Chapter 13
Function Approximation

13.1 Introduction

There are numerous instances in which we need to approximate functions of one or several variables. Examples include the perturbation methods considered in Chapters 2-4, the discrete state space methods of Chapter 7, and the weighted residuals methods of Chapter 5. This chapter gathers the most useful tools for this purpose. First, we clarify the notion of function approximation in the next section. We distinguish between local and global methods. The local approximation of the policy functions of the canonical DSGE model in Chapters 2-4 relies on the combination of two tools: Taylor's theorem in one or more dimensions and the implicit function theorem. We cover them in Sections 13.3 and 13.4, respectively.

Global methods approximate a given function with a combination of simple basis functions. Various kinds of polynomials can serve as basis functions. Polynomials also play an important role in numerical integration, which we consider in Section 14.3. Section 13.5 considers monomials as basis functions and the related Lagrange interpolation. One way to overcome the poor performance of the Lagrange polynomial on equally spaced grids is spline interpolation, which employs low-order polynomials piecewise on the domain of a function. We use this technique to implement the finite element methods in Chapter 5 and consider it in Section 13.6. In Section 13.7, we introduce the family of orthogonal polynomials. Members of this family include the Hermite and Chebyshev polynomials. The former play an important role in the numerical approximation of expectations considered in Section 14.4.2. The latter are key to the spectral methods of Chapter 5, so we devote Section 13.8 to them.

B. Heer and A. Maußner, *Dynamic General Equilibrium Modeling*, Springer Texts in Business and Economics, https://doi.org/10.1007/978-3-031-51681-8_13

In Section 13.9, we turn to the problem of approximating a function of several variables. We consider tensor product bases, complete bases, and the Smolyak polynomial on a sparse grid.

13.2 Function Spaces

The tools introduced in Chapter 12 focus on Euclidean space, i.e., n-tuples of real numbers $\mathbf{x} := (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$. With the rules of vector addition and scalar multiplication Euclidean space is a vector space.[1] With one of the vector norms considered in Section 12.4, it is also a normed vector space. This allows one to define neighborhoods and limits. Functional analysis develops the same concepts on spaces, whose elements are functions. Accordingly, it enables us to speak of a function f_n, $n \in \mathbb{N}$, approaching another function f as its limit, which is the concern of function approximation.

Our sketch of concepts is conducted without the mathematical rigor of textbooks on this subject because we require only a few definitions to understand the theorems that characterize the tools in this chapter.

The elements of a function space must have the same domain. Their other properties depend on the specific question at hand. As an example, consider all continuous functions $f : [a, b] \to \mathbb{R}$, which map the closed interval $[a, b] \in \mathbb{R}$ to the real line $\mathbb{R}$. The common symbol for this set is $\mathscr{C}^0([a, b], \mathbb{R})$. For two members f and g of this set, define addition as

$$h(x) := f(x) + g(x), \; x \in [a, b]$$

and scalar multiplication as

$$h(x) := af(x), \; a \in \mathbb{R}, \; x \in [a, b]$$

so that continuity is preserved and the function $h : [a, b] \to \mathbb{R}$ is also a member of the set $\mathscr{C}^0([a, b], \mathbb{R})$. Furthermore, let $|x|$ denote the absolute value of $x \in [a, b]$, and consider the definition

$$\|f\|_1 := \int_a^b |f(x)| \, \mathrm{d}x. \tag{13.1}$$

$\|f\|_1$ satisfies the conditions for a norm (see (12.3)) and is known as the L^1 norm. The space $\mathscr{C}^0([a, b], \mathbb{R})$ with this norm, denoted by $L^1([a, b], \mathbb{R})$,

[1] For the complete list of axioms that define a vector space, see, e.g., Lang (1997), p. 129.

is a normed vector space. A second norm on the space of continuous functions is the maximum or sup norm

$$\|f\|_\infty := \max_{x\in[a,b]} |f(x)|, \tag{13.2}$$

which is used to define uniform convergence. We say that the sequence of functions f_n converges uniformly to f if for $\epsilon > 0$ there exists N such that for all $n > N$

$$\|f_n - f\|_\infty < \epsilon.$$

In the section on orthogonal polynomials, we shall encounter the space of square integrable functions $f : [a,b] \to \mathbb{R}$, which are defined by the condition:

$$\int_a^b |f(x)|^2 \, \mathrm{d}x < \infty. \tag{13.3}$$

Similar to the inner product $\mathbf{x}^T\mathbf{y} \in \mathbb{R}$ of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ in Euclidean n space, there is an inner product denoted by

$$\langle f, g\rangle := \int_a^b f(x)g(x)\,\mathrm{d}x.$$

This gives rise to the least-squares norm

$$\|f\|_2 := \sqrt{\int_a^b f(x)g(x)\,\mathrm{d}x.} \tag{13.4}$$

The vector space of functions with this norm is denoted by $L^2([a,b],\mathbb{R})$.

13.3 Taylor's Theorem

Consider a function f of a single variable $x \in U$, where U is an open subset of $\mathbb{R}$. Taylor's theorem states the following:[2]

[2] Statements of this theorem appear in calculus textbooks and in most texts on mathematics for economists. See, e.g., Lang (1997), p. 111 for the single-variable and p. 408 for the n-variable case. Judd (1998) states the theorem for a singe variable on p. 23. We employ his notation for the n-variable case on p. 239 in our statement below.

Theorem 13.3.1 *Let $f : [a,b] \to \mathbb{R}$ be a $k+1$ times continuously differentiable function on (a,b), let $\bar{x}$ be a point in (a,b), and let $h \in \mathbb{R}$ with $\bar{x}+h \in [a,b]$. Then:*

$$f(\bar{x}+h) = f(\bar{x}) + f^{(1)}(\bar{x})h + f^{(2)}(\bar{x})\frac{h^2}{2} + \cdots + f^{(n)}(\bar{x})\frac{h^k}{k!}$$
$$+ f^{(k+1)}(\xi)\frac{h^{k+1}}{(k+1)!}, \quad \xi \in (\bar{x}, \bar{x}+h).$$

In this statement, $f^{(i)}$ is the i–th derivative of f evaluated at the point $\bar{x}$. The derivative that appears in the rightmost term of this formula is evaluated at some unknown point between $\bar{x}$ and $\bar{x}+h$. If we neglect this term, the formula approximates the function f at $\bar{x}$, and the approximation error is of order $k+1$. By this we mean that the error is proportional to h^{k+1}, where the constant of proportionality is given by $C = f^{k+1}(\xi)/(k+1)!$.

For the n-variable case, the theorem is:

Theorem 13.3.2 *Let $X \subset \mathbb{R}^n$ be an open subset, $\mathbf{x} := [x_1, x_2, \dots, x_n]^T \in X$, $\mathbf{h} := [h_1, h_2, \dots, h_n]^T \in \mathbb{R}^n$ such that $\mathbf{x} + t\mathbf{h} \in X$ for all $t \in [0,1]$. Assume that $f : X \to \mathbb{R}$ is $(k+1)$-times continuously differentiable. Then, there is a $\lambda \in [0,1]$, such that*

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + \sum_{i_1=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_{i_1}} h_{i_1} + \frac{1}{2}\sum_{i_1=1}^{n}\sum_{i_2=1}^{n} \frac{\partial^2 f(\mathbf{x})}{\partial x_{i_1}\partial x_{i_2}} h_{i_1}h_{i_2}$$
$$+ \frac{1}{k!} \underbrace{\sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n}}_{k \text{ summation signs}} \frac{\partial^k f(\mathbf{x})}{\partial x_{i_1}\partial x_{i_2}\dots\partial x_{i_k}} \underbrace{h_{i_1}h_{i_2}\dots h_{i_k}}_{k \text{ terms}}$$
$$+ \frac{1}{(k+1)!}\sum_{i_1=1}^{n} \cdots \sum_{i_{k+1}=1}^{n} \frac{\partial^{k+1} f(\mathbf{x}+\lambda\mathbf{h})}{\partial x_{i_1}\partial x_{i_2}\dots\partial x_{i_{k+1}}} h_{i_1}h_{i_2}\dots h_{i_{k+1}}.$$

An immediate corollary of this theorem is the quadratic approximation of f at $\mathbf{x}$, which follows from this theorem for $k = 2$. Let us arrange the first-order partial derivatives in the row vector

$$\nabla f(\mathbf{x})^T := \left[\tfrac{\partial f(\mathbf{x})}{\partial x_1} \dots \tfrac{\partial f(\mathbf{x})}{\partial x_n} \right],$$

the gradient of f at the point $\mathbf{x}$, and the second-order partial derivatives in the matrix

$$H(\mathbf{x}) := \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_n} \end{bmatrix}, \tag{13.5}$$

the Hessian matrix of f at $\mathbf{x}$. Then,

$$f(\mathbf{x}+\mathbf{h}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T H(\mathbf{x})\mathbf{h}, \tag{13.6}$$

where the approximation error $\phi(\mathbf{h})$, with $\phi(\mathbf{0}) = 0$, has the property

$$\lim_{\substack{\mathbf{h}\to\mathbf{0} \\ \mathbf{h}\neq 0}} \frac{\phi(\mathbf{h})}{||\mathbf{h}||^2} = 0.$$

Similarly, the linear approximation is given by:

$$f(\mathbf{x}+\mathbf{h}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{h}, \tag{13.7}$$

where the error $\phi(\mathbf{h})$ now obeys

$$\lim_{\substack{\mathbf{h}\to\mathbf{0} \\ \mathbf{h}\neq 0}} \frac{\phi(\mathbf{h})}{||\mathbf{h}||} = 0.$$

Consider a map $\mathbf{f} : X \to Y$ that maps points $\mathbf{x}$ of the open subset $X \subset \mathbb{R}^n$ into points $\mathbf{y}$ of the open subset $Y \subset \mathbb{R}^m$:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \Leftrightarrow \begin{cases} y_1 = f^1(x_1, x_2, \ldots, x_n), \\ y_2 = f^2(x_1, x_2, \ldots, x_n), \\ \;\vdots = \vdots \\ y_m = f^m(x_1, x_2, \ldots, x_n). \end{cases} \tag{13.8}$$

If we apply Taylor's theorem to each component of $\mathbf{f}$, we obtain the linear approximation of the nonlinear map $\mathbf{f}$:

$$\mathbf{y} \approx \mathbf{f}(\mathbf{x}) + J(\mathbf{x})\mathbf{h}. \tag{13.9}$$

The matrix

$$J(\mathbf{x}) := \begin{bmatrix} \frac{\partial f^1(\mathbf{x})}{\partial x_1} & \frac{\partial f^1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f^1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f^2(\mathbf{x})}{\partial x_1} & \frac{\partial f^2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f^2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f^m(\mathbf{x})}{\partial x_1} & \frac{\partial f^m(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f^m(\mathbf{x})}{\partial x_n} \end{bmatrix} \tag{13.10}$$

is called the Jacobian matrix of $\mathbf{f}$ at the point $\mathbf{x}$.

13.4 Implicit Function Theorem

In this book, many if not all systems of nonlinear equations that we have encountered are not given in explicit form $\mathbf{y} = \mathbf{f}(\mathbf{x})$, as considered in the previous paragraph. Rather, the relation between $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$ is implicitly defined via

$$\mathbf{g}(\mathbf{x}, \mathbf{y}) = \mathbf{0}_{m\times 1}, \tag{13.11}$$

where $\mathbf{g} : U \to V$, U is a subset of $\mathbb{R}^n \times \mathbb{R}^m$ and V a is subset of $\mathbb{R}^m$. The implicit function theorem[3] allows us to compute the derivative of $\mathbf{y} = \mathbf{f}(\mathbf{x})$ at a solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ of (13.11).

