

Chapter 2: Interpolation, Approximation and Extrapolation

Whenever copious data are recorded and reported sooner or later the need arises to represent the data in terms of curves and surfaces. If the curves and surfaces pass through the data points we speak of interpolating the data. If the data are not exactly reproduced but only in some average way, for example with a least squares fit, then we have an approximation. In either case, if the curves and surfaces are used to predict an outcome beyond the domain containing the data then we speak of extrapolation.

Interpolation and approximation at times have to be quite sophisticated. To illustrate this point let us consider the following scenario. Suppose that we are given, at time $t = 0$, a set of N call values $\{C_j\}$ corresponding to discrete strikes prices $\{K_j\}$. Suppose the value of the underlying asset is S_0 . But for hedging purposes we need the call prices as a function of the value S of the asset for a given strike price K_0 .

A look at the Black Scholes formula of Chapter 1 allows the conclusion if we set

$$x = S/K, \quad u(x) = C/K$$

then the given data correspond to $x_i = S_0/K_i$ and $u_i = C_i/K_i$. If we interpolate (or approximate) these data with a curve $u(x)$ we obtain an approximation to

$$C(S, K_0) = K_0 u(S/K_0).$$

What makes this problem challenging is the fact that for delta and gamma hedging we need not only $C(S, K_0)$ but also $\partial C/\partial S$ and $\partial^2 C/\partial^2 S$ so that the usual piecewise linear approximation of the data is not in general useful. We need an approximation in terms of a once or twice differentiable function (unless we want to entrust ourselves to the numerical differentiation of discrete data and its numerical instability as discussed in Chapter X).

We begin by discussing the one-dimensional interpolation problem: Find a function $Pu(x)$ which interpolates the data $\{(x_i, u_i)\}_{i=0}^N$. It will always be assumed that the data $\{u_i\}$ represent measured or computed values of an underlying function $u(x)$ which, of course, is not known to us. Thus Pu can always be considered an approximation to u . The notation

Pu is chosen to suggest that the interpolant is a projection of u in the sense that if we take the interpolant $P(Pu)$ of Pu we end up with Pu itself.

Polynomial interpolation: There are infinitely many functions which pass through all the given data points. Conceptually a very simple function is a polynomial of degree N

$$P_N(x) = \sum_{j=0}^N a_j x^j$$

where the $N + 1$ coefficients must be chosen such that

$$P_N(x_i) = u_i, \quad i = 0, \dots, N.$$

These $N + 1$ equations can be written in matrix form as

$$\mathcal{N}a = u$$

where

$$\mathcal{N}_{ij} = x_i^j, \quad i, j = 0, \dots, N$$

and

$$u = (u_0, \dots, u_N).$$

The matrix \mathcal{N} is known as a Vandermonde matrix and can be shown to be non-singular for distinct interpolation points $\{x_i\}$ so that the interpolating polynomial exists and is unique. It is, in fact, not even necessary to set up the matrix system. The interpolating polynomial can be written in terms of the so-called Lagrange coefficients

$$\ell_i(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_N)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_N)}$$

or in short

$$\ell_i(x) = \frac{\prod_{\substack{j=0 \\ j \neq i}}^N (x - x_j)}{\prod_{\substack{j=0 \\ j \neq i}}^N (x_i - x_j)}.$$

We observe that $\ell_i(x)$ is a polynomial of degree N which satisfies

$$\ell_i(x_j) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases}$$

Thus the interpolating polynomial for $\{x_i, u_i\}$ is

$$P_N(x) = \sum_{j=0}^N u_j \ell_j(x)$$

since the sum of N th order polynomials is an N th order polynomial and since at $x = x_i$ its value is u_i . Uniqueness guarantees that we obtain the same interpolant as with the Vandermonde matrix. For example, the data $(x_0, u_0), (x_1, u_1), (x_2, u_2)$ are interpolated by

$$P_2(x) = u_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + u_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + u_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}.$$

