Projection Methods

Numerical Methods

Cezar Santos

FGV/EPGE

Acknowledgements

- This set of slides is heavily based on notes by Georg Dürnecker and Grey Gordon.

Projection Methods

- Also known as weighted residual methods
- The projection method is a general method used to derive the (approximate) solution to functional equations
- Projection methods focus on finding the function (and not just a sequence of values) which solves a given functional equation
- Examples for functional equations: Bellman equation, Euler equation, linear ordinary and partial differential equations, conditional expectations, etc.

Projection Methods

Two types of projection methods:

- Global basis $\rightarrow$ spectral methods
- Local basis $\rightarrow$ finite elements methods

Let's start with spectral methods!

A Motivating Example

- Let's consider the following ordinary differential equation:

$$\dot{x}(t) + x(t) = 0 \qquad \text{with} \quad x(0) = 1$$

- This is a functional equation, which implicitly determines the solution $x(t)$

- Our objective is to find the approximate solution, $\widehat{x}(t)$, to the differential equation over the interval $[0, 2]$

- The true solution is, of course, $x(t) = e^{-t}$, but suppose it is unknown

$$\dot{x}(t) + x(t) = 0 \qquad \text{with} \quad x(0) = 1$$

- Projection uses a finite sum of polynomials $p \in P = \{p_i\}_{i=0}^{\infty}$ to approximate the solution.

$$\widehat{x}(t) = \sum_{i=0}^{d} \gamma_i p_i(t)$$

- For the case at hand, we use the monomials $(1, t, t^2)$ to approximate the solution in the interval $[0, 2]$

$$\widehat{x}(t) = 1 + \gamma_1 t + \gamma_2 t^2$$

- The parameters (γ_1, γ_2) are unknown. How shall we choose them?

- From

$$\widehat{x}(t) = 1 + \gamma_1 t + \gamma_2 t^2 \qquad \text{We get } \Rightarrow \qquad \dot{\widehat{x}}(t) = \gamma_1 + 2\gamma_2 t$$

- Let's rewrite the functional equation using the approximation

FE to be solved $\qquad \dot{x}(t) + x(t) = 0$

FE using approximation $\qquad \underbrace{\gamma_1 + 2\gamma_2 t}_{\dot{\hat{x}}(t)} + \underbrace{1 + \gamma_1 t + \gamma_2 t^2}_{\hat{x}(t)} = \underbrace{R(\gamma, t)}_{\text{Residual function}}$

- The residual function measures the deviation - induced by the approximation - from zero, the target value

- Want that $\widehat{x}(t)$ is as close as possible to $x(t)$ in the interval $[0, 2]$

- Projection: Adjust $\gamma = \{\gamma_1, \gamma_2\}$ until we find a "good" γ that makes $R(\gamma, t)$ "nearly" the zero function

Different Projection Methods

How to operationalize the notions of "good" and "nearly"?

1. Least squares method
2. Galerkin method
3. Collocation

Least Squares Method

- Choose (γ_1, γ_2) to minimize the total squared residuals over the interval

$$\min_{\gamma_1,\gamma_2} \int_0^2 R(\gamma,t)^2 dt \qquad \text{where} \qquad \gamma = \{\gamma_1, \gamma_2\}$$

- First-order conditions to this problem w.r.t. (γ_1, γ_2) are given by

$$\gamma_1 : \quad \int_0^2 R(\gamma,t)\frac{\partial R(\gamma,t)}{\partial \gamma_1} dt = 0 \qquad \gamma_2 : \quad \int_0^2 R(\gamma,t)\frac{\partial R(\gamma,t)}{\partial \gamma_2} dt = 0$$

- Using the residual function $R(\gamma,t) = \gamma_1 + 2\gamma_2 t + 1 + \gamma_1 t + \gamma_2 t^2$ to solve the integrals delivers a linear system of equations in the unknowns (γ_1, γ_2)

$$\begin{aligned} -6 &= 13\gamma_1 + 24\gamma_2 \\ && \Rightarrow \gamma_1, \gamma_2 \\ -25 &= 60\gamma_1 + 124\gamma_2 \end{aligned}$$

Galerkin method

- If $R(\gamma, t)$ were the zero function, then the integral of its product with any other function would also be zero
- Galerkin method: Takes the functions used in the approximation to impose moment conditions
- In the current example, those functions are given by t and t^2
- Pick the γ which makes $R(\gamma, t)$ orthogonal to t and t^2

$$\int_0^2 R(\gamma, t) t dt = 0 \qquad \int_0^2 R(\gamma, t) t^2 dt = 0$$

- Using the expression for $R(\gamma, t)$ to solve the integrals delivers

$$\begin{aligned} -3 &= 7\gamma_1 + 14\gamma_2 \\ -10 &= 25\gamma_1 + 54\gamma_2 \end{aligned} \quad \Rightarrow \gamma_1, \gamma_2$$

Collocation

- So far: Minimize the residual function $R(\gamma, t)$ over a given interval
- Collocation: Choose γ so that $R(\gamma, t)$ is zero at a finite set of predetermined points $\Rightarrow$ The functional equation holds exactly at these points
- Suppose we want $R(\gamma, t_i) = 0$ for $t_i \in \{t_1 = 1, t_2 = 2\}$

$$R(\gamma, t_1) = \gamma_1 + 2\gamma_2 t_1 + 1 + \gamma_1 t_1 + \gamma_2 t_1^2 = 0$$

$$R(\gamma, t_2) = \gamma_1 + 2\gamma_2 t_2 + 1 + \gamma_1 t_2 + \gamma_2 t_2^2 = 0$$

$\Rightarrow$

$$0 = 2\gamma_1 + 3\gamma_2 + 1 \qquad 0 = 3\gamma_1 + 8\gamma_2 + 1 \qquad \Rightarrow \gamma_1, \gamma_2$$

Collocation

- Preceding methods all involve integrals of the residual function. For the problem at hand, these are easy to compute directly. Is not true generally
- Main advantage of collocation: No integral needs to be evaluated

General Discussion of Projection Methods

- Want to approximate an unknown function

$$f : X \to Y \qquad \text{where} \quad X \subset \mathbb{R}^n, Y \subset \mathbb{R}^m$$

- The function f is not known. It is defined, implicitly, by the functional equation[1]

$$F(f) = 0 \qquad \text{with} \quad F : C_1 \to C_2$$

[1] Where C_1, C_2 are given function spaces

- Let $P = \{p_i\}_{i=0}^{\infty}$ denote a collection of polynomials

- We represent the approximate solution to $f(x)$ by a finite linear combination of $d+1$ polynomials

$$\widehat{f}(\gamma, \mathbf{x}) = \sum_{i=0}^{d} \gamma_i p_i(\mathbf{x}) \qquad \text{with} \quad \mathbf{x} \in X$$

- The residual function is obtained by substituting the approximate solution $\hat{f}$ into the functional equation $F(f) = 0$

$$R(\gamma, \mathbf{x}) = F(\widehat{f}(\gamma, \mathbf{x})) \qquad \text{where} \quad \gamma = (\gamma_0, \ldots., \gamma_p)$$

  Construct a norm (a measure of distance) based on the inner product of $R(\gamma, \mathbf{x})$

$$\int_X \omega(\mathbf{x}) R(\gamma, \mathbf{x}) g_i(\mathbf{x}) d\mathbf{x}$$

  where $\omega(\mathbf{x})$ is a *weighting function* and $\{g_i(\mathbf{x})\}_{i=0}^{d}$ is a set of *test functions*

- The task is to choose the elements of γ such that

$$\int_X \omega(\mathbf{x}) R(\gamma, \mathbf{x}) g_i(\mathbf{x}) d\mathbf{x} = 0 \qquad \forall i = 0, 1, ..., d$$

- Each of the three methods we have seen before can be considered as a special case:
  1. Least-squares: $g_i \equiv \partial R / \partial \gamma_i$ and $\omega \equiv 1$
  2. Galerkin: $g_i \equiv p_i$ and $\omega \equiv 1$
  3. Collocation: $g_i \equiv 1$ and $\omega(\mathbf{x}) = \begin{cases} 0 & if \mathbf{x} \neq \mathbf{x}_i \\ 1 & if \mathbf{x} = \mathbf{x}_i \end{cases}$

Choosing Your Basis Function

- In the previous example, we used monomials, as our basis function: $x, x^2, x^3, \ldots$
- This might not be the best solution since, for large n, x^n and x^{n+1} are highly collinear.
- Thus, it's best to use orthogonal polynomials.
- Great solution: Chebyshev polynomials.

