Very short notes on approximating functions

Josep Pijoan-Mas
CEMFI, January 2009

Let's define $\mathcal{F}$ as the space of continuous real-valued functions with domain $\mathbf{x} \subset \mathbb{R}$. $\mathcal{F}$ is a vector space. We define also the following inner-product operation,

$$< g, h >= \int_{\mathbf{x}} g(x)\, h(x)\, w(x)\, dx$$

where $g, h, w \in \mathcal{F}$ and w is a proper weighting function (this means that it is positive almost everywhere in $\mathbf{x}$ and has a finite integral in $\mathbf{x}$). The pair $\{\mathcal{F}, < \cdot, \cdot >\}$ form an inner-product vector space.

Now, let's assume we want to approximate a known function $f : \mathbf{x} \to \mathbb{R}$ in $\{\mathcal{F}, < \cdot, \cdot >\}$. We will use a finite number of elements of a basis $\Psi_n \equiv \{\psi_i\}_{i=0}^{n-1} \subset \mathcal{F}$ such that we can write,

$$f(x) \simeq \hat{f}(x, \Lambda) \equiv \sum_{i=0}^{n-1} \lambda_i \psi_i(x)$$

where Λ is an n-dimensional vector whose entries are $\lambda_i \in \mathbb{R}$. We can define a residual function,

$$\chi(x, \Lambda) \equiv f(x) - \sum_{i=0}^{n-1} \lambda_i \psi_i(x)$$

that tells us the size of the error of the approximation at any point $x \in \mathbf{x}$. We want to find the vector Λ that makes the residual function $\chi(x, \Lambda)$ small.

We have to decide what is the exact meaning of *small*. If the approximation was perfect, small means that the residual function would be zero in all its domain $\mathbf{x}$. This typically will be impossible. There are different alternatives we can follow.

1 Minimize the sum of squared errors

One possible choice is the following,

$$\Lambda^* = \text{argmin} \int_{\mathbf{x}} \chi(x, \Lambda)^2\, w(x)\, dx$$

If we wanted a pure OLS, then we could set $w(x) = 1$. The optimality conditions of this minimization state,

$$\begin{cases} < \chi(x, \Lambda), \psi_0(x) > & = & 0 \\ < \chi(x, \Lambda), \psi_1(x) > & = & 0 \\ \quad \dots & & \\ < \chi(x, \Lambda), \psi_{n-1}(x) > & = & 0 \end{cases}$$

which can be rewritten as,

$$\begin{cases} < f(x), \psi_0(x) > - \sum_{i=0}^{n-1} \lambda_i < \psi_i(x), \psi_0(x) > & = & 0 \\ < f(x), \psi_1(x) > - \sum_{i=0}^{n-1} \lambda_i < \psi_i(x), \psi_1(x) > & = & 0 \\ \dots & & \\ < f(x), \psi_{n-1}(x) > - \sum_{i=0}^{n-1} \lambda_i < \psi_i(x), \psi_{n-1}(x) > & = & 0 \end{cases}$$

Let's move to matrix form. We define F as the $n \times 1$ vector with entries $< f(x), \psi_i(x) >$. We define the $n \times n$ matrix M as follows,

$$M \equiv \begin{bmatrix} < \psi_0(x), \psi_0(x) > & < \psi_1(x), \psi_0(x) > & \dots & < \psi_{n-1}(x), \psi_0(x) > \\ < \psi_0(x), \psi_1(x) > & < \psi_1(x), \psi_1(x) > & \dots & < \psi_{n-1}(x), \psi_1(x) > \\ \quad \dots & & & \\ < \psi_0(x), \psi_{n-1}(x) > & < \psi_1(x), \psi_{n-1}(x) > & \dots & < \psi_{n-1}(x), \psi_{n-1}(x) > \end{bmatrix}$$

Then, the previous system of equations can be written as,

$$F - M\Lambda = 0$$

The solution to our minimization problem is given by,

$$\Lambda = M^{-1}F$$

If the chosen basis Ψ_n is such that its elements share a lot of information (for example, monomials) we might have the problem of M being close to singular and therefore very difficult to invert. The contrary is also true. If the elements Ψ_n share very little information, then the inversion of M will be very easy. In particular, if the elements Ψ_n of our basis are orthogonal the matrix M is a diagonal matrix, since all the entries outside the diagonal are zero. Note that any two elements $g, h \in \mathcal{F}$ are orthogonal with respect to the inner-product $< \cdot, \cdot >$ if $g \neq h \Rightarrow < g, h >= 0$. With M diagonal, the entries of our vector Λ are just given by the following expressions,

$$\lambda_i = \frac{< f(x), \psi_i(x) >}{< \psi_i(x), \psi_i(x) >} \qquad \forall\ 0 \leq i \leq n-1 \tag{1}$$