Theorem 13.4.1 (Implicit Function Theorem) *Let U be an open subset in a product $U_1 \times U_2$, $U_1 \subset \mathbb{R}^n$, $U_2 \subset \mathbb{R}^m$, and let $\mathbf{g} : U \to V \subset \mathbb{R}^m$ be a p-times continuously differentiable map. Let $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in U$ with $\bar{\mathbf{x}} \in U_1$ and $\bar{\mathbf{y}} \in U_2$. Let $\mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \mathbf{0}_{m\times 1}$. Assume that $D_y(\mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{y}})) \in \mathbb{R}^{m\times n} : U_2 \to V$ is invertible. Then, there exists an open ball $B \subset U_1$, centered at $\bar{\mathbf{x}} \in U_1$ and a continuous map $\mathbf{f} : B \to U_2$ such that $\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$ and $\mathbf{g}(\mathbf{x}, \mathbf{f}(\mathbf{x})) = \mathbf{0}_{m\times 1}$ for all $\mathbf{x} \in B$. If B is a sufficiently small ball, then $\mathbf{f}$ is uniquely determined and p-times continuously differentiable.*

The expression D_y denotes the Jacobian matrix of $\mathbf{g}$ with respect to the elements of the vector $\mathbf{y}$. This is an $m \times m$ matrix of partial derivatives:

$$D_y(\mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{y}})) = \begin{bmatrix} \frac{\partial g^1(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_1} & \frac{\partial g^1(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_2} & \cdots & \frac{\partial g^1(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_m} \\ \frac{\partial g^2(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_1} & \frac{\partial g^2(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_2} & \cdots & \frac{\partial g^2(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g^m(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_1} & \frac{\partial g^m(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_2} & \cdots & \frac{\partial g^m(\bar{\mathbf{x}},\bar{\mathbf{y}})}{\partial y_m} \end{bmatrix}.$$

[3] See, e.g., Lang (1997) p. 529, Theorem 5.4.

If this matrix is invertible (as required by Theorem 13.4.1), we obtain the Jacobian matrix of $\mathbf{f}$ at $\bar{\mathbf{x}}$ by differentiating $\mathbf{g}(\mathbf{x}, \mathbf{f}(\mathbf{x}))$ with respect to $\mathbf{x}$:

$$J(\bar{\mathbf{x}}) := \mathbf{f}_x(\bar{\mathbf{x}}) = -D_y^{-1}(\bar{\mathbf{x}}, \bar{\mathbf{y}})D_x(\bar{\mathbf{x}}, \bar{\mathbf{y}}), \tag{13.12}$$

where $D_x(\cdot)$ is analogously defined as $D_y(\cdot)$.

13.5 Lagrange Interpolation

13.5.1 Polynomials and the Weierstrass Approximation Theorem

Suppose that we are given a function $f : [a, b] \to \mathbb{R}$ and want to approximate it by a simpler one. Unlike the Taylor series approximation at a point $x \in [a, b]$ we want to employ information from the shape of f on its entire domain. The simpler function we shall consider is a polynomial of degree n defined by

$$p_n(x) := c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n, \; c_n \neq 0, x \in \mathbb{R}. \tag{13.13}$$

The functions $1, x, x^2, ..., x^n$ in this formula are called the monomials, and the $c_k \in \mathbb{R}$ are called the coefficients. Note that the degree n of the polynomial is the largest exponent in this formula with a nonzero coefficient. The set of polynomials of degree at most n build a finite-dimensional vector space. The monomials span this space or, equivalently, build a $n+1$-dimensional base of this space. We can construct other bases $\{\varphi_k\}_{k=0}^n$ from the monomial base via

$$\begin{bmatrix} \varphi_0(x) \\ \varphi_1(x) \\ \vdots \\ \varphi_n(x) \end{bmatrix} = A \begin{bmatrix} 1 \\ x \\ \vdots \\ x^n \end{bmatrix}$$

for some given $(n+1) \times (n+1)$ nonsingular matrix A.

One reason to employ polynomials for function approximation is given by the Weierstrass approximation theorem.[4]

Theorem 13.5.1 (Weierstrass Approximation Theorem) *Let $[a, b]$ be a closed interval, and let f be a continuous function on $[a, b]$. Then, f can be*

[4] See, e.g., Lang (1997), Theorem 2.1, p. 287 and his definition of uniform convergence on p. 179.

uniformly approximated by polynomials; i.e., for each $\epsilon > 0$, there exists N such that for all $n \geq N$

$$\|f(x) - p_n(x)\|_\infty < \epsilon, \ \forall x \in [a, b].$$

This theorem assures us that polynomials are useful for function approximation; however, it does not give us a way to find the appropriate polynomial. A simple requirement on $p_n(x)$ is that it passes through $n+1$ given points. Accordingly, suppose that we have decided to sample $n+1$ distinct points, called the nodes, $x_0 < x_1 < \cdots x_n$, from the interval $[a, b]$ and can compute the related ordinate values $y_i = f(x_i)$. Equivalently, we may be given $n+1$ data points (x_i, y_i), $i = 0, 1, \dots, n$, which we want to approximate by the polynomial (13.13). This latter problem arises, e.g., in Chapter 7, where we know the value function on a grid but also want to evaluate the function between the grid points. The problem is to choose the coefficients c_i of $p_n(x)$ so that $y_i = p_n(x_i)$, $i = 0, 1, \dots n$. This give rise to $n+1$ linear equations:

$$\underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}}_{=:\mathbf{y}} = \underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}}_{=:X} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ \dots \\ c_n \end{bmatrix}}_{=:\mathbf{c}}. \tag{13.14}$$

Since the points $x_0, x_1, \dots, x_n$ are distinct, the $n+1$ rows of matrix X are linearly independent and the system has a unique solution given by

$$\mathbf{c} = X^{-1}\mathbf{y}. \tag{13.15}$$

13.5.2 Lagrange Interpolating Polynomial

A second approach to finding the interpolating polynomial is Lagrange interpolation. Consider Figure 13.1 and the problem of computing the straight line from (x_0, y_0) to (x_1, y_1) that approximates the function $f(x)$ in the interval $[x_0, x_1]$. This is equivalent to solving the system

$$\begin{aligned} y_0 &= c_0 + c_1 x_0 \\ y_1 &= c_0 + c_2 x_1 \end{aligned}$$

for c_0 and c_1 yielding

$$\begin{aligned}\hat{f}(x) := p_1(x) &= f(x_0) + \frac{f(x_1)-f(x_0)}{x_1-x_0}(x-x_0)\\ &\equiv \underbrace{\frac{x_1-x}{x_1-x_0}}_{=:L_{1,0}(x)} y_0 + \underbrace{\frac{x-x_0}{x_1-x_0}}_{=:L_{1,1}(x)} y_1.\end{aligned} \tag{13.16}$$

This is the linear interpolation formula. It has the property of being shape preserving, which is important if we want to approximate a function with a known shape, e.g., the value function in Chapter 7. In particular, if f is monotonically increasing and concave on $[a,b]$, then $\hat{f}(x)$ preserves these properties.

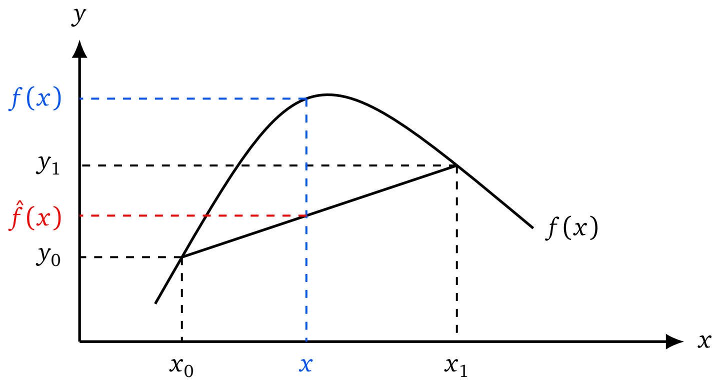

Figure 13.1 Linear Interpolation

We use the second equation in (13.16) to introduce the Lagrange interpolating polynomial. First, note that both $L_{1,0}(x)$ and $L_{1,1}(x)$ as defined in (13.16) are polynomials of degree one, which explains the first index. Second, observe that the term $L_{1,0}$ is equal to one at the point $x = x_0$, while the term $L_{1,1}$ is equal to one at the point $x = x_1$, which explains the second index. Furthermore, $L_{1,0}(x_1) = 0$ and $L_{1,1}(x_0) = 0$. To generalize this formula to a polynomial of degree n that passes trough $n+1$ distinct points (x_i, y_i), $i = 0, 1, \dots n$, define the k-th Lagrange polynomial by

$$L_{n,k}(x) := \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \tag{13.17}$$

so that $L_{n,k}(x_k) = 1$ and $L_{n,k}(x_i) = 0$ for $i \neq k$. Then, it is apparent that the polynomial

$$p_n(x) = L_{n,0}(x)y_0 + L_{n,1}(x)y_1 + \cdots + L_{n,n}(x)y_n \tag{13.18}$$

passes through the $n+1$ points (x_i, y_i). Note that the polynomial (13.13) with coefficients given by (13.15) and the Lagrange polynomial (13.18) are equivalent ways to write the solution of the interpolation problem.

13.5.3 Drawbacks

Lagrange polynomials have several drawbacks: They can oscillate near the end points of the interval $[a, b]$, low- and high-order monomials differ greatly in size, and bases from higher-order monomials are nearly multicollinear.

OSCILLATORY BEHAVIOR. First, consider the interpolation error, which is given by the term on the right-hand side of the following equation:[5]

$$f(x) - p_n(x) = \frac{f^{(1+n)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n}(x - x_i), \quad \xi(x) \in (a, b).$$

This error is mainly determined by the polynomial $l(x) = (x - x_0)\ldots(x - x_n)$. On equally spaced grids, this polynomial oscillates near the boundary of the interval $[a, b]$ for large n. Figure 13.2 illustrates the consequences of this behavior for the Runge function $f(x) := 1/(1 + x^2)$ on the interval $[-5, 5]$. The polynomial $p_n(x)$ employs the nodes $x_i = -5 + 10i/n$, $i = 0, 1, \ldots, n$. Clearly, the higher-degree polynomial $p_{10}(x)$ performs worse than the polynomial $p_5(x)$ at both the left and right ends of the interval.[6]

A careful choice of the interpolation nodes x_i may overcome this problem. Figure 13.3 compares the Lagrange polynomial $p_{10}(x)$ on an equally spaced grid with a Lagrange polynomial of the same order on a grid that consists of the 11 zeros of the Chebyshev polynomial $T_{11}(x)$ (see Section 13.8) shown as green dots. Note that there are more points near the left and right boundaries of the interval $[-5, 5]$ and that the large swings around the true function disappear.

[5] For this formula, see, e.g., Stoer and Bulirsch (2002), p. 49 or Quarteroni et al. (2007), equation (8.7) on p. 335.

[6] See, e.g., Walter (2014), pp. 93f. and Quarteroni et al. (2007), p. 337 for similar examples.

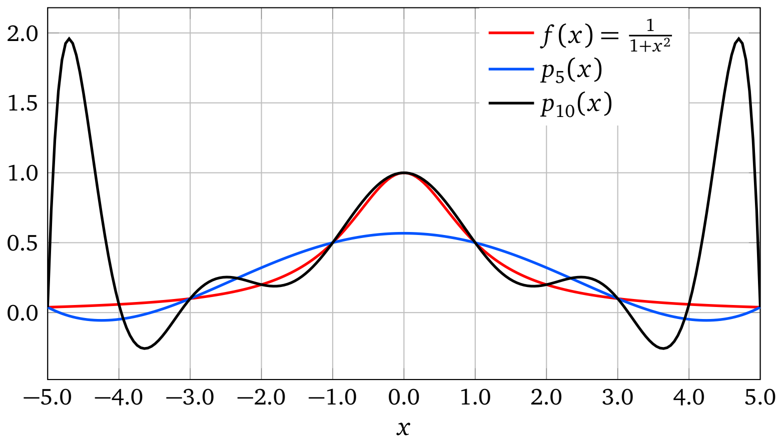

Figure 13.2 Polynomial Approximation of the Runge Function on an Equally Spaced Grid

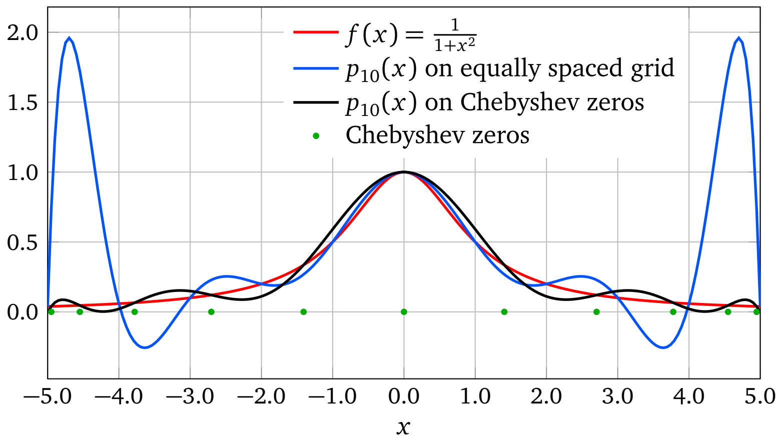

Figure 13.3 Polynomial Approximation of the Runge Function on Chebyshev Zeros

Size Differences and Near Multicollinearity. Figure 13.4 shows the graphs of several monomials over the interval $[0, 1.2]$. Within the interval $[0, 1]$, the graphs are close together and they separate quickly from each other to the right of the point $x = 1$. The large differences in size are an obstacle to root-finding and minimization algorithms that operate on algebraic polynomials. The similarity, in particular of the higher order members of this family, means that the columns of matrix X in equation (13.14) are nearly multicollinear so that the solution $\mathbf{c}$ is numerically unstable: Small differences in the elements of X can induce large changes in the elements of the coefficients $\mathbf{c}$. An indicator of this problem is the condition number of $X^T X$.[7] Greene (2012) considers values in excess of 20 as indicative of a problem. For example, the matrix $X = [x^1, x^2, x^5, x^{10}, x^{15}]$ with 100 nodes equally spaced on the interval $x \in [0, 1.2]$ has a condition number of over 208.

