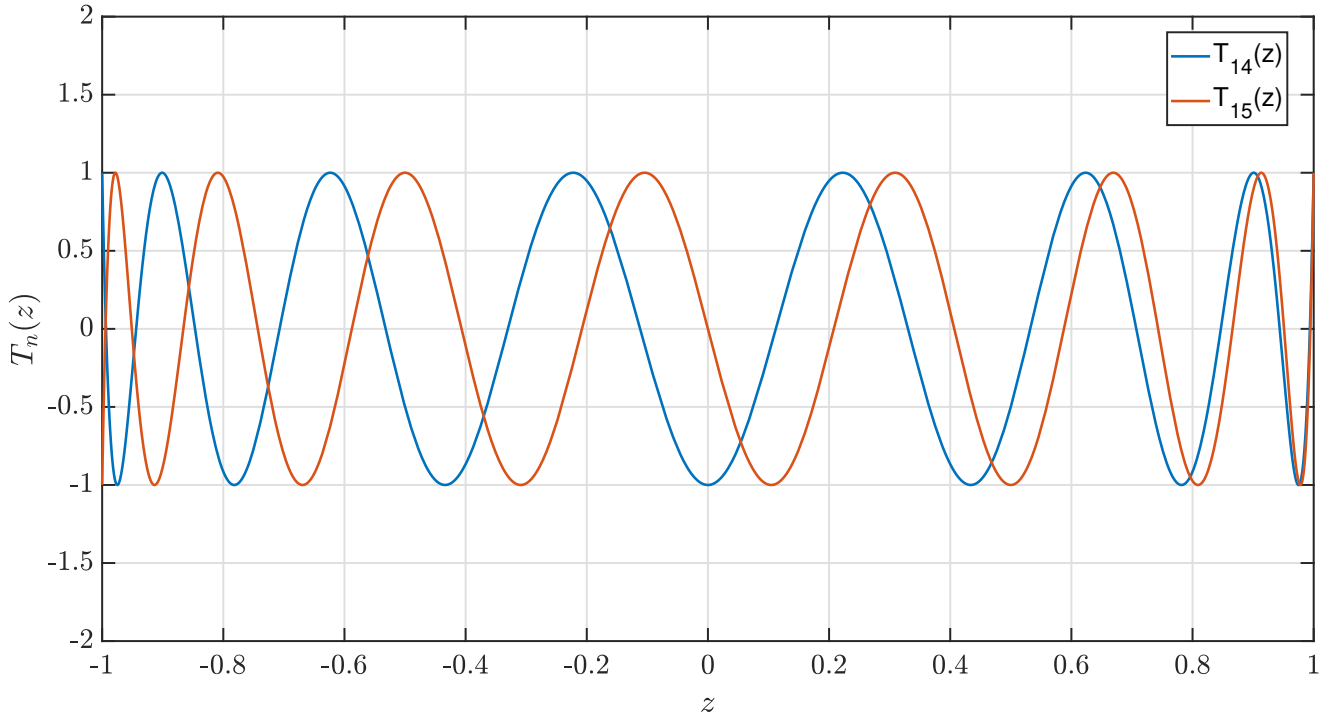

Figure 9: Chebyshev polynomials for $T_{14}(x)$ and $T_{15}(x)$

Spectral methods

- Orthogonality of Chebyshev polynomials with respect to $w(x) = \frac{1}{\sqrt{1-x^2}}$ is the reason why they are so useful in various function approximation problems.

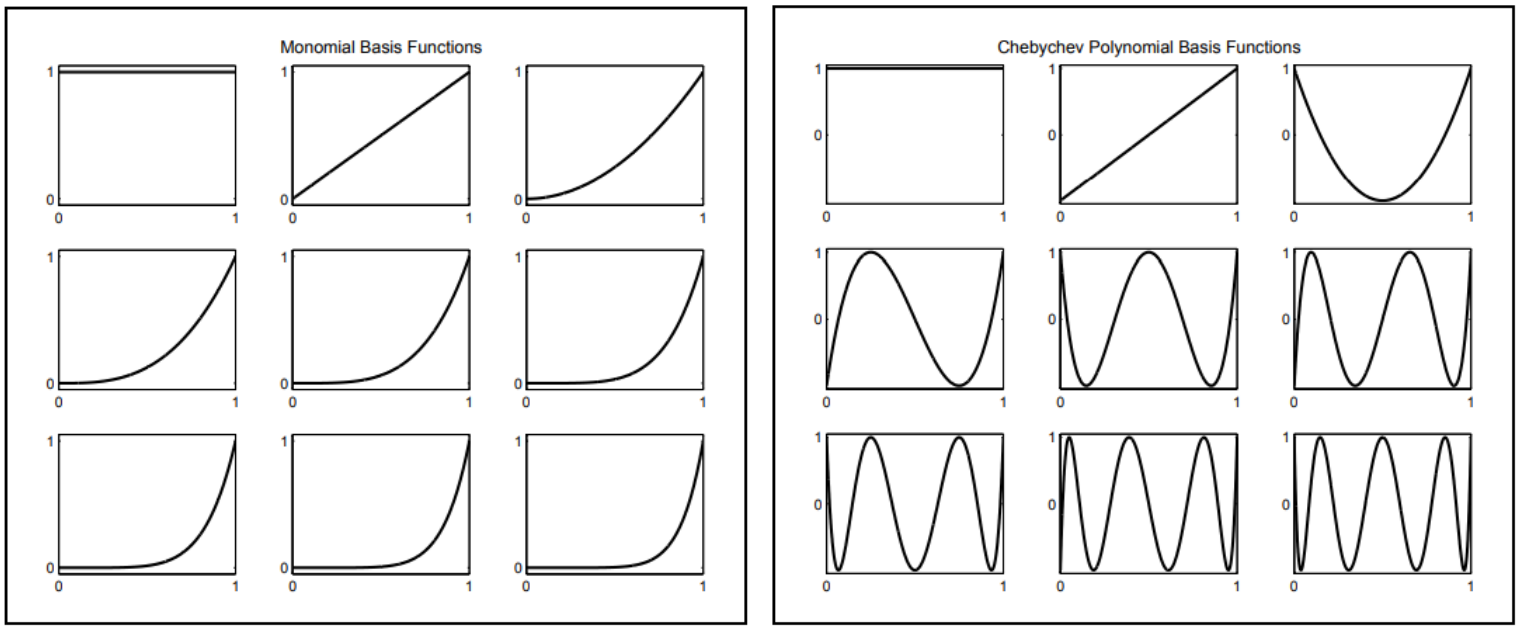

Figure 10: First nine Monomials and Chebyshev polynomials (see Miranda and Fackler, 2004).

Spectral methods

- Spectral methods may not be the best choice if we have some weird local behavior in the state space.

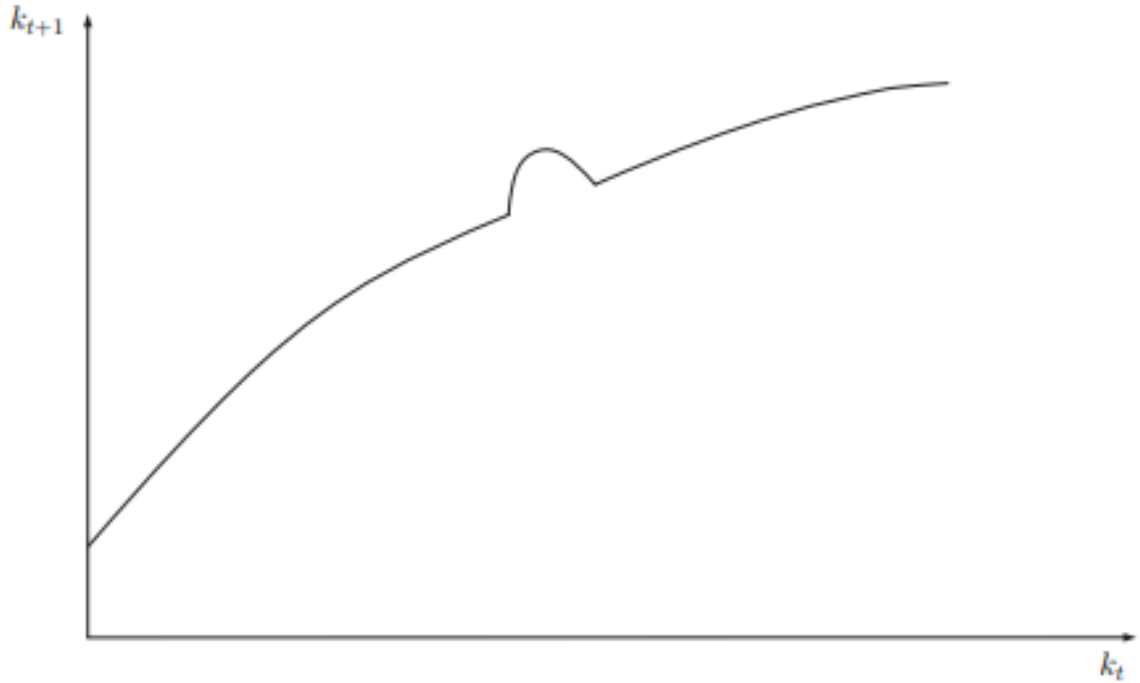

Figure 11: Decision Rule for Capital with a locally hump-shaped behavior. See Fernández-Villaverde, Rubio Ramírez, and Schorfheide (2016).

