Lecture XI

Global Approximation Methods

Gianluca Violante

New York University

Quantitative Macroeconomics

Spectral methods

- Basically, it means that we use polynomial basis. Polynomials are generically nonzero. The motivation for using polynomials come from:

- Weierstrass Theorem: If $C[a,b]$ is the set of all continuous function on $[a,b]$ then for all $f \in C[a,b]$ and $\varepsilon > 0$ there exists a polynomial q for which

$$\sup_{x\in[a,b]} |f(x) - q(x))| < \varepsilon$$

- I.e., there exists a polynomial that approximates any continuous function over a compact domain arbitrarily well in a uniform sense (the approximation error as measured by the sup norm is arbitrarily small.)

Spectral methods

- How do we find a good approximating polynomial and the degree of interpolation needed to reach a desired level of accuracy?

- Another theorem comes handy and tells us some properties of such polynomial. Define:

$$\rho_n(f) = \inf_{q_n} ||f - q_n||_\infty$$

  the smallest of the all the supnorm distances between a given function f and the polynomials q of degree n. Then:

- Equioscillation theorem: If $C[a,b]$ is the set of all continuous function on $[a,b]$ and $f \in C[a,b]$, then there is a unique polynomial of degree n, q_n^* that achieves $\rho_n(f)$. In addition, the polynomial q_n^* is also the unique polynomial for which there are at least $n+2$ points $a \leq x_0 < x_1 < ...x_{n+1} \leq b$ such that for $m = 1$ or $m = -1$

$$f(x_j) - q_n^*(x_j) = m(-1)^j \rho_n(f) \qquad , j = 0, ..., n+1$$

Equioscillation theorem

- This last equation is called the equioscillation property.
- It means that, if we have the best, say, cubic polynomial approximation of f, the maximum error should be achieved at least five times and that the sign of the error should alternate between these points.
- This is a useful theorem because it tells us what our errors should look like when we are trying to find the best approximation.
- If we plot the error and have a very different shape, we know that it is theoretically possible to do better. A general rule of thumb is the closer our error looks to this shape the better the overall approximation.

Jackson’s theorem

- Finally, we have:

- Jackson’s theorem: For all k, if $f \in C^k[a,b]$, then for $n \geq k$

$$\rho_n(f) \leq \frac{(n-k)!}{n!}\left(\frac{\pi}{2}\right)^k\left(\frac{b-a}{2}\right)^k ||f^{(k)}||_\infty$$

- It establishes an upper bound for $\rho_n(f)$

- Under the best polynomial approximation (i) the higher the order of the polynomial and (ii) the smoother the function the better the approximation.

Monomial basis

- The most naive procedure to approximate $y = f(x)$ would be to use as basis functions the monomials, i.e:

$$\phi_j(x) = x^j, \quad j = 0, 1, ..., n$$

with interpolating polynomials of the form:

$$\tilde{f}(x) \equiv q_n(x) \equiv w_0 + w_1 x + w_2 x^2 + ... + w_n x^n$$

and a linear system of $n+1$ equations to solve of the form:

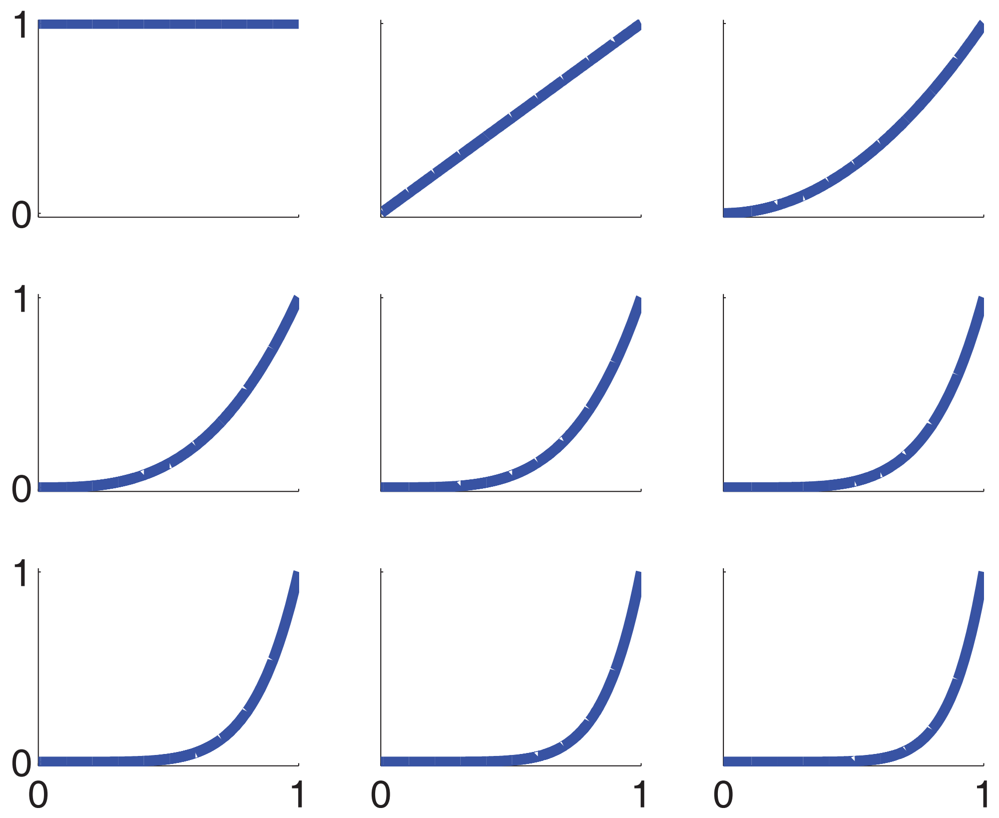
$$\begin{bmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & \dots & x_1^n \\ & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^n \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Monomial basis