Suppose we wanted to obtain derivative information from the data at $x = x_0$. Then it is natural to expect that $P_2'(x_0)$ and $P_2''(x_0)$ are reasonable approximations to $u'(x_0)$ and $u''(x_0)$. For the special case of $(x_1 - x_0) = (x_2 - x_1) = \Delta x$ we obtain

$$u'(x_0) = \frac{-3u_0 + 4u_1 - u_2}{2\Delta x},$$

and

$$u''(x_0) = \frac{u_0 - 2u_1 + u_2}{\Delta x^2}$$

(which, as we shall see later, is a much better approximation to $u''(x_1)$ than to $u''(x_0)$). If the data come from an $N + 1$ times continuously differentiable function u then it is natural to ask how close the polynomial interpolant comes to reproducing u . This question is answered by the following theorem:

Let $\{x_j\}_{j=0}^N$ be distinct real numbers and let u be a given real valued function with $N + 1$ continuous derivatives on an interval I containing the points $\{x_j\}$ and the variable t . Then there exists a point $\zeta \in I$ such that

$$u(t) - Pu(t) = \frac{(t - x_0) \cdots (t - x_N)}{(N + 1)!} u^{(N+1)}(\zeta).$$

For a proof see, e.g. \square .

Intuitively one would expect to approximate u better and better the more interpolation points are used. That is not necessarily the case because the derivatives of u in our error formula can increase rapidly with N . In practice, polynomial interpolation is highly useful

for approximating a few data points, with $N = 3, 4,$ or 5 . It becomes problematical for large N because the interpolant can oscillate wildly. A classic example is the interpolation of the following nice function

$$u(x) = \frac{1}{1+x^2}$$

on the interval $[-5, 5]$. Let us set

$$x_i = -5 + 10i/N, \quad u_i = u(x_i), \quad i = 0, \dots, N$$

and plot u and Pu as a function of x when we interpolate for $N = 10$, i.e., when we interpolate at the integers $-5, -4, \dots, 4, 5$.

It is apparent from the plot that the 10th order polynomial Pu reproduces u very poorly near the endpoints of the interval. The interpolant is useless for $N = 10$ and becomes worse as N increases.

Piecewise cubic spline interpolation: In order to avoid the instabilities which can occur in high order polynomial interpolation but still be able to provide derivative information the theory of spline interpolation has been developed in the past 30 years. Splines have become the dominant tool for interpolating data which are thought to come from a smooth function of a single variable. We shall discuss here the piecewise cubic spline.

We again suppose that we have data $\{x_i, u_i\}$ for $i = 0, \dots, N$. The basic idea is to define over each subinterval $[x_{i-1}, x_i]$ a smooth function $P_i(x)$ which interpolates u_{i-1} at x_{i-1} and u_i at x_i so that

$$P_i(x_i) = P_{i+1}(x_i) = u_i.$$

In addition, we want $P_i u(x)$ and $P_{i+1} u(x)$ to come together at $x = x_i$ so that

$$P'_i(x_i) = P'_{i+1}(x_i)$$

$$P''_i(x_i) = P''_{i+1}(x_i).$$

The interpolant of u is then

$$Pu(x) = P_i(x) \quad \text{for } x \in [x_{i-1}, x_i].$$

By construction Pu is twice continuously differentiable on $[x_0, x_N]$. It turns out that the simplest function which can satisfy these conditions is a cubic polynomial over each subinterval. It is straightforward to compute if we write

$$P_i''(x) = M_i \frac{(x - x_{i-1})}{(x_i - x_{i-1})} + M_{i-1} \frac{(x_i - x)}{(x_i - x_{i-1})}, \quad i = 1, \dots, N$$

for as yet unknown constants $\{M_i\}$. By inspection $P_i''(x_i) = M_i$ and $P_{i+1}''(x_i) = M_i$ so that Pu will have a continuous second derivative on $[x_0, x_N]$ regardless of how we pick $\{M_i\}$. We now find $P_i(x)$ by integrating twice so that

$$P_i(x) = \frac{M_i}{6} \frac{(x - x_{i-1})^3}{(x_i - x_{i-1})} + \frac{M_{i-1}}{6} \frac{(x_i - x)^3}{(x_i - x_{i-1})} + c_i x + d_i.$$