Chebyshev Polynomials

The Chebyshev polynomials, $\{T_0(x), T_1(x), \ldots\}$, take take $[-1,1]$ to $[-1,1]$.

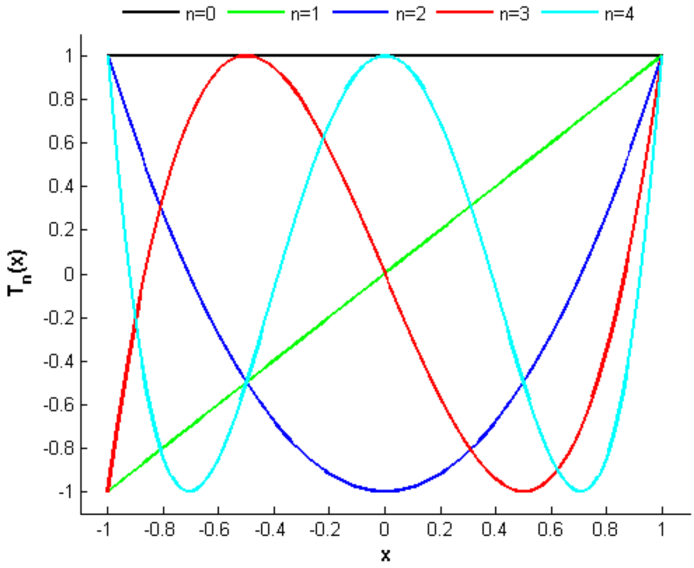
They are defined by a recurrence relation:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_n(x) &= 2xT_{n-1}(x) - T_{n-2}(x). \end{aligned}$$

$T_n(x)$ is called the Chebyshev polynomial of degree n.

There is also a non-recursive way: $T_j(x) = \cos(j \cos^{-1} x)$

Chebyshev Polynomials



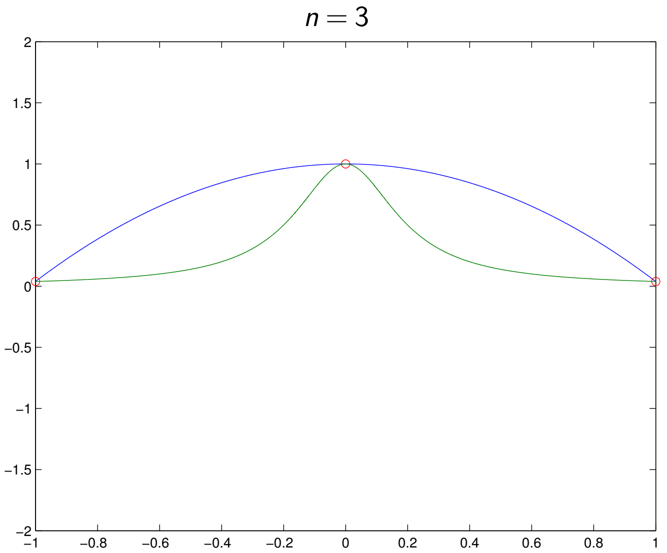
Chebyshev Polynomials

- If using collocation, it's a great idea to pick the roots of an order m Chebyshev polynomial as your interpolation nodes.
- These roots are given by:

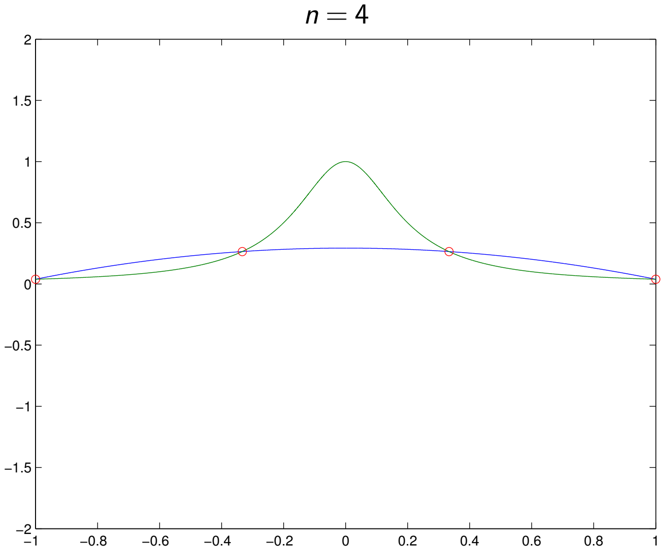
$$z_i = -\cos\left(\frac{2i-1}{2m}\pi\right).$$

- Why? Chebyshev's theorem
  - It avoids the Runge phenomenon.