- They span space of continuous functions, so in theory they work
- Matrix associated with this linear system is called a Vandermonde matrix, i.e. a matrix with the terms of a geometric progression in each row. It can be shown that it's determinant is always non-zero. Thus theoretically, the system is well-defined.
- Unfortunately, in practice, the Vandermonde matrix is known for being ill-conditioned, especially for high-degree polynomials. This is because, as the degree increases, the functions become progressively less distinguishable making the columns of the Vandermonde matrix nearly linearly dependent.
- In fact, if we define $V_{ij} = x_{i-1}^{j-1}$ as the generic element of the matrix, then

$$\det(V) = \Pi_{1\leq i<j\leq n+1}(x_{i-1} - x_{j-1})$$

Monomials in [0,1]



As n increases these functions look more and more similar

Orthogonal polynomials

- The reason why the monomials are not a good basis for the space of continuous functions is that they are far from being orthogonal.
- Orthogonality is a good property for polynomial basis because (i) they do a good job of sampling the interval —thus the problem is unlikely to be poorly conditioned— and (ii) orthogonal polynomials can be obtained recursively, which is very convenient (Theorem 6.3.1 in Judd)
- Weighting functions. A weighting function, ω on $[a, b]$ is a positive function almost everywhere, such that $\int_a^b \omega(u)\, du < \infty$
- Inner Product. Consider two functions, f and g defined at least on $[a, b]$. The inner product with respect to the weighting function ω is given by $\langle f, g\rangle = \int_a^b f(u)g(u)\omega(u)du$.

Orthogonal polynomials

- Orthogonal Polynomials. The collection of polynomials $\{\phi_j\}_{j=0}^{n}$ is orthogonal on $[a, b]$ with respect to the weighting function ω if and only if $\langle \phi_i, \phi_k \rangle = 0$ for all $i \neq k$ and they are orthonormal if $\langle \phi_i, \phi_i \rangle = 1$ for all i.

- Thus a basis is orthonormal if

$$\int_a^b \phi_i(u)\,\phi_k(u)\,\omega(u)\,du = \begin{cases} 0 & \text{if } i \neq k \\ 1 & \text{if } i = k \end{cases}$$

- Common families of orthogonal polynomials: Chebyshev, Lagendre, Laguerre, Hermite

- Easy to integrate through quadrature!

Weighted residual methods: collocation

- Since we have $n+1$ undetermined coefficients, we need at least $n+1$ conditions to determine these coefficients.

- The most basic conditions imposed are that $\tilde{f}$ interpolates or matches the value of the original function f at $n+1$ selected interpolation nodes $\{x_i\}_{i=0}^{n}$ or:

$$\tilde{f}\left(x_i\right) \equiv \sum_{j=0}^{n} w_j \phi_j\left(x_i\right) = f\left(x_i\right), \quad i = 0, ..., n$$

- Let Φ be the $(n+1 \times n+1)$ matrix with element $\phi_{ij} = \phi_j\left(x_i\right)$, w be the $(n+1 \times 1)$ vector of weights with element w_j and y be the $(n+1 \times 1)$ vector of $f\left(x_i\right)$ or in matrix notation:

$$\Phi w = y$$

  Note that Φ the interpolation matrix should be nonsingular, so that we can obtain $w = \Phi^{-1}y$.

