THE UNIVERSITY OF WESTERN ONTARIO
LONDON ONTARIO

Paul Klein
Office: SSC 4028
Phone: 661-2111 ext. 85227
Email: paul.klein@uwo.ca
URL: www.ssc.uwo.ca/economics/faculty/klein/

Economics 613/614

Advanced Macroeconomics I & 2

Fall 2007

Multi-dimensional interpolation

This discussion is based on that in Miranda and Fackler (2002), but is a little bit more explicit.

1 A cookbook

1.1 Interpolation in one dimension

Function approximation in one dimension involves defining an approximating function $\widehat{f}$ of a given function $f : A \to \mathcal{R}$, where $A \subset \mathcal{R}$. The function $\widehat{f}$ is a member of a finite-dimensional set of functions with basis $\{\psi_i\}_{i=0}^n$. This means that, for any $x \in A$, we have

$$\widehat{f}(x) = \sum_{j=0}^{n} \theta_j \psi_j(x). \tag{1}$$

In vector notation, we may write (1) as

$$\widehat{f}(x) = \psi(x)\theta \tag{2}$$

where

$$\psi(x) = \begin{bmatrix} \psi_0(x) & \psi_1(x) & \dots & \psi_n(x) \end{bmatrix}$$

and

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}.$$

The interpolation problem in one dimension is to find $\{\theta_i\}_{i=0}^n$ such that

$$f(x_i) = \sum_{j=0}^{n} \theta_j \psi_j(x_i) \tag{3}$$

for all $i = 0, 1, \dots, n$. In vector notation, we may write (3) as

$$y = \Psi\theta \tag{4}$$

where

$$\Psi = \begin{bmatrix} \psi_0(x_0) & \psi_1(x_0) & \psi_2(x_0) & \dots & \psi_n(x_0) \\ \psi_0(x_1) & \psi_1(x_1) & \dots & \dots & \psi_n(x_1) \\ \psi_0(x_2) & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \psi_0(x_n) & \dots & \dots & \dots & \psi_n(x_n) \end{bmatrix}$$

and

$$y = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

1.2 Interpolation in d dimensions

Now suppose $f : A \to \mathcal{R}$ where $A \subset \mathcal{R}^d$. Then we have a set of $n_k + 1$ basis functions $\{\psi_i^k\}_{i=0}^{n_k}$ for each dimension $k = 1, 2, \ldots, d$. We also have a grid $\{x_i^k\}_{i=0}^{n_k}$ for each dimension $k = 1, 2, \ldots, d$.

The set of known function values is naturally organized in a multi-dimensional array as follows.

$$Y(i_1, i_2, \ldots, i_d) = f(x_{i_1}^1, x_{i_2}^2, \ldots, x_{i_d}^d).$$

We would like (2) and (4) to be valid in the multi-dimensional case as well. To achieve that, we begin by defining

$$y = Y(:).$$

This is Matlabese for hyper-columnwise vectorization. What does this mean exactly? In two dimensions it is just the usual vec operator. In three dimensions, you first take the two-dimensional matrices $Y(:, :, i)$ for $i = 0, 1, \ldots, n_3$, apply the vec operator to each of them, then stack the results on top of each other, with $\mathrm{vec}(Y(:, :, 0))$ on top of course.[1] In four dimensions, $Y(:)$ begins with vec of $Y(:, :, 0, 0)$, followed by vec of $Y(:, :, 1, 0)$, followed by vec of $Y(:, :, 0, 1)$, followed by vec of $Y(:, :, 1, 1)$. In

[1]0-based indexing is not actually allowed in Matlab, but it is allowed in Fortran.

d dimensions, the order is defined by incrementing the first index until the end is reached, then increment the second index by one step, reset the first index and again increment it until the end, then increment the second index one step... etc. etc. until the end of the first index is reached. Then increment the third index by one step, reset the second index... etc. etc.

Meanwhile, we define, for $k = 1, 2, \ldots, d$,

$$\Psi_k = \begin{bmatrix} \psi_0^k(x_0^k) & \psi_1^k(x_0^k) & \psi_2^k(x_0^k) & \ldots & \psi_{n_k}^k(x_0^k) \\ \psi_0^k(x_1^k) & \psi_1^k(x_1^k) & \ldots & \ldots & \psi_{n_k}^k(x_1^k) \\ \psi_0^k(x_2^k) & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \psi_0^k(x_{n_k}^k) & \ldots & \ldots & \ldots & \psi_{n_k}^k(x_{n_k}^k) \end{bmatrix}$$

and

$$\Psi = \Psi_d \otimes \Psi_{d-1} \otimes \cdots \otimes \Psi_1.$$

Finally, define

$$\psi(x) = \psi^d(x^d) \otimes \psi^{d-1}(x^{d-1}) \otimes \cdots \otimes \psi^1(x^1)$$

where $x = (x^1, x^2, \ldots, x^d)$.

With these definitions, (2) and (4) remain valid in the multidimensional case as well in the following sense. If θ solves 4 then for all $0 \leq i_1 \leq n_1$, $0 \leq i_2 \leq n_2$, $\ldots$, $0 \leq i_d \leq n_d$ we have

$$\psi(x_{i_1}^1, x_{i_2}^2, \ldots, x_{i_d}^d)\theta = Y(i_1, i_2, \ldots, i_d).$$

Notice also that solving (4) in the multi-dimensional case is not as hard as it looks, since

$$\Psi^{-1} = \Psi_d^{-1} \otimes \Psi_{d-1}^{-1} \otimes \cdots \otimes \Psi_1^{-1}.$$

In the case of Chebyshev interpolation, we have an explicit expression for Ψ_k^{-1} so that case is particularly simple.

2 Why it works

It is surprisingly hard to show that this works in general. But let's show it in the simplest possible non-trivial case. Let the basis functions in each of two dimensions be 1 and x and 1 and y, respectively. Let the grids be $\{x_0, x_1\}$ and $\{y_0, y_1\}$. Organize the known function values in a 2×2 matrix Z. Notice first that the interpolating function is written as

$$\widehat{f}(x, y) = \psi(x, y)\theta$$

where

$$\psi(x, y) = \begin{bmatrix} 1 & y \end{bmatrix} \otimes \begin{bmatrix} 1 & x \end{bmatrix} = \begin{bmatrix} 1 & x & y & xy \end{bmatrix}.$$

Now form the matrix Ψ via

$$\Psi_y \otimes \Psi_x = \begin{bmatrix} 1 & y_0 \\ 1 & y_1 \end{bmatrix} \otimes \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} = \begin{bmatrix} 1 & x_0 & y_0 & x_0y_0 \\ 1 & x_1 & y_0 & x_1y_0 \\ 1 & x_0 & y_1 & x_0y_1 \\ 1 & x_1 & y_1 & x_1y_1 \end{bmatrix}.$$

Now consider the expression $\Psi\theta$. Evidently it should equal

$$\begin{bmatrix} Z_{00} \\ Z_{10} \\ Z_{01} \\ Z_{11} \end{bmatrix},$$

but this is precisely $\mathrm{vec}(Z)$.

References

Miranda, M. J. and P. L. Fackler (2002). *Applied Computational Economics and Finance*. MIT Press.