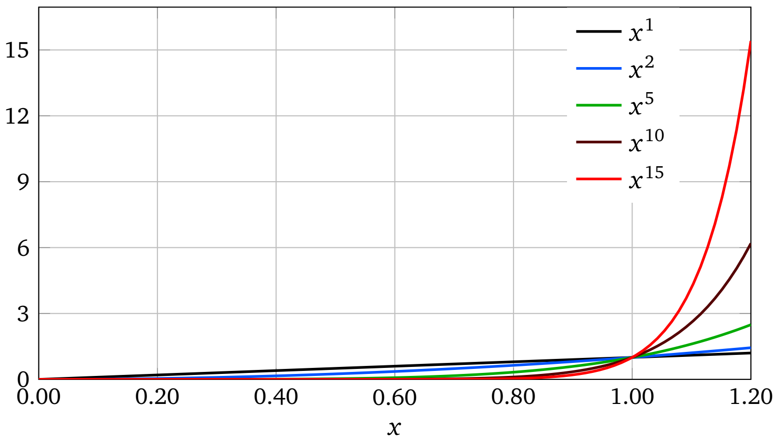

Figure 13.4 Monomials on [0,1.2]

One way to address these drawbacks is spline interpolation, which is considered in the next section.

13.6 Spline Interpolation

Spline interpolation employs low-order monomials to approximate the function $f : [a, b] \to \mathbb{R}$ on subintervals of $[a, b]$ and joins these local approximations to build $\hat{f}(x)$ on $[a, b]$. Figure 13.5 illustrates this principle. We choose $n+1$ points x_i, $i = 0, 1, \ldots n$ that divide the interval $[a, b]$ into n subintervals, which are not necessarily of the same length. On each of them, a low-order polynomial

$$p_i(x) := c_{i0} + c_{i1}(x - x_i) + \cdots + c_{in}(x - x_i)^n, \; x \in [x_i, x_{i+1}]$$

locally approximates the function f.

[7] See Section 12.9.6 on the definition of this concept.

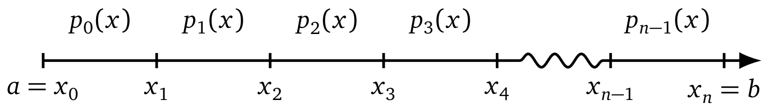

Figure 13.5 Spline Interpolation

Usually, the order of the local polynomials is not larger than three. For ease of exposition, we use the first two, three, and four letters of the alphabet to represent the coefficients of a linear, a quadratic, and a cubic polynomial. Furthermore we take as given $n + 1$ pairs of observations (x_i, y_i), $y_i = f(x_i)$.

13.6.1 Linear Splines

Linear polynomials are defined by

$$p_i(x) = a_i + b_i(x - x_i),\ x \in [x_i, x_{i+1}].$$

Their $2n$ coefficients are determined by two conditions:

1) At nodes x_i, $i = 0, 1, \dots n-1$, the polynomial agrees with the values of the function $y_i = f(x_i)$.
2) At the $n-1$ neighboring nodes, the polynomials agree:

$$p_i(x_{i+1}) = p_{i+1}(x_{i+1}).$$

It is easy to see (and therefore left to the reader) that these conditions uniquely determine the parameters. For $i = 0, 1, \dots, n$, the parameters are determined by

$$a_i = y_i,$$
$$b_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

so that on each subinterval, the polynomial $p_i(x)$ coincides with the linear algebraic polynomial encountered in the previous section (see equation (13.16)).

In the GAUSS programming language, we implement linear splines with two procedures. `LSpline_coef` computes the coefficients and must be called first. Afterwards, the user can evaluate the spline repeatedly with the procedure `LSpline_eval`. The latter function employs binary search, encoded in the procedure `Find`, to find the index i with the property $x \in [x_i, x_{i+1}]$ for $x \in [a, b]$ (see Figure 13.5).

Linear splines preserve the continuity of a function, its monotonicity and its convexity. In many applications, for instance in root-finding and optimization, it is advantageous for the approximate function to be differentiable everywhere and not only within the subintervals. Quadratic splines, defined by

$$p_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2, \ x \in [x_i, x_{i+1}].$$

have one more degree of freedom that can be used to ensure that the derivatives of $p_i(x)$ and $p_{i+1}(x)$ agree at neighboring nodes:

$$b_i + 2c_i(x_{i+1} - x_i) = b_{i+1}.$$

The problem with quadratic polynomials, however, is that these additional $n-1$ conditions together with the $2n$ conditions from $p_i(x_i) = y_i$ and $p_i(x_{i+1}) = p_{i+1}(x_{i+1})$, $i = 0, \dots, n$, determine only $3n-1$ of the $3n$ parameters and it is difficult to argue for a specific additional condition. For instance, one can either supply a condition for the derivative at x_0 or at x_n but not for both endpoints. Cubic splines do not suffer from this problem and are thus widely applied in interpolation.

13.6.2 Cubic Splines

The cubic polynomial

$$p_i(x) := a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \ x_i \in [x_i, x_{i+t}]$$

has four free parameters. To determine their values we impose the following conditions:

1) The approximation is exact at the grid points, $y_i = p_i(x_i)$:

$$y_i = a_i \ \ i = 0, \dots, n.$$

2) At the $n-1$ neighboring nodes, the polynomials agree:

$$\begin{aligned}&p_i(x_{i+1}) = p_{i+1}(x_{i+1})\\ &\Rightarrow a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 = a_{i+1}.\end{aligned}$$

3) At the $n-1$ neighboring nodes, the first and second derivatives agree:

$$\begin{aligned}
&p_i'(x_{i+1}) = p_{i+1}'(x_{i+1})\\
&\quad \Rightarrow b_{i+1} = b_i + 2c_i(x_{i+1}-x_i) + 3d_i(x_{i+1}-x_i)^2,\\
&p_i''(x_{i+1}) = p_{i+1}''(x_{i+1})\\
&\quad \Rightarrow c_{i+1} = c_i + 3d_i(x_{i+1}-x_i).
\end{aligned}$$

These conditions yield $4n-2$ linear equations in the $4n$ unknowns a_i, b_i, c_i, d_i, leaving us two conditions short of fixing the coefficients. If we know the derivative of f at both endpoints, we can supply the clamped boundary condition $f'(x_0) = p_1'(x_0)$ and $f'(x_n) = p_n'(x_n)$. In most applications, this information is not available. Thus, we may use the secant over the first and the last subinterval, giving the conditions $p_1'(x_0) = (y_1-y_0)/(x_1-x_0)$ and $p_n'(x_n) = (y_n-y_{n-1})/(x_n-x_{n-1})$. The respective cubic spline is called the secant Hermite spline. A third solution is the natural spline that sets the second derivative at the endpoints equal to zero: $p_1''(x_0) = p_n''(x_n) = 0$. Finally, the 'not-a-knot' condition demands that the third derivative agrees on the boundaries of the first and the last subinterval: $p_1'''(x_1) = p_2'''(x_1)$ and $p_{n-2}'''(x_{n-1}) = p_n'''(x_{n-1})$.

The coefficients of cubic splines are fast to compute since the set of equations sketched above is not only linear but can be reduced to a tridiagonal system in the coefficients c_i. We implement the cubic spline interpolation following Press et al. (1992) Section 3.3 in two GAUSS procedures. The procedure `CSpline_coef` computes the coefficients of the quadratic part of the spline from given vectors **x** and **y** that supply the data point (x_i, y_i). It uses the not-a-knot condition. After a call to this function we can use `CSpline_eval` , which returns the value of the spline at a given point $x \in [a,b]$.

In MATLAB®, the command `griddedinterpolant` implements linear and cubic splines. For the latter, it also uses the not-a-knot condition.

Figure 13.6 shows the linear and cubic spline approximations of the Runge function. The grid is determined from the same 11 values of $x \in [-5,5]$ employed in Section 13.5.3 to determine the Lagrange polynomial. A glance at Figure 13.3 should convince the reader that both splines are more appropriate for approximating this function than the Lagrange polynomials.

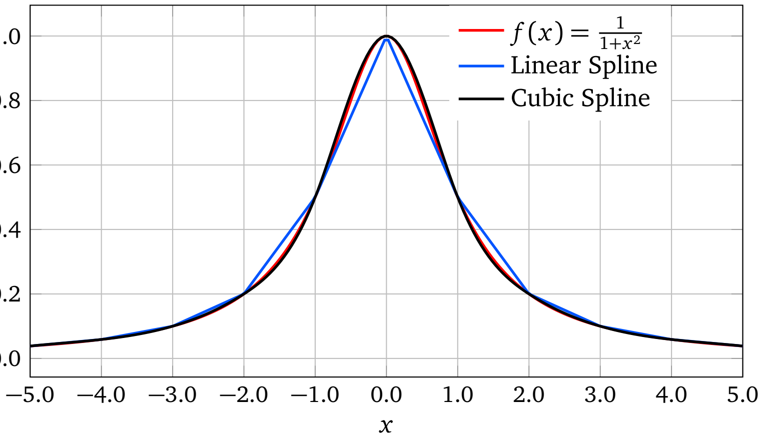

Figure 13.6 Spline Interpolation of the Runge Function

13.7 Orthogonal Polynomials

13.7.1 Orthogonality in Euclidean Space

Recall from the geometry of the Euclidean n-space that the angle θ between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ satisfies (see, e.g., Lang (1986), p. 25):

$$\cos\theta \, ||\mathbf{x}||_2 ||\mathbf{y}||_2 = \mathbf{x}^T \mathbf{y}.$$

Accordingly, two vectors are perpendicular or orthogonal, $\cos\theta = 0$, if their scalar product $\langle \mathbf{x}, \mathbf{y} \rangle$ vanishes (see Figure 13.7):

$$0 = \langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^T \mathbf{y}.$$

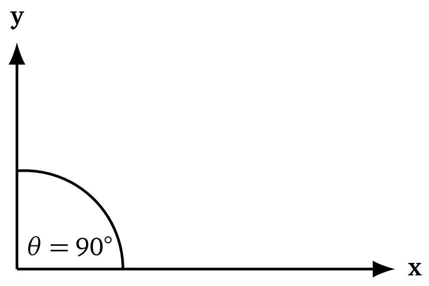

Figure 13.7 Orthogonal Vectors

The orthogonal projection of a vector $\mathbf{z} \in \mathbb{R}^n$ onto the space spanned by two linearly independent vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, therefore, is defined by

$$[\mathbf{x}, \mathbf{y}]^T (\mathbf{z} - \hat{\mathbf{z}}) = 0, \quad \hat{\mathbf{z}} = \gamma_1 \mathbf{x} + \gamma \mathbf{y}$$

and illustrated in Figure 13.8.

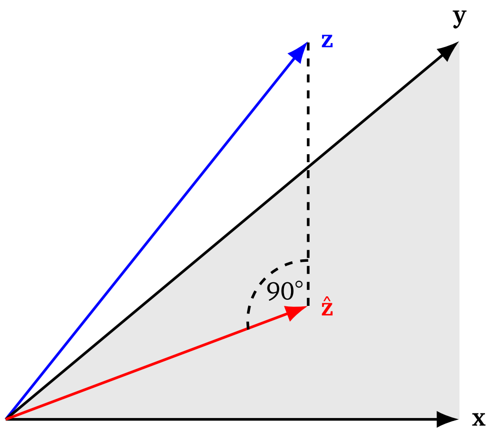

Figure 13.8 Orthogonal Projection in Euclidean Space

A reader acquainted with the linear regression model will recall that this is the principle that determines the least-squares estimator. Given an $n \times m$ matrix X of observations of $m \leq n$ independent variables and an n-dimensional vector $\mathbf{y}$ of observations of one dependent variable, the vector of estimated coefficients $\mathbf{b}$, say, has the property

$$X'(\mathbf{y} - X\mathbf{b}) = \mathbf{0}_{m \times 1}.$$

If they are linearly independent, the columns $\mathbf{x}_i$, $i = 1, \ldots, m$ of the matrix X build a base for the m dimensional subspace of all their linear combinations. If the base consists of mutually orthogonal vectors of unit length, it is called an orthonormal base. With appropriate changes, these definitions carry over to spaces of functions to which we turn now.