The two constants of integration are determined such that $P_i(x)$ interpolates u at x_{i-1} and x_i . It is straightforward to verify that the interpolant is

$$P_i(x) = \frac{M_i}{6} \frac{(x - x_{i-1})^3}{(x_i - x_{i-1})} + \frac{M_{i-1}}{6} \frac{(x_i - x)^3}{(x_i - x_{i-1})} + \left(u_i - \frac{M_i(x_i - x_{i-1})^2}{6} \right) \frac{(x - x_{i-1})}{(x_i - x_{i-1})} + \left(u_{i-1} - \frac{M_{i-1}(x_i - x_{i-1})^2}{6} \right) \frac{(x_i - x)}{(x_i - x_{i-1})}.$$

The as yet unknown coefficients $\{M_i\}$ are still available to enforce continuity of the derivative. A straightforward calculation shows that

$$P_i'(x_i) = P_{i+1}'(x_i)$$

leads to the linear equation

$$\frac{1}{6} M_{i-1} \Delta x_i + \frac{1}{3} M_i \Delta x_i + \frac{1}{3} M_i \Delta x_{i+1} + \frac{1}{6} M_{i+1} \Delta x_{i+1} = \frac{u_{i+1} - u_i}{\Delta x_{i+1}} - \frac{u_i - u_{i-1}}{\Delta x_i}$$

for $i = 1, \dots, N - 1$, where we have set $\Delta x_i = x_i - x_{i-1}$. We need to impose an additional condition at x_0 and x_N in order to obtain altogether $N + 1$ equations for the $N + 1$ unknowns $\{M_i\}$. Two choices are commonly available. We can define a “natural spline” satisfying

$$P_1''(x_0) = P_N''(x_N) = 0$$

so that

$$M_0 = M_N = 0$$

or we can fit derivative information for u' so that

$$P'_1(x_0) = u'(x_0) \quad \text{and} \quad P'_N(x_N) = u'(x_N).$$

(If u' is not available one can approximate u with a low order Lagrange polynomial at the end points and take its derivative as an approximation to u' .) These conditions lead to the two equations

$$\begin{aligned} \frac{1}{3} M_0 \Delta x_1 + \frac{1}{6} M_1 \Delta x_1 &= \frac{u_1 - u_0}{\Delta x_1} - u'(x_0) \\ \frac{1}{6} M_{N-1} \Delta x_N + \frac{1}{3} M_N \Delta x_N &= \frac{u_N - u_{N-1}}{\Delta x_N} + u'(x_N). \end{aligned}$$

In both cases we end up with a linear system

$$\mathcal{N}M = b.$$

For the natural spline the matrix \mathcal{N} has dimension $(N - 1) \times (N - 1)$ and the unknowns are $\{M_j\}_{j=1}^{N-1}$. In the second case the matrix \mathcal{N} has dimension $(N + 1) \times (N + 1)$. In both cases the matrix \mathcal{N} satisfies

$$|\mathcal{N}_{ii}| > \sum_{\substack{j=0 \\ j \neq i}}^N |\mathcal{N}_{ij}|.$$

Such a matrix is called strictly diagonally dominant which guarantees the existence of a unique solution of the linear system and makes its numerical solution on a computer straightforward.

We can again ask how well the cubic spline will approximate a smooth function u by interpolating its values. One now has the following theorem:

Let u be four times continuously differentiable on the interval $[a, b]$. Let $a = x_0 < x_1 \cdots < x_N = b$ and set

$$h = \max_i \Delta x_i.$$

Let Pu be the piecewise cubic spline which interpolates u at the points $\{x_j\}$ and $u'(a)$ and $u'(b)$. Then

$$\max_x |u^{(n)}(x) - Pu^{(n)}(x)| < c_n h^{4-n} \max |u^{(4)}(x)|, \quad n = 0, 1, 2$$

where

$$c_0 = 5/384, \quad c_1 = 1/24, \quad c_2 = 3/8.$$

This theorem shows that the piecewise cubic spline is a high order approximation of the function u and its first two derivatives. For example, if $h = 10^{-1}$ then error in approximating u is of order 10^{-4} .

For a proof see again [].