Spectral methods

- Spectral methods may not be the best choice if we have some weird local behavior in the state space.
  - Think about zero/effective lower bound. We have already seen the picture below.

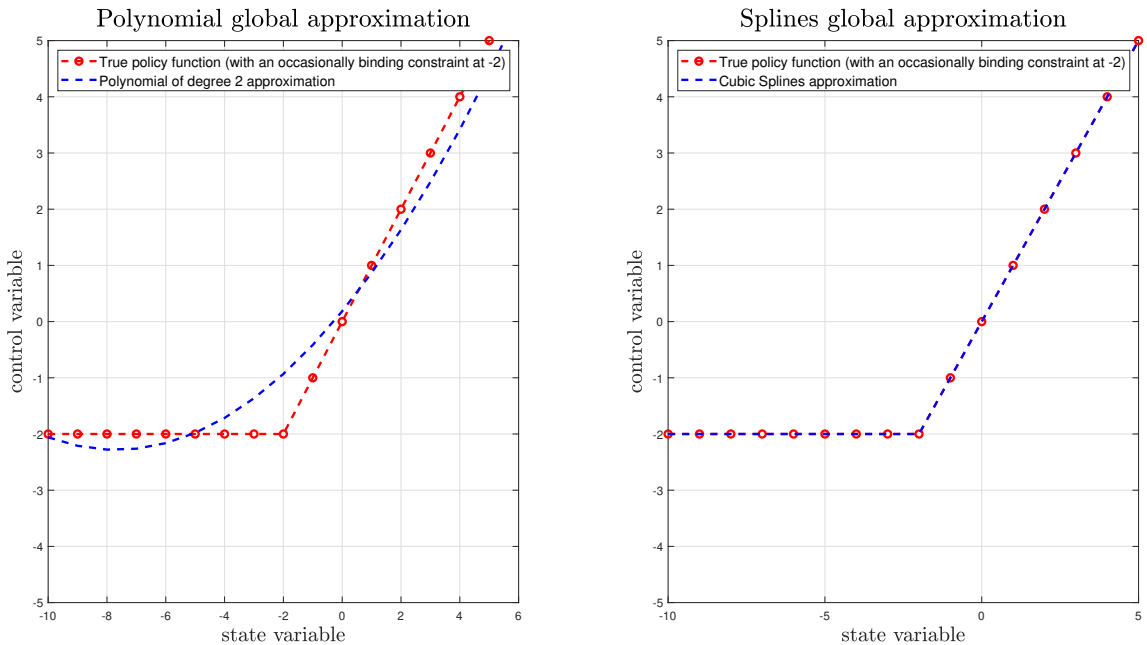

Figure 12: Graphical comparison of spectral and finite element approach.

Finite element methods

Finite element approximation

When working with an approximant $\hat{f}(\alpha, x)$ using the finite element approach, each parameter in vector α affects $\hat{f}(\alpha, x)$ only across a given subinterval of $x \in X$

- Hence, assuming that we split the state space into I subdomains, we have:

$$\alpha = [\alpha_1 \quad \alpha_2 \quad ... \quad \alpha_{I+1}],$$

- and we approximate each subdomain using a low-degree polynomials.
- This approach is called spline approximation.

Finite element methods

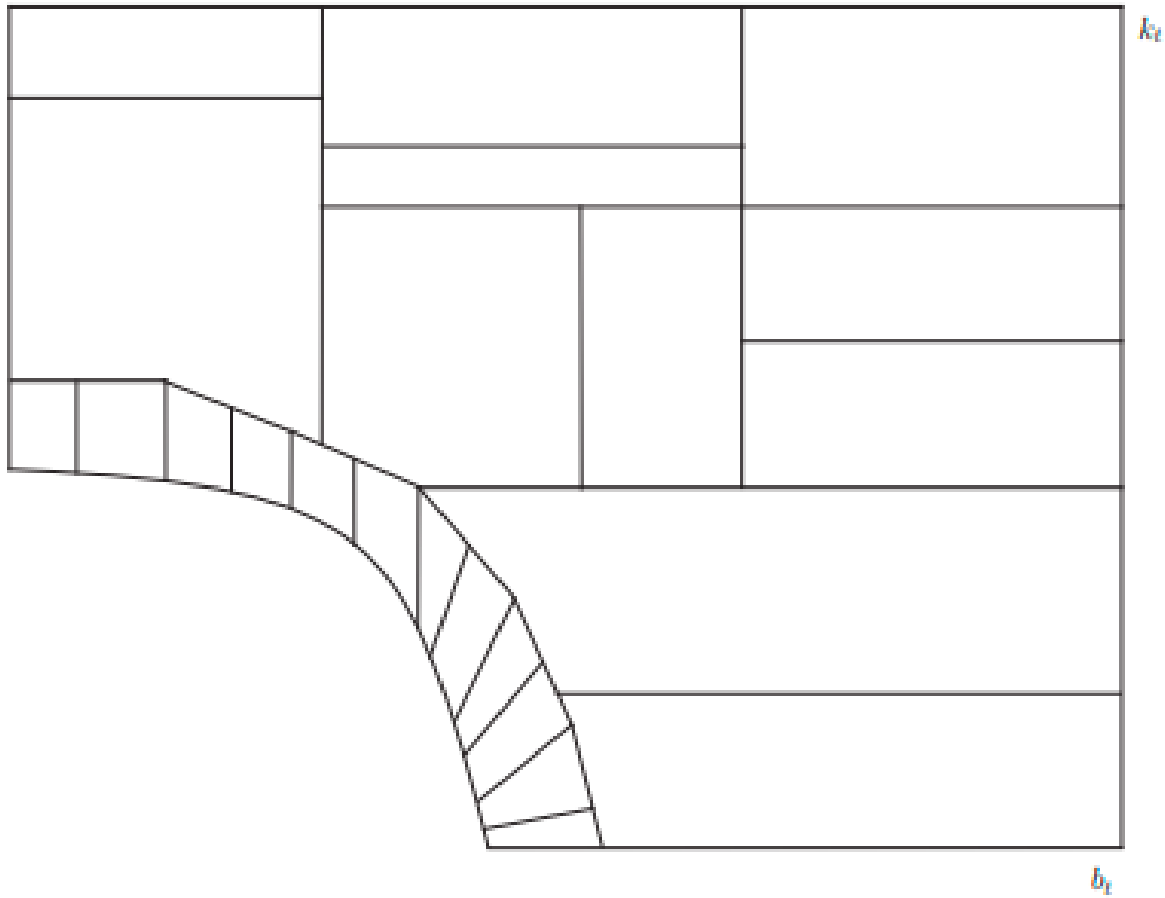

Figure 13: An example of cutting grid in a finite element method. Source Fernández-Villaverde, Rubio Ramírez, and Schorfheide (2016).

Finite element methods

- Splines may be expressed as a linear combination of basis functions. Yet they are not polynomials themselves.
  - Across most of the domain, basis functions are zero.
- The most straightforward approach is a piecewise linear. Recall:

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

- for $x \in [x_i, x_{i+1}]$. This expression is equivalent to the one we saw at the very beginning.
- By definition, linear splines suffer from non-differentiability at nodes.

Finite element methods

- The most straightforward approach is a piecewise linear.

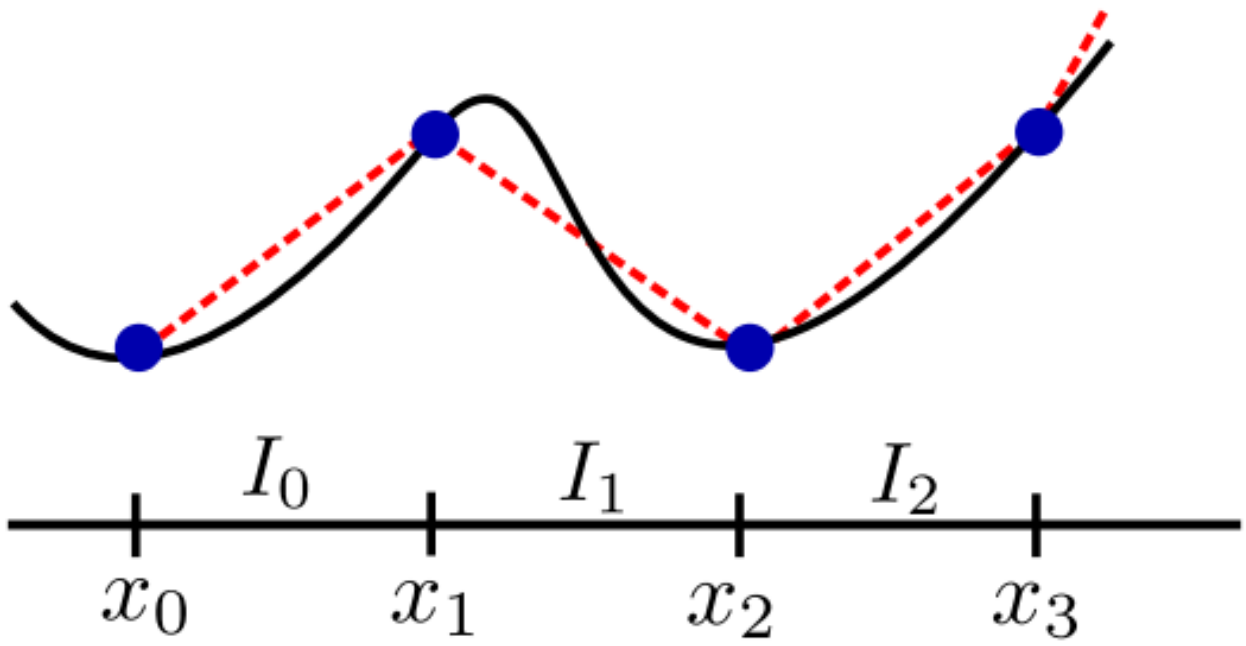

Figure 14: An example of linear splines.