13.7.2 Orthogonality in Function Spaces

Consider the space of functions $\mathscr{C}^k([a, b], \mathbb{R})$ that map the interval $[a, b]$ to the real line $\mathbb{R}$ and that are k-times continuously differentiable. This set is a vector space. Unlike Euclidean n-space, however, there is no finite dimensional set of basis functions $\varphi_i(x)$, $i = 0, 1, \ldots$ that allows us to represent $f \in \mathscr{C}^k([a,b],\mathbb{R})$ as a linear combination of them. This can be seen by taking the limit $k \to \infty$ in the statement of Taylor's theorem 13.3.1.

To motivate the notion of orthogonality in a function space, consider the following problem. Assume we want to approximate $f \in \mathscr{C}^0([a,b],\mathbb{R})$ by a linear combination of base functions taken from some family of polynomials $\varphi_k : [a,b] \to \mathbb{R}$:

$$\hat{f}_\gamma(x) := \sum_{k=0}^{K} \gamma_i \varphi_k(x).$$

Suppose further that there is a weight function. This is an integrable function with positive values almost everywhere[8] on $[a,b]$, i.e., $w : [a,b] \to \mathbb{R}_{>0}$, and a finite integral:

$$\int_a^b w(x)\,\mathrm{d}x < \infty.$$

Our goal is to choose parameters $\gamma_k, k = 0, 1, \ldots K$, such that the weighted sum of squared errors $R(\boldsymbol{\gamma}, x) := f(x) - \hat{f}_\gamma(x)$ over all $x \in [a,b]$ reaches a minimum:[9]

$$\min_{\boldsymbol{\gamma}} \int_a^b w(x) \left[f(x) - \sum_{k=0}^{K} \gamma_k \varphi_k(x) \right]^2 \mathrm{d}x. \tag{13.19}$$

The first-order conditions for this problem are

$$0 = 2 \int_a^b w(x) \left[f(x) - \sum_{j=0}^{K} \gamma_j \varphi_j(x) \right] \varphi_k(x)\,\mathrm{d}x,\ k = 0, 1, \ldots, K,$$

which may be rewritten as

$$\int_a^b w(x) f(x) \varphi_k(x)\,\mathrm{d}x = \sum_{j=0}^{K} \gamma_j \int_a^b w(x) \varphi_j(x) \varphi_k(x)\,\mathrm{d}x, \quad k = 0, 1, \ldots, K. \tag{13.20}$$

[8] Intuitively, the qualifier 'almost everywhere' allows $w(x)$ to be nonpositive on a very small set of points. This set must be so small that its size – technically, its measure – equals zero.

[9] This is called a continuous least squares approximation of $f(x)$. We assume that this and the following integrals are finite.

If the integral on the rhs of (13.20) vanishes for $k \neq j$ and equals a constant c_j for $k = j$, it will be easy to compute γ_k from

$$\gamma_k = \frac{1}{c_k} \int_a^b w(x) f(x) \varphi_k(x) \, \mathrm{d}x.$$

This motivates the following definition of orthogonal polynomials: A set of functions $(\varphi_0, \varphi_1, \ldots)$ is called orthogonal with respect to the weight function w if and only if:

$$\int_a^b w(x) \varphi_k(x) \varphi_j(x) \, \mathrm{d}x = 0 \text{ for } k \neq j. \tag{13.21}$$

If, in addition, the integral equals unity for all $k = j$, the polynomials are called orthonormal.[10]

13.7.3 Orthogonal Interpolation

Polynomial interpolation on the zeros of orthogonal polynomials does not suffer from the oscillatory behavior encountered on equidistant grids in Section 13.5.3. In contrast, the following theorem (Theorem 6.5 in Mason and Handscomb (2003)) states that on the zeros of a family of orthogonal polynomials the polynomial (13.13) approximates a given function arbitrarily well in the least-squares norm:

Theorem 13.7.1 *Given a function $f \in \mathscr{C}^0([a, b], \mathbb{R})$, a system of polynomials $\{\varphi_k(x), k = 0, 1, \ldots\}$ (with exact degree k) that are orthogonal with respect to $w(x)$ on $[a, b]$, and a polynomial $p_n(x)$ that interpolates $f(x)$ in the zeros of $\varphi_{n+1}(x)$, then*

$$\lim_{n \to \infty} \int_a^b w(x) (f(x) - p_n(x))^2 \, \mathrm{d}x = 0.$$

[10] Note the analogy with the definition of orthogonality in Euclidean n-space. If we define the scalar product of square integrable functions as in (5.3), condition (13.21) can be written as

$$< \phi_k(x), \phi_j(x) >_w = 0.$$

13.7.4 Families of Orthogonal Polynomials

Orthogonal polynomials satisfy a three-term recurrence relationship:[11]

$$\varphi_k(x) = (a_k x + b_k)\varphi_{k-1}(x) - c_k\varphi_{k-2}(x). \tag{13.22}$$

For our purposes, two members of the family of orthogonal polynomials are of particular relevance: Hermite and Chebyshev polynomials. The former play an important role in the numerical approximation of agent's expectations discussed in Section 14.4. They are defined by (see, e.g., Boyd (2000), p. 505):

$$\begin{aligned} H_0 &= 1,\\ H_1(x) &= 2x,\\ H_k(x) &= 2xH_{k-1}(x) - 2(k-1)H_{k-2}(x) \end{aligned} \tag{13.23}$$

and are orthogonal with respect to the weight function $w(x) := e^{-x^2}$:

$$\int_{-\infty}^{\infty} e^{-x^2} H_k(x)H_j(x)\,\mathrm{d}x = \begin{cases} \sqrt{\pi}2^j j!, & \text{for } k=j,\\ 0, & \text{otherwise .}\end{cases} \tag{13.24}$$

Chebyshev polynomials are key to our implementation of the spectral methods of Chapter 5, and therefore, we consider them in more depth in the next section.

13.8 Chebyshev Polynomials

13.8.1 Definition

There are four kinds of Chebyshev polynomials (named after the Russian mathematician Pafnuty Lvovich Chebyshev (1821-1892)). Our focus here is on polynomials of the first kind.[12] The degree-n polynomial is defined on the domain $z \in [-1,1]$ by the formula[13]

$$T_n(z) := \cos(n\theta(z)), \quad \theta(z) = \cos^{-1}(z). \tag{13.25}$$

Using the properties of the cosine function on the interval $\theta \in [0,\pi]$ as

[11] See, e.g. Golub and Welsch (1969), equation (2.1).

[12] See Mason and Handscomb (2003), Section 1.2 for the second through fourth kind.

[13] We use z rather than x as an argument of the polynomial to remind the reader that the domain of the Chebyshev polynomials is the interval $[-1,1]$ rather than the entire real line $\mathbb{R}$.

Table 13.1
Tabulated Values of the Sine and Cosine Function

θ	$\sin(\theta)$	$\cos\theta$
0	0	1
$\frac{1}{8}\pi$	$\frac{\sqrt{2-\sqrt{2}}}{2}$	$\frac{\sqrt{2+\sqrt{2}}}{2}$
$\frac{1}{4}\pi$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$
$\frac{3}{8}\pi$	$\frac{\sqrt{2+\sqrt{2}}}{2}$	$\frac{\sqrt{2-\sqrt{2}}}{2}$
$\frac{1}{2}\pi$	1	0
$\frac{5}{8}\pi$	$\frac{\sqrt{2+\sqrt{2}}}{2}$	$-\frac{\sqrt{2-\sqrt{2}}}{2}$
$\frac{3}{4}\pi$	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$
$\frac{7}{8}\pi$	$\frac{\sqrt{2-\sqrt{2}}}{2}$	$-\frac{\sqrt{2+\sqrt{2}}}{2}$
π	0	-1

The analytic expressions in the entries of the table employ standard formulas for the sine and cosine functions. See, e.g., Sydsæter et al. (1999), pp. 14f. For instance, the value of $x = \cos(\pi/4) = \sqrt{2}/2$ follows from $x = \sin(\pi/2 - \pi/4) = \sin(\pi/4) = y$ and $x^2 + y^2 = 1$ so that $x = \sqrt{1/2} = 0.5\sqrt{2}$.

presented in Table 13.1, the degree-zero and the degree one polynomial are equal to

$$T_0(z) = 1, \tag{13.26a}$$

$$T_1(z) = \cos(\cos^{-1}(z)) = z. \tag{13.26b}$$

Further members of this family of polynomials follow from the recurrence relation

$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z). \tag{13.26c}$$

To see this, recall the following relation between the cosine and sine functions[14]

$$\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y),$$
$$\cos(x-y) = \cos(x)\cos(y) + \sin(x)\sin(y).$$

Therefore,

$$\begin{aligned} T_{n+1}(z) &= \cos((n+1)\theta(z)) \qquad (13.27a)\\ &= \cos(n\theta(z))\cos(\theta(z)) - \sin(n\theta(z))\sin(\theta(z)),\\ T_{n-1}(z) &= \cos((n-1)\theta(z)) \qquad (13.27b)\\ &= \cos(n\theta(z))\cos(\theta(z)) + \sin(n\theta(z))\sin(\theta(z)). \end{aligned}$$

Adding equations (13.27a) and (13.27b) yields

$$T_{n+1}(z) + T_{n-1}(z) = 2\underbrace{\cos(\theta(z))}_{=z}\underbrace{\cos(n\theta(z))}_{=T_n(z)} = 2zT_n(z).$$

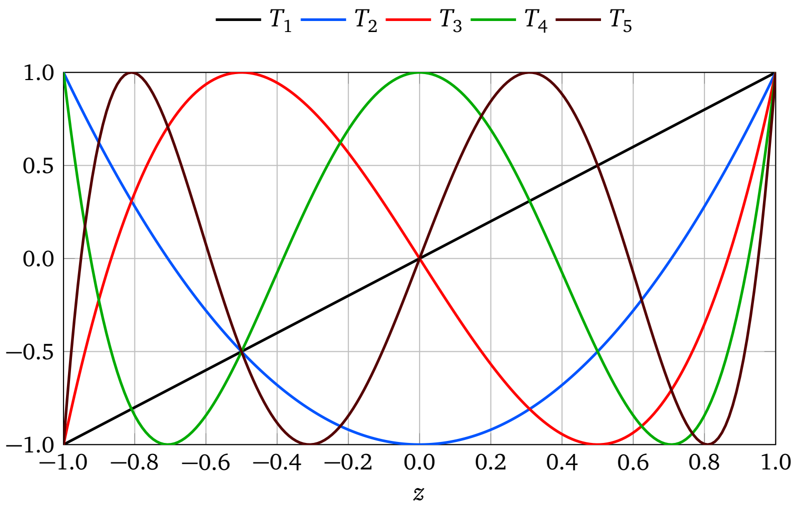

Figure 13.9 Chebyshev Polynomials T_1 through T_5.

Figure 13.9 displays the polynomials T_1 through T_5. The graph of the degree-one polynomial is the 45°line in the plane $[-1,1]\times[-1,1]$. The

[14] See, e.g., Sydsæter et al. (1999), p. 15

degree-two polynomial is a parabola with a vertex at $(0, -1)$. The polynomials of degrees three and above oscillate between -1 and 1.

In general, we will be concerned with functions whose domains are the closed interval $[a, b]$, i.e., $f : [a, b] \to \mathbb{R}$. We employ the bijections

$$\xi : [-1, 1] \to [a, b],\ z \mapsto x = \xi(z) := a + \frac{(1+z)(b-a)}{2}, \tag{13.28a}$$

$$\xi^{-1} : [a, b] \to [-1, 1],\ x \mapsto z = \xi^{-1}(x) := \frac{2(x-a)}{b-a} - 1. \tag{13.28b}$$

to map points between the two domains.