This is a clear advantage. Which bases are orthogonal? It depends on the weighting $w(x)$ function that defines our inner product $< \cdot, \cdot >$. Some examples are the Chebyshev polynomials

(see Section 1.1 below), the Hermite polynomials (with respect to the normal distribution) or the Legendre polynomials (with respect to the uniform probability distribution).

1.1 Chebyshev regression

An element i of the Chebyshev polynomials is defined as $\psi_i(x) = \cos(i \arccos(x))$. They are defined in the set $\mathbf{x} \equiv [-1, 1]$, but this is no limitation as long as $\mathbf{x}$ is a compact set. They are orthogonal under the inner-product with the weighting function $w(x) = (1 - x^2)^{-1/2}$. This means that,

$$< \psi_i(x), \psi_j(x) >= 0 \qquad \forall\ i \neq j$$

Chebyshev polynomials have another useful property: using the weighting function just mentioned, their products for $i = j$ are very easy to compute,

$$\begin{aligned} < \psi_0(x), \psi_0(x) > &= \pi \\ < \psi_i(x), \psi_i(x) > &= \pi/2 \qquad \forall\ 0 < i \leq n-1 \end{aligned} \tag{2}$$

This simplifies the calculations of the denominator terms in (1). Furthermore, using Gauss-Chebyshev quadrature to approximate the integrals in the vector F makes Λ even easier to compute. Let's pick $m \geq n$ points for the integral approximation. Gauss-Chebyshev quadrature requires us to choose these $\{x_k\}_{k=1}^m$ points as the m roots of the m^{th} order Chebyshev polynomial. Then the integrals in the numerator of equation (1) are given by

$$< f(x), \psi_i(x) >= \int_{\mathbf{x}} f(x)\, \psi_i(x)\, w(x)\, dx \simeq \frac{\pi}{m} \sum_{k=1}^{m} f(x_k)\, \psi_i(x_k) \tag{3}$$

And hence, the elements of Λ are really simple to obtain,

$$\begin{aligned} \lambda_0 &= \frac{1}{m} \sum_{k=1}^{m} f(x_k) \\ \lambda_i &= \frac{2}{m} \sum_{k=1}^{m} f(x_k)\, \psi_i(x_k) \qquad \forall\ 0 < i \leq n-1 \end{aligned}$$

Is there any easy way to obtain the m roots x_k of the m^{th} order Chebyshev polynomial? The answer is yes. They are given by the formula,

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

1.2 Chebyshev regression with general domain

Chebyshev polynomials are defined in the compact set $\mathbf{z} \equiv [-1, 1]$. If we want to approximate a function $f(x)$ defined over a different domain $\mathbf{x} \equiv [\underline{x}, \bar{x}]$, we can still use Chebyshev polynomials by mapping $\mathbf{x}$ into $\mathbf{z}$. We use the linear function $\varphi : \mathbf{x} \to \mathbf{z}$,

$$z = \varphi(x) = \frac{2(x - \underline{x})}{\bar{x} - \underline{x}} - 1$$

Hence,

$$f(x) \simeq f(x, \Lambda) = \sum_{i=0}^{n-1} \lambda_i \psi_i(\varphi(x))$$

In order to find the vector Λ that best approximates our function $f(x)$, we need to convert the m roots $\{z_k\}_{k=1}^m$ of the m^{th} order Chebyshev polynomial to the new domain $\mathbf{x}$. We do so by use of the inverse of function φ,

$$x = \varphi^{-1}(z) = \frac{z + 1}{2}(\bar{x} - \underline{x}) + \underline{x}$$

So, let's call $\{x_k\}_{k=1}^m$ the Chebyshev points in the domain $\mathbf{x}$ such that $x_k = \varphi^{-1}(z_k)$. Then, to obtain the elements λ_i we only need to do:

$$\begin{aligned} \lambda_0 &= \frac{1}{m} \sum_{k=1}^{m} f(x_k) \\ \lambda_i &= \frac{2}{m} \sum_{k=1}^{m} f(x_k) \psi_i(z_k) \qquad \forall\ 0 < i \leq n - 1 \end{aligned}$$