Weighted residual methods: collocation

- One can also choose to match the value of the function at $n_1 + 1$ points and its derivative at $n_2 + 1$ points. Thus, we solve a system of $(n_1 + n_2 + 2) = n + 1$ linear equations and same number of unknowns defined by:

$$\begin{aligned} \sum_{j=0}^{n} w_j \phi_j (x_i) &= f(x_i), i = 0, ..., n_1 \\ \sum_{j=0}^{n} w_j \phi_j' (x_i) &= f'(x_i), i = 0, ..., n_2 \end{aligned}$$

- We then still set the residuals on the nodes exactly to zero:

$$r = y - \Phi w = 0 \to w = \Phi^{-1} y$$

where now some rows of Φ have basis $\{\phi\}$, and others have derivatives of the basis $\{\phi'\}$

Weighted residual methods: LS

- If we have more interpolation nodes than basis functions $(n_i > n_j)$, then we have a “curve fitting” problem: we can define residuals

$$r \equiv y - \Phi w$$

- By minimizing the SSR, we get:

$$w = (\Phi'\Phi)^{-1}\,\Phi' y$$

  which equals $\Phi^{-1}y$ when Φ is squared and invertible since $(\Phi'\Phi)^{-1} = \Phi^{-1}\,(\Phi')^{-1}$

- Basically, we set to zero the weighted residuals, with weight $\frac{\partial r}{\partial w}$:

$$\Phi' r = \Phi'(y - \Phi w) = 0$$

Weighted residual methods: Galerkin

- Define residuals:

$$r(x) \equiv f(x) - \sum_{j=0}^{n} w_j \phi_j(x)$$

- Solve the $n+1$ equations:

$$\int_a^b r(x)\phi_j(x)dx = 0, \quad \text{for} \quad j = 0, 1, ..., n$$

  into $n+1$ unknowns $\{w_j\}$

- The Galerkin choice for the weights are the basis functions used to represent $\tilde{f}$. Similar to LS but done on entire domain.

- Idea: a continuous function $(r(x))$ is zero if it is orthogonal to each member of a complete set of functions

Chebyshev polynomials

- They are trigonometric polynomials defined by

$$\begin{aligned} T_j &: [-1,1] \to [-1,1] \\ T_j &= \cos\left(n \cos^{-1}(x)\right) \end{aligned}$$

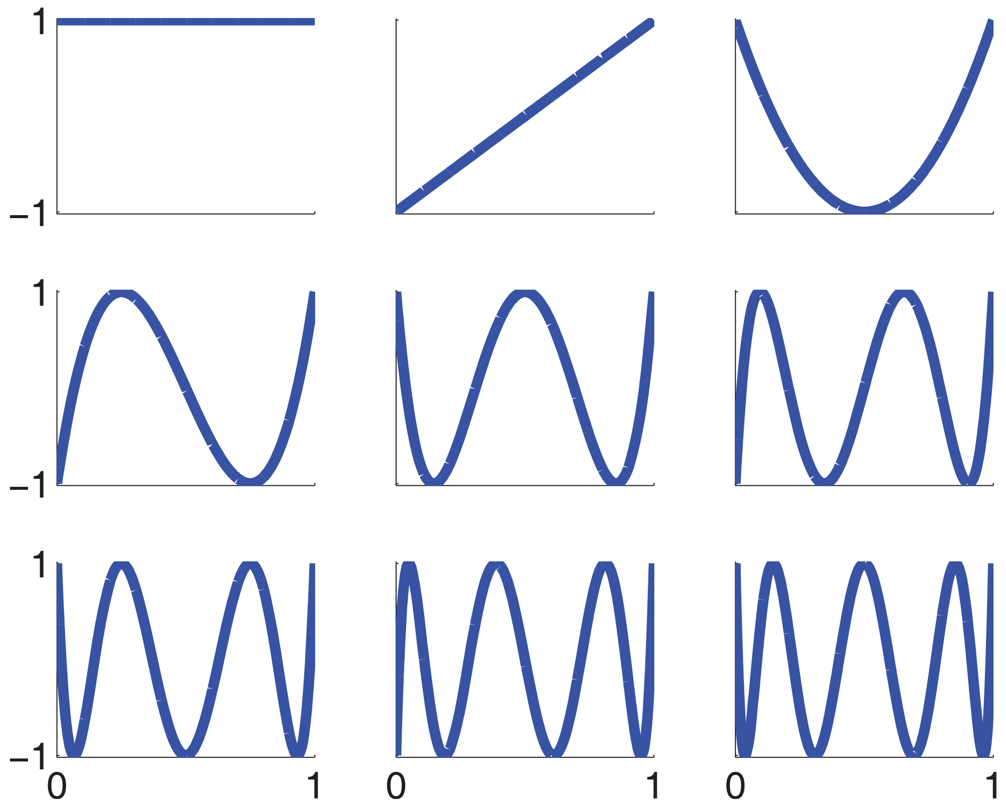
and can be simply constructed recursively as