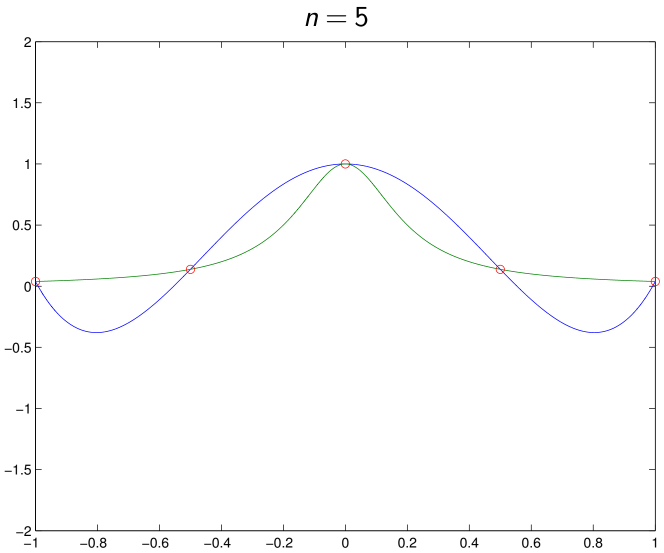
Runge Phenomenon



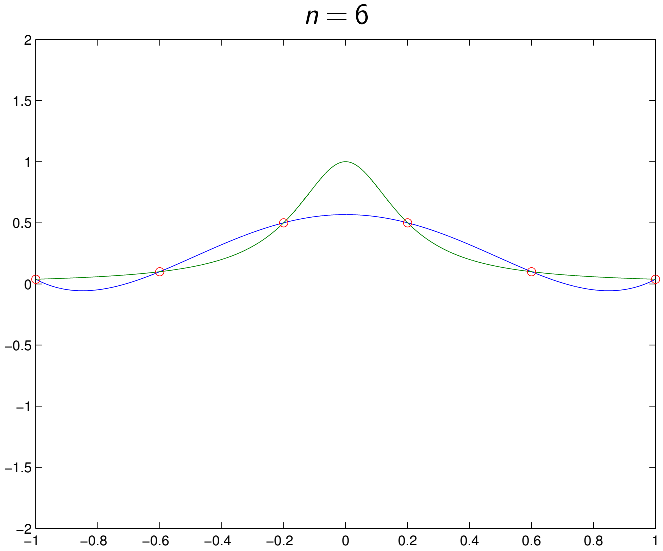
Runge Phenomenon



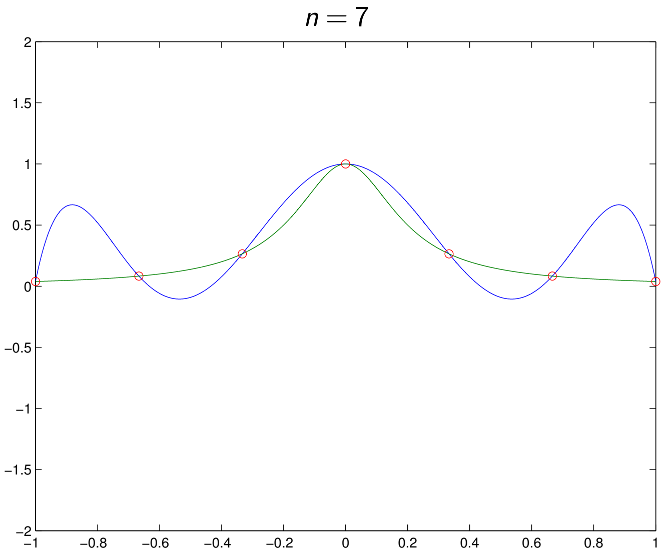
Runge Phenomenon



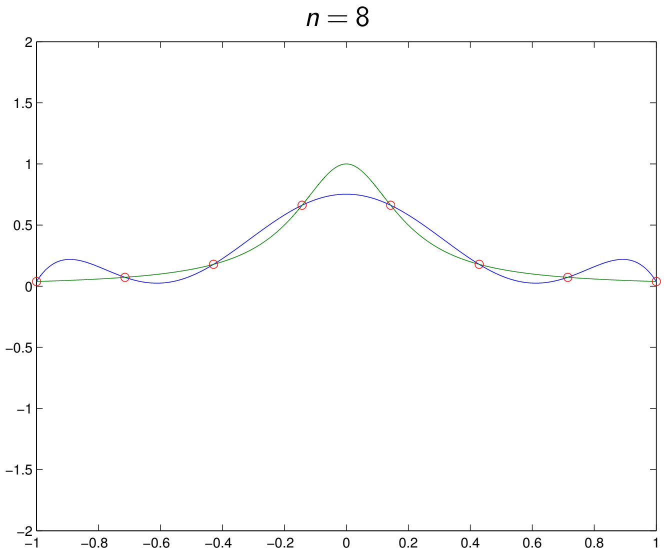
Runge Phenomenon



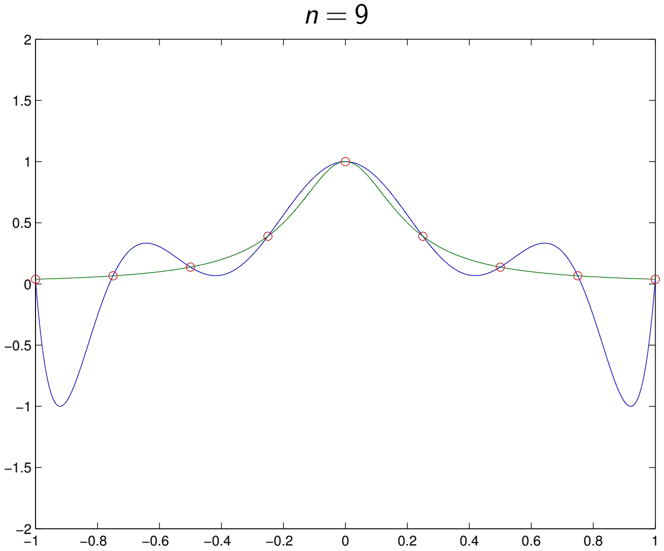
Runge Phenomenon



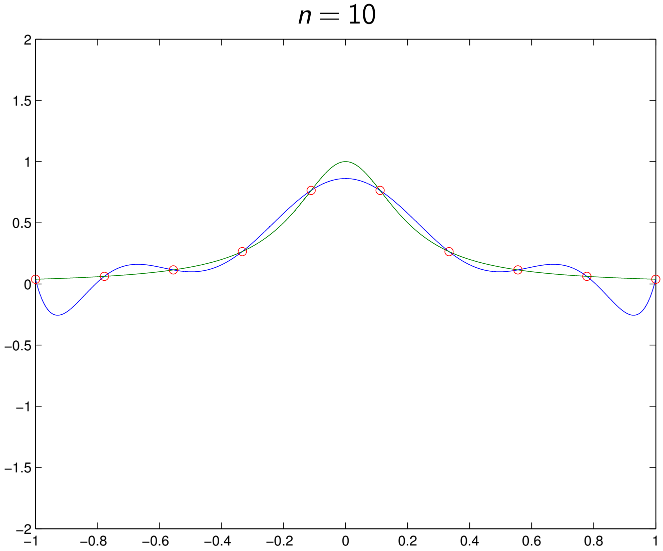
Runge Phenomenon



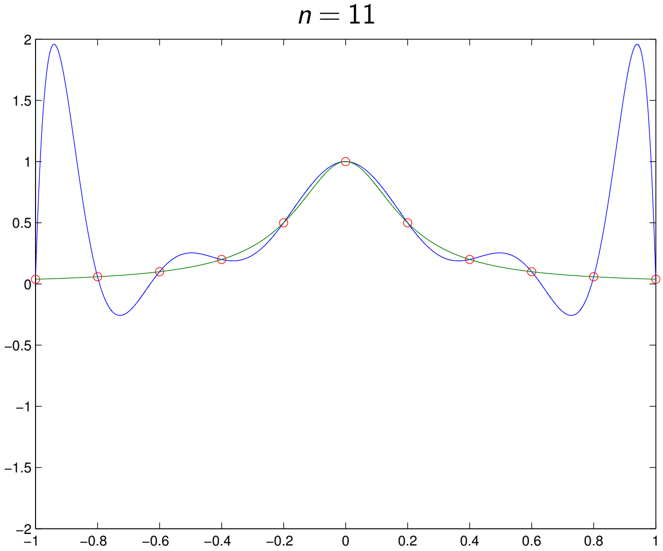
Runge Phenomenon



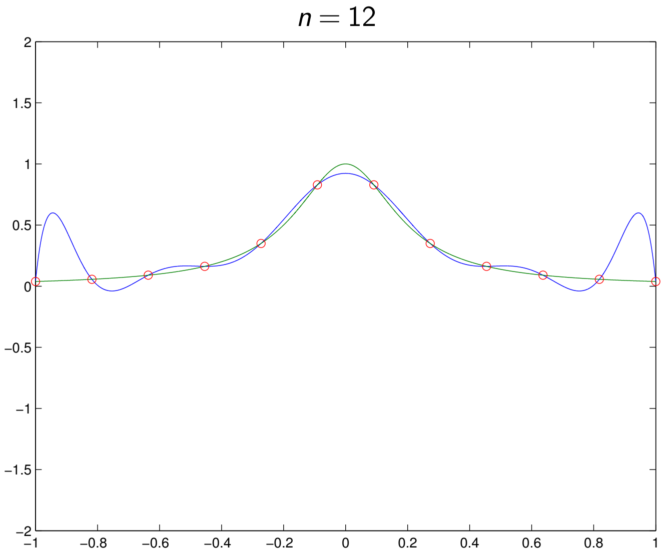