Finite element methods

- Another option may be using higher order polynomials. The most often used is cubic splines.
  - Within each subdomain that we split up the state space into, we fit the third degree polynomial.
- In cubic splines approximation, we have:

$$S_1(x) = y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \quad \forall x \in [x_1, x_2]$$

$$S_2(x) = y_2 + b_2(x - x_1) + c_2(x - x_1)^2 + d_2(x - x_1)^3 \quad \forall x \in [x_2, x_3]$$

$$\dots$$

$$S_{n-1}(x) = y_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3$$

$$\forall x \in [x_{n-1}, x_n]$$

- Hence, we have 3(n-1) unknowns to choose in order to satisfy the interpolation and smoothness constraints.

Finite element methods

- Hence, we have 3(n-1) unknowns to choose in order to satisfy the interpolation and smoothness constraints. Let's disentangle it specifically.
  - Interpolation conditions. We want the continuous functions. Thus, $S_i(x_{i+1}) = S_{i+1}(x_{i+1}) = y_{i+1}$ holds $\forall$ i =1,2,...,n-2. On top of that, we have $S_{n-1}(x_n) = y_n$. Therefore, in total we get n-1 constraints here.
  - Smoothness conditions. To satisfy smoothness conditions, we need the following to hold at the interior points: $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ and $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$. This gives us 2(n-2) constraints.
- Sum it up, we end up with 3n - 5 constraints so far. But remember that we have 3n-3 coefficients to estimate. To define the problem uniquely, we need to add two constraints.

Finite element methods

- Sum it up, we end up with 3n - 5 constraints so far. But remember that we have 3n-3 coefficients to estimate. To define the problem uniquely, we need to add two constraints.
- We can use different approaches. Mostly, the ones below are common:
  - The natural splines. $S_1''(x_1) = 0 = S_{n-1}''(x_n)$.
  - The clamped splines. You define $S_1'(x_1)$ and $S_{n-1}'(x_n)$.
  - Not-A-Knot. $S_1'''(x_2) = S_1'''(x_2)$ and $S_{n-2}'''(x_{n-2}) = S_{n-1}'''(x_{n-2})$.
- So we have different cubic splines given our choice of the two additional constraints.

Finite element methods

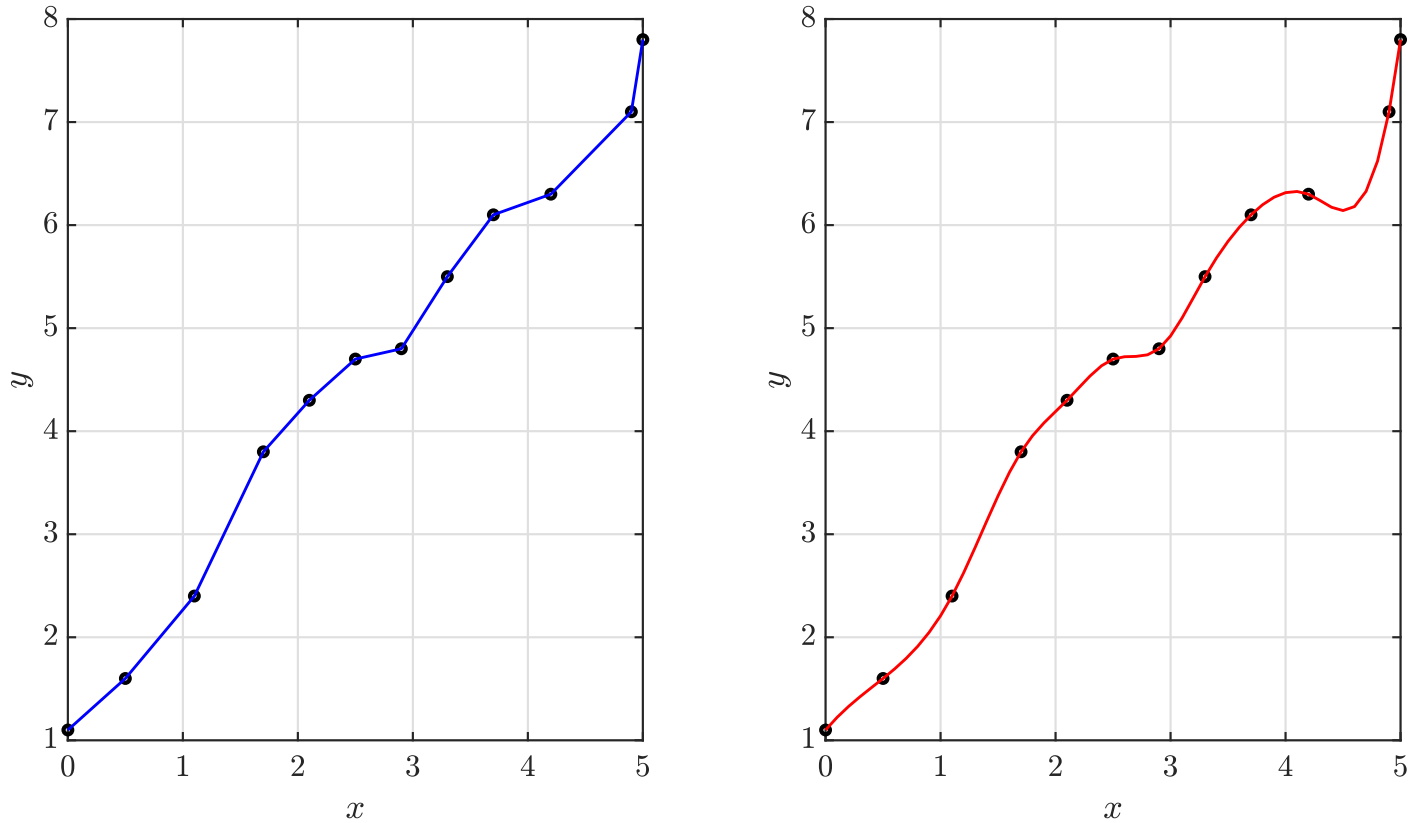

Figure 15: A comparison of linear and cubic splines.

Finite element methods

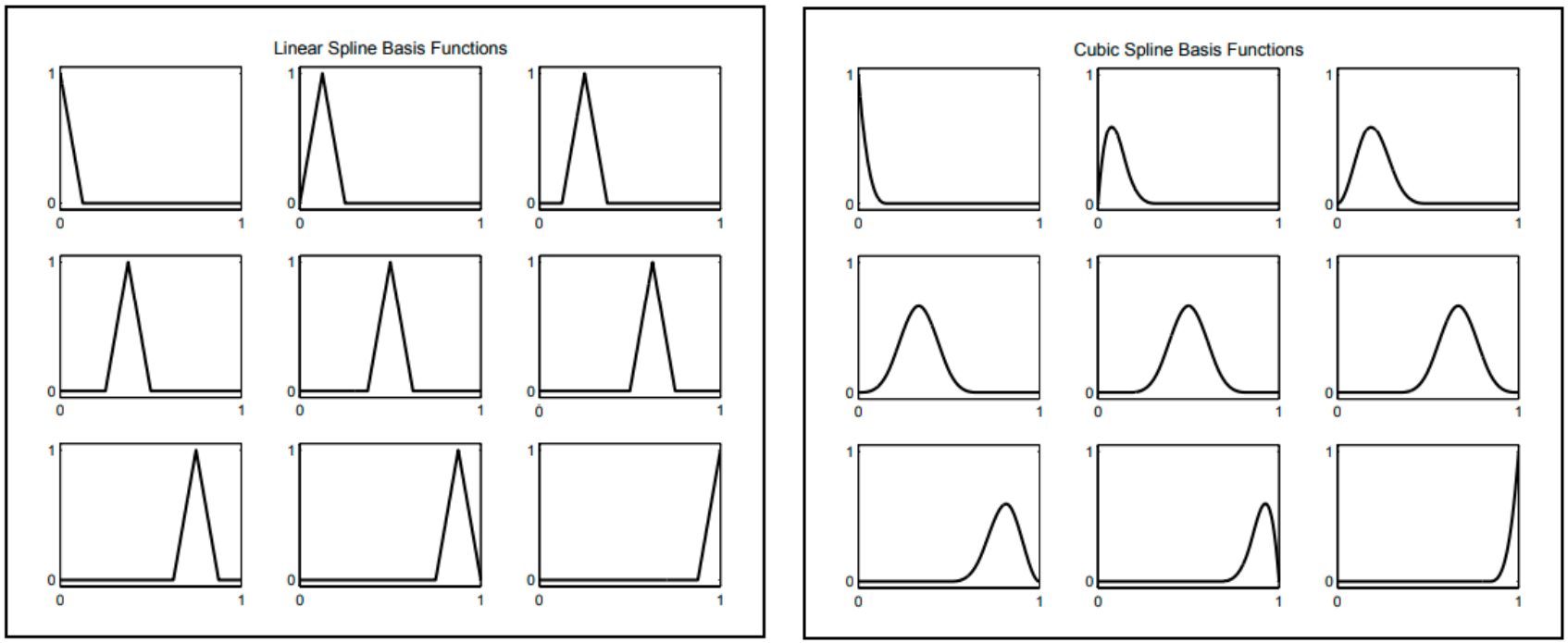

Figure 16: A comparison of linear and cubic splines. See Miranda and Fackler (2004).