$$\begin{aligned} T_0(x) &= \cos(0) = 1, \\ T_1(x) &= \cos\left(\cos^{-1}(x)\right) = x, \\ T_{j+1}(x) &= 2xT_j(x) - T_{j-1}(x), \quad \text{for } j \geq 1. \end{aligned}$$

- Therefore:

$$\begin{aligned} T_2(x) &= 2xT_1(x) - T_0(x) = 2x^2 - 1 \\ T_3(x) &= 2xT_2(x) - T_1(x) = 4x^3 - 3x \\ T_4(x) &= 2xT_3(x) - T_2(x) = 8x^4 - 8x^2 + 1 \end{aligned}$$

Chebyshev basis functions in [0,1]



Chebyshev polynomials

- $T_n(x)$ has n distinct roots in $[-1, 1]$ with expression:

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

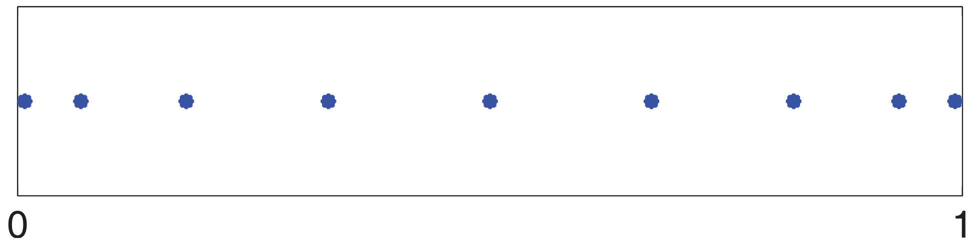
  and it can be shown that the zeros are clustered quadratically towards -1 and $+1$

- $T_n(x)$ has $n+1$ distinct extrema with value of either -1 or 1. Check for $j = 2$ for example.

- They are orthogonal with respect to the weighting function

  $$\omega(x) = \left(1 - x^2\right)^{-1/2}.$$

Chebyshev zeros on [0,1]



- One can show that the interpolation nodes that minimize the error of the Chebyshev interpolation are the zeros of the Chebyshev polynomials themselves.
- More Cheybshev points near the endpoints of the interval where we see larger errors when using an evenly-spaced grid.

Chebyshev polynomials: quality of appx

- Rivlin Theorem: If $\mathcal{T}_n(x)$ is the nth degree Chebyshev approximation of $f \in C^1[-1,1]$ then:

$$||f - \mathcal{T}_n||_\infty \leq \left(4 + \frac{4}{\pi^2}\ln n\right)\rho_n(f)$$