Runge Phenomenon



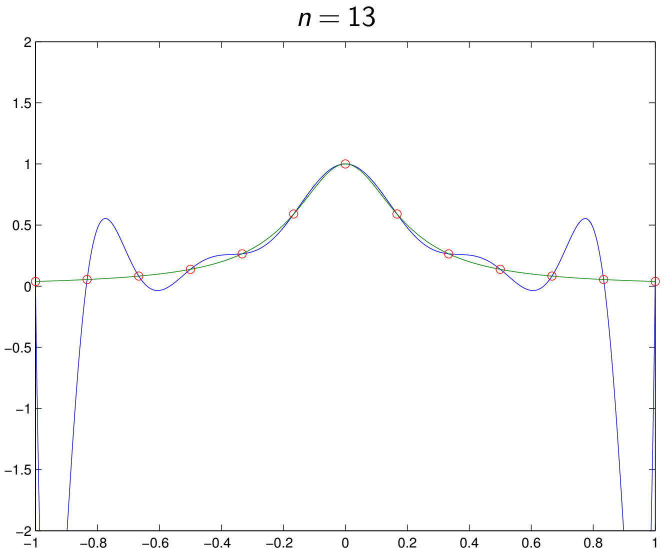
Runge Phenomenon



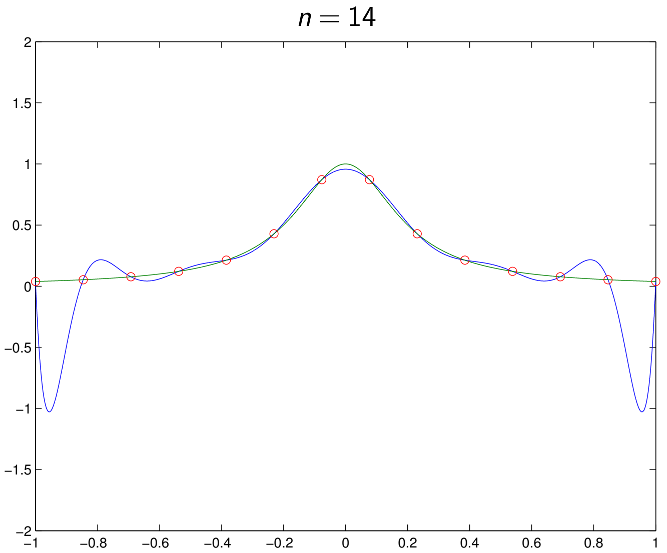
Runge Phenomenon



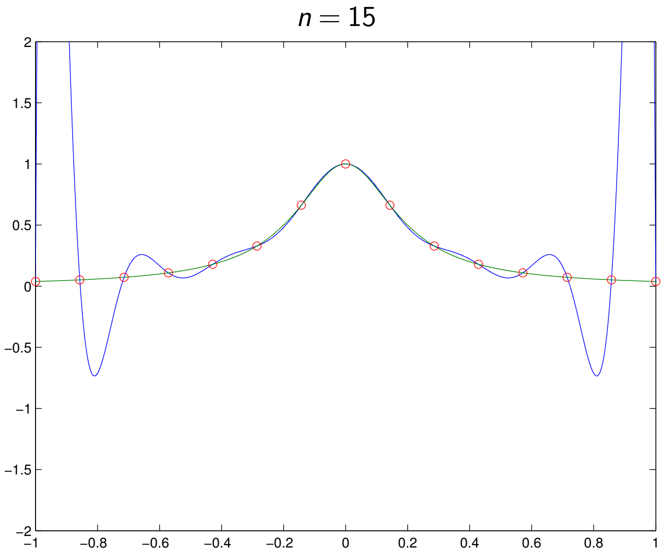
Runge Phenomenon



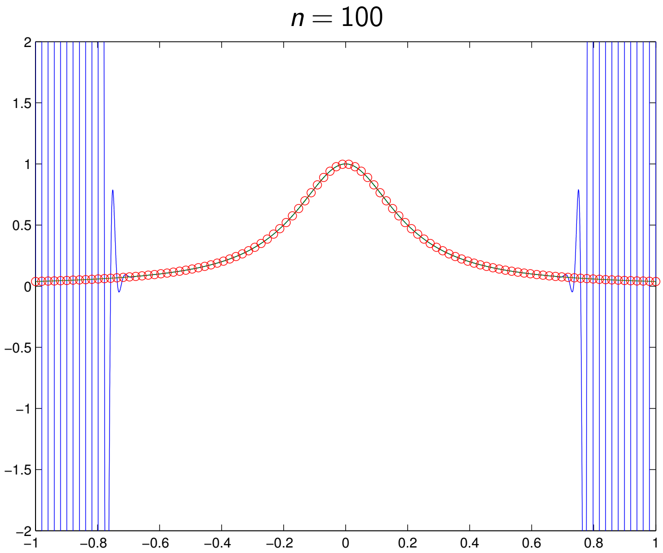
Runge Phenomenon



Runge Phenomenon



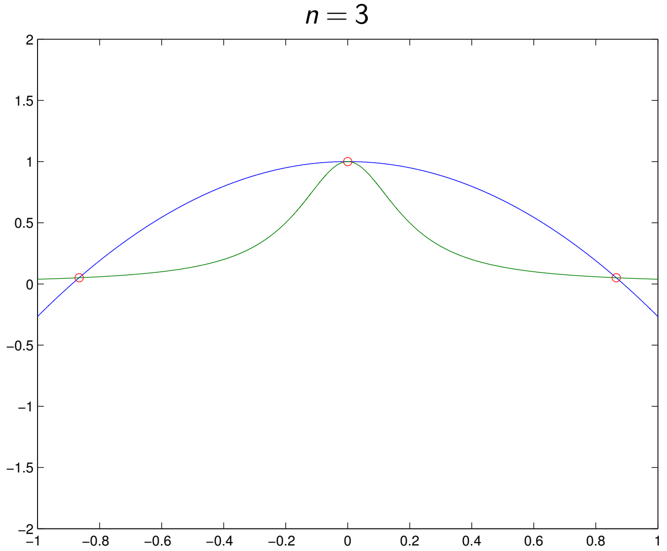
Runge Phenomenon



Chebyshev Polynomials

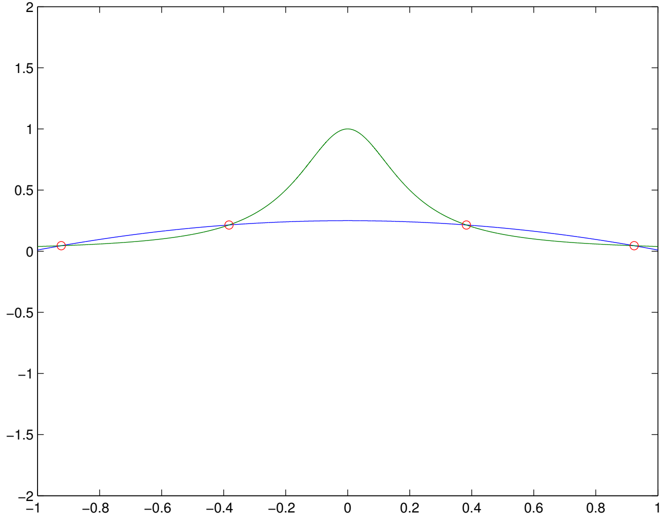
Now let's try to do the same using the roots of the Chebyshev polynomial.