The theory of splines goes much beyond the case of simple piecewise cubic interpolants. Matlab provides tools of splines up to order 9 (i.e., piece-wise ninth order polynomials for a smooth interpolation of data. In addition, interpolation in terms of functions other than polynomials can be applied, such as in the so-called theory of “splines under tension” which can be applied when cubic splines show oscillations.

Here we shall pursue interpolation in the opposite direction and drop the order of the spline to 1; we now are talking about piecewise linear interpolation (which is routinely employed when plotting data). In analogy to the polynomial interpolation based on Lagrange coefficients we can write the linear interpolant through $\{x_i, u_i\}$ in the form

$$Pu = \sum_{j=0}^N u_j \phi_j(x)$$

where $\phi_i(x)$ is the piecewise linear continuous function defined by

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & x \in [x_{i-1}, x_i] \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise.} \end{cases}$$

We see that $\phi_i(x)$ is piecewise linear between any two interpolation points and that

$$\phi_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases}$$

$\phi_i(x)$ is often referred to as a “hat” function. Piecewise linear interpolation requires little work compared to cubic spline interpolation. On the other hand, the approximation error is notably increased.

Theorem: Let u be twice continuously differentiable on the interval $[a, b]$. Let $a = x_0 < \dots < x_N = b$ and set

$$h = \max_i \Delta x_i.$$

Let Pu be the piecewise linear function which interpolates u at the points $\{x_i\}$. Then

$$\max_x |u^{(n)}(x) - Pu^{(n)}(x)| \leq c_n h^{2-n} \max_x |u''(x)|, \quad n = 0, 1$$

where $c_0 = 1/8$ and $c_1 = 1/2$. Hence the error in approximating u is of order h^2 compared to h^4 for the piecewise cubic spline.

Interpolation in the plane: Let us suppose now that we need to find a surface $Pu(x, y)$ which interpolates given points $\{(x_i, y_j), u_{ij}\}$ for $i = 0, \dots, M$ and $j = 0, \dots, N$. Because the data are given on a regular grid one can write down immediately interpolating functions, such as

$$Pu(x, y) = \sum_{\substack{i=0 \\ j=0}}^{\substack{j=N \\ i=M}} u_{ij} \ell_i(x) \ell_j(y)$$

or

$$Pu(x, y) = \sum_{\substack{i=0 \\ j=0}}^{\substack{j+N \\ i=M}} u_{ij} \phi_i(x) \phi_j(y)$$

where $\ell_i(x)$ is the M th order Lagrange coefficient in x and $\ell_j(y)$ is the N th order Lagrange coefficient in y . Similarly, $\phi_i(x)$ and $\phi_j(y)$ are the one-dimensional piecewise linear hat functions.

Note that when Lagrange coefficients are used then an infinitely differentiable interpolant results. As in the one-dimensional case artificial oscillations rule out the polynomial interpolation for more than a few points in each direction. We also observe that in piecewise linear interpolation the function

$$\phi_i(x) \phi_j(y)$$

vanishes outside the rectangle $[x_{i-1}, x_{i+1}]x[y_{j-1}, y_{j+1}]$, while inside it is piecewise linear in x for fixed y , piecewise linear in y for fixed x and piecewise quadratic in x along any line

$y = mx + b$ crossing the rectangle. The function looks like a tent or hat with height 1 at the center and zero on and outside the boundary of the rectangle.

The essential assumption for our interpolation schemes is the requirement that the data are given on a rectangular grid so that the interpolant can be expressed in product form. Unfortunately, data may well be scattered. For example, if option prices $C(S, t)$ are to be interpolated from recorded prices at different times $\{t_j\}$ then the nodes $\{x_i\}$ change from time level to time level since the value of the underlying asset will change with time. Hence in general all we may assume is that data $\{(x_i, y_i, u_i)\}_{i=1}^N$ are given where each (x_i, y_i) simply is a point with known coordinates. If all we need is a surface which interpolates the data but which need not have continuous derivatives then the basic idea underlying the hat function can be generalized.