2 Make the error function zero at n different points

Another obvious choice to make $\chi(x, \Lambda)$ small is the following. Since we have n unknown parameters, we may want to make the residual function zero at n different points in $\mathbf{x}$. This would give us a well-defined system of n equations in n unknowns. In particular, choose a sequence of points $\{x_k\}_{k=1}^n \subset \mathbf{x}$. Then, our vector Λ^* is given by,

$$\begin{cases} \chi(x_1, \Lambda^*) &= 0 \\ \chi(x_2, \Lambda^*) &= 0 \\ \quad \cdots & \\ \chi(x_n, \Lambda^*) &= 0 \end{cases}$$

Therefore,

$$\begin{cases} f(x_1) - \sum_{i=0}^{n-1} \lambda_i \psi_i(x_1) &= 0 \\ f(x_2) - \sum_{i=0}^{n-1} \lambda_i \psi_i(x_2) &= 0 \\ \cdots & \\ f(x_n) - \sum_{i=0}^{n-1} \lambda_i \psi_i(x_n) &= 0 \end{cases}$$

Let's move to matrix form and define F as the $n \times 1$ vector with entries $f(x_i)$ and the $n \times n$ matrix M as follows,

$$M \equiv \begin{bmatrix} \psi_0(x_1) & \psi_1(x_1) & \dots & \psi_{n-1}(x_1) \\ \psi_0(x_2) & \psi_1(x_2) & \dots & \psi_{n-1}(x_2) \\ \dots & & & \\ \psi_0(x_n) & \psi_1(x_n) & \dots & \psi_{n-1}(x_n) \end{bmatrix}$$

Then, the system of equations becomes,

$$F - M\Lambda = 0$$

which yields the solution,

$$\Lambda = M^{-1}F$$

As before, if the elements Ψ_n share a lot of information this matrix M might be difficult to invert. However, the matrix M is a different object here: it is just the n basis functions evaluated at the n different points. In this situation, what may help is to use basis functions that are non zero in just a small set of $\mathbf{x}$. For example, in the extreme case, using the tent functions as basis would avoid the inversion of M altogether.

2.1 Chebyshev collocation

The use of Chebyshev polynomials leads to a very easy way of computing the elements in Λ that does not require matrix inversion. The inner product expressions for the Chebyshev polynomials in equation (2) and the Gauss-Chebyshev quadrature formula in equation (3) lead to the very useful property known as *discrete orthogonality*. If $\{x_k\}_{k=1}^n$ are the n roots of the n^{th} order Chebyshev polynomial, then

$$\sum_{k=1}^{n} \psi_i(x_k)\,\psi_j(x_k) = \begin{cases} 0 & i \neq j \\ \frac{m}{2} & i = j \neq 0 \\ m & i = j = 0 \end{cases}$$

Now, let's go back to our system of equations. For every point x_k we have

$$f(x_k) - \sum_{i=0}^{n-1} \lambda_i \psi_i(x_k) = 0$$

Let's take $0 \leq j \leq n-1$ and multiply both sides by $\psi_j(x_k)$ such that,

$$f(x_k)\psi_j(x_k) - \sum_{i=0}^{n-1} \lambda_i \psi_i(x_k)\psi_j(x_k) = 0$$

and sum up across all points x_k,

$$\sum_{k=1}^{n} f(x_k)\psi_j(x_k) = \sum_{k=1}^{n}\sum_{i=0}^{n-1} \lambda_i \psi_i(x_k)\psi_j(x_k)$$

In the right hand side, all the terms such that $i \neq j$ disappear due to the discrete orthogonality. So, we get,

$$\sum_{k=1}^{n} f(x_k)\psi_j(x_k) = \lambda_j \sum_{k=1}^{n} \psi_j(x_k)\psi_j(x_k)$$

And using the discrete orthogonality once more we obtain that,

$$\begin{aligned} \lambda_0 &= \frac{1}{n}\sum_{k=1}^{n} f(x_k) \\ \lambda_j &= \frac{2}{n}\sum_{k=1}^{n} f(x_k)\psi_j(x_k) \qquad \forall\ 0 < j \leq n-1 \end{aligned}$$

Notice that these expressions are identical to the ones for the Chebyshev regression whenever $m = n$. That is to say, Chebyshev collocation is the particular case of Chebyshev regression when, to approximate the integrals, we use as many points as elements of the basis.