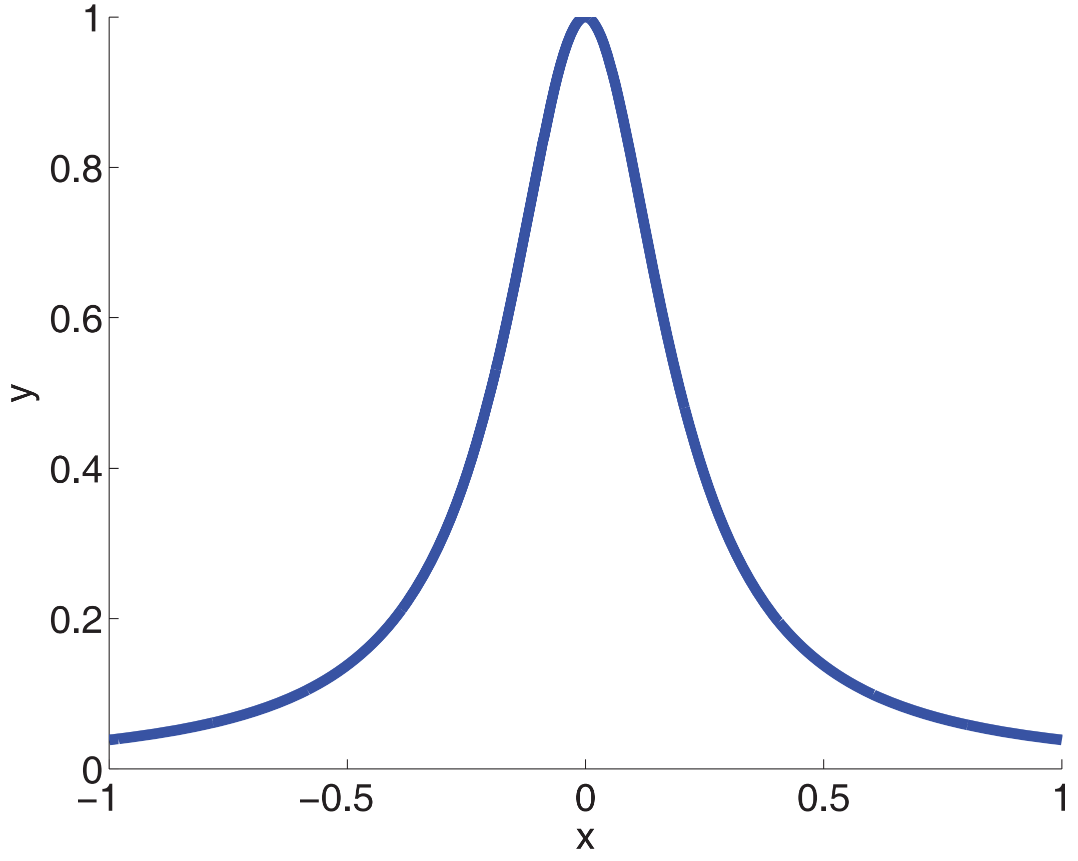
and therefore the Chebyshev approximation is not much worse than the best polynomial appx of the same order in L^∞.

- Worried about the $\log n$ term? You should not be, because remember that by Jackson theorem for a function $f \in C^k$:

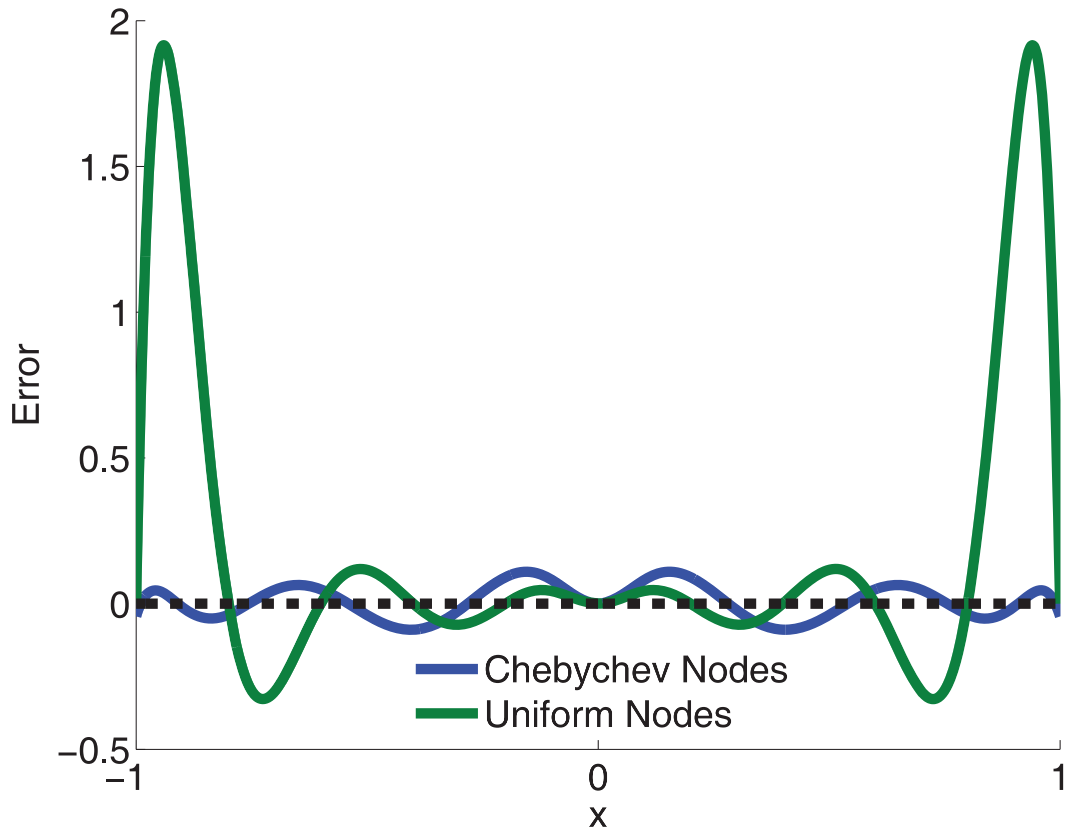
$$\left(4 + \frac{4}{\pi^2}\ln n\right)\rho_n(f) \leq \left(4 + \frac{4}{\pi^2}\ln n\right)\frac{(n-k)!}{n!}\left(\frac{\pi}{2}\right)^k\left(\frac{b-a}{2}\right)^k |f^{(k)}||_\infty$$