First we “triangularize,” i.e., break up into triangles, the region containing the points $\{(x_i, y_i)\}$. Computer subroutines are available which generate triangles in the $x - y$ plane such that each point (x_i, y_i) is the vertex of one or more triangles. Note that in general these points are scattered in the plane. We cannot assume in general that (x_k, y_k) and (x_{k+1}, y_{k+1}) belong to the same triangle or adjacent triangles. Then for each vertex (x_n, y_n) we are going to find the analog to the hat function which was defined above over the rectangular grid. This function is defined over all triangles. We shall denote it by $\phi_n(x, y)$. It has the shape of a plane over each triangle. This plane is uniquely determined by its height at the three vertices of the triangle. The plane will have height one at the chosen vertex (x_n, y_n) and height zero at all other vertices. We note that if none of the three vertices of a triangle corresponds to the chosen vertex (x_n, y_n) then $\phi_n(x, y) = 0$ over the triangle. If we were to plot ϕ_n we have a tent of unit height at (x_n, y_n) which slopes linearly to zero at adjacent vertices and is continued as zero over the remaining triangles. The interpolant of a function $u(x, y)$ then is

$$Pu(x, y) = \sum_{n=1}^N u(x_n, y_n)\phi_n(x, y)$$

which is a continuous piecewise linear function defined over the whole domain. The actual shape of each $\phi_n(x, y)$ over a given triangle is easily found. Suppose we have the unit triangle with vertices 1, 2 and 3 given by $(0, 0)$, $(1, 0)$ and $(0, 1)$, respectively. Then by inspection

we see that over this triangle in the $X - Y$ plane

$$\phi_1(X, Y) = 1 - X - Y$$

$$\phi_2(X, Y) = X$$

$$\phi_3(X, Y) = Y.$$

Now let (x_0, y_0) , (x_1, y_1) and (x_2, y_2) be the coordinates of the three vertices of a triangle T . We can map them to the vertices of the unit triangle with an affine transformation of the form

$$\begin{pmatrix} X \\ Y \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix} + b$$

where the 2×2 matrix A and the vector b are chosen such that

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = A \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + b$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} = A \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + b$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} = A \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} + b.$$

These equations allow us to compute A and b from

$$A \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$b = -A \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}.$$

Over T the interpolant through $u(x_0, y_0)$, $u(x_1, y_1)$ and $u(x_2, y_2)$ is

$$Pu(x, y) = u(x_0, y_0)\phi_0(x, y) + u(x_1, y_1)\phi_1(x, y) + u(x_2, y_2)\phi_2(x, y)$$

where

$$\phi_0(x, y) = 1 - [a_{11}x + a_{12}y + b_1] - [a_{21}x + a_{22}y + b_2]$$

$$\phi_1(x, y) = [a_{11}x + a_{12}y + b_1]$$

$$\phi_2(x, y) = [a_{21}x + a_{22}y + b_2].$$

Clearly, an efficient implementation of the interpolation of scattered data places a high demand on computing skills because of the triangulation and the bookkeeping skills of where the $\{(x_n, y_n)\}$ are located and what functions are associated with each triangle. Building

blocks for a linear (or higher order) interpolation on triangulated domains may be found in “finite element” program libraries which exist for the numerical integration of partial differential equations.

Approximation

Interpolation is not a suitable representation of data if the data are noisy due to measurement errors, if the mathematical model requires that a specific form of function with few degrees of freedom reproduce the measurements as well as possible, or if an extrapolation of data well beyond the domain is required for which measurements are available. In this case the data are approximated by an a priori chosen function in some average manner.

A common method for approximating data is the least squares method. To illustrate this approach consider the following problem. Suppose we have M measurements $\{(x_i, u_i)\}_{i=1}^M$, with $M \gg 1$, which we expect to be normally distributed. However, we know neither the mean α_1 nor the standard deviation α_2 . A least squares approach would require the calculation of α_1 and α_2 such that

$$F(\alpha_1, \alpha_2) = \sum_{j=1}^M w_j (u_j - N(x_j, \alpha_1, \alpha_2))^2$$

is minimized. Here

$$N(x, \alpha_1, \alpha_2) = \frac{1}{\sqrt{2\pi} \alpha_2} \int_{-\infty}^x e^{-\frac{1}{2} \left(\frac{x-\alpha_1}{\alpha_2}\right)^2}$$