13.8.2 Zeros and Extrema

The cosine function has zeros in the interval $[0, 2\pi]$ at $\theta_1 = \pi/2$ and $\theta_2 = \theta_1 + \pi$ and is periodic with period 2π. Therefore, its zeros occur at

$$\theta_k = \tfrac{1}{2}\pi + (k-1)\pi = \frac{2k-1}{2}\pi,\ k = 1, 2, \ldots,$$

and the Chebyshev polynomial $T_n(z)$, $n = 1, 2, \ldots$ has its n zeros at the points

$$z_k^0 = \cos\left(\frac{2k-1}{2n}\pi\right),\ k = 1, 2, \ldots, n. \tag{13.29}$$

The extrema of the cosine function occur at $k\pi$, $k = 0, 1, \ldots$ (see Table 13.1). Accordingly, the $n+1$ extrema of $T_n(z)$, denoted by $\bar{z}_k$, follow from the condition

$$n \cos^{-1}(\bar{z}_k) = k\pi$$

and are given by

$$\bar{z}_k = \cos\left(\frac{k\pi}{n}\right),\ k = 0, 1, \ldots, n \tag{13.30}$$

with $T_n(\bar{z}_k) = (-1)^k$. They are often referred to as Gauss-Lobatto points (see, e.g., Boyd (2000), p. 570).

13.8.3 Orthogonality

The weight function of the Chebyshev polynomials is the function

$$w(z) := \frac{1}{\sqrt{1-z^2}}$$

plotted in Figure 13.10. In particular, Chebyshev polynomials satisfy:[15]

$$\int_{-1}^{1} \frac{T_i(z)T_j(z)}{\sqrt{1-z^2}}\,\mathrm{d}z = \begin{cases} \pi, & \text{if } i=j=0, \\ \frac{1}{2}\pi, & \text{if } i=j=1,2,\ldots, \\ 0, & \text{if } i \neq j. \end{cases} \tag{13.31}$$

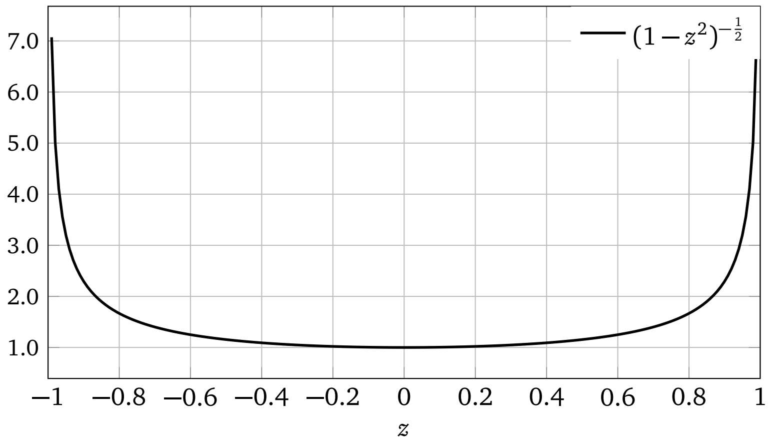

Figure 13.10 Weight Function of the Chebyshev Polynomials.

There is also a discrete version of this property. Let z_k^0 denote the n zeros of $T_n(z)$, and consider two polynomials $T_i(z)$ and $T_j(z)$, $i,j < n-1$. Then:

$$\sum_{k=1}^{n} T_i(z_k^0)T_j(z_k^0) = \begin{cases} n & \text{if } i=j=0, \\ \frac{1}{2}n & \text{if } i=j=1,2,\ldots \\ 0 & \text{if } i \neq j. \end{cases} \tag{13.32}$$

[15] For a proof, see, e.g., Burden and Faires (2016), pp. 508f. or Mason and Handscomb (2003), p. 83.

Mason and Handscomb (2003), pp. 107f. proves this relation from the properties of the trigonometric functions $\sin(\theta)$ and $\cos(\theta)$. We will see in Section 14.3.2 on Gauss-Chebyshev integration that (13.32) is simply the exact Gauss-Chebyshev integral of (13.31).

With respect to the extrema of T_n, the discrete orthogonality relation is (see Mason and Handscomb (2003), p. 109):

$$\sum_{k=0}^{n} \frac{T_i(\bar{z}_k)T_j(\bar{z}_k)}{c_k} = \begin{cases} n, & \text{if } i = j = 0, n, \\ \frac{1}{2}n, & \text{if } i = j = 1, 2, \dots n-1, , \\ 0, & \text{if } i \neq j. \end{cases} \tag{13.33}$$

$$c_k = \begin{cases} 2, & \text{for } k = 0, n, \\ 1, & \text{for } k = 2, \dots, n-1. \end{cases}$$

13.8.4 Chebyshev Regression

We are now prepared to consider the approximation of a given function $f : [a, b] \to \mathbb{R}$ by a linear combination of Chebyshev polynomials. Without loss of generality, we assume for the moment $[a, b] \equiv [-1, 1]$ and define:

$$p_K(z) := \gamma_0 T_0(z) + \gamma_1 T_1(z) + \gamma_2 T_2(z) + \cdots + \gamma_K T_K(z). \tag{13.34}$$

For $\gamma_K \neq 0$, $p_K(z)$ is a Chebyshev polynomial of degree K. We choose the coefficients γ_k, $k = 0, \dots, K$, to minimize the weighted integral

$$I(\gamma_0, \dots, \gamma_K) := \int_{-1}^{1} \frac{[f(z) - p_K(z)]^2}{\sqrt{1 - z^2}} \, dz.$$

The first-order conditions of this problem are

$$\frac{\partial I(\gamma_0, \dots, \gamma_K)}{\partial \gamma_k} = 0, \; k = 0, 1, \dots K.$$

The respective system of equations is:

$$\begin{bmatrix} a_{01} & a_{02} & \dots & a_{0K} \\ a_{11} & a_{12} & \dots & a_{1K} \\ \vdots & \vdots & \ddots & \vdots \\ a_{K1} & a_{K2} & \dots & a_{KK} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_K \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_K \end{bmatrix},$$

$$a_{kj} := \int_{-1}^{1} \frac{T_k(z)T_j(z)}{\sqrt{1 - z^2}} \, dz, \; b_k = \int_{-1}^{1} \frac{f(z)T_k(z)}{\sqrt{1 - z^2}} \, dz.$$

Using the orthogonality (13.31) and Gauss-Chebyshev integration (14.19) on the $m \geq K+1$ nodes of $T_m(z)$ to evaluate the integrals yields:

$$\begin{bmatrix} \pi & 0 & \dots & 0 \\ 0 & \frac{\pi}{2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\pi}{2} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_K \end{bmatrix}, \; b_k = \frac{\pi}{m} \sum_{j=1}^{m} f(z_j^0) T_k(z_j^0), k = 0, \dots, K.$$

so that the coefficients are given by:[16]

$$\gamma_k = \frac{c_k}{m} \sum_{j=1}^{m} f(z_j^0) T_k(z_j^0), \; c_k = \begin{cases} 1, & \text{for } k = 0, \\ 2, & \text{for } k > 0. \end{cases} \tag{13.35}$$

An alternative version, derived from the integration formula (14.20), is:

$$\begin{aligned} \bar{\gamma}_k &= \frac{c_k}{m} \sum_{j=0}^{m} \frac{1}{d_j} f(\bar{z}_j) T_k(\bar{z}_j), \\ c_k &= \begin{cases} 1, & \text{for } k = 0, \\ 2 & \text{for } k > 0, \end{cases} \; d_j = \begin{cases} 2, & \text{for } j \in \{0, m\}, \\ 1, & \text{for } j \in \{1, \dots, m-1\}. \end{cases} \end{aligned} \tag{13.36}$$

We summarize the computation of the coefficients in equation (13.35) in terms of an algorithm:

Algorithm 13.8.1 (Chebyshev Regression)

Purpose: *Approximate a continuous function $f : [a, b] \to \mathbb{R}$ with a linear combination of Chebyshev polynomials.*

Steps:

Step 1: Choose the degree K of the approximating Chebyshev polynomial. Compute $m \geq K+1$ Chebyshev interpolation nodes z_k^0, $k = 1, \dots, m$, from (13.29).

Step 2: For $k = 1, 2, \dots, m$, employ the bijection $\xi(\cdot)$ from equation (13.28a) *and compute $f(\xi(z_k^0))$.*

Step 3: Compute the Chebyshev coefficients:

$$\gamma_0 = \frac{1}{m} \sum_{j=1}^{m} f(\xi(z_j^0)),$$

$$\gamma_k = \frac{2}{m} \sum_{j=1}^{m} f(\xi(z_j^0)) T_k(z_j^0), \quad k = 1, \dots, K.$$

[16] Some authors multiply the coefficient of T_0 by 0.5 so that the distinction of cases in (13.35) is unnecessary.

The algorithm is easy to implement. The GAUSS procedure `Cheb_coef` illustrates this point. Let $T_{K,m}$ denote the matrix of base functions evaluated at the m Chebyshev zeros,

$$T_{K,m} := \begin{bmatrix} 1 & 1 & \dots & 1 \\ z_1^0 & z_2^0 & \dots & z_m^0 \\ T_2(z_1^0) & T_2(z_2^0) & \dots & T_2(z_m^0) \\ \vdots & \vdots & \ddots & \vdots \\ T_K(z_1^0) & T_K(z_2^0) & \dots & T_K(z_m^0) \end{bmatrix}, \tag{13.37}$$

D be the diagonal matrix

$$D = \begin{bmatrix} \frac{1}{m} & 0 & 0 & \dots & 0 \\ 0 & \frac{2}{m} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & \dots & \frac{2}{m} \end{bmatrix}, \tag{13.38}$$

and $\mathbf{y}$ be the vector of function values

$$\mathbf{y} := \begin{bmatrix} f(\xi(z_1^0)) \\ f(\xi(z_2^0)) \\ \dots \\ f(\xi(z_m^0)) \end{bmatrix}.$$

Then, the vector of coefficients is given by

$$\gamma = DT\mathbf{y}. \tag{13.39}$$

Theorem 13.7.1 implies that the Chebyshev polynomial (13.34) converges to the true function f in the least-squares norm. In practice, however, we cannot use an infinite series to approximate a given function. The Chebyshev truncation theorem (Boyd (2000), Theorem 6, p. 47) provides an estimate of the error:

Theorem 13.8.1 (Chebyshev Truncation Theorem) *The error in approximating $f(z)$ by*

$$f_K(z) := \sum_{k=0}^{K} \gamma_k T_k(z)$$

is bounded by the sum of the absolute values of all neglected coefficients:

$$|f(z) - f_K(z)| \le \sum_{i=K+1}^{\infty} |\gamma_i|$$

for all $f(z)$, K, and $z \in [-1, 1]$.

Suppose we observe the rapid convergence of the series $\{\gamma_k\}_{k=0}^K$ with γ_K being close to zero, then we can be confident in ignoring the higher-order terms in (13.34). Boyd (2000), p. 51, summarizes this finding in his Rule-of-Thumb 2:

"we can loosely speak of the last retained coefficient as being the truncation error".

13.8.5 Chebyshev Evaluation

Suppose that we have computed the coefficients $\gamma_0, \gamma_1, \dots, \gamma_K$ of the Chebyshev polynomial (13.34) and want to evaluate it on some point x in the domain of the function $f : [a, b] \to \mathbb{R}$, which the polynomial approximates. The recursive definition (13.26c) implies an efficient method, which we present in terms of an algorithm (see Judd (1998), Algorithm 6.1):

Algorithm 13.8.2 (Chebyshev Evaluation)

Purpose: *Evaluate a K-th degree Chebyshev polynomial at z given the coefficients $\gamma_0, \dots, \gamma_K$*

Steps:

```
Step 1: Initialize: set the elements of the $K+1$-vector $\mathbf{y} = [y_1, \dots, y_{K+1}]^T$
        equal to zero, and set $y_1 = 1$ and $y_2 = z$.
Step 2: For $k = 2, 3, \dots, K-1$, compute:

            $y_{k+1} = 2zy_k - y_{k-1}$.

Step 3: Return $p_n(z) = \sum_{k=0}^K \gamma_k y_k$.
```

A simple way to implement this procedure for an entire vector of points $\mathbf{x} := [x_1, x_2, \dots, x_n]$ consists of three steps:

1) Use (13.28b) and map each element of $\mathbf{x}$ to the corresponding element of the vector $\mathbf{z} := [\xi^{-1}(x_1), \xi^{-1}, \dots, \xi^{-1}(x_n)]$.