Using Chebyshev Roots

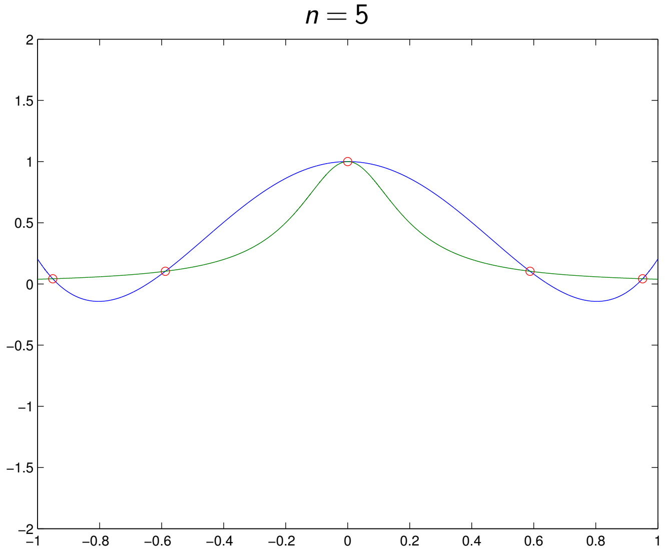


Using Chebyshev Roots

$n = 4$

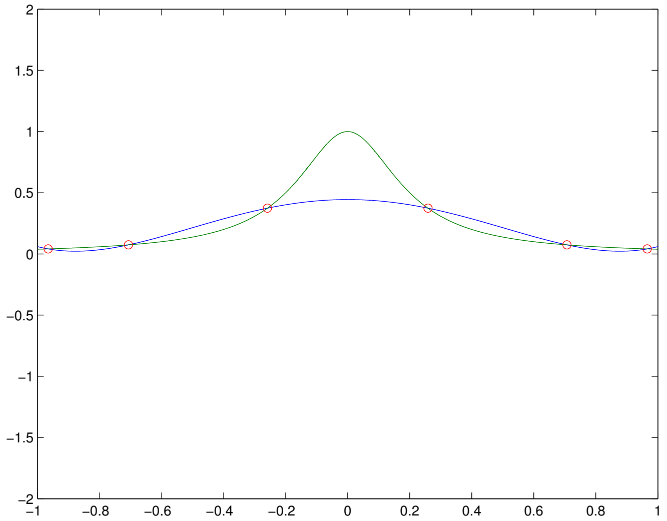


Using Chebyshev Roots

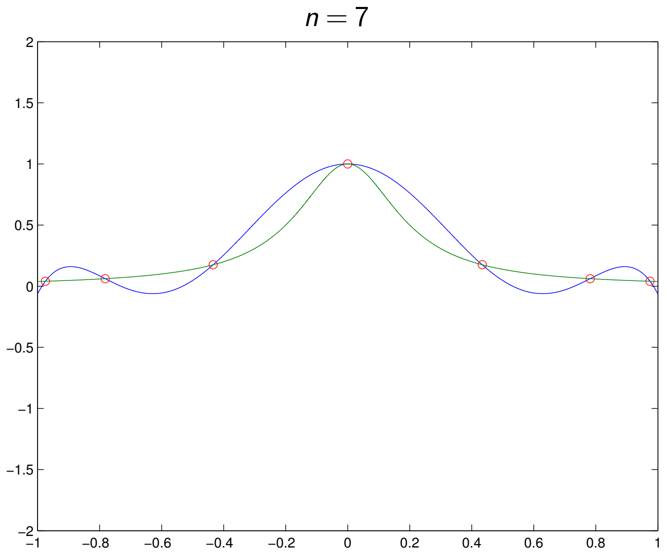


Using Chebyshev Roots

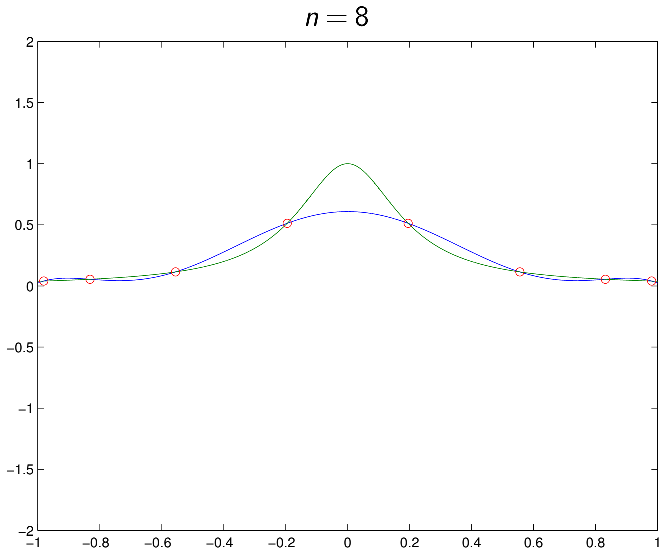
$n = 6$



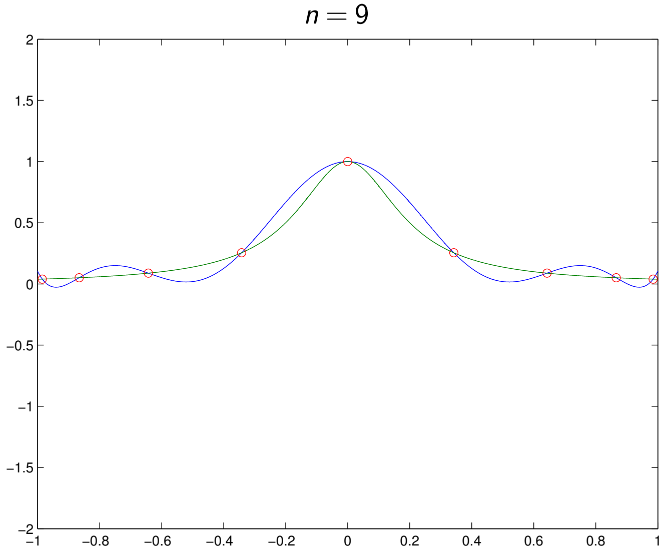
Using Chebyshev Roots



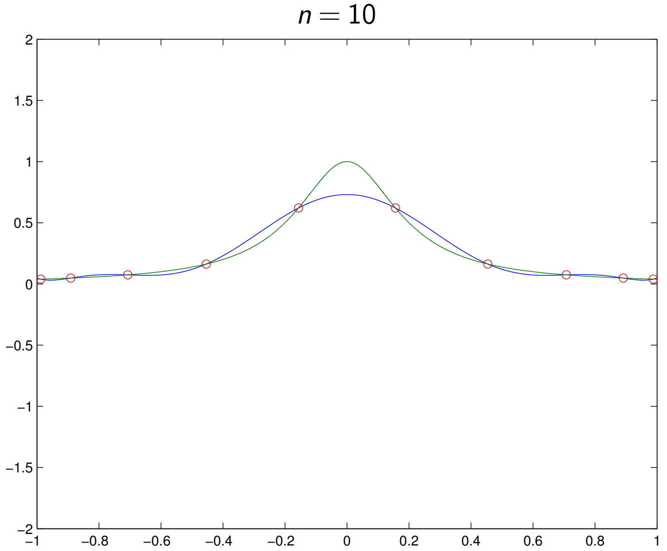
Using Chebyshev Roots



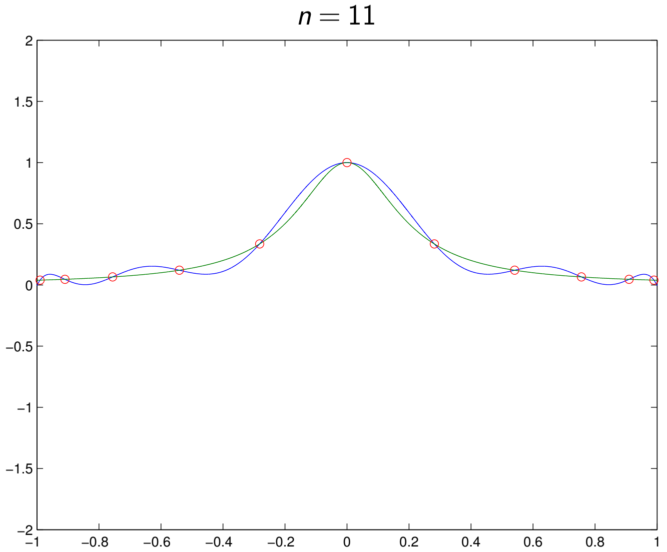
Using Chebyshev Roots



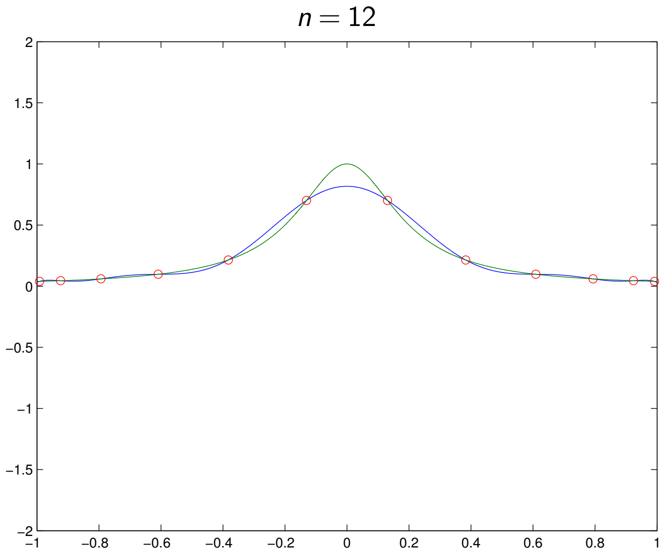
Using Chebyshev Roots



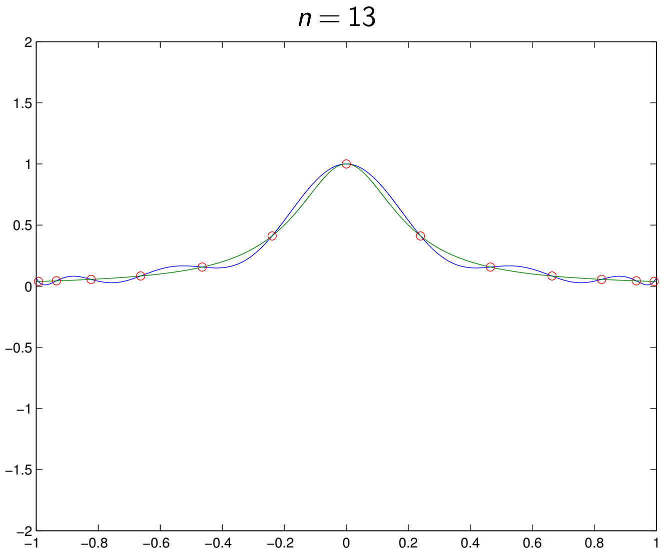
Using Chebyshev Roots



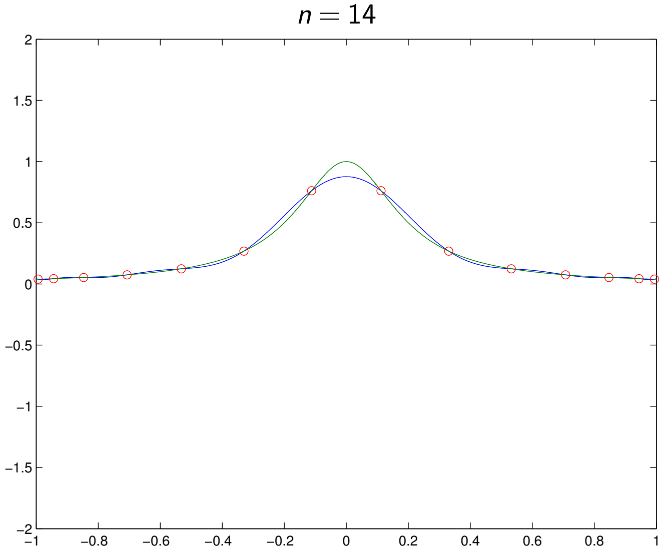
Using Chebyshev Roots



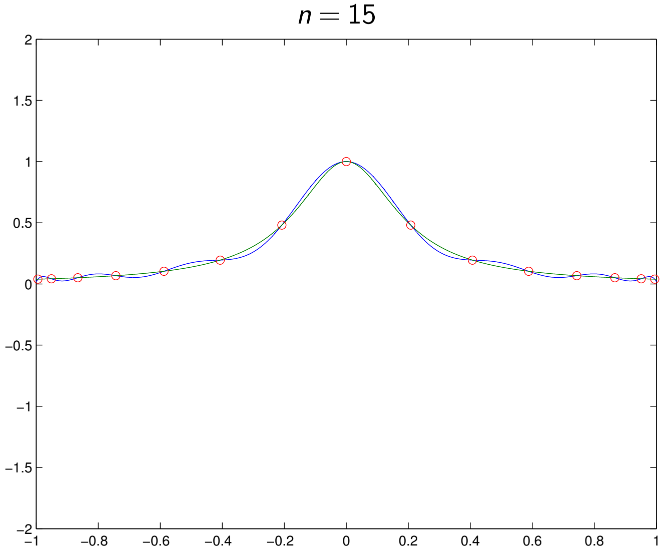
Using Chebyshev Roots



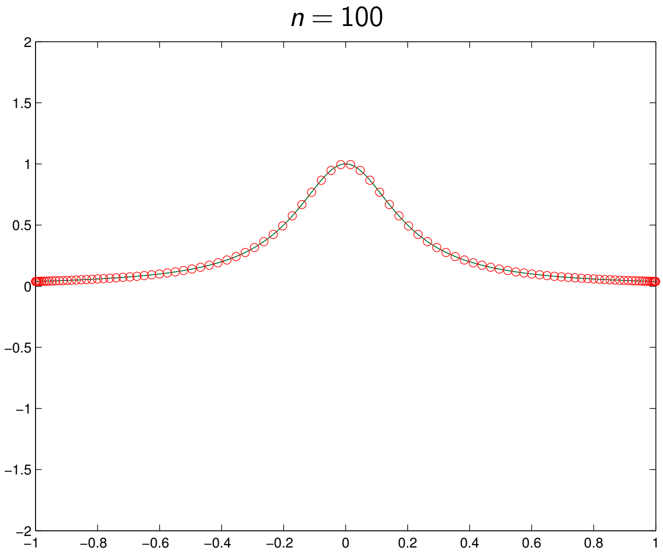
Using Chebyshev Roots



Using Chebyshev Roots



Using Chebyshev Roots



Chebyshev Polynomials

- Chebyshev polynomials can be used in a variety of scenarios.
- Essentially, for anything that you want to fit a smooth function through.
- I always try to use it when possible
    - It's fast and accurate
- Let's look at how we'd implement its use within a familiar framework

Example:
Solving the Euler equation with Chebyshev Collocation

- The Euler equation of the deterministic growth model with
  - preferences: $u(C_t) = \frac{C_t^{1-\eta}}{1-\eta}$
  - production: $F(K_t) = K_t^{\alpha}$
  - capital accumulation: $K_{t+1} = (1-\delta)K_t + F(K_t) - C_t$ is given by

$$\left[\frac{C_{t+1}}{C_t}\right]^{-\eta} \beta \left(1 - \delta + \alpha K_{t+1}^{\alpha-1}\right) - 1 = 0$$

- Task: Compute/approximate the consumption policy function $C(K)$ by Chebychev collocation

Implementation

1. State space is one-dimensional and consists of capital $\Rightarrow$ Choose space $X = [\underline{K}, \bar{K}]$ over which $C(K)$ is approximated