while the number w_i represents the weight one wishes to assign to the i th measurement. In a sense the function $N(x, \alpha_1, \alpha_2)$ is the mathematical model one wishes to fit to the data. In general, the model is either determined on theoretical grounds (e.g., by the laws of physics) or represents the experimenter’s choice. Once a least squares formulation has been chosen the problem becomes purely mathematical. The above model has two unknowns α_1 and α_2 , i.e., two degrees of freedom, which are to be pinned down to minimize a weighted sum of the errors between the measurements and the values predicted by the model. As is often the case, the problem is compounded by constraints imposed on the parameters. For example, while α_1 may take on any value the standard deviation α_2 is required to be positive. This means that we have a so-called constrained minimization problem

$$\text{minimize } F(\alpha_1, \alpha_2)$$

over the set

$$-\infty < \alpha_1 < \infty$$

$$\alpha_2 > 0.$$

There is an extensive theory of unconstrained and constrained optimization. Relatively small problems are often solvable with sophisticated search methods like the Nelder-Mead simplex search algorithm found in the Matlab library. Other approaches can be based on descent like techniques. In addition, one knows that at an interior minimum the gradient of F has to vanish so that one may attempt to solve the non-linear system

$$\frac{\partial F}{\partial \alpha_1} = 0$$

$$\frac{\partial F}{\partial \alpha_2} = 0$$

with Newton's method or its many variants. However, non-linear least squares is not for the faint of heart and requires good mathematical insight and/or a powerful program library.

The situation is much improved if the mathematical model is linear in $\vec{\alpha}$ so that it has the general form

$$L(x)\vec{\alpha} = \sum_{j=1}^N g_j(x)\alpha_j$$

where the $\{g_j(x)\}$ are given functions and the $\{\alpha_j\}$ represent N unknown parameters which have to be found such that $(Lx_i)\vec{\alpha}$ approximates u_i for all i . Typically $M \gg N$, i.e., we have many more observations than degrees of freedom. The weighted least squares minimization problem is then

$$\text{minimize } F(\vec{\alpha}) = \sum_{i=1}^M w_i (u_i - (Lx_i)\vec{\alpha})^2$$

If we define the so-called residual

$$r_i(\vec{\alpha}) = u_i - (Lx_i)\vec{\alpha}$$

then the least squares problem can be rewritten in vector-matrix form as

$$\text{minimize } \langle Wr, r \rangle$$

where

W is the diagonal matrix $W = \text{diag}\{w_1, w_2, \dots, w_M\}$

$$r = (r_1, \dots, r_M)$$

and $\langle \cdot, \cdot \rangle$ is the usual dot product for vectors in R_M . Since

$$L(x_i)\vec{\alpha} = \sum_{j=1}^N g_j(x_i)\alpha_j$$

we can express the residual r in the form

$$r = \vec{u} - A\vec{\alpha}$$

where $\vec{u} = (u_1, \dots, u_M)$ and A is an $M \times N$ matrix with entries

$$A_{ij} = g_j(x_i).$$

Hence we need to

$$\text{minimize } F(\vec{\alpha}) = \langle W(u - A\vec{\alpha}), u - A\vec{\alpha} \rangle.$$

A necessary condition is that

$$\frac{\partial F(\vec{\alpha})}{\partial \alpha_j} = 0 \quad \text{for all } j.$$

We compute

$$\frac{\partial F(\vec{\alpha})}{\partial \alpha_j} = \langle -W A_j, u - A\vec{\alpha} \rangle + \langle W(u - A\vec{\alpha}), -A_j \rangle, \quad j = 1, \dots, N$$

where A_j is the j th column of A .

The N equations can be written in matrix form as

$$(WA)^T(\vec{u} - A\vec{\alpha}) + A^T W(\vec{u} - A\vec{\alpha}) = 0$$

which is identical to

$$A^T W A \vec{\alpha} = A^T W \vec{u}.$$

This is a square system which has to be solved for $\vec{\alpha}$. These N equations are sometimes referred to as the normal equations of the linear least squares method. Let us illustrate the linear least squares method in the following setting. Suppose at times $\{t_i\}$ a portfolio has observed values $\{\mathcal{M}_i\}$ for $i = 1, \dots, M$.