2) Employ the recursion (13.26c) to construct the matrix

$$T_{K,n}(\mathbf{z}) := \begin{bmatrix} 1 & 1 & \dots & 1 \\ z_1 & z_2 & \dots & z_n \\ T_2(z_1) & T_2(z_2) & \dots & T_2(z_n) \\ \vdots & \vdots & \ddots & \vdots \\ T_K(z_1) & T_K(z_2) & \dots & T_K(z_n) \end{bmatrix}.$$

3) Compute

$$\begin{bmatrix} \hat{f}(x_1) \\ \hat{f}(x_2) \\ \vdots \\ \hat{f}(x_n) \end{bmatrix} = T'_{K,n}(\mathbf{z})\boldsymbol{\gamma}.$$

An even faster way to solve this problem is the following algorithm (see Corless and Fillion (2013) Algorithm 2.2):

Algorithm 13.8.3 (Clenshaw Algorithm)

Purpose: *Evaluate a K-th degree Chebyshev polynomial at z, given the coefficients $\gamma_0, \dots, \gamma_K$*

Steps:

Step 1: *Initialize: set the $K+1$ elements of the vector $\mathbf{y} = [y_1, \dots, y_{K+1}]^T$ equal to zero and replace y_K with γ_K.*

Step 2: *For $k = K-2, K-3, \dots, 1$, compute*

$$y_k = \gamma_{k+1} + 2zy_{k+1} - y_{k+2}.$$

Step 3: *Return $p_K(z) = (\gamma_0 - y_2) + y_1 z$.*

If we count each multiplication and addition as one floating point operation (FLOP), steps 2 and 3 involve a total of $4n$ FLOPs. Steps 2 and 3 of Algorithm 13.8.2 involve a total of $5n-3$ FLOPs. Our GAUSS procedure `Cheb_eval` implements this algorithm.

13.8.6 Examples

Figure 13.11 illustrates the convergence of Chebyshev polynomials to the Runge function shown in Figures 13.3 and 13.6. Note that the degree

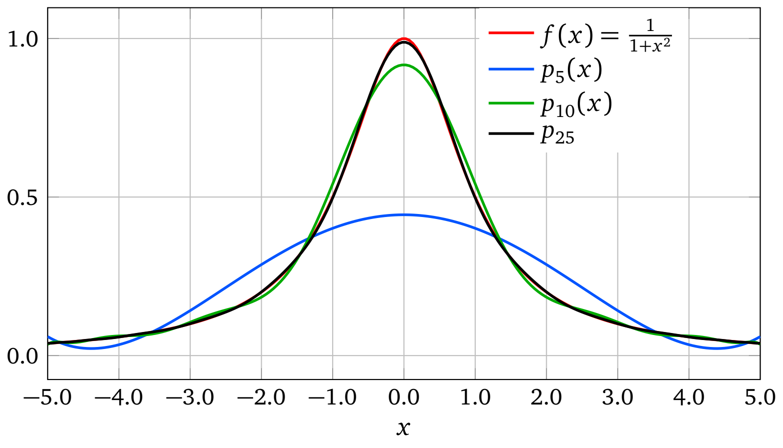

Figure 13.11 Approximation of the Runge Function with Chebyshev Polynomials

$K = 25$ polynomial still has many fewer free parameters than the cubic spline on the 10 subintervals shown in Figure 13.6.

As a second example of the performance of Chebyshev polynomials, we consider a function with a kink:

$$f(x) := \begin{cases} 0, & \text{for } x \in [0,1), \\ x-1, & \text{for } x \in [1,2]. \end{cases} \tag{13.40}$$

This kind of function appears in economic problems with constraints. For instance, in Chapter 7, we consider a model with a binding constraint on investment. As another example, assume that agents supply labor elastically. If the wage is below unemployment compensation, they do not work and the labor supply is equal to zero. For a wage exceeding unemployment compensation, they supply labor, and if the substitution effect dominates the income effect the labor supply increases with the wage rate. The optimal labor supply may look similar to the function in equation (13.40).

Figure 13.12 shows that with increasing order, Chebyshev polynomials approach this function. The degree $K = 55$ polynomial has a maximal error of less than 0.0066 on an equally spaced grid of 200 points over the interval $[0, 2]$.

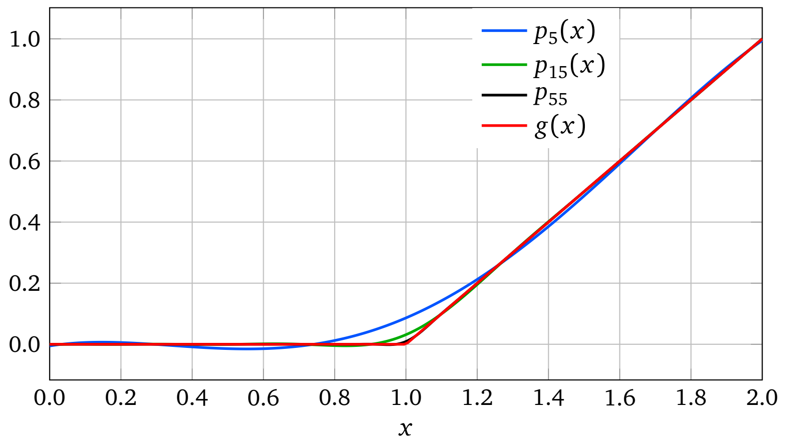

Figure 13.12 Approximation of a Kinked Function with Chebyshev Polynomials

13.9 Multivariate Extensions

The policy functions of DSGE models usually have more than one independent variable. This section considers the extension of the approximation of a given function $f : X \to \mathbb{R}$ defined on a compact subset of the d-dimensional space of real numbers, $X \subset \mathbb{R}^d$. The next subsection employs a straightforward approach. We approximate f by Cartesian products of one-dimensional polynomials. However, this approach suffers from the curse of dimensionality; i.e., the number of coefficients of the respective polynomials grows exponentially with d. A smaller set is the complete set of polynomials. On this set, however, there are more nodes than required to determine the parameters of the collocation solution considered in Chapter 5, and there is no guidance for their selection. The Smolyak algorithm presented in Subsection 13.9.4 solves this problem. It selects collocation points on sparse grids, whose number grows only polynomially with the dimension d.

13.9.1 Tensor Product and Complete Polynomials

Suppose that we want to approximate the function

$$f : X \subset \mathbb{R}^d \to \mathbb{R}, \ \mathbf{x} := [x_1, x_2, \dots, x_d] \mapsto f(\mathbf{x})$$

with polynomials. Let $\varphi_k : [a,b] \to \mathbb{R}$ denote a univariate polynomial from some family of polynomials, and consider the d univariate polynomials p_{K_j}, $j = 1, 2, \dots, d$:

$$p_{K_j}(x) := \gamma_0^j \varphi_0(x) + \cdots + \gamma_{K_j}^j \varphi_{K_j}(x).$$

Let $\mathbf{K} = [K_1, \dots, K_d]^T$ denote the vector of univariate dimensions. The Cartesian product of these d polynomials, $p_{K_1} \times p_{K_2} \times \cdots \times p_{K_d}$, is the d-fold sum

$$p^{\mathbf{K}}(\mathbf{x}) := \sum_{l_1=0}^{K_1} \cdots \sum_{l_d=0}^{K_d} \gamma_{l_1,l_2,\dots,l_d} \varphi_{l_1}(x_1)\varphi_{l_2}(x_2)\cdots\varphi_{l_d}(x_d),$$

$$\gamma_{l_1,l_2,\dots,l_d} := \prod_{j=1}^{d} \gamma_{l_j}^j.$$

For $K_j = K$, $j = 1, \dots, d$, this polynomial has $(1+K)^d$ coefficients, a number that grows exponentially with the number of dimensions d. The single products $\varphi_{l_1} \dots \varphi_{l_d}$ are members of the set

$$\Phi := \left\{ \prod_{j=1}^{d} \varphi_{l_j}(x_j) \middle| l_j = 0, 1, \dots, K \right\}, \tag{13.41}$$

which is an example of a d-fold tensor product base of polynomials.

The complete set of polynomials has fewer members. An example of this set is build by the monomials involved in the Taylor series expansion of f according to Theorem 13.3.2 up to degree K:

$$\mathscr{P}_K^d := \left\{ (x_1^{k_1} x_2^{k_2} \cdots x_d^{k_d}) \middle| \sum_{i=1}^{d} k_i = j, k_i \geq 0, j = 0, 1, \dots, K \right\}.$$

For $d = 2$ and $K = 2$, this set has 6 members:

$$\mathscr{P}_2^2 := \left\{1, x_1, x_2, x_1x_2, x_1^2, x_2^2\right\}.$$

With $\varphi_k(x) := x^k$, the tensor product set has 9 members

$$\Phi = \left\{1, x_2, x_2^2, x_1, x_1x_2, x_1x_2^2, x_1^2, x_1^2x_2, x_1^2x_2^2\right\}.$$

The complete set of polynomials grows only polynomially in the dimension d. For instance, for $K = 2$, the set has $1 + d + d(d+1)/2$ elements (Judd (1998), Table 6.6).

13.9.2 Multidimensional Splines

A spline on a d-dimensional subset $X \subset \mathbb{R}^d$ consists of d-dimensional polynomials restricted to the elements of a d-dimensional grid over X. It is beyond the scope of this book to consider d-dimensional splines in general. Instead, we focus on linear and cubic splines in two dimensions. They are helpful in many of the applications considered in this book.

Let $X := [a_1, b_1] \times [a_2, b_2]$ denote a rectangular subset of $\mathbb{R}^2$, and let $f : X \to \mathbb{R}$ denote a function we want to approximate with a two-dimensional spline on suitably chosen rectangles. To accomplish this, we subdivide both intervals as shown in Figure 13.13. The shaded rectangle corresponds to the subinterval $[x_{1i}, x_{1i+1}] \times [x_{2j}, x_{2j+1}]$.

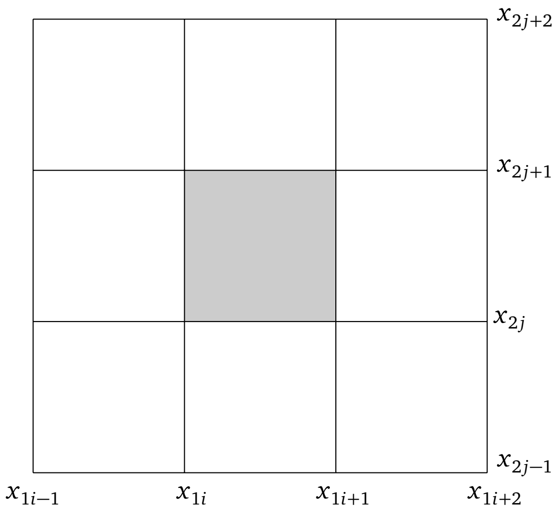

Figure 13.13 Rectangular Grid

Bilinear Interpolation. A bilinear, tensor-product-base polynomial on this subinterval can be written as

$$p_2^{ij}(\mathbf{x}) := c_0 + c_1(x_1 - x_{1i}) + c_2(x_2 - x_{2j}) + c_3(x_1 - x_{1i})(x_2 - x_{2j}).$$

Alternatively, we can write the polynomial as a tensor product of Lagrange polynomials so that we need not compute the coefficients in a separate step. To develop the respective formula, we start with the univariate interpolation of x_1 at the points x_{2j} and x_{2j+1}:

$$\hat{f}(x_1, x_{2j}) = L_{1,0}(x_1) f(x_{1i}, x_{2j}) + L_{1,1}(x_1) f(x_{1i+1}, x_{2j}),$$
$$\hat{f}(x_1, x_{2j+1}) = L_{1,0}(x_1) f(x_{1i}, x_{2j+1}) + L_{1,1}(x_1) f(x_{1i+1}, x_{2j+1}),$$

where[17]

$$L_{1,0}(x_1) = \frac{x_{1i+1} - x_1}{x_{1i+1} - x_{1i}},$$
$$L_{1,1}(x_1) = \frac{x_1 - x_{1i}}{x_{1i+1} - x_{1i}},$$

are the Lagrange polynomials $k = 0$ and $k = 1$ of the variable x_1 seen in equation (13.16). In the second and final step, we interpolate the variable x_2 on $\hat{f}(x_1, x_{2j})$ and $\hat{f}(x_1, x_{2j+1})$,

$$\hat{f}(x_1, x_2) = L_{2,0}(x_2)\hat{f}(x_1, x_{2j}) + L_{2,1}(x_2)\hat{f}(x_1, x_{2j+1})$$

giving the solution:

$$\begin{aligned}\hat{f}(x_1, x_2) =& L_{1,0}(x_1) L_{2,0}(x_2) f(x_{1i}, x_{2j}) \\ &+ L_{1,1}(x_1) L_{2,0}(x_2) f(x_{1i+1}, x_{2j}) \\ &+ L_{1,0}(x_1) L_{2,1}(x_2) f(x_{1i}, x_{2j+1}) \\ &+ L_{1,1}(x_1) L_{2,1}(x_2) f(x_{1i+1}, x_{2j+1}).\end{aligned} \tag{13.42}$$

This procedure can be extended to more than two dimensions. Our GAUSS procedure BLIP implements equation (13.42). MATLAB® provides the command `griddedinterpolant` for linear and cubic spline interpolation for functions of more than one independent variable.

