The previous two chapters have introduced the Matlab and R code needed to specify basis function systems and then to define curves by combining these coefficient arrays. For example, we saw how to construct a basis object such as `heightbasis` to define growth curves and how to combine it with a matrix of coefficients such as `heightcoef` so as to define growth functional data objects such as were plotted in Figure 1.1.

We now turn to methods for computing these coefficients with more careful consideration of measurement error. For example, how do we compute these coefficients to obtain an optimal fit to data such as the height measurements for 54 girls in the Berkeley growth study stored in the 31 by 54 matrix that we name `heightmat`? Or how do we replace the rather noisy mean daily precipitation observations by smooth curves?

Two strategies are discussed. The simplest revisits the use of regression analysis that concluded Chapter 4, but now uses a special function for this purpose. The second and more elaborate strategy aims to miss nothing of importance in the data by using a powerful basis expansion, but avoids overfitting the data by imposing a penalty on the “roughness” of the function, where the meaning of “rough” can be adapted to special features of the application from which the data were obtained.

**5.1 Regression Splines: Smoothing by Regression Analysis**

We tend, perhaps rather too often, to default to defining data fitting as the minimization of the sum of squared errors or residuals,

\[
SSE(x) = \sum_{j}^{n} [y_j-x(t_j)]^2. \tag{5.1}
\]

When smoothing function \( x \) is defined as a basis function expansion (3.1), the least-squares estimation problem becomes...
\[ \text{SSE}(c) = \sum_{j} [y_j - \sum_{k} c_k \phi_k(t_j)]^2 = \sum_{j} [y_j - \phi(t_j)'c]^2. \]  

(5.2)

The approach is motivated by the error model

\[ y_j = x(t_j) + \varepsilon_j = c'\phi(t) + \varepsilon_j = \phi'(t_j)c + \varepsilon_j \]  

(5.3)

where the true errors or residuals \( \varepsilon_j \) are statistically independent and have a normal or Gaussian distribution with mean 0 and constant variance. Of course, if we look closely, we often see that this error model is too simple. Nevertheless, the least-squares estimation process can be defended on the grounds that it tends to give nearly optimal answers relative to “best” estimation methods so long as the true error distribution is fairly short-tailed and departures from the other assumptions are reasonably mild.

Readers will no doubt recognize (5.3) as the standard regression analysis model, along with its associated least-squares solution. Using matrix notation, let the \( n \)-vector \( y \) contain the \( n \) values to be fit, vector \( \varepsilon \) contain the corresponding true residual values, and \( n \) by \( k \) matrix \( \Phi \) contain the basis function values \( \phi_k(t_j) \). Then we have

\[ y = \Phi c + \varepsilon \]

and the least-squares estimate of the coefficient vector \( c \) is

\[ \hat{c} = (\Phi'\Phi)^{-1}\Phi'y. \]  

(5.4)

R and Matlab already have the capacity to smooth data through their functions for regression analysis. Here is how we can combine these functions with the basis creation functions available in the fda package. Suppose that we want a basis system for the growth data with \( K = 12 \) basis functions using equally spaced knots. This can be accomplished in R with the following command:

```r
heightbasis12 = create.bspline.basis(c(1,18), 12, 6)
```

If we evaluate the basis functions at the ages of measurement in vector object \( \text{age} \) by the command \( \text{basismat} = \text{eval.basis}(\text{age}, \text{heightbasis12}) \) (in R), then we have a 31 by 12 matrix of covariate or design values that we can use in a least-squares regression analysis defined by commands such as

```r
heightcoef = lsfit(basismat, heightmat, intercept=FALSE)$coef
heightcoef = basismat\heightmat
```

in R and Matlab, respectively. Spline curves fit by regression analysis are often referred to as regression splines in statistical literature.

However, the function \text{smooth.basis} (R) and \text{smooth.basis} (Matlab) are provided to produce the same results as well as much more without the need to explicitly evaluate the basis functions, through the R command

```r
heightList = smooth.basis(age, heightmat,
```