It is our belief that $\mathcal{M}(t)$ should grow exponentially according to

$$\mathcal{M}(t) = \alpha_1 e^{\alpha_2 t}.$$

The task is to find a least squares estimate of the parameters $\{\alpha_1, \alpha_2\}$.

We note that our model is not linear so that the matrix formalism would appear not to apply. However, if instead of $\mathcal{M}(t)$ we fit $\log \mathcal{M}(t)$ then we obtain

$$u_i = \log \mathcal{M}(t_i), \quad i = 1, \dots, M$$

$$L(t)\vec{\beta} = 1\beta_1 + t\beta_2$$

where

$$\beta_1 = \log \alpha_1$$

$$\beta_2 = \alpha_2$$

so that A is a $M \times 2$ matrix with entries

$$A_{i1} = 1$$

$$A_{i2} = t_i.$$

The choice of weights will reflect our judgment which of the data points are more significant. If all share equally then we would set W equal to the identity. The normal equations for $W = I$ can now be written in the form of a 2×2 system

$$\begin{pmatrix} \langle A_1, A_1 \rangle & \langle A_1, A_2 \rangle \\ \langle A_1, A_2 \rangle & \langle A_2, A_2 \rangle \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \langle A_1, u \rangle \\ \langle A_2, u \rangle \end{pmatrix}$$

where A_i is the i th column of A and $\langle \cdot, \cdot \rangle$ again denotes the dot product of two vectors. We solve for (β_1, β_2) and then find (α_1, α_2) .

Closely related to the least squares fitting of data is the approximation of a given function defined on an interval $[a, b]$ in terms of a set of other functions.

For example, a financial record may be observed to be periodic in time with a period of, say one month. The question may now be asked whether the overall record is the superposition of processes which are periodic with shorter periods, say half a month, a week, a day, in short, with periods T/n for $n = 1, 2, \dots, N$. If $f(t)$ is the observed record with t measured in days, then the mathematical problem can be stated loosely as: Can f be approximated in terms of the set of functions $\{\sin 2\pi nt/30, \cos 2\pi nt/30\}$ for $n = 1, 2, \dots, N$. If so then, for example, the terms involving $\sin 4\pi t/30$ and $\cos 4\pi t/30$ would roughly be the contribution to the total record of the processes whose period is half a month since both the sine and cosine function go through two complete cycles in every 30 day time span.

In general, the mathematical problem is: Given f defined on $[a, b]$ and functions $\{\phi_n(t)\}_{n=1}^N$ find an approximation Pf to f of the form

$$Pf(t) = \sum_{n=1}^N \alpha_n \phi_n(t).$$

It usually will be the goal to find the coefficients $\vec{\alpha} = (\alpha_1, \dots, \alpha_N)$ which make Pf a “good” approximation to f . However, we can find Pf only after we have decided what makes an approximation “good”. Clearly,

$$E(t, \vec{\alpha}) = f(t) - Pf(t)$$

is the error of the approximation. One may now attempt to determine $\vec{\alpha}$ such that

$$\max_t |E(t, \vec{\alpha})| \leq \max_t |E(t, \vec{\alpha})| \quad \text{for any choice of } \vec{\alpha}.$$

This means the maximum error on $[a, b]$ is minimized which would seem a good criterion for determining $\{\alpha_1, \dots, \alpha_N\}$. Unfortunately, this Pf is not easy to calculate because the standard tools of calculus do not apply since the absolute value function is not differentiable.