2. Choose class of polynomials for the approximation. Here: Chebyshev polynomial[2] of order d

$$\widehat{C}(\gamma, K) = \sum_{j=0}^{d} \gamma_j T_j(K)$$

[2]Recall: Chebyshev polynomials are defined over $[-1, 1] \Rightarrow K$ needs to be transformed first.

Implementation

1. Pick $K_0 \in X$, and evaluate $\widehat{C}_0 = \widehat{C}(\gamma, K_0)$, which we use together with the resource constraint to get
$$K_1 = K_0^\alpha + (1-\delta)K_0 - \widehat{C}_0$$
2. Given K_1 we evaluate $\widehat{C}$ again to get $\widehat{C}_1 = \widehat{C}(\gamma, K_1)$
3. Using $\widehat{C}_0$, $\widehat{C}_1$ and K_1 we compute the residual function
$$R(\gamma, K_0) = \beta \left[\frac{\widehat{C}_1}{\widehat{C}_0}\right]^{-\eta} \left(1 - \delta + \alpha K_1^{\alpha-1}\right)$$ Falta um -1 aqui!!
4. There are $d+1$ unknown coefficients $\Rightarrow$ need as many conditions
5. Use the zeros of a degree $d+1$ Chebyshev polynomial as collocation points and solve the system of non-linear equations for γ
$$R(\gamma, K_i) = 0 \qquad \forall i = 0, ..., d$$

Issues

- Choice of polynomials: Use of monomials, e.g. $\{1, x, x^2\}$ is not advisable since higher degree monomials are indistinguishable numerically
- Chebyshev polynomials are the preferred choice, are orthogonal to each other when evaluated at the roots. Also Chebyshev roots as collocation nodes minimize the maximal interpolation error (Chebyshev interpolation theorem)
- Accuracy check of the solution: Plot residuals, Euler equation errors, etc.

Finite Element Methods

Our basis function examples have mostly been polynomials.

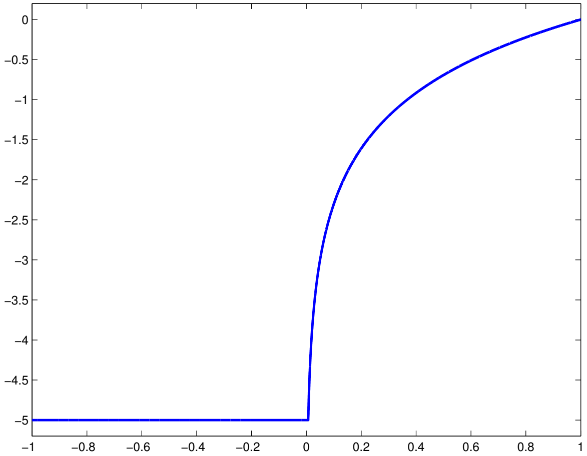
These basis functions are global, non-zero everywhere.