How to set a grid

Equidistant grid

- Until now we always implicitly assumed that we work with an equidistant grid.
- If x is a state variable with a range $x_j \in [a, b]$, nodes in the equidistant grid are simply:

$$x_j = a + \frac{j-1}{n-1}\Big(b - a\Big), \quad \forall j = 1, 2, ..., n$$

- for n grid points.
- Anyway, such a setup is a terrible idea!

Equidistant grid

- Why not using the equidistant grid for a polynomial approximation? Runge's phenomenon (polynomial wiggle).
  - The higher degree polynomial we use, the more weird behavior starts to happen between grid points. Especially wild oscillations in tails are common.
- We gotta be more careful about the nodes. Weierstrass' theorem tells us that there exists a uniformly converging polynomial approximation. But to find it, we need to do some work with the grid.

Equidistant grid

- Consider the Runge function:

$$f(x) = \frac{1}{1 + 25x^2},$$

- and try to approximate it by polynomials of n degree using the equidistant nodes.

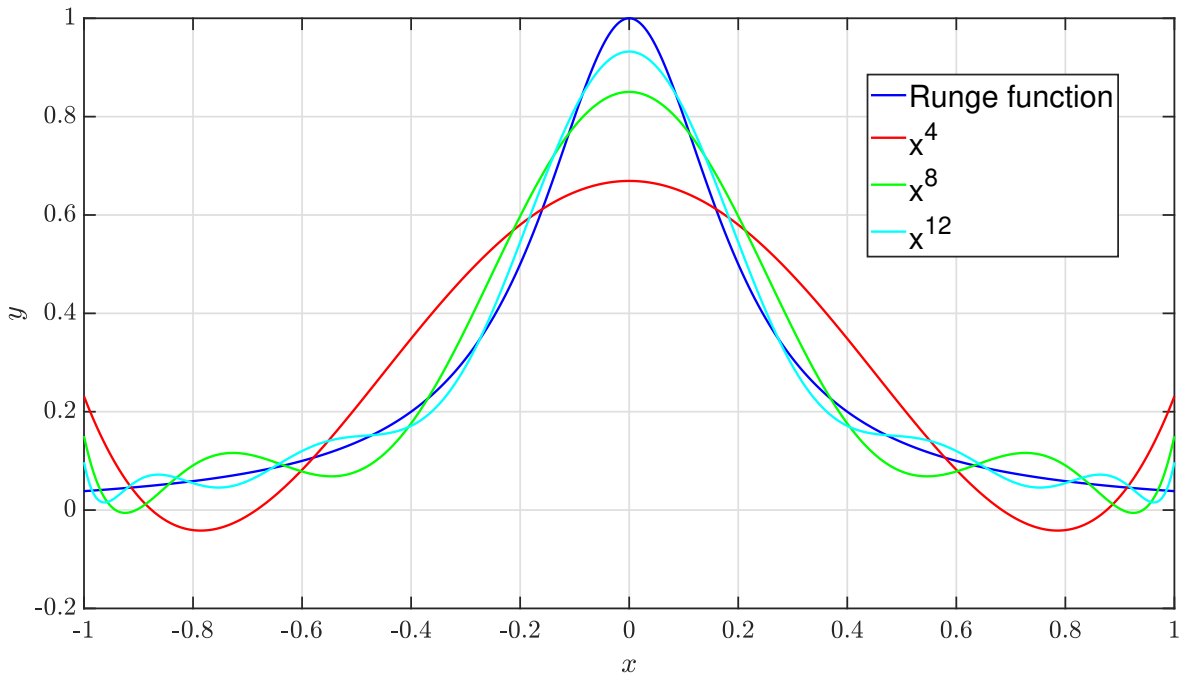

Figure 17: Runge's phenomenon.

Equidistant grid

- The higher degree polynomial we use, the more weird behavior starts to happen between grid points. Especially wild oscillations in tails are common.

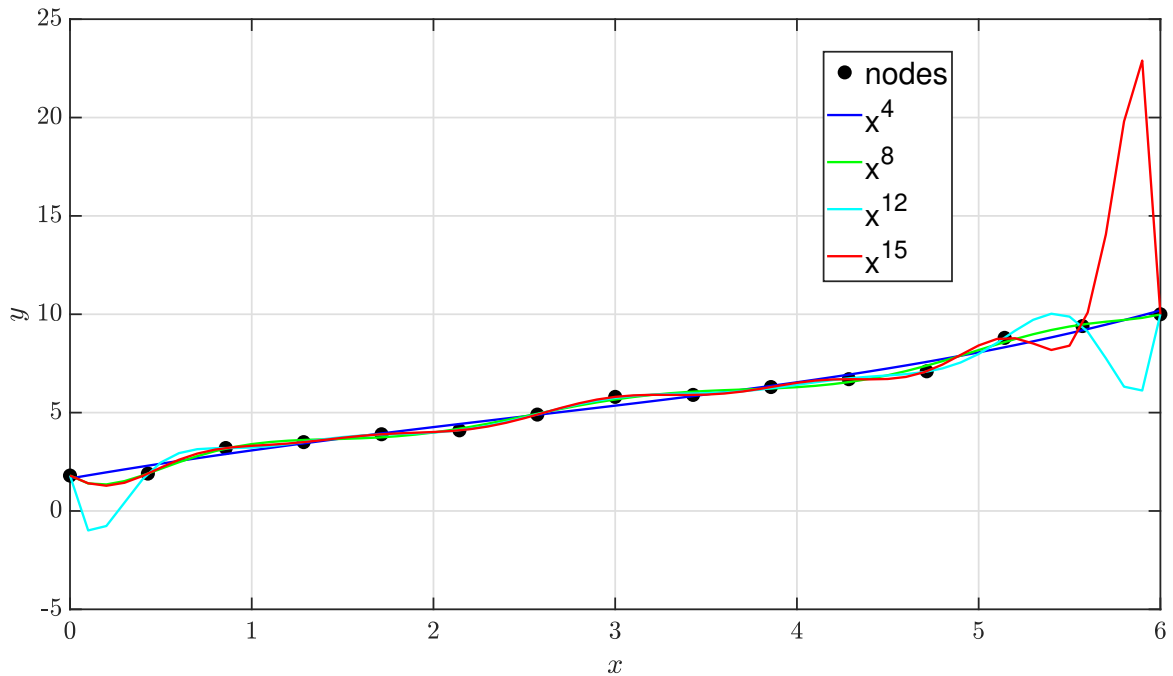

Figure 18: Swing between nodes in tails.

Chebyshev nodes

- Strongly preferred option is to use Chebyshev nodes.
  - Chebyshev nodes are the values of x for which basis functions equal to zero. Hence, for $T_2 = 2x^2 - 1$ we get $-\sqrt{1/2}$ and $+\sqrt{1/2}$.
- Fitting polynomials at the Chebyshev nodes delivers an uniform convergence.
  - Aka it rules out the problems with swing in tails (minimizes the effect of Runge's phenomenon, to be more accurate).

Chebyshev nodes

- Consider a given integer n representing the number of grid points and an iterator j = 1,2,...,n. Then within the interval [-1,1], we can write the Chebyshev nodes as:

$$x_j = cos\left(\frac{2j-1}{2n}\pi\right)$$

- Regarding an arbitrary interval $[a, b]$, we can transform it into Chebyshev nodes as:

$$x_j = \frac{1}{2}\Big(a+b\Big) + \frac{1}{2}\Big(b-a\Big)cos\left(\frac{2j-1}{2n}\pi\right)$$

Chebyshev nodes

- Fitting polynomials at the Chebyshev nodes delivers an uniform convergence.
  - Aka it rules out the problems with swing in tails (minimizes the effect of Runge's phenomenon, to be more accurate).

Chebyshev nodes are more spread towards the boundaries

The reason behind is due to the fact that the Chebyshev nodes are not evenly spaced. Instead they are more spread towards the boundaries of the interval (and less in the center of it).

Chebyshev nodes

- Chebyshev nodes are more spread towards the boundaries.