and thus, as n grows, the error goes to zero uniformly under the Chebyshev approximation

Runge function on [-1,1]: $f(x) = 1/(1+25x^2)$



Approx. errors with Chebyshev poly (11th order)



Chebyshev polynomials: extended grid

- The Chebyshev nodes are always internal thus these optimal nodes exclude the endpoints.
- In practice, we need to have a grid point at endpoint it because we expect a mass of agents on the constraint for example.
- Then, if we want the first gridpoint to be -1

$$-1 = -\cos\left(\frac{\pi}{2n}\right)\kappa \rightarrow \kappa = \frac{1}{\cos\left(\frac{\pi}{2n}\right)} = \sec\left(\frac{\pi}{2n}\right)$$

- Then, one should use the expanded Chebyshev grid:

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

Chebyshev polynomials: extended grid

- Also note that even though the Chebyshev polynomials are defined over the interval $[-1, 1]$ this interval can be easily converted to $[a, b]$ by linearly transforming the data.
- If $x \in [-1, 1]$ and $z \in [a, b]$ then the nodes become:

$$z_j = \frac{a+b}{2} + \frac{b-a}{2} x_j$$

How to use Chebyshev poly in practice

- Theorem 8.2.3 in Heer and Maussner: If $f \in C^k[-1,1]$ has a Chebyshev representation $f = \sum_{j=0}^{\infty} w_j T_j(x)$ then there is a constant c s.t.:

$$|w_j| \leq \frac{c}{j^k}, \quad j \geq 1$$

- Coefficients (weights) of Chebyshev poly rapidly decreasing in the order of polynomials $\rightarrow$ we can confidently ignore higher order basis

How to use Chebyshev poly in practice

- Here comes handy the discrete orthogonality property.
- Let $\{x_k\}_{k=0}^{m}$ be the first $m+1$ roots of the Chebyshev polynomial. Then for all $i, j < m$:

$$\sum_{k=0}^{m} T_i(x_k) T_j(x_k) = \begin{cases} 0 & \text{if } i \neq j \\ (m+1)/2 & \text{if } i = j \neq 0 \\ (m+1) & \text{if } i = j = 0 \end{cases}$$

- This property allows us to derive simple expressions for the interpolation weights when the grid points used are the Cheybshev nodes.

How to use Chebyshev poly in practice

- Suppose we want to approximate a function f on $[-1,1]$ with an n^{th} order Chebyshev polynomial.
- Compute $m+1 \geq n+1$ Chebyshev interpolation nodes indexed by $k=0,...,m$
- Evaluate the function:

$$y_k = f\left(x_k\right)$$

- The problem is to find the $(n+1)$ weights $\{w_j\}_{j=0}^{n}$ such that:

$$\sum_{j=0}^{n} w_j T_j\left(x_k\right) = y_k, \quad k=0,...,m \geq n$$

i.e., collocation or LS (depending whether $m=n$ or $m>n$)

How to use Chebyshev poly in practice

- First pick an $i \in \{0, 1, ..., n\}$ and multiply both sides by $T_i(x_k)$ and then sum across k

$$\begin{aligned}
\sum_{k=0}^{m} T_i(x_k) y_k &= \sum_{k=0}^{m}\sum_{j=0}^{n} w_j T_i(x_k) T_j(x_k) \\
&= \sum_{j=0}^{n} w_j \sum_{k=0}^{m} T_i(x_k) T_j(x_k)
\end{aligned}$$

From the discrete orthogonality the terms in $i \neq j$ are zero, so

$$\begin{aligned}
\sum_{k=0}^{m} T_i(x_k) y_k &= w_j \sum_{k=0}^{m} T_j(x_k) T_j(x_k) \\
w_j &= \frac{\sum_{k=0}^{m} T_j(x_k) y_k}{\sum_{k=0}^{m} T_j(x_k) T_j(x_k)}
\end{aligned}$$

How to use Chebyshev poly in practice

- Note that the expression above for w_j is that of an OLS estimator that solves

$$\min_{\{w_j\}_{j=0}^{n}} \left\{ \sum_{k=0}^{m} \left[f\left(x_k\right) - \sum_{j=0}^{n} w_j T_j\left(x_k\right) \right]^2 \right\}$$

i.e. the vector of $\{w_j\}$ minimizes the sum of squared residuals between the true function and the approximated function.