The approximation problem becomes tractable if we choose to minimize the error in the mean square sense. This means we wish to find $\vec{\alpha}$ such that

$$\int_a^b E(t, \vec{\alpha})^2 w(t) dt \leq \int_a^b E(t, \vec{\alpha})^2 w(t) dt$$

where we have included a weight function w with the property that w is continuous and that $w(t) > 0$ on $[a, b]$ except at finitely many points where its value may be zero. This problem is solvable because

$$F(\vec{\alpha}) = \int_a^b E(t, \vec{\alpha})^2 w(t) dt$$

is differentiable with respect to each α_i . Hence at a minimum of F we have

$$\frac{\partial F}{\partial \alpha_i} = 0 \quad \text{for } i = 1, \dots, N.$$

From

$$F(\vec{\alpha}) = \int_a^b \left(f(t) - \sum_{n=1}^N \alpha_n \phi_n(t) \right)^2 w(t) dt$$

we find

$$\frac{\partial F}{\partial \alpha_i} = \int_a^b -2 \left(f(t) - \sum_{n=1}^N \alpha_n \phi_n(t) \right) \phi_i(t) w(t) dt = 0.$$

These N equations can be written in matrix form as

$$A\vec{\alpha} = b$$

where the $N \times N$ matrix A and the N -vector b have the entries

$$A_{ij} = \int_a^b \phi_i(t) \phi_j(t) w(t) dt$$

$$b_i = \int_a^b f(t) \phi_i(t) w(t) dt.$$

Under the natural condition that the functions $\{\phi_n(t)\}$ are linearly independent, i.e., that no function is a linear combination of the remaining functions, the matrix A will be invertible and the linear system is uniquely solvable. Since the error can be made large by taking very large coefficients it is clear that the critical point where $\partial F / \partial \alpha_i = 0$ is in fact a minimizer for the mean square error.

As an illustration consider the following problem: Find the polynomial of degree ≤ 2 which approximates best in the mean square sense (with weight function $w(t) \equiv 1$) the function $f(t) = t^3$ over the interval $[-1, 1]$. In this case

$$Pf(t) = \alpha_0 1 + \alpha_1 t + \alpha_2 t^2$$

where $\phi_0(t) \equiv 1$, $\phi_1(t) \equiv t$ and $\phi_2(t) \equiv t^2$. The system $A\vec{\alpha} = b$ (with indices running from 0 to 2) has the following entries

$$\begin{aligned}
 A_{00} &= \int_{-1}^1 1 \cdot 1 \, dt = 2 & A_{01} &= \int_{-1}^1 1t \, dt = 0 & A_{02} &= \int_{-1}^1 1t^2 \, dt = 2/3 \\
 A_{10} &= \int_{-1}^1 1t \, dt = 0 & A_{11} &= \int_{-1}^1 tt \, dt = 2/3 & A_{12} &= \int_{-1}^1 tt^2 \, dt = 0 \\
 A_{20} &= \int_{-1}^1 1t^2 \, dt = 2/3 & A_{21} &= \int_{-1}^1 tt^2 \, dt = 0 & A_{22} &= \int_{-1}^1 t^2t^2 \, dt = 2/5 \\
 b_0 &= \int_{-1}^1 t^3 \cdot 1 \, dt \\
 b_1 &= \int_{-1}^1 t^3 t \, dt = 2/5 \\
 b_2 &= \int_{-1}^1 t^3 t^2 \, dt = 0.
 \end{aligned}$$

The equation

$$\begin{pmatrix} 2 & 0 & 2/3 \\ 0 & 2/3 & 0 \\ 2/3 & 0 & 2/5 \end{pmatrix} \vec{\alpha} = \begin{pmatrix} 0 \\ 2/5 \\ 0 \end{pmatrix}$$

has the unique solution

$$\vec{\alpha} = \begin{pmatrix} 0 \\ 3/5 \\ 0 \end{pmatrix}$$

so that

$$Pf(t) = 3/5t$$

In many applications the entries of A have the property that

$$A_{ij} = 0 \quad \text{for } i \neq j.$$

The functions $\{\phi_n\}$ are then said to be orthogonal with respect to the weight function w .

In this case it follows that

$$\alpha_i = \frac{\int_a^b f(t)\phi_i(t)w(t)dt}{\int_a^b \phi_i(t)\phi_i(t)w(t)dt}.$$

In this setting the α_i is known as a Fourier coefficient. This situation arises routinely when f is approximated by a Fourier series where the functions $\{\phi_n\}$ are trigonometric functions as outlined above.