Cubic Splines. A cubic tensor product polynomial on the rectangle

$$[x_{1i}, x_{1i+1}] \times [x_{2j}, x_{2j+1}]$$

with coefficients c_{kl}^{ij} can be written as

$$p_4^{ij}(x_1, x_2) := \sum_{k=1}^{4} \sum_{l=1}^{4} c_{kl}^{ij} (x_1 - x_{1i})^{k-1} (x_2 - x_{2j})^{l-1}.$$

[17] In a slight abuse of the notation introduced in (13.17), we use the first index to refer to the argument of the function $y = f(x_1, x_2)$ and ignore the dependence on the chosen rectangle (i, j).

Similar to bilinear interpolation, we can construct the cubic spline in two dimensions stepwise (see Press et al. (1992) pp. 120ff.). Let n_1 and n_2 denote the numbers of grid points on $[a_1, b_1]$ and $[a_2, b_2]$, respectively. In the first step, we compute for each grid point $x_{1i} \in [a_1, b_1]$ the quadratic coefficients of the cubic spline on $x_{20}, x_{21}, \dots, x_{2n_2}$. They approximate the function $f(x_{1i}, x_2)$. Next, we evaluate the n_1 splines for a given value $x_2 \in [a_2, b_2]$. This yields the points $\hat{f}(x_{10}, x_2), \hat{f}(x_{11}, x_2), \dots, \hat{f}(x_{1n_1}, x_2)$. In the third and final step, we employ these values to compute the spline over $x_{10}, \dots, x_{1n_1}$ and evaluate this spline at x_1. For the GAUSS programming environment, we implement this approach in two functions encoded in Fortran. They can be called via the GAUSS foreign language interface. The procedure `CSpline2_coef` must be called first. It solves the first step. Afterwards, the procedure `CSpline3_eval` provides the interpolated value.

13.9.3 Multidimensional Chebyshev Regression

The coefficients of a tensor product Chebyshev polynomial follow from formulas similar to (13.35). We will demonstrate their derivation for the case of $d = 2$. For notational convenience, we assume the same number of polynomials $K_1 = K_2 = K$ and (with the obvious change of coordinates, $z_i = \xi^{-1}(x_i)$) consider $f(z_1, z_2)$ on the square $[-1, 1]^2$:

$$p^{K,K}(z) := \sum_{k_1=0}^{K} \sum_{k_2=0}^{K} \gamma_{k_1,k_2} T_{k_1}(z_1) T_{k_2}(z_2).$$

Let $\gamma = [\gamma_{0,0}, \dots, \gamma_{0,K}, \gamma_{1,0}, \dots, \gamma_{1,K}, \dots, \gamma_{K,K}]^T$ denote the vector of coefficients. We wish to choose these to minimize the double integral:

$$I(\gamma) := \int_{-1}^{1} \int_{-1}^{1} \left[f(z_1, z_2) - \sum_{k_1=0}^{K} \sum_{k_2=0}^{K} \gamma_{k_1,k_2} T_{k_1}(z_1) T_{k_2}(z_2) \right]^2 \frac{\mathrm{d}z_1 \, \mathrm{d}z_2}{\sqrt{1-z_1^2}\sqrt{1-z_2^2}}.$$

The first-order conditions are:

$$\frac{\partial I(\gamma)}{\partial \gamma_{r_1,r_2}} = 0, \; r_1, r_2 = 0, 1, \dots, K.$$

The respective system of linear equations is:

$$A\gamma = \mathbf{b}.$$

The vector **b** has the typical element:

$$b_i := \int_{-1}^{1}\int_{-1}^{1} \frac{T_{r_1}(z_1)T_{r_2}(z_2)f(z_1,z_2)}{\sqrt{1-z_1^2}\sqrt{1-z_2^2}}\,\mathrm{d}z_1\,\mathrm{d}z_2,$$
$$i = 1 + r_2 + r_1(1+K).$$

The typical element of the matrix A is:

$$a_{i,j} := \int_{-1}^{1} \frac{T_{r_1}(z_1)T_{k_1}(z_1)}{\sqrt{1-z_1^2}}\left[\int_{-1}^{1} \frac{T_{r_2}(z_2)T_{k_2}(z_2)}{\sqrt{1-z_2^2}}\,\mathrm{d}z_2\right]\mathrm{d}z_1,$$
$$j = 1 + k_2 + k_1(1+K).$$

Hence, from the orthogonality of the Chebyshev polynomials (13.31),

$$a_{i,j} = \begin{cases} \pi^2, & \text{for } i = j = 1, \\ \frac{1}{2}\pi^2, & \text{for } i = j = 2,\dots,3+K, \\ \frac{1}{4}\pi^2, & \text{for } i = j = 4+K,\dots(1+K)^2, \\ 0, & \text{for } i \neq j. \end{cases}$$

We approximate the elements of the vector **b** by Gauss-Chebyshev sums on $m \geq 1+K$ nodes of $T_m(z)$:

$$b_i = \frac{\pi}{m}\frac{\pi}{m}\sum_{k_1=1}^{m}\sum_{k_2=1}^{m} T_{r_1}(z_{k_1}^0)T_{r_2}(z_{k_2}^0)f(z_{k_1}^0, z_{k_2}^0).$$

Therefore, the formula:

$$\gamma_{r_1,r_2} = \frac{4}{c_{r_1}c_{r_2}}\frac{1}{m^2}\sum_{k_1=1}^{m}\sum_{k_2=1}^{m} T_{r_1}(z_{k_1}^0)T_{r_2}(z_{k_2}^0)f(z_{l_1}^0, z_{l_2}^0),$$
$$c_{r_1}, c_{r_2} = \begin{cases} 2, & \text{for } r_1, r_2 = 0, \\ 1, & \text{otherwise.} \end{cases} \tag{13.43}$$

determines the coefficients of $p^{K,K}(z_1,z_2)$. Similar to the univariate case, we can compute the coefficients from a matrix product. Let

$$F := \begin{bmatrix} f(z_1^0, z_1^0) & \dots & f(z_1^0, z_m^0) \\ \vdots & \ddots & \vdots \\ f(z_m^0, z_1^0) & \dots & f(z_m^0, z_m^0) \end{bmatrix}, \Gamma := \begin{bmatrix} \gamma_{0,0} & \cdots & \gamma_{0,K} \\ \vdots & \ddots & \vdots \\ \gamma_{K,0} & \cdots & \gamma_{K,K} \end{bmatrix}$$

denote the matrix of function values at the m^2 nodes of $T_m(z)$ and the coefficient matrix, respectively. Let the matrices $T_{K,m}$ and D be defined as in (13.37) and (13.38), respectively. Then, the coefficient matrix follows from:

$$\Gamma = DT_{K,m}FT_{K,m}'D. \tag{13.44}$$

The obvious generalization of (13.43) to the d-dimensional case is:

$$\begin{aligned} \gamma_{r_1,\dots,r_d} &= \frac{2^d}{c_{r_1}\cdots c_{r_d}} \frac{1}{m^d} \sum_{k_1=1}^{m} \cdots \sum_{k_d=1}^{m} T_{r_1}(z_{k_1}^0)\cdots T_{r_d}(z_{k_d}^0) f(z_{k_1}^0, \dots, z_{k_d}^0), \\ c_{r_j} &= \begin{cases} 2, & \text{for } r_j = 0, j = 1, \dots, d, \\ 1, & \text{otherwise.} \end{cases} \end{aligned} \tag{13.45}$$

13.9.4 The Smolyak Polynomial

The seminal paper of Smolyak (1963) presents a method to construct a polynomial with a relatively small number of coefficients that interpolates a function of d variables.[18] The collocation nodes form a sparse grid within the d-dimensional hypercube $[-1, 1]^d$. The first application of Smolyak's method to an economic problem was an article by Krueger and Kubler (2004). The presentation here follows Malin, Krueger, and Kubler (2011) and Judd, Maliar, Maliar, and Valero (2014). We begin with the construction of the grid.

Construction of the Grid. Let $\mathcal{G}^i$, with $\mathcal{G}^1 = \{0\}$ denote the set that consists of the extrema of the Chebyshev polynomial of degree $m_i - 1$, where

$$m_i := 2^{i-1} + 1, \ i = 2, 3, \dots. \tag{13.46}$$

[18] Smolyak's paper is written in Russian. We include the reference to give credit to the inventor of the method.

Therefore, the first four unidimensional grids are (see also Table 13.1):

$$\begin{aligned}
\mathscr{G}^1 &= \{0\},\\
\mathscr{G}^2 &= \{-1, 0, 1\},\\
\mathscr{G}^3 &= \{-1, -0.5\sqrt{2}, 0, 0.5\sqrt{2}, 1\},\\
\mathscr{G}^4 &= \Big\{-1, -0.5\sqrt{2+\sqrt{2}}, -0.5\sqrt{2}, -0.5\sqrt{2-\sqrt{2}}, 0,\\
&\qquad 0.5\sqrt{2-\sqrt{2}}, 0.5\sqrt{2}, 0.5\sqrt{2+\sqrt{2}}, 1\Big\}.
\end{aligned}$$

Note that $\mathscr{G}^1 \subset \mathscr{G}^2 \subset \mathscr{G}^3 \subset \mathscr{G}^4$. In general, each of the sets

$$\mathscr{G}^i = \{\bar{z}_1, \bar{z}_2, \dots, \bar{z}_{m_i}\} \tag{13.47}$$

has an odd number of elements, and the locations of the extrema of T_{m_i-1} and T_{m_i} imply $\mathscr{G}^{i-1} \subset \mathscr{G}^i$.

We need two more definitions before we can construct the multidimensional Smolyak grid. Let $\mathbf{i} := [i_1, i_2, \dots, i_d]$ denote the vector of indices of unidimensional grids in dimensions 1 through d, and abbreviate the sum of its elements by

$$|\mathbf{i}| = i_1 + i_2 + \dots + i_d.$$

Additionally, we introduce the integer $\lambda = 0, 1, \dots$. It determines the degree of exactness (or the level of the approximation) in the sense that the Smolyak polynomial introduced below can reproduce polynomials of degree at most λ (see Barthelmann et al. (2000), Theorem 4). The Smolyak grid of dimension d and level of approximation λ is defined as:

$$\mathscr{H}^{d,\lambda} = \bigcup_{|\mathbf{i}|=d+\lambda} \mathscr{G}^{m_{i_1}} \times \mathscr{G}^{m_{i_2}} \times \dots \times \mathscr{G}^{m_{i_d}}. \tag{13.48}$$

In words, it consists of the unions of Cartesian products of unidimensional grids with the property that the sum of the indices considered in the construction is equal to $d + \lambda$. For $d = 1$, the set $\mathscr{H}^{1,\lambda}$ is equal to the set $\mathscr{G}^{m_{1+\lambda}}$. As an example, we construct the set $\mathscr{H}^{2,2}$ on $[-1, 1]^2$. The union consists of the tensor products that satisfy $i_1 + i_2 = 4$. These are:

$$\begin{aligned}
i_1 = 1 \text{ and } i_2 = 3 : \mathscr{G}^1 \times \mathscr{G}^3 &= \big\{(0,-1);(0,-0.5\sqrt{2});(0,0);(0,0.5\sqrt{2});\\
&\quad (0,1)\big\},\\
i_1 = 2 \text{ and } i_2 = 2 : \mathscr{G}^2 \times \mathscr{G}^2 &= \big\{(-1,-1);(-1,0);(-1,1);(0,-1);(0,0);\\
&\quad (0,1);(1,-1);(1,0);(1,1)\big\},\\
i_1 = 3 \text{ and } i_2 = 1 : \mathscr{G}^3 \times \mathscr{G}^1 &= \big\{(-1,0);(-0.5\sqrt{2},0);(0,0);(0.5\sqrt{2},0);\\
&\quad (1,0)\big\}.
\end{aligned}$$

Therefore, $\mathscr{H}^{2,2}$ is a set of 13 pairs of Chebyshev extrema:

$$\begin{aligned}
\mathscr{H}^{2,2} = \big\{&(0,-1);(0,-0.5\sqrt{2});(0,0);(0,0.5\sqrt{2});(0,1);\\
&(-1,-1);(-1,0);(-1,1);(1,-1);(1,0);\\
&(1,1);(-0.5\sqrt{2},0);(0.5\sqrt{2},0)\big\}.
\end{aligned} \tag{13.49}$$

Figure 13.14 displays these pairs as circles together with the elements of the tensor product $\mathscr{G}^3 \times \mathscr{G}^3$ as blue dots. The latter set consists of 25 elements, almost twice the number of elements of the Smolyak grid.