3 Generalization: projection methods

The approach in the previous sections can be generalized. A general definition of making the residual function $\chi(x, \Lambda)$ *small* is given by making zero its projections in different directions. In particular, as we have n parameters, we may want n projecting directions. Let's define $P \equiv \{p_i\}_{i=0}^{n-1} \subset \mathcal{F}$ as a set of projecting directions. Then, we might want our vector Λ of parameters to solve the following system,

$$\begin{cases} < \chi(x,\Lambda), p_0(x) > &= 0 \\ < \chi(x,\Lambda), p_1(x) > &= 0 \\ \quad \cdots & \\ < \chi(x,\Lambda), p_{n-1}(x) > &= 0 \end{cases}$$

In matrix form, the vector F is defined by the elements $< f(x), p_i(x) >$ and the matrix M is given by,

$$M \equiv \begin{bmatrix} < \psi_0(x), p_0(x) > & < \psi_1(x), p_0(x) > & \dots & < \psi_{n-1}(x), p_0(x) > \\ < \psi_0(x), p_1(x) > & < \psi_1(x), p_1(x) > & \dots & < \psi_{n-1}(x), p_1(x) > \\ \dots & & & \\ < \psi_0(x), p_{n-1}(x) > & < \psi_1(x), p_{n-1}(x) > & \dots & < \psi_{n-1}(x), p_{n-1}(x) > \end{bmatrix}$$

Therefore,

$$\Lambda = M^{-1} F$$

Under this generalization, we see that the minimization of the mean squared error is given by choosing as projection directions the basis functions themselves (which are the gradient of the error function). Making the error function zero at a given set of points $\{x_i\}_{i=1}^{n} \subset \mathbf{x}$ corresponds to choosing as projection directions the Dirac deltas,

$$p_i(x) \equiv \begin{cases} +\infty & \text{if } x = x_i \\ 0 & \text{otherwise} \end{cases}$$

and a constant weighting function.

4 Functions of more than one variable

Let's define $\mathcal{G}$ as the space of continuous real-valued functions with domain $\mathbf{x} \times \mathbf{z} \subset \mathbb{R}^2$. $\mathcal{G}$ is a vector space. We can define the inner-product operation,

$$< g, h >= \int_{\mathbf{x} \times \mathbf{z}} g(x, z)\, h(x, z)\, \omega(x, z)\, dx\, dz$$

where $g, h, \omega \in \mathcal{G}$ and ω is a proper weighting function. The pair $\{\mathcal{G}, < \cdot, \cdot >\}$ form an inner-product vector space.

Now, let's assume we want to approximate a function $f : \mathbf{x} \times \mathbf{z} \to \mathbb{R}$ in $\{\mathcal{G}, < \cdot, \cdot >\}$. We can use tensor products of univariate functions to approximate f. Recall that in previous sections we defined Ψ as a basis of $\mathcal{F}$. In our two-dimensional setting we can define a basis Φ of the space $\mathcal{G}$ as the tensor product of Ψ against itself:

$$\Phi \equiv \Psi \otimes \Psi = \{\psi(x)\, \psi(z) \mid \psi \in \Psi\}$$

And a finite dimension approximation Φ_{n_1,n_2} to Φ as:

$$\Phi_{n_1,n_2} \equiv \left\{ \{\psi_i \psi_j\}_{i=0}^{n_1-1} \right\}_{j=0}^{n_2-1}$$

Hence, we can approximate

$$f(x,z) \simeq \hat{f}(x,z,\Lambda) \equiv \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} \lambda_{i,j}\psi_i(x)\psi_j(z)$$

where Λ is a $n_1 n_2 \times 1$ vector whose entries are $\lambda_{i,j} \in \mathbb{R}$. As in the one-dimensional case we can define a residual function,

$$\chi(x,z,\Lambda) \equiv f(x,z) - \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} \lambda_{i,j}\psi_i(x)\psi_j(z)$$

and we will look for the matrix Λ that makes the residual function $\chi(x,z,\Lambda)$ small.