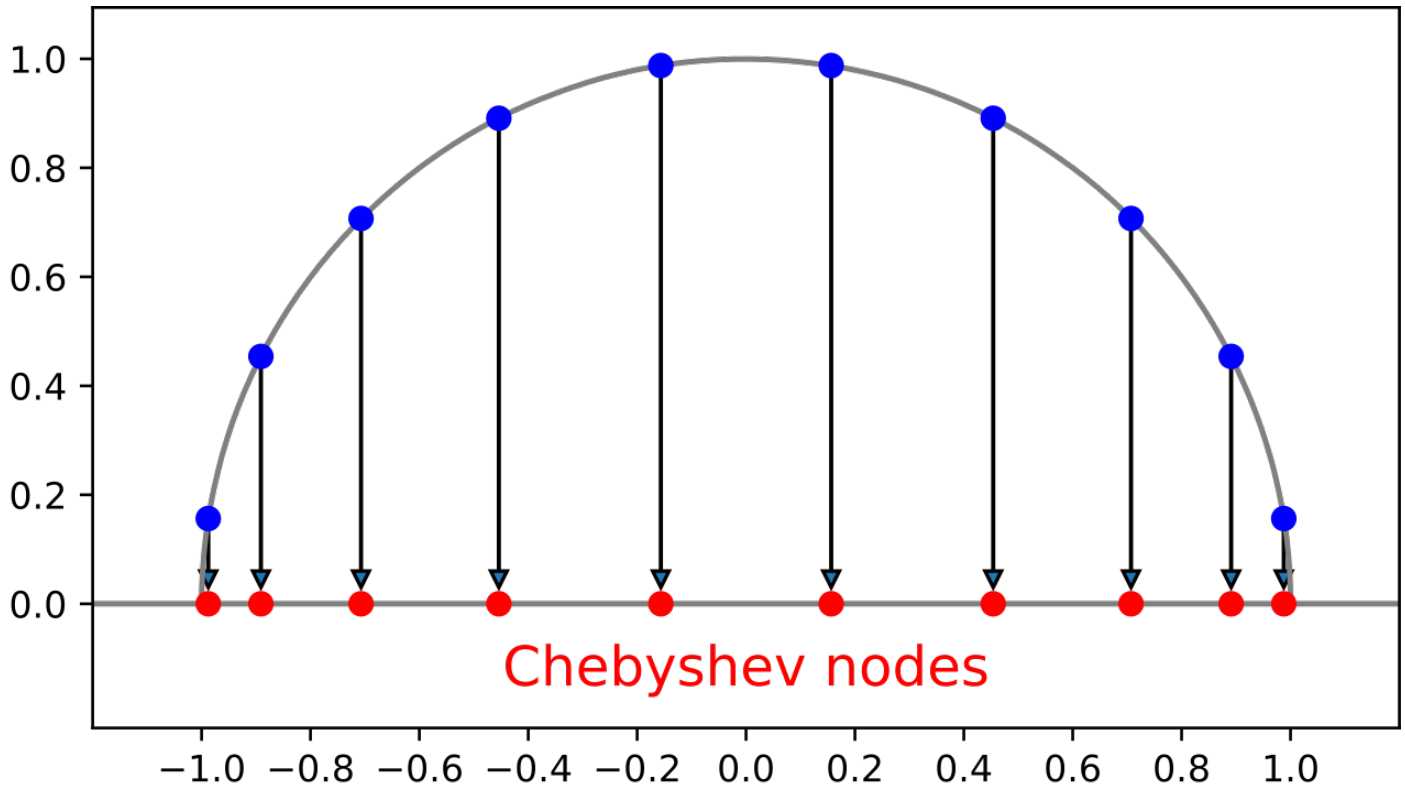

Figure 19: The Chebyshev nodes are equivalent to the coordinates of the evenly spaced grid for the case of a unit semicircle.

Chebyshev nodes

- Consider the Runge function:

$$f(x) = \frac{1}{1 + 25x^2},$$

- and this time try to approximate it by polynomials of n degree using the Chebyshev nodes.

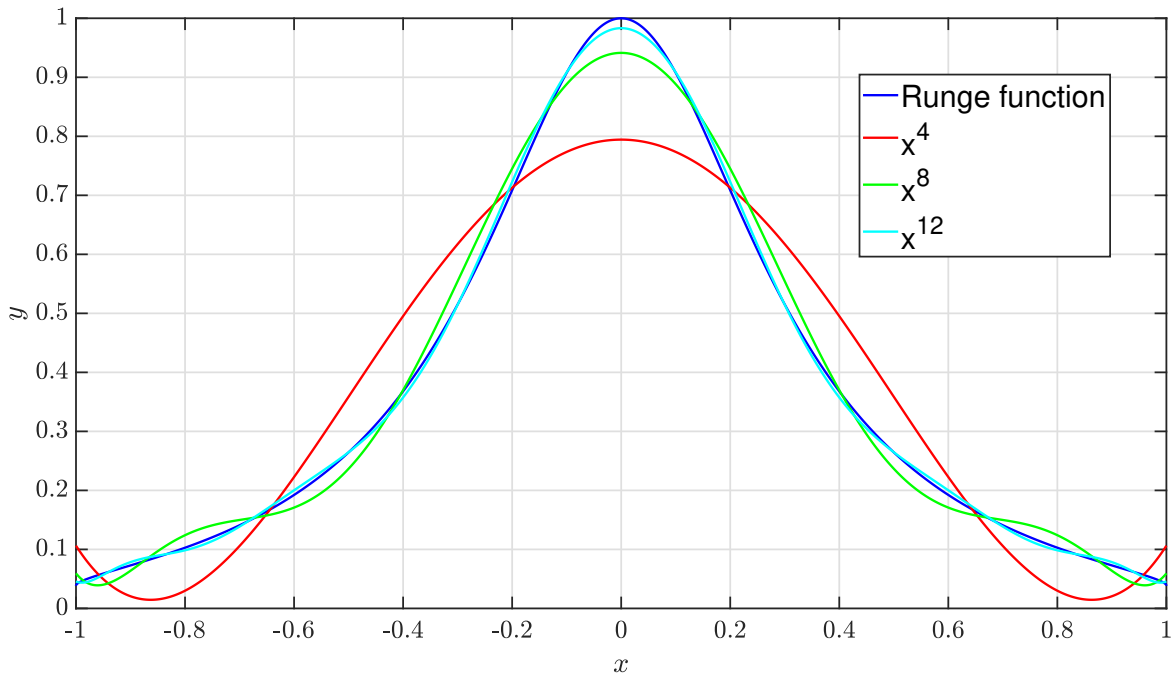

Figure 20: Runge's phenomenon using the Chebyshev nodes.

Chebyshev nodes

- In the case of the equidistant grid, the higher degree polynomial we use, the more weird behavior starts to happen between grid points. Especially wild oscillations in tails are common. We can handle this by deploying the Chebyshev nodes.

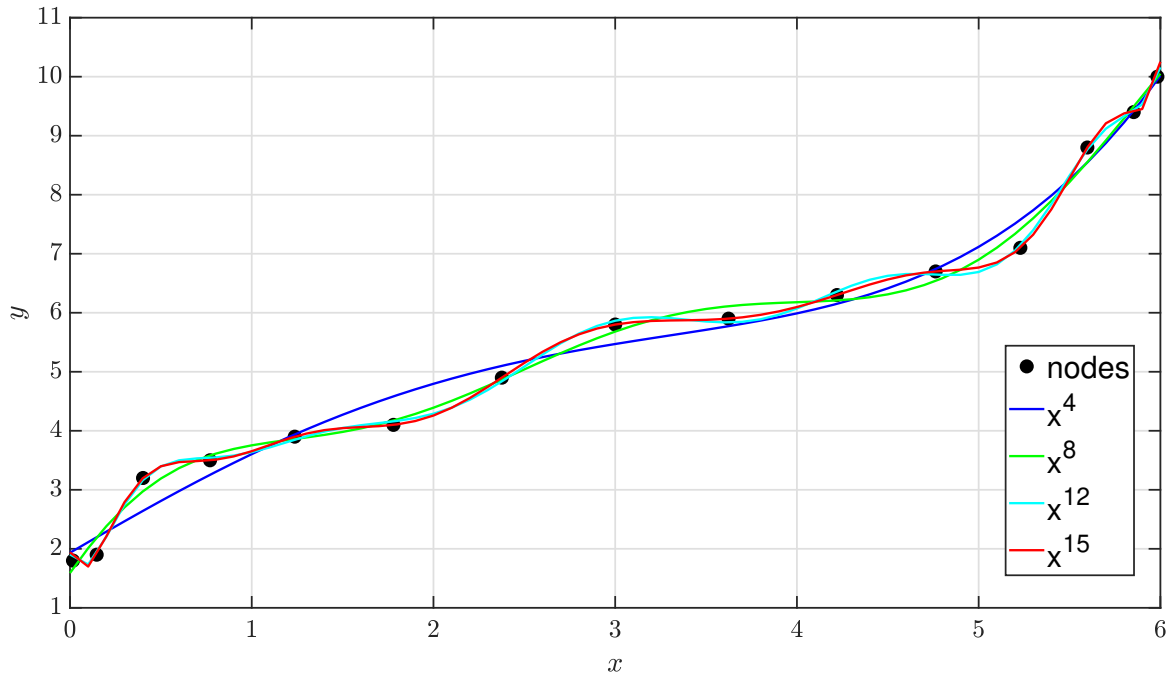

Figure 21: Swing between nodes in tails are not present anymore when applying the Chebyshev nodes.