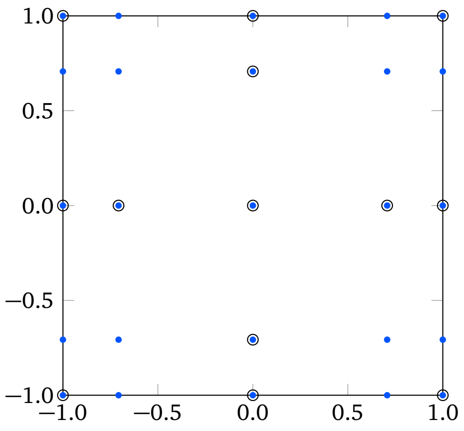

Figure 13.14 Tensor and Smolyak Grid

Interpolating Polynomial. The polynomial that interpolates a given function

$$f : [-1,1]^d \to \mathbb{R}, \mathbf{z} := [z_1, z_2, \ldots, z_d] \mapsto f(\mathbf{z})$$

is a linear combination of polynomials, which are themselves sums of the tensor products of Chebyshev polynomials:

$$p^{\mathbf{i}}(\mathbf{z}) := \sum_{l_1=0}^{m_{i_1}-1} \cdots \sum_{l_d=1}^{m_{i_d}-1} \alpha_{l_1,\dots,l_d} T_{l_1}(z_1)\cdots T_{l_d}(z_d). \tag{13.50}$$

The Smolyak polynomial combines the polynomials $p^{\mathbf{i}}$ according to:[19,20]

$$p^{d,\lambda}(\mathbf{z}) := \sum_{\max\{d,1+\lambda\}\leq|\mathbf{i}|\leq d+\lambda} (-1)^{d+\lambda-|\mathbf{i}|}\binom{d-1}{d+\lambda-|\mathbf{i}|} \sum_{i_1+\cdots+i_d=|\mathbf{i}|} p^{\mathbf{i}}(\mathbf{z}). \tag{13.51}$$

To understand these formulas, we consider the case of $d=2$ and $\lambda=2$. Therefore, we have to consider combinations of indices i_1 and i_2 that satisfy $3=\max\{2,1+2\}\leq i_1+i_2\leq 4=2+2$. The respective numbers of basis functions m_{i_1} and m_{i_2} follow from equation(13.46) and the weights attached to each combination satisfy:

$$w_{i_1,i_2} = (-1)^{4-i_1-i_2}\binom{1}{4-i_1-i_2}.$$

Accordingly, the single terms of the sum (13.51) are:

$$\begin{aligned}
&i_1=1 \text{ and } i_2=2 \Rightarrow m_{i_1}=1, m_{i_2}=2, w_{1,2}=(-1)^1\binom{1}{1}=-1,\\
&i_1=2 \text{ and } i_2=1 \Rightarrow m_{i_1}=3, m_{i_2}=1, w_{2,1}=(-1)^1\binom{1}{1}=-1,\\
&i_1=1 \text{ and } i_2=3 \Rightarrow m_{i_1}=1, m_{i_2}=5, w_{1,3}=(-1)^0\binom{1}{0}=1,\\
&i_1=2 \text{ and } i_2=2 \Rightarrow m_{i_3}=1, m_{i_2}=3, w_{2,2}=(-1)^0\binom{1}{0}=1,\\
&i_1=3 \text{ and } i_2=1 \Rightarrow m_{i_5}=1, m_{i_2}=1, w_{3,1}=(-1)^0\binom{1}{0}=1,
\end{aligned}$$

and the five respective polynomials are:

[19] See Judd et al. (2014), equations (5)-(7).

[20] The symbol

$$\binom{n}{k}=\frac{n!}{k!(n-k)!}$$

in this formula denotes the binomial coefficient, which gives the weight assigned to $p^{\mathbf{i}}$.

$$\begin{aligned}
p^{1,2}(z_1,z_2) &= \gamma_{0,0}T_0(z_1)T_0(z_2) + \gamma_{0,1}T_0(z_1)T_1(z_2) + \gamma_{0,2}T_0(z_1)T_2(z_2),\\
p^{2,1}(z_1,z_2) &= \gamma_{0,0}T_0(z_1)T_0(z_2) + \gamma_{1,0}T_1(z_1)T_0(z_2) + \gamma_{2,0}T_2(z_1)T_1(z_2),\\
p^{1,3}(z_1,z_2) &= \gamma_{0,0}T_0(z_1)T_0(z_2) + \gamma_{0,1}T_0(z_1)T_1(z_2) + \gamma_{0,2}T_0(z_1)T_2(z_2)\\
&\quad + \gamma_{0,3}T_0(z_1)T_3(z_2) + \gamma_{0,4}T_0(z_1)T_4(z_2),\\
p^{2,2}(z_1,z_2) &= \gamma_{0,0}T_0(z_1)T_0(z_2) + \gamma_{0,1}T_0(z_1)T_1(z_2) + \gamma_{0,2}T_0(z_1)T_2(z_2)\\
&\quad + \gamma_{1,0}T_1(z_1)T_0(z_2) + \gamma_{1,1}T_1(z_1)T_1(z_2) + \gamma_{1,2}T_1(z_1)T_1(z_2)\\
&\quad + \gamma_{2,0}T_2(z_1)T_0(z_2) + \gamma_{2,1}T_2(z_1)T_1(z_2) + \gamma_{2,2}T_2(z_1)T_2(z_2),\\
p^{3,1}(z_1,z_2) &= \gamma_{0,0}T_0(z_1)T_0(z_2) + \gamma_{1,0}T_1(z_1)T_0(z_2) + \gamma_{2,0}T_2(z_1)T_0(z_2)\\
&\quad + \gamma_{3,0}T_3(z_1)T_0(z_2) + \gamma_{4,0}T_4(z_1)T_0(z_2).
\end{aligned}$$

Adding the five polynomials together with their weights leaves the final interpolating polynomial with 13 elements:

$$\begin{aligned}
p^{d=2,\lambda=2}(\mathbf{z}) &= \gamma_{0,0}T_0(z_1)T_0(z_2) + \gamma_{0,1}T_0(z_1)T_1(z_2) + \gamma_{0,2}T_0(z_1)T_2(z_2)\\
&\quad + \gamma_{0,3}T_0(z_1)T_3(z_2) + \gamma_{0,4}T_0(z_1)T_4(z_2) + \gamma_{1,0}T_1(z_1)T_0(z_2)\\
&\quad + \gamma_{1,1}T_1(z_1)T_1(z_2) + \gamma_{1,2}T_1(z_1)T_2(z_2) + \gamma_{2,0}T_2(z_1)T_0(z_2)\\
&\quad + \gamma_{2,1}T_2(z_1)T_1(z_2) + \gamma_{2,2}T_2(z_1)T_2(z_2) + \gamma_{3,0}T_3(z_1)T_0(z_2)\\
&\quad + \gamma_{4,0}T_4(z_1)T_0(z_2).
\end{aligned}$$

Note that the set $\mathscr{H}^{2,2}$ presented in (13.49) also has 13 elements. Therefore, we can determine the 13 coefficients of the polynomial by solving the linear equation

$$\begin{bmatrix} f(\bar{z}_{1_1},\bar{z}_{2_1}) \\ f(\bar{z}_{1_2},\bar{z}_{2_2}) \\ \vdots \\ f(\bar{z}_{1_{13}},\bar{z}_{2_{13}}) \end{bmatrix} = \begin{bmatrix} T_0(\bar{z}_{1_1})T_0(\bar{z}_{2_1}) & \dots & T_4(\bar{z}_{1_1})T_0(\bar{z}_{2_1}) \\ T_0(\bar{z}_{1_2})T_0(\bar{z}_{2_2}) & \dots & T_4(\bar{z}_{1_2})T_0(\bar{z}_{2_2}) \\ \vdots & \ddots & \vdots \\ T_0(\bar{z}_{1_{13}})T_0(\bar{z}_{2_{13}}) & \dots & T_4(\bar{z}_{1_{13}})T_0(\bar{z}_{2_{13}}) \end{bmatrix} \begin{bmatrix} \gamma_{0,0} \\ \gamma_{0,1} \\ \vdots \\ \gamma_{4,0} \end{bmatrix}, \tag{13.53}$$

where $(\bar{z}_{1_i},\bar{z}_{2_i})$, $i = 1,\dots,13$ denote the elements of $\mathscr{H}^{2,2}$. Therefore, the difficult part of the implementation of the Smolyak polynomial is the construction of the matrix on the rhs of equation (13.53) for a given dimension d and level of approximation λ. Our MATLAB® function `ChebBase2.m` performs this task. It is based on the code that accompanies Judd, Maliar, Maliar, and Valero (2014).

13.9.5 Neural Networks

Neural networks provide an alternative to linear combinations of polynomials. A single-layer neural network is a nonlinear function of the form

$$\Phi(\mathbf{a},\mathbf{x}) := h\left(\sum_{i=1}^{n} a_i g(x_i)\right),$$

where h and g are scalar functions. In the left panel of Figure 13.15, the first row of nodes represents the function g processing the inputs x_i. The result is aggregated via summation, as indicated by the arrows toward the single node that represents the function h that delivers the final output y. In the single hidden-layer feed-forward network displayed in the right panel of Figure 13.15, the function g delivers its output to a second row of nodes. There, this input is processed by another function, say G, before it is aggregated and passed on to the function h.

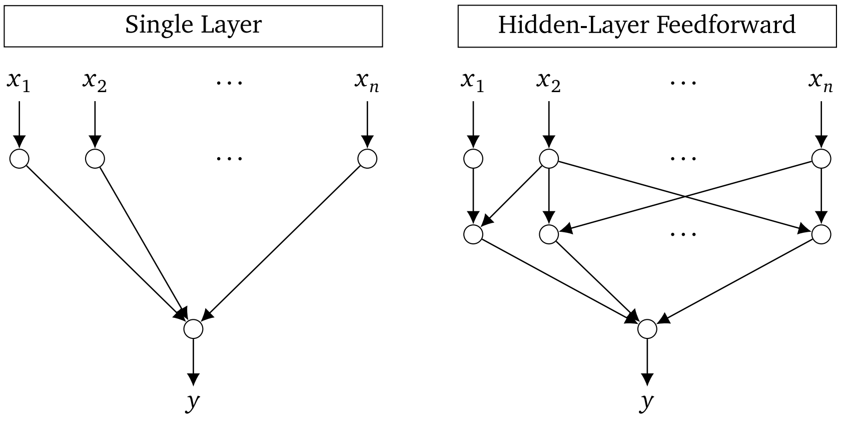

Figure 13.15 Neural Networks

Formally, the single hidden-layer feedforward network is given by:

$$\Phi(\mathbf{a},\mathbf{b},\mathbf{x}) := h\left(\sum_{j=1}^{m} b_j G\left[\sum_{i=1}^{n} a_{ij} g(x_i)\right]\right).$$

The function G is called the hidden-layer activation function. A common choice for G is the sigmoid function

$$G(x) = \frac{1}{1+e^{-x}}.$$

Neural networks are efficient functional forms for approximating multi-dimensional functions. Often, they require fewer parameters for a given accuracy than polynomial approximations.[21]

Duffy and McNelis (2001) solve the stochastic growth model by approximating the rhs of the Euler equation for capital by a function of capital and log TFP using either neural networks or polynomials and find that the former may be preferred to the latter. Lim and McNelis (2008) use neural networks throughout their book to solve increasingly complex models of small open economies. The deep-learning algorithm of Maliar et al. (2021) for solving dynamic economic models involves neural networks to perform model reduction and to handle multicollinearity. Fernández-Villaverde et al. (2023) employ neural networks to solve a heterogenous agent model with an occasionally binding constraint on the nominal interest rate set by the monetary authority.

[21] See Sargent (1993), pp. 58f. and the literature cited there.