4.1 Minimize the sum of squared errors

As in the one-dimensional case, we may want to choose the Λ such that,

$$\Lambda^* = \text{argmin} \int_{\mathbf{x}\times\mathbf{z}} \chi(x,z,\Lambda)^2 \, dx\, dz$$

This leads to the optimality conditions,

$$< \chi(x,z,\Lambda), \psi_i(x)\psi_j(z) >= 0 \qquad \forall\ 0 \le i \le n_1 - 1,\ 0 \le j \le n_2 - 1$$

Let's define F as the $n_1 n_2 \times 1$ vector with entries $< f(x,z), \psi_i(x)\psi_j(z) >$ and M as the $n_1 n_2 \times n_1 n_2$ matrix:

$$\begin{bmatrix}
< \psi_0(x)\psi_0(z), \psi_0(x)\psi_0(z) > & < \psi_0(x)\psi_1(z), \psi_0(x)\psi_0(z) > & \dots & < \psi_{n_1-1}(x)\psi_{n_2-1}(z), \psi_0(x)\psi_0(z) > \\
< \psi_0(x)\psi_0(z), \psi_0(x)\psi_1(z) > & < \psi_0(x)\psi_1(z), \psi_0(x)\psi_1(z) > & \dots & < \psi_{n_1-1}(x)\psi_{n_2-1}(z), \psi_0(x)\psi_1(z) > \\
\dots & & & \\
< \psi_0(x)\psi_0(z), \psi_{n_1-1}(x)\psi_{n_2-1}(z) > & < \psi_0(x)\psi_1(z), \psi_{n_1-1}(x)\psi_{n_2-1}(z) > & \dots & < \psi_{n_1-1}(x)\psi_{n_2-1}(z), \psi_{n_1-1}(x)\psi_{n_2-1}(z) >
\end{bmatrix}$$

Then, the solution to the system of equations is given by,

$$F - M\Lambda = 0 \quad \Rightarrow \quad \Lambda = M^{-1}F$$

The orthogonality properties that we applied in the one-dimensional case can be extended to the multidimensional problem. We need to define the weighting function $\omega(x,z)$ as the product of the weighting functions $w(x)$ and $w(z)$ that make the one-dimensional basis Ψ orthogonal.

Then, the elements of M become the product of the one-dimensional inner-products:

$$\begin{aligned} < \psi_i(x)\,\psi_j(z)\,, \psi_k(x)\,\psi_l(z) > &= \int_{\mathbf{x}\times\mathbf{z}} \psi_i(x)\,\psi_j(z)\,\psi_k(x)\,\psi_l(z)\,w(x)\,w(z)\,dx\,dz \\ &= \int_{\mathbf{x}\times\mathbf{z}} \psi_i(x)\,\psi_k(x)\,w(x)\,\psi_j(z)\,\psi_l(z)\,w(z)\,dx\,dz \\ &= \int_{\mathbf{x}} \psi_i(x)\,\psi_k(x)\,w(x)\,dx \int_{\mathbf{z}} \psi_j(z)\,\psi_l(z)\,w(z)\,dz \\ &= < \psi_i(x)\,, \psi_k(x) > < \psi_j(z)\,, \psi_l(z) > \end{aligned}$$

Hence, all the non-diagonal entries of the matrix M are zero and the elements in Λ are easy to compute:

$$\lambda_{i,j} = \frac{< f(x,z)\,, \psi_i(x)\,\psi_j(z) >}{< \psi_i(x)\,, \psi_i(x) >< \psi_j(x)\,, \psi_j(x) >} \qquad \forall\ 0 \le i \le n_1 - 1,\ 0 \le j \le n_2 - 1 \tag{4}$$

4.2 Chebyshev regression

When we use the Chebyshev polynomials as the elements of the basis, we obtain the elements in Λ in a further simpler way. The elements in the denominator of equation (4) are given by the expressions in (2). To compute the elements in the numerator we need to approximate the integrals. Let's pick $m_1 \ge n_1$ and $m_2 \ge n_2$ points in each dimension. Following the Gauss-Chebyshev quadrature procedure take the $\{x_k\}_{k=1}^{m_1}$ points as the m_1 roots of the Chebyshev polynomial of order m_1 and do likewise with the m_2 points $\{z_l\}_{l=1}^{m_2}$. Then, the integrals are approximated as,

$$< f(x,z)\,, \psi_i(x)\,\psi_j(z) >= \int_{\mathbf{x}\times\mathbf{z}} f(x,z)\,\psi_i(x)\,\psi_j(z)\,dx\,dz \simeq \frac{\pi^2}{m_1 m_2} \sum_{k=1}^{m_1}\sum_{l=1}^{m_2} f(x_k, z_l)\,\psi_i(x_k)\,\psi_j(z_l)$$