Chebyshev nodes vs evenly spaced nodes

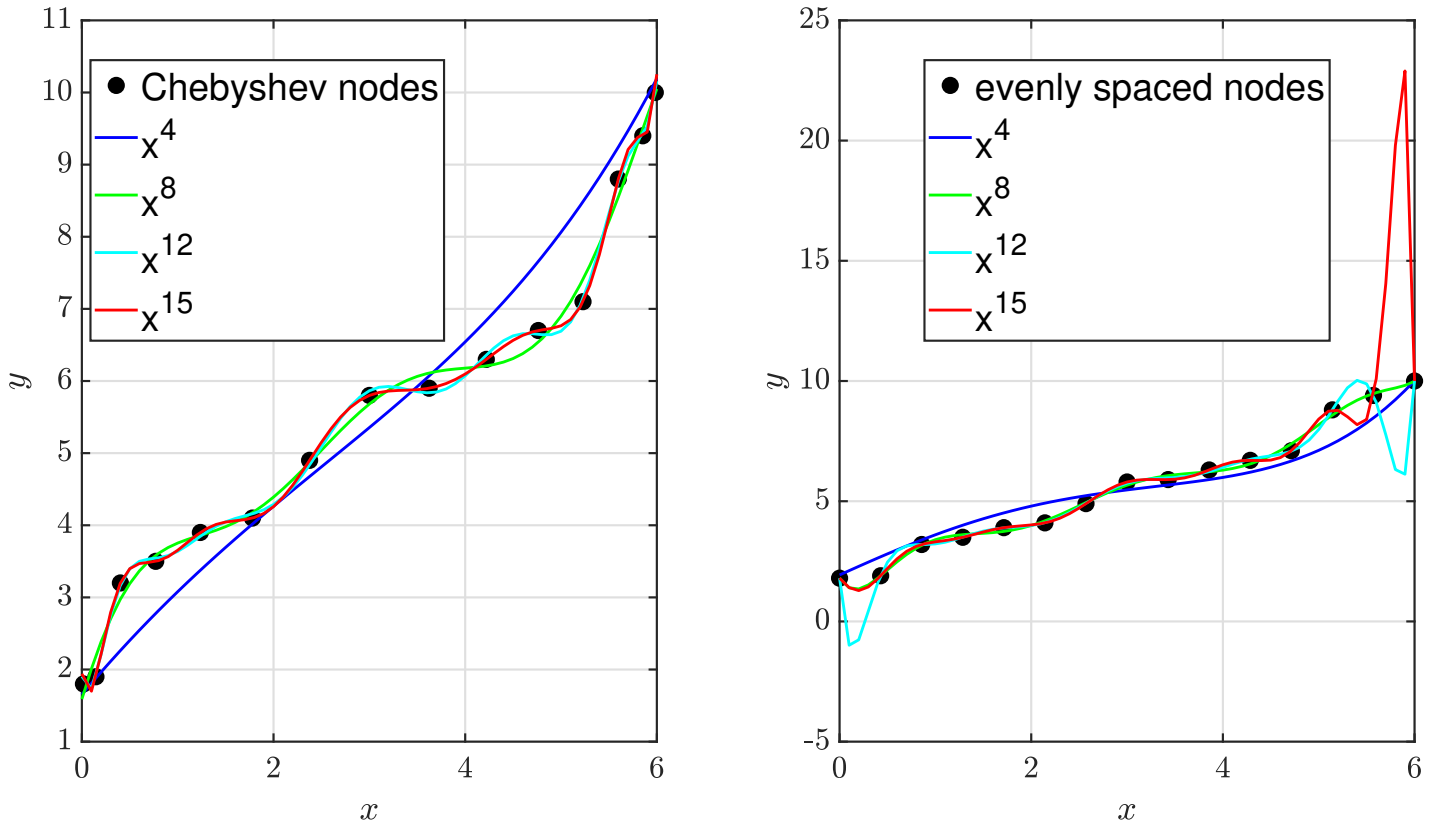

Figure 22: A comparison of polynomial approximation using the Chebyshev nodes and evenly spaced nodes.

Numerical integration

Numerical integration

- As y'all certainly know, there are integrals all over the place in economics.
- Sometimes, it may seem to be a tricky to handle integrals in our problems (even though as we find out in a while, it actually is not so much).
  - Sometimes the functional form we are after is tricky. Even worse, there may be occasions we do not even have the functional form. People familiarized with Bayesian estimation know that we can have a problem with drawing out of an integral.
- One can come up with different methods to overcome the problem.
  - Monte-Carlo integration.
  - Quadrature methods.
    - Newton-Cotes quadrature.
    - Gaussian quadrature.

Numerical integration

- As y'all certainly know, there are integrals all over the place in economics.
- Sometimes, it may seem to be a tricky to handle integrals in our problems (even though as we find out in a while, it actually is not so much).
  - In economics, we often need to compute the definite integral of a real-valued function $f(x)$ given some weighting function $w(x)$ over an interval $[a, b] \in \mathbb{R}^n$. Formally:

$$\int_a^b f(x)w(x)dx$$

- Changing $w(x)$ gives us a different sort of problem. The integral represents the area under the function $f(x)$ in the case we set $w(x) = 1$.

Numerical integration

- Changing $w(x)$ gives us a different sort of problem. The integral represents the expectation of $f(X)$ in the case we express $w(X)$ being the probability density of a random variable X considering that $[a, b]$ describes the whole support of X.
- To allow for some intuition related to our models, assume that we have a model where y is our policy function and x_t is a state vector. What is more, the agents in the economy face some innovations ϵ and $x_t = g(\epsilon)$ holds. Then:

$$\mathbb{E}_t y(x_{t+1}) = \int_{\epsilon} y(x_{t+1}) f(\epsilon_{t+1}) d\epsilon_{t+1}$$

  - The agents make their decision based on the expected value of the state space which is a function of the shock.
  - An equilibrium of the model is determined by a policy function and expectation function satisfying the model's equilibrium conditions (FOCs, market clearing, etc.).

Monte-Carlo integration

- Let's come back to different approaches of handle integrals. Many of you would come up with a Monte-Carlo integration as a solution.

  Just a quick recap:You want to compute the integral $\int_a^b f(x)dF(x)$ with x denoting a random variable with the cumulative distribution function F(x). Knowing $\{x_t\}_{t=1}^T$ is a series drawn from a random number generator, we proceed as follows:´

$$\int_a^b f(x)dF(x) \approx \frac{\sum_{t=1}^T f(x_t)}{T}$$

- The problem is that MC integration is not so accurate and for the case of solving general equilibrium models we have better procedures.

Quadrature Methods

- In numerical quadrature methods, we approximate the integral by a weighted sum of function values:

$$\int_a^b f(x)w(x)dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

  - Where specific methods differ in the quadrature weights w_i and the quadrature nodes x_i choice.

Quadrature Methods

- Specific methods differ in the quadrature weights w_i and the quadrature nodes x_i choice.
- Newton-Cotes quadrature methods divide interval into equidistant subintervals and approximate f(x) with a low-order polynomials within the subintervals. Then they use integrals of these polynomials as the approximated solution to the integration.
  - You might have come across Trapezoidal rule or Simpson's rule.
- Gaussian quadrature methods are based on the same idea as Newton-Cotes methods (approximate the function by low-order polynomials and integrate over these) yet they deploy way smarter choice of quadrature nodes and weights.
  - Gauss–Legendre quadrature as a generic example. Anyway, we are more interested in Gauss–Hermite quadrature.

Trapezoidal rule

- Consider a bounded interval $[a, b] \in \mathbb{R}$ where nodes $x_i \in [a, b]$ are defined as $x_i = a + (i-1)h, \quad \forall i$. We also know that $h = (b-a)/n$.
  - Hence, the nodes x_i divide the interval into $n-1$ equidistant subintervals of length h.
- The function f(x) may be then approximated within each subinterval i ($[x_i, x_{i+1}]$ by the line connecting the two points $\Big(x_i, f(x_i)\Big)$ and $\Big(x_{i+1}, f(x_{i+1})\Big)$. Once we sum up these approximations over the subintervals, we get the trapezoidal approximation.

Trapezoidal rule

- Keep it more formal, in each subinterval, we have:

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \int_{x_i}^{x_{i+1}} \hat{f}(x)dx = \frac{h}{2}\Big[f(x_i) + f(x_{i+1})\Big]$$

- Then if we sum up the approximated areas (trapezoids) over the subintervals, we get the final approximation in generic form:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i),$$

- where in the case of trapezoidal integration $w_1 = w_n = h/2$ and otherwise $w_i = h$.

Trapezoidal rule

- To compute an approximate integral of a real-valued function $f(x)$ over a bounded interval $[a, b] \in \mathbb{R}$
  - Partition the interval into equidistant subintervals.
  - Approximate the function across each of them by a straight line that linearly interpolates the function values at the subinterval boundaries.
  - Sum up the areas under the lines over the subintervals.