We will now consider local basis functions, which are zero over most of the domain.

Projection with local basis functions is called Finite Element Analysis (FEA).

Finite Element Methods

Why use FEA? Suppose you expected a solution like this:



Trying to approximate this with one large polynomial is a bad idea.
It's better to use a more flexible structure.

Finite Element Methods

All FEA have the same ingredients:

1. Partition the state space into separate elements.
2. Use local basis functions to project the functions of interest.
3. Solve for the coefficients.

FEA is used frequently in engineering and physics.

Finite Element Methods

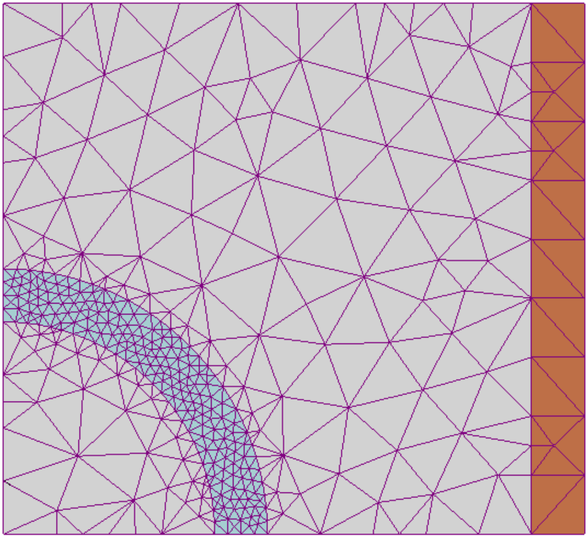
Example from physics, Faraday cages:

The occupant is protected from electricity by being surrounded in a conductive enclosure. This is how microwaves work also.

Source: Wikipedia.

Finite Element Methods

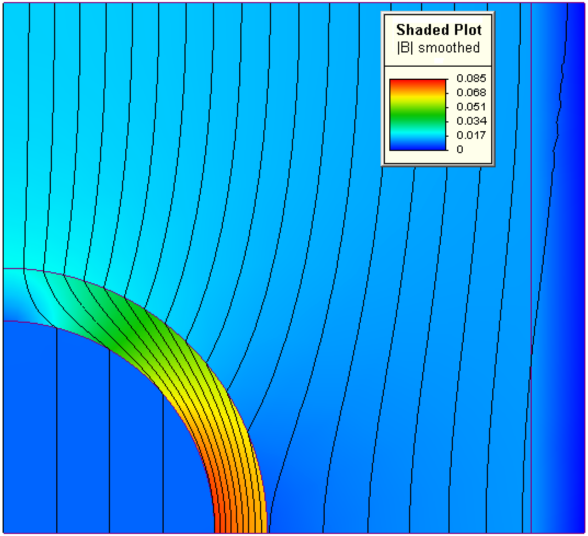
Partition the state space into elements:



Source: Wikipedia.

Finite Element Methods

Find the solution:



Source: Wikipedia.

Finite Element Methods

In this physics example,

- each element has different properties (the light blue is iron, the grey is air, the orange is an electric source)
- the laws of physics are differential equations with boundary conditions which differ by region on the state space
- the mesh is not close to uniform.

In economics, usually

- every element has the same properties (e.g., some region of k and z)
- there is usually a few first order conditions governing the entire space
- the mesh is rectangular and finer where things are expected to be more non-linear.

FEA provides a lot of flexibility and can handle quite challenging problems.

Finite Element Methods

Let's begin with a concrete example and generalize later.

You learned one way to represent a linear interpolant c was to define

$$\psi_i(k) = \begin{cases} \frac{k-k_{i-1}}{k_i-k_{i-1}} & \text{if } k \in [k_{i-1}, k_i] \\ \frac{k_{i+1}-k}{k_{i+1}-k_i} & \text{if } k \in [k_i, k_{i+1}] \\ 0 & \text{o/w} \end{cases}$$

and use

$$c(k;a) = \sum_{i=1}^{n} a_i \psi_i(k).$$

In this formulation, $a_i = c(k_i; a)$ and so a_i is the value of consumption at node k_i.

Finite Element Methods

Let $n = \#\{k_i\}$. In this case,

- There are $n-1$ elements $D_i = [k_i, k_{i+1}]$.
- There are n basis functions are the ψ_i.
- There are n coefficients a_i.

Finite Element Methods

Let's define the residual function:

$$R(k;a) = u'(c(k;a)) - \beta u'(c(\tilde{k}';a))(1 + \alpha \tilde{k}'^{\alpha-1} - \delta)$$

where $\tilde{k}' = -c(k;a) + k^\alpha + (1-\delta)k$.

The Galerkin method chooses a so that

$$\langle R(\cdot;a), \psi_i \rangle = 0 \, \forall \, i$$

or, in different notation,

$$\int_{k_1}^{k_n} R(k;a)\psi_i(k)\omega(k)dk = 0 \, \forall \, i$$

where ω is some weighting function.

Finite Element Methods

The key aspect of FEA is that the basis functions are local. In this case, the Galerkin method reduces from

$$\int_{k_1}^{k_n} R(k;a)\psi_i(k)\omega(k)dk = 0 \,\forall\, i$$

to

$$\int_{[k_{i-1},k_i)\cup[k_i,k_{i+1})} R(k;a)\psi_i(k)\omega(k)dk = 0 \,\forall\, i$$

ignoring the special cases at $i = 1$ and $i = n$.

One advantage of FEA is that evaluating the integral accurately requires only a few quadrature points because the support is small $[k_{i-1}, k_{i+1})$ rather than large $[k_1, k_n]$.

Finite Element Methods

For simplicity, set $\omega = 1$. Then, substituting in ψ_i and rearranging, Define

$$\overline{E}_i(a) = \int_{[k_i, k_{i+1})} R(k;a) \frac{k_{i+1} - k}{k_{i+1} - k_i} dk$$

$$\underline{E}_i(a) = \int_{[k_{i-1}, k_i)} R(k;a) \frac{k - k_{i-1}}{k_i - k_{i-1}} dk.$$

Finite Element Methods

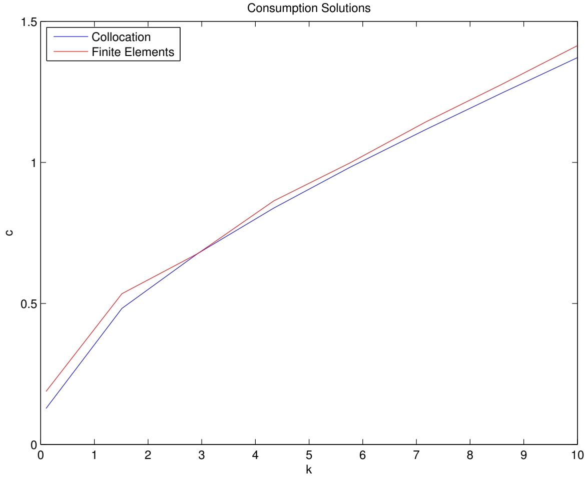
This system of equations can be expressed as

$$E(a) := \begin{bmatrix} \overline{E}_1(a) \\ \underline{E}_2(a) + \overline{E}_2(a) \\ \vdots \\ \underline{E}_{n-1}(a) + \overline{E}_{n-1}(a) \\ \underline{E}_n(a) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

FEA / Galerkin with local basis function tries to find an a such that $E(a) = 0$.