and hence,

$$\begin{aligned} \lambda_{0,0} &= \frac{1}{m_1 m_2} \sum_{k=1}^{m_1}\sum_{l=1}^{m_2} f(x_k, z_l) \\ \lambda_{0,j} &= \frac{2}{m_1 m_2} \sum_{k=1}^{m_1}\sum_{l=1}^{m_2} f(x_k, z_l)\,\psi_j(z_l) \qquad \forall\ 0 < j \le n_2 - 1 \\ \lambda_{i,0} &= \frac{2}{m_1 m_2} \sum_{k=1}^{m_1}\sum_{l=1}^{m_2} f(x_k, z_l)\,\psi_i(x_k) \qquad \forall\ 0 < i \le n_1 - 1 \\ \lambda_{i,j} &= \frac{4}{m_1 m_2} \sum_{k=1}^{m_1}\sum_{l=1}^{m_2} f(x_k, z_l)\,\psi_i(x_k)\,\psi_j(z_l) \qquad \forall\ 0 < i \le n_1 - 1,\ 0 < j \le n_2 - 1 \end{aligned}$$

It is easy to show that whenever $m_1 = n_1$ and $m_2 = n_2$ we would have Chebyshev collocation.

Finally, we can express the $n_1 n_2 \times 1$ vector Λ in a more compact notation. Define the $m_1 m_2 \times 1$ vector $\overline{F}$ as the vector whose entries are given by $\overline{F}_{k,l} = f(x_k, z_l)$. Define the $n_1 n_2 \times m_1 m_2$ matrix $\overline{\Psi}$ as follows,

$$\overline{\Psi} = \begin{bmatrix} \psi_0(x_1)\psi_0(z_1) & \psi_0(x_1)\psi_0(z_2) & \dots & \psi_0(x_{m_1})\psi_0(z_{m_2}) \\ \psi_0(x_1)\psi_1(z_1) & \psi_0(x_1)\psi_1(z_2) & \dots & \psi_0(x_{m_1})\psi_1(z_{m_2}) \\ \dots & & & \\ \psi_{n_1-1}(x_1)\psi_{n_2-1}(z_1) & \psi_{n_1-1}(x_1)\psi_{n_2-1}(z_2) & \dots & \psi_{n_1-1}(x_{m_1})\psi_{n_2-1}(z_{m_2}) \end{bmatrix}$$

And finally, define a $n_1 n_2 \times n_1 n_2$ diagonal matrix D whose entries in the diagonal are given by,

$$D_{i,j} = \frac{2^{I_{i>0}} 2^{I_{j>0}}}{m_1 m_2}$$

where I_s is a indicator function that takes value 1 when the statement s is true and zero otherwise. Then,

$$\Lambda = D\,\overline{\Psi}\,\overline{F}$$

This notation should make it straightforward to extend Chebyshev regression or Chebyshev collocation to higher dimensional problems.

5 Integration and differentiation

Approximating a function $f(x)$ as a linear combination of a finite number of some simple and known basis functions has the important advantage that it becomes quite cheap to either differentiate or integrat. The reason is that both differentiation and integration are linear operators. Note that,

$$\frac{d}{dx} f(x) \simeq \frac{d}{dx} \hat{f}(x, \Lambda) = \sum_{i=0}^{n-1} \lambda_i \frac{d}{dx} \psi_i(x)$$

So, imagine we are doing some value function iteration. We approximate the value function as a linear combination of some finite number of elements of a basis. Hence, we can write our Bellman operator T as,

$$\Lambda^1 = T\left(\Lambda^0\right)$$

Our first guess of the approximation is given by the vector Λ^0. Then to obtain Λ^1 we need to evaluate the Bellman operator at some preselected $\{x_k\}_{k=1}^m$ points. If the Bellman operator implies taking derivatives or integrals of the value function at each iteration, we do not need to compute the derivative or the integral of the basis functions at each iteration. It suffices to compute at the beginning of the program $\frac{d}{dx}\psi_i(x)$ at all the values of the set $\{x_k\}_{k=1}^m$, tabulate

these values such that $\delta_{i,k} \equiv \frac{d}{dx}\psi_i(x_k)$ and then at each iteration compute,

$$\frac{d}{dx}\hat{f}\left(x_k, \Lambda^0\right) = \sum_{i=0}^{n-1} \lambda_i^0 \delta_{i,k}$$

At each new iteration, only the elements in Λ^0 will be different. The $\delta_{i,k}$ will be unchanged and hence we will not need to differentiate or integrate the basis functions every time.