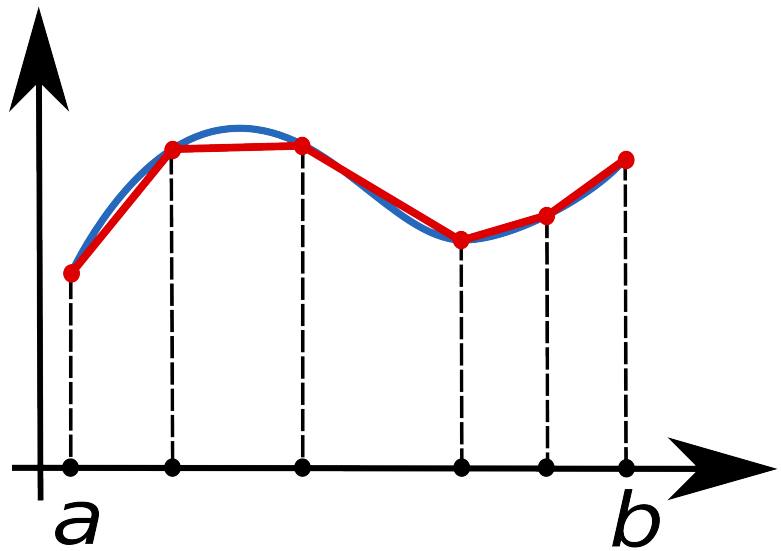

Figure 23: An example of Trapezoidal rule.

Simpson's rule

- Consider a bounded interval $[a, b] \in \mathbb{R}$ where nodes $x_i \in [a, b]$ are defined as $x_i = a + (i-1)h, \quad \forall i$. This time n is an odd integer and we have $h = (b-a)/(n-1)$.
  - Hence, the nodes x_i divide the interval into $n-1$ equidistant subintervals of length h where $n-1$ is an even number.
- The function f(x) may be then approximated within the j^{th} pair of subinterval ($[x_{2j-1}, x_{2j}]$ and $[x_{2j}, x_{2j+1}]$) by the quadratic function (x) connecting the three points $\left(x_{2j-1}, f(x_{2j-1})\right)$ and $\left(x_{2j}, f(x_{2j})\right)$, and $\left(x_{2j+1}, f(x_{2j+1})\right)$. Once we sum up these approximations over the subintervals, we get the quadratic approximation.

Simpson's rule

- Keep it more formal, in each subinterval, we have:

$$\int_{x_{2j-1}}^{x_{2j+1}} f(x)dx \approx \int_{x_{2j-1}}^{x_{2j+1}} \hat{f}(x)dx = \frac{h}{3}\Big[f(x_{2j-1})+4f(x_{2j})+f(x_{2j+1})\Big]$$

- Then if we sum up the approximated areas (quadratic approximants) over the subintervals, we get the final approximation in generic form:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i),$$

- where in the case of Simpson's integration $w_1 = w_n = h/3$ and otherwise $w_i = 4h/3$ or $w_i = 2h/3$ depending whether i is odd or even.

Simpson's rule

- Simpson's rule follows the cookbook as the Trapezoidal rule except for the way of approximating the subintervals. Here we use a piece-wise quadratic approximation, not the piece-wise linear as in the previous case of Trapezoidal approximation.
  - Simpson's rule yields higher degree of accuracy if the integrand is smooth.

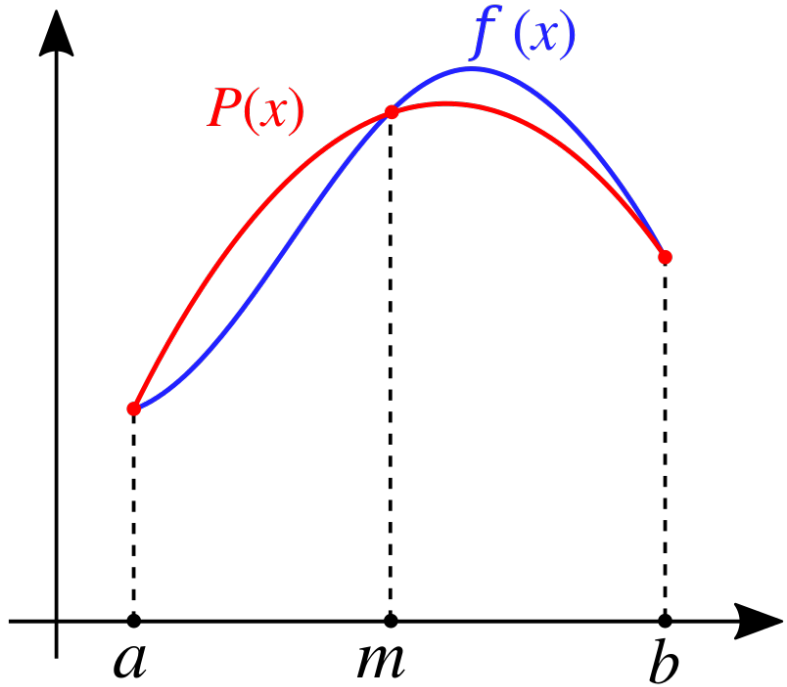

Figure 24: An example of Simpson's rule.

Gaussian quadrature methods

- The underlying idea behind is basically the same as for Newton-Cotes quadrature methods. However, Gaussian quadrature methods are way smarter in choosing the nodes (and the weights).
- Newton-Cotes quadrature delivers the exact solution only in the case that the true function is polynomial of degree up to $n-1$.
- Nevertheless, if we are more careful about setting the nodes, we can increase the ability to retrieve the exact solution up to the polynomial degree of $2n-1$!

Gaussian quadrature methods

- Imagine we are after integrating a scalar function defined on an interval $[-1, 1]$. We decide to use the Gaussian quadrature methods. Therefore, our problem may be written as:

$$\int_{-1}^{1} f(x)d(x) \approx \sum_{i=1}^{n} \omega_i f(\zeta_i),$$

- where n denotes the number of nodes. We end up with $2n$ free parameters (the weights and nodes) while we want to get the exact solution for any polynomial up to degree $2n - 1$.
  - To disentangle this problem, we need to choose the values of the nodes and weights such that we get the exact answer for all basis functions $(1, x, x^2, ...x^{2n-1})$.

Gaussian quadrature methods

- In order of finding the values of ω_i and ζ_i for all i, the following must hold:

$$\int_{-1}^{1} x^j dx = \sum_{i=1}^{n} \omega_i \zeta_i^j \quad \forall j = 0, 1, 2..., 2n-1$$

- Thus, we have a system of 2n equations in 2n unknowns.
  - Note that the solution of the system does not depend on the f. Hence, practically it is not a problem to use these methods as one can find dozens of publicly available material to solve for the nodes and weights.

Gaussian quadrature methods

- The type of problem we have worked with until now when integrating over an interval $[-1, 1]$ is called Gauss-Legendre procedure.

$$\int_{-1}^{1} f(x)d(x) \approx \sum_{i=1}^{n} \omega_i^{GL} f(\zeta_i)^{GL}$$

- Let's be utterly practical. What you really need to do to obtain this numerical integration approximation:
  - Use some procedure or material out there to get the values of ω_i^{GL} and $f(\zeta_i)^{GL}$ for all i.
  - These values will satisfy that the expression from the previous slide holds.
  - Honestly you do not need take care much about the details behind generating the values (if you want to study more, you will find a public material though).
  - You just have to use the values and calculate the functional values to get the integral approximation. You are done!

Gauss-Hermite

- Gauss-Legendre works nicely if our function $f(x)$ can be approximated well by some polynomial.
- However, what if we want to get an approximation of $f(x) = g(x)W(x)$, where $g(x)$ can be approximated well but $g(x)W(x)$ cannot.
- There are different Gaussian quadrature that adjust for this given different weighting function and different domain.
- We want to work with Gauss-Hermite which works with $W(x) = e^{-x^2}$ and the real line as a domain.

Gauss-Hermite

- In the case of Gauss-Hermite, the nodes and weights are chosen such that the following holds:

$$\int_{-\infty}^{\infty} x^j e^{-x^2} dx = \sum_{i=1}^{n} \omega_i \zeta_i^j \quad \forall j = 0, 1, 2..., 2n-1,$$

- and hence we can denote our approximation as follows:

$$\int_{-\infty}^{\infty} g(x) e^{-x^2} d(x) \approx \sum_{i=1}^{n} \omega_i^{GH} f(\zeta_i)^{GH}$$

Gauss-Hermite

- Let me stress one last important point. This will be our cookbook of what to do in practice when approximating innovations in some shock variable within our models.
- Assume one wants to approximate some expectation function $h(y)$ (i.e. $\mathbb{E}\Big[h(y)\Big]$) with y being a random variable with a distribution $N(\mu, \sigma^2)$.
- Presuming that $h(y)$ can be approximated by some polynomial without any problem, we handle the following integral:

$$\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} h(y) \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right] dy$$

- But some changes are necessary to do.

Gauss-Hermite

- Presuming that $h(y)$ can be approximated by some polynomial without any problem, we handle the following integral:

$$\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} h(y) \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right] dy$$

- But some changes are necessary to do.
- We cannot just approximate the following expression:

$$\int_{-\infty}^{\infty} \bar{h}(y) \exp(-y^2) dy,$$

- where we adapted $\bar{h}(y) = \frac{h(y)}{\sqrt{2\pi}\sigma} \frac{exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)}{exp(-y^2)}$.
  - But this is a problem because even though $h(y)$ may be approximated nicely by polynomial, the same does not hold for $\bar{h}(y)$. Note that $\exp(-y^2)$ is inside of $\bar{h}(y)$. Therefore, given $\exp(-y^2)$ presence, $\bar{h}(y)$ would be approximated very poorly by polynomial.

Gauss-Hermite

- To approximate the integral, we need to do some changes of the variables. Specifically:

$$x = \frac{y - \mu}{\sqrt{2}\sigma},$$

- which gives us:

$$y = \sqrt{2}\sigma x + \mu$$

- Now we just substitute back to the initial problem.