Finite Element Methods

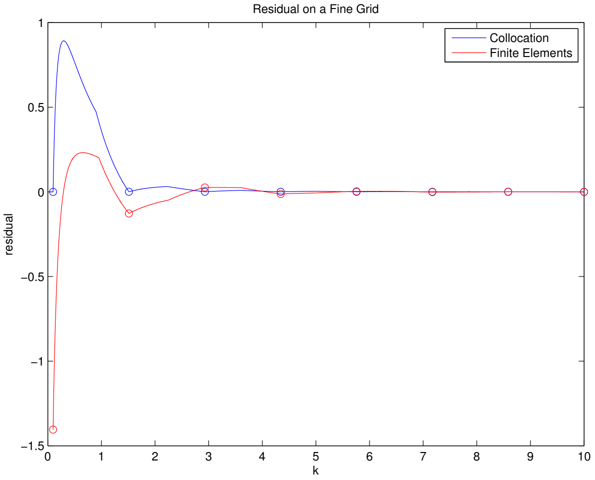
Compare the solution for FEA versus collocation for 8 points.



Noticeable differences. Which one is better?

Finite Element Methods

Here is the residual plotted on a fine grid:



FEA errors integrate to zero. Collocation errors do not.

Finite Element Methods

Now let's generalize this just a little bit:

Suppose our domain D is partitioned into some "elements" D_i, $i = 1, \ldots, I$. That is,

- $D = \bigcup_i D_i$ and
- $D_i \cap D_j = \emptyset$ for $i \neq j$ (or is measure zero).

Further, suppose the basis functions $\{\psi_i\}$ are of the form

$$\psi_i(x) = \mathbf{1}[x \in \bigcup_{j \in J(i)} D_j] g_i(x)$$

and $J(i) \neq \{1, \ldots, I\}$ unless I is small. The g_i function could be anything at this point.

The ψ_i are then local basis functions. They are non-zero on just a few elements (D_j).

Finite Element Methods

Lastly, assume the ψ_i (and in particular the g_i) are such that

$$f(x) = \sum_{i=1}^{n} a_i \psi_i(x) = \{\ p_i(x) \quad \text{if } x \in D_i$$

where p_i is a low-degree polynomial (usually just linear).

Again, let our residual be

$$R(x; a) := R(f(\cdot; a))(x)$$

Apply Galerkin to get the conditions

$$\langle R(\cdot; a), \psi_i \rangle = 0$$

and we're done (after finding an a such that this holds)!

Finite Element Methods

The "devil is in the details" here. f can also be a

- piecewise-polynomial: for a degree 2 polynomial, need an extra coefficient for each interval (i.e., if $n-1$ elements, need $2(n-1)+1$ coefficients.

There are essentially two ways to improve accuracy with FEA:

1. Adding more elements, "h-refinement."
2. Increasing the basis function order, "p-refinement."

Multi-dimensional projection

Multi-dimensional projection

- So far, all the examples have been one-dimensional.
- The multi-dimensional case is a fairly easy extension.

Multi-dimensional projection

The easiest way is tensor-product:

$$f(\mathbf{x}) = \sum_{\mathbf{0} \leq \mathbf{i} \leq \mathbf{n}} a_{\mathbf{i}} (\psi^1_{i_1}(x_1) \dots \psi^d_{i_d}(x_d))$$

where

$$\psi_{\mathbf{i}}(\mathbf{x}) := \psi^1_{i_1}(x_1) \dots \psi^d_{i_d}(x_d)$$

and

$$\mathbf{x} := (x_1, \dots, x_d), \mathbf{i} := (i_1, \dots, i_d), \text{ and } \mathbf{n} := (n_1, \dots, n_d).$$

- If the ψ^j are polynomials, then ψ is complete for C^0 functions.
- If the ψ^j are piece-wise linear, then ... ? speculation: ψ is complete for C^0 functions.
- If ψ^j is orthogonal with respect to ω^j, then ψ is orthogonal with respect to $\omega^1 \dots \omega^d$. (This is easy to show.)

Multi-dimensional projection

Example in two dimensions, tensor-product formulation:

$$f(x,y) = \sum_{i,j} a_{ij} \psi_i^x(x) \psi_j^y(y)$$

For a bilinear interpolant, one can use the tent functions from before for each ψ_i^z:

$$\psi_i^z(z) = \begin{cases} \frac{z - z_{i-1}}{z_i - z_{i-1}} & \text{if } z \in [z_{i-1}, z_i] \\ \frac{z_{i+1} - z}{z_{i+1} - z_i} & \text{if } z \in [z_i, z_{i+1}] \\ 0 & \text{o/w} \end{cases}$$

for $z \in \{x, y\}$ (with grids for both x and y). By inspection,

- $f(x_i, y_j) = a_{ij}$.
- $f(\cdot, y)$ is piecewise linear for all y.
- $f(x, \cdot)$ is piecewise linear for all x.

Multi-dimensional projection

For some residual function

$$R(x, y; a) := R(f(\cdot, \cdot); a))(x, y)$$

Applying Galerkin, we need

$$\langle R(\cdot, \cdot; a), \psi_i(\cdot)\psi_j(\cdot)\rangle = 0$$

or

$$\int_{y_1}^{y_{ny}} \int_{x_1}^{x_{nx}} R(x, y; a)\psi_i(x)\psi_j(y)\omega(x, y)dxdy = 0$$

Sparse grids

An alternative to tensor-product projection that is in fact easier than using complete polynomials is Smolyak sparse grid collocation.

Reference: Krueger, Kubler, and Malin (JEDC 2011).

This is really only applicable to smooth functions (unless using a huge number of points).

Sparse grids

This approximates $f : [-1,1]^d \to \mathbb{R}$ with

$$\hat{f}^{d,\mu}(x) = \sum_{\max(d,\mu+1)\leq|\mathbf{i}|\leq d+\mu} (-1)^{d+\mu-|\mathbf{i}|} \binom{d-1}{d+\mu-|\mathbf{i}|} p^{\mathbf{i}}(x)$$

where

- $\mathbf{i} = (i_1, \ldots, i_d)$ is a multi-index
- d is the dimension of the problem
- μ is an "order of approximation"
- $|\mathbf{i}| := \sum_j i_j$ and
- $p^{\mathbf{i}}(x)$ is a tensor-product interpolant on some grid (specified by the method).

Sparse grids

Typically,

1. the grids used are tensor-products of the Chebyshev-Gauss-Lobatto (CGL) nodes (the extrema of a Chebsyhev polynomial) and
2. the tensor-product interpolant is the Chebyshev polynomial interpolant.

This is because

1. these grids result in nested sets and
2. polynomial interpolants at the CGL nodes have good properties (similar to those at the Chebyshev zeros).

Sparse grids

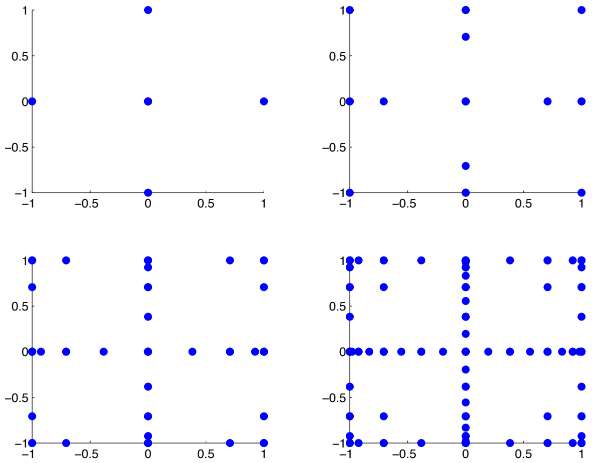


Figure 1: The Grids $\mathcal{H}(3,2)$, $\mathcal{H}(4,2)$, $\mathcal{H}(5,2)$ and $\mathcal{H}(6,2)$

Sparse Grids

- This has obviously been a very gentle introduction to the subject.
- It appears to be a very powerful tool though.
- For Matlab implementation:
  - Grey Gordon has some code on his website: https://sites.google.com/site/greygordon/code
  - There's also this free Matlab toolbox (spinterp): http://www.ians.uni-stuttgart.de/spinterp/
- For Fortran implementation:
  - Dirk Krueger (Penn) has the code for his original paper on his website: http://economics.sas.upenn.edu/∼dkrueger/