The extension of this ideas to the multi-dimensional problem is straightforward. Imagine we have a function of two variables that we approximate with a tensor product of a finite number of elements of the one-dimensional bases:

$$f(x,z) \simeq \hat{f}(x,z,\Lambda) \equiv \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} \lambda_{i,j}\psi_i(x)\psi_j(z)$$

Then, if we want to take the derivative of this function with respect to one of its arguments,

$$\frac{\partial}{\partial z} f(x,z) \simeq \frac{\partial}{\partial z}\hat{f}(x,z,\Lambda) \equiv \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} \lambda_{i,j}\psi_i(x)\frac{d}{dz}\psi_j(z)$$

And again, we can tabulate the derivatives of the basis functions at the points in which we will want to evaluate our value function.

Finally, if we want to take derivatives of functions that are defined in a domain $\mathbf{x} \equiv [\underline{x}, \bar{x}]$ we need to map $\mathbf{x}$ into $\mathbf{z} \equiv [-1, 1]$. As mentioned before, we use the linear function $\varphi : \mathbf{x} \to \mathbf{z}$. Hence,

$$\frac{d}{dx}\hat{f}(x,\Lambda) = \sum_{i=0}^{n-1} \lambda_i \frac{d}{dx}\psi_i(\varphi(x)) = \sum_{i=0}^{n-1} \lambda_i \psi_i'(\varphi(x))\varphi'(x) = \left(\frac{2}{\bar{x}-\underline{x}}\right)\sum_{i=0}^{n-1} \lambda_i \psi_i'(\varphi(x))$$

Likewise,

$$\int_a^{\bar{x}} \hat{f}(x,\Lambda)\,dx = \sum_{i=0}^{n-1} \lambda_i \int_a^{\bar{x}} \psi_i(\varphi(x))\,dx$$

6 Practicalities

Some functions may be difficult to approximate by a linear combination of polynomials. In particular, it may take a large number of elements to obtain *acceptable* approximations to the function, let alone to its derivatives. Using a large number of elements makes the polynomial approximations expensive in terms of computing time and hence it may undo its main advantage. To help produce good approximations with a small number of elements of our basis we can do some non-linear transformation to our approximation. For instance, let's define the

approximation $\hat{f}$ to the function f as:

$$f(x,z) \simeq g\left(\hat{f}(x,z,\Lambda)\right) \equiv g\left(\sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\lambda_{i,j}\psi_i(x)\psi_j(z)\right)$$

where $g:\mathbb{R}\to\mathbb{R}$ is an invertible function. For instance, if f is the value function of a problem with instantaneous log utility, we may want to define g as the log function. Then, we can also write,

$$g^{-1}\Big(f(x,z)\Big) \simeq \hat{f}(x,z,\Lambda) \equiv \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\lambda_{i,j}\psi_i(x)\psi_j(z)$$

And our residual function becomes,

$$\chi(x,z,\Lambda) \equiv g^{-1}\Big(f(x,z)\Big) - \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\lambda_{i,j}\psi_i(x)\psi_j(z)$$

So, everything described in the previous sections regarding how to find the values in Λ can be applied easily by just replacing f by $g^{-1}\circ f$. To evaluate the derivatives one has to remember that,

$$\frac{d}{dx}f(x,z) \simeq \frac{d}{dx}\,g\left(\sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\lambda_{i,j}\psi_i(x)\psi_j(z)\right) = g'\left(\sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\lambda_{i,j}\psi_i(x)\psi_j(z)\right)\sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\lambda_{i,j}\frac{d}{dx}\psi_i(x)\psi_j(z)$$

A Recursive definitions for the Chebyshev polynomials

The Chebyshev polynomials, as all orthogonal polynomials, have a recursive definition:

$$\psi_i(x) = \begin{cases} 1 & i = 0 \\ x & i = 1 \\ 2x\psi_{i-1}(x) - \psi_{i-2}(x) & i > 1 \end{cases}$$

It is also useful the recursive characterization of their first derivatives:

$$\frac{d\psi_i(x)}{dx} = \begin{cases} 0 & i = 0 \\ 1 & i = 1 \\ 4x & i = 2 \\ 2i\psi_{i-1}(x) + \frac{i}{i-2}\frac{d\psi_{i-2}(x)}{dx} & i > 2 \end{cases}$$