How to use Chebyshev poly in practice

- The vector of $\{w_j\}$ can be obtained in one step as:

$$\begin{bmatrix} T_0(x_0) & T_1(x_0) & \dots & T_n(x_0) \\ T_0(x_1) & T_1(x_1) & \dots & T_1(x_1) \\ & \vdots & \ddots & \vdots \\ T_0(x_m) & T_1(x_m) & \dots & T_n(x_m) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_m \end{bmatrix}$$

  where the $\mathbf{T}$ matrix is $(m+1 \times n+1)$ and

$$\mathbf{w} = (\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{y}$$

- Finally, the polynomial approximation of f is

$$f(x) = \sum_{j=0}^{n} w_j T_j(x)$$

Application when function is unknown

- In practice, function f is unknown: value function or decision rule
- How do we adapt this method? Easy.
- Represent a decision rule by polynomial approximation:

$$x' = g(x) = \sum_{j=0}^{n} w_j T_j(x)$$

- Build a residual function from Euler equation

$$\mathcal{R}(x, \mathbf{w}) \equiv \mathcal{R}(x, g(x), g(g(x)))$$

- Choose a weighted residual method to solve for $\mathbf{w} = \{w_j\}$ s.t.:

$$\int_{x_{\min}}^{x_{\max}} \pi(x)\mathcal{R}(x, \mathbf{w})dx = 0$$

Solving the income fluctuation problem

$$\begin{aligned} V(a, y_k) &= \max_{\{c,a'\}} u(c) + \beta \sum_{k' \in Y} \pi(y_{k'}|y_k) V(a', y_{k'}) \\ & \quad s.t. \\ c + a' &\leq Ra + y_k \\ a' &\geq \underline{a} \end{aligned}$$

The Euler equation, once we substitute in the budget constraint, reads:

$$u_c(Ra + y_k - a') - \beta R \sum_{y_{k'} \in Y} \pi(y_{k'}|y_k) u_c(Ra' + y_{k'} - a'') \geq 0.$$

where the strict inequality holds when the constraint binds.

Solving the income fluctuation problem

1. Choose order n of the Chebyshev polynomial for approximation

2. Recall the expression for Chebyshev nodes on $[-1,1]$ as

$$z_i = \cos\left(\frac{2i+1}{2\,(m+1)}\pi\right) \qquad i = 0, ..., m \geq n$$

3. Set the upper bound $\bar{a}$. Define the grid on $[\underline{a}, \bar{a}]$ as:

$$a_i = \frac{\bar{a}+\underline{a}}{2} + z_i \cdot \left(\frac{\bar{a}-\underline{a}}{2}\right) \qquad i = 0, ..., m$$

   Note that we have no control over point allocation on $[a, \bar{a}]$.

4. Let $t\,(a_i)$ be the function that converts a value a_i on $[\underline{a}, \bar{a}]$ back to a value z_i on $[-1, 1]$

Solving the income fluctuation problem

1. Express the saving decision rule as:

$$a' = g\left(a, \mathbf{w}_k\right) = \sum_{j=0}^{n} w_{jk} T_j\left(t\left(a\right)\right)$$

   where T_j are Chebyshev basis of order $j = 0, 1, ..., n$.

2. There is one decision rule for each value of the shock $k = 1, ..., K$, i.e., the Chebyshev coefficients depend on the value of y_k

3. Define the residual function from the Euler equation

$$\mathcal{R}_k\left(a, \mathbf{w}_k\right) \equiv u_c\left(Ra + y_k - \sum_{j=0}^{n} w_{jk} T_j\left(t\left(a\right)\right)\right)$$
$$-\beta R \sum_{y_{k'} \in Y} \pi\left(y_{k'} | y_k\right) u_c\left(R\sum_{j=0}^{n} w_{jk} T_j\left(t\left(a\right)\right) + y_{k'} - \sum_{j=0}^{n} w_{jk'} T_j\left(t\left(\sum_{j=0}^{n} w_{jk} T_j\left(t\left(a\right)\right)\right)\right)\right)$$