Gauss-Hermite

- Now we just substitute back to the initial problem.

$$\mathbb{E}\Big[h(y)\Big] = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} h(y) \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right] dy,$$

- which leads to:

$$\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} h(\sqrt{2}\sigma x + \mu) \exp(-x^2)\sigma\sqrt{2} dy,$$

- or eventually:

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi}} h(\sqrt{2}\sigma x + \mu) \exp(-x^2) dy$$

Gauss-Hermite

- Here we are! We have finally arrived into the practical cookbook. Thus, in practice once we have some $\mathbb{E}\Big[h(y)\Big]$ that we want to approximate, we follow these steps.
  - Obtain the values of the nodes and weights using some numerical procedure.
  - Retrieve the functional values and sum them up to get the approximation of the following form:

$$\mathbb{E}\Big[h(y)\Big] \approx \sum_{i=1}^{n} \frac{1}{\sqrt{\pi}} \omega_i^{GH} h(\sqrt{2}\sigma\zeta_i^{GH} + \mu)$$

- And that's all. We have approximated the integral using Gauss-Hermite quadrature. I like the quote from the notes of Wouter den Haan:
  - *"Well, how often do you get something in life so complex as an integral so easily?"*

Examples of econ models solved by projection methods

Solving econ models

- Let's finally do some economics. I will show you how to use projection methods to solve three models - the RBC model we worked with at the very beginning, the Lucas Asset Pricing Model, and the 3-equation New Keynesian model with an occasionally binding constraint (ELB).
- Let's dive into it!

RBC Model example

- What other model than the growth model extended by the stochastic innovations can we start with. So coming back to the beginning of the presentation, we work with:

$$\boxed{C_t^{-\eta} = \beta\, \mathbb{E}_t\, C_{t+1}^{-\eta} \alpha Z_{t+1} k_{t+1}^{\alpha-1}}$$

$$C_t + k_{t+1} = Z_t k_t^{\alpha}$$

$$Z_t = (1-\rho)Z + \rho Z_{t-1} + \sigma\epsilon_t,$$

- Do you remember what we said? If not, read again the following slide.

RBC model example

- Have a look at the system of equations above. What do we know?
  - We surely know Z_t. We also know the initial given value of k. However, we need to choose C_t -> the policy function! Once we obtain C_t, k_{t+1} comes out straight from the budget constraint equation.
- We need to solve out (approximate) for C_t.

RBC Model example

- So we need to approximate C_t by some family of polynomials. It is by definition that for the true rational expectation solution, the following must hold:

$$C_t = c(K_t, Z_t),$$

- and hence also $K_{t+1} = k(K_t, Z_t)$.
- Firstly, we need to choose the family of polynomials to approximate the policy function:

$$C_t = c(K_t, Z_t) \approx \varphi^n(K_t, Z_t; \theta_n),$$

- where n is the polynomial degree. Consequently, we have to solve for θ_n which is a finite-dimensional object.
  - Note that we have N_n elements of θ_n.
  - The problem is that our policy function is a whole brand new equation at each given point in the state space.

RBC Model example

- So we need to approximate C_t by some family of polynomials. It is by definition that for the true rational expectation solution, the following must hold:

$$C_t = c(K_t, Z_t),$$

- and hence also $K_{t+1} = k(K_t, Z_t)$.
- Or put it differently, we can express the problem in terms of the residual function:

$$r(K_t, Z_t) = -C_t^{-\eta} + \mathbb{E}_t \beta C_{t+1}^{-\eta} \alpha Z_{t+1} K_{t+1}^{\alpha-1}$$

- Thus, at the true solution $C_t = c(K_t, Z_t)$ and $K_{t+1} = k(K_t, Z_t)$, we have:

$$r(K_t, Z_t) = 0 \quad \forall K_t, Z_t$$

RBC Model example

- Consider having M grid points $\{K_i, Z_i\}$ in the state space where $M \geq N_n$. Then we can write:

$$r(K_i, Z_i; \theta_n) = -\varphi^n(K_i, Z_i; \theta_n)^{-\eta} + \mathbb{E}\begin{bmatrix} \alpha\beta \times \\ \varphi^n(K', Z'; \theta_n)^{-\eta} \times \\ Z' \times \\ (K')^{\alpha-1} \end{bmatrix}$$

RBC Model example

- Substitute in what we know gives us (bear in mind that we know the grid values as we set it up):

$$r(K_i, Z_i; \theta_n) = -\varphi^n(K_i, Z_i; \theta_n)^{-\eta} +$$
$$\mathbb{E}\left[\begin{array}{c} \alpha\beta\times \\ \varphi^n\left(Z_i K_i^{\alpha} - \varphi^n(K_i, Z_i; \theta_n), \exp\{\rho \log(Z_i) + \epsilon'\}; \theta_n\right)^{-\eta} \times \\ \rho \log(\mathrm{Z_i} + \epsilon' \times \\ \left(Z_i K_i^{\alpha} - \varphi^n(K_i, Z_i; \theta_n)\right)^{\alpha-1} \end{array}\right]$$

RBC Model example

- We need somehow get rid of ϵ'. Fortunately, after circa 20 slides about the numerical integration it should not be a big deal for us anymore. We use Gauss-Hermite quadrature and get:

$$r(K_i, Z_i; \theta_n) = -\varphi^n(K_i, Z_i; \theta_n)^{-\eta} + \alpha\beta \times$$
$$\sum_{j=1}^{J} \left[\begin{array}{c} \varphi^n\left(Z_i K_i^{\alpha} - \varphi^n(K_i, Z_i; \theta_n), \exp\{\rho \log(Z_i) + \sqrt{2}\sigma\zeta_j\}; \theta_n\right)^{-\eta} \times \\ \rho \log(\mathrm{Z_i} + + \sqrt{2}\sigma\zeta_{\mathrm{j}} \times \\ \left(Z_i K_i^{\alpha} - \varphi^n(K_i, Z_i; \theta_n)\right)^{\alpha-1} \times \\ \frac{\omega_j}{\sqrt{\pi}} \end{array} \right]$$

- where ω_j and ζ_j are the Gauss-Hermite weights and nodes, respectively.

RBC Model example

- Now just follow the cookbook described in the previous sections. particularly, we solve the model by utilizing the method of collocation.

The Lucas Asset Pricing Model example

- Let me use basically isomorphic kind of problem we handled in the RBC model in order to get more intuition by repetition. In the Lucas Asset Pricing Model, we once again approximate an Euler equation. However, this time it has slightly different interpretation.

The Lucas Asset Pricing Model example

- We have a representative infinitely-lived agent allocating her wealth between consumption and investment. She can invest in shares S_t and thus increase her wealth. The shares give you a right to obtain a dividend of D_t units of a consumption good per share. The share price is P_t.
- The problem is then defined as:

$$V(s_t, D_t, P_t) = \max_{C_t \in [0, d_t s_t]} \mathbb{E}_0[\sum_{t=0}^{\infty} \delta^t U(C_t]$$

- s.t.

$$C_t + P_t S_{t+1} = D_t S_t + P_t S_t$$

The Lucas Asset Pricing Model example

- The agent's dynamic optimization problem has a unique solution satisfying the first-order Euler condition:

$$U'(C_t)P_t = \delta \mathbb{E}_t \Big[U'(C_{t+1})(P_{t+1} + D_{t+1}) \Big],$$

- which for the case of CRRA utility function:

$$C_t^{-\eta} P_t = \delta \mathbb{E}_t \Big[C_{t+1}^{-\eta} (P_{t+1} + D_{t+1}) \Big],$$

- where η is the inverse of the intertemporal elasticity of substitution (risk aversion). The equation tells us that the expected marginal utility from saving is equal to zero.

The Lucas Asset Pricing Model example

- Normalizing the total number of shares being equal to the population size allows us reaching the clearing condition $C_t = D_t$.
- To close the model, we need to impose some assumption on D_t. We will assume that it behaves according to an exogenous Markov process:

$$D_{t+1} = g(D_t, \epsilon_{t+1})$$

- So to put it into the context. The model may be characterized by one state variable D_t, one control variable P_t, and one equilibrium condition in the form of the Euler equation.

The Lucas Asset Pricing Model example

- A solution is a function P(D). One can express the asset return function by the functional equation:

$$U'(D)P(D) - \delta \mathbb{E}_\epsilon \left[U'\Big(g(D,\epsilon)\Big)\Big(P(g(D,\epsilon)\Big) + g(D,\epsilon) \right] = 0,$$

- From here, follow the well-known steps.
  - Choose some family of polynomials and some degree of approximation.
  - Approximate either the policy function or the expectation function.
  - Use Gaussian quadrature to discretize the innovations in the shock equation.
  - Use some method to project the approximant into the unknown function (we use collocation).
  - Find the coefficients of the polynomial writing the collocation equation as a fixed-point problem or as a root-finding problem.

The textbook 3-equation NK model with an effective lower bound

- Time permitting, I show also how to solve the benchmark New Keynesian model with an occasionally binding constraint (OBC). We deploy the finite element approach.
- Okay, let's switch to some coding. Anyway, do not hesitate to drop me an email if you want to help out with something or to ask about some extensions.
  - frantisek.masek@uniroma1.it