Solving the income fluctuation problem

4. Choose your favorite weighted residual method. Set:

$$\int_{\underline{a}}^{\bar{a}} \pi_k(a)\,\mathcal{R}_k(a,\mathbf{w}_k) = 0 \quad k = 1,2,...,K$$

4.a Collocation: Set $m = n$ and residuals equal to zero on the $n+1$ grid points. Equivalent to choosing Dirac as weighting function:

$$\pi_j(a) = \delta(a - a_i) = \begin{cases} 1 & \text{if } a = a_i \\ 0 & \text{otherwise} \end{cases}$$

i.e., appx. exact on the nodes but nothing imposed off nodes.

In practice, solve a $(n+1)K$ eqns into $(n+1)K$ unknowns w:

$$\mathcal{R}_k(a_i,\mathbf{w}_k) = 0, \quad i = 0,1,...,n \text{ and } k = 1,2,...,K$$

by using a multidimensional rootfinding algorithm

Solving the income fluctuation problem

4.b Least square: Define

$$T(t(a)) = \begin{bmatrix} T_0(t(a_0)) & T_0(t(a_1)) & \ldots & T_0(t(a_m)) \\ T_1(t(a_0)) & T_1(t(a_1)) & \vdots & T_1(t(a_m)) \\ \vdots & \vdots & \vdots & \vdots \\ T_n(t(a_0)) & T_n(t(a_1)) & & T_n(t(a_m)) \end{bmatrix} \text{ and } \mathcal{R}_k(a, \mathbf{w}_k) = \begin{bmatrix} R_k(a_0, \mathbf{w}_k) \\ R_k(a_1, \mathbf{w}_k) \\ \vdots \\ R_k(a_m, \mathbf{w}_k) \end{bmatrix}$$

and write the system of $n+1$ equations as

$$T(t(a))\,\mathcal{R}_k(a, \mathbf{w}_k) = 0 \quad \text{for } k = 1, 2, ..., K$$

Once again, use a quasi-Newton method to solve for the vector of Chebyshev coefficients. You can do it separately for each k.

For each k you have a system of $n+1$ equations in $n+1$ unknowns, the coefficient vector $\mathbf{w}_k$

How to deal with occasionally binding constraints

Suppose $a' \geq 0$. Three approaches:

1. Penalty function

$$u_c\left(Ra + y_k - g\left(a, \mathbf{w}_k\right)\right) - P_{a'}\left(g\left(a, \mathbf{w}_k\right)\right) =$$
$$\beta R \sum_{y_{k'} \in Y} \pi\left(y_{k'} | y_k\right) u_c\left(Rg\left(a, \mathbf{w}_k\right) + y_{k'} - g\left(g\left(a, \mathbf{w}_k\right), \mathbf{w}_{k'}\right)\right).$$

2. Impose constraint on policy, i.e. define

$$g = \max\left\{0, \tilde{g}\left(a, \mathbf{w}_k\right)\right\}$$

where $\tilde{g}$ is the solution to the unconstrained problem (thus defined on a grid that starts well below $a_{\min} = 0$) and the EE becomes:

How to deal with occasionally binding constraints

$$u_c\left(Ra + y_k - g\left(a, \mathbf{w}_k\right)\right) + \lambda\left(a, y_k\right) =$$
$$\beta R \sum_{y_{k'} \in Y} \pi\left(y_{k'} | y_k\right) u_c\left(Rg\left(a, \mathbf{w}_k\right) + y_{k'} - g\left(g\left(a, \mathbf{w}_k\right), \mathbf{w}_{k'}\right)\right)$$

with $\lambda\left(a, y_k\right)$ obtained residually on points where constraint is binding

3. Solve for Lagrange multiplier explicitly, i.e., solve for functions $\left\{g\left(a, \mathbf{w}_k\right), \lambda\left(a, \pi_k\right)\right\}$ both approximated by Chebyshev polynomials such that:

$$u_c\left(Ra + y_k - g\left(a, \mathbf{w}_k\right)\right) + \lambda\left(a, \pi_k\right) =$$
$$\beta R \sum_{y_{k'} \in Y} \pi\left(y_{k'} | y_k\right) u_c\left(Rg\left(a, \mathbf{w}_k\right) + y_{k'} - g\left(g\left(a, \mathbf{w}_k\right), \mathbf{w}_{k'}\right)\right)$$
$$\lambda\left(a, \pi_k\right) g\left(a, \mathbf{w}_k\right) = 0$$