Numerical Analysis Lecture Notes

Peter J. Olver

13. Approximation and Interpolation

We will now apply our minimization results to the interpolation and least squares fitting of data and functions.

13.1. Least Squares.

Linear systems with more equations than unknowns typically do not have solutions. In such situations, the least squares solution to a linear system is one means of getting as close as one can to an actual solution.

Definition 13.1. A *least squares solution* to a linear system of equations

$$A\mathbf{x} = \mathbf{b} \tag{13.1}$$

is a vector $\mathbf{x}^\star \in \mathbb{R}^n$ that minimizes the Euclidean norm $\| A\mathbf{x} - \mathbf{b} \|$.

If the system (13.1) actually has a solution, then it is automatically the least squares solution. Thus, the concept of least squares solution is new only when the system does not have a solution.

To find the least squares solution, we need to minimize the quadratic function

$$\begin{aligned} \| A\mathbf{x} - \mathbf{b} \|^2 &= (A\mathbf{x} - \mathbf{b})^T (A\mathbf{x} - \mathbf{b}) = (\mathbf{x}^T A^T - \mathbf{b}^T)(A\mathbf{x} - \mathbf{b}) \\ &= \mathbf{x}^T A^T A\mathbf{x} - 2\,\mathbf{x}^T A^T \mathbf{b} + \mathbf{b}^T \mathbf{b} = \mathbf{x}^T K \mathbf{x} - 2\,\mathbf{x}^T \mathbf{f} + c, \end{aligned}$$

where

$$K = A^T A, \qquad \mathbf{f} = A^T \mathbf{b}, \qquad c = \| \mathbf{b} \|^2. \tag{13.2}$$

According to Theorem 12.10, the Gram matrix $K = A^T A$ is positive definite if and only if $\ker A = \{0\}$. In this case, Theorem 12.12 supplies us with the solution to this minimization problem.

Theorem 13.2. *Assume that* $\ker A = \{\mathbf{0}\}$. *Set* $K = A^T A$ *and* $\mathbf{f} = A^T \mathbf{b}$. *Then the least squares solution to the linear system* $A\mathbf{x} = \mathbf{b}$ *is the unique solution* $\mathbf{x}^\star$ *to the so-called* normal equations

$$K\mathbf{x} = \mathbf{f} \qquad \textit{or, explicitly,} \qquad (A^T A)\,\mathbf{x} = A^T \mathbf{b}, \tag{13.3}$$

namely

$$\mathbf{x}^\star = (A^T A)^{-1} A^T \mathbf{b}. \tag{13.4}$$

The least squares error is

$$\| A\mathbf{x}^\star - \mathbf{b} \|^2 = \| \mathbf{b} \|^2 - \mathbf{f}^T \mathbf{x}^\star = \| \mathbf{b} \|^2 - \mathbf{b}^T A (A^T A)^{-1} A^T \mathbf{b}. \tag{13.5}$$

Note that the normal equations (13.3) can be simply obtained by multiplying the original system $A\mathbf{x} = \mathbf{b}$ on both sides by A^T. In particular, if A is square and invertible, then $(A^T A)^{-1} = A^{-1}(A^T)^{-1}$, and so (13.4) reduces to $\mathbf{x} = A^{-1}\mathbf{b}$, while the two terms in the error formula (13.5) cancel out, producing zero error. In the rectangular case — when inversion is *not* allowed — (13.4) gives a *new* formula for the solution to a compatible linear system $A\mathbf{x} = \mathbf{b}$.

Example 13.3. Consider the linear system

$$\begin{aligned} x_1 + 2x_2 \qquad &= 1, \\ 3x_1 - x_2 + x_3 &= 0, \\ -x_1 + 2x_2 + x_3 &= -1, \\ x_1 - x_2 - 2x_3 &= 2, \\ 2x_1 + x_2 - x_3 &= 2, \end{aligned}$$

consisting of 5 equations in 3 unknowns. The coefficient matrix and right hand side are

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 3 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & -2 \\ 2 & 1 & -1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 2 \\ 2 \end{pmatrix}.$$

A direct application of Gaussian Elimination shows that the system is incompatible — it has no solution. Of course, to apply the least squares method, we are not required to check this in advance. If the system has a solution, it is the least squares solution too, and the least squares method will find it.

To form the normal equations (13.3), we compute

$$K = A^T A = \begin{pmatrix} 16 & -2 & -2 \\ -2 & 11 & 2 \\ -2 & 2 & 7 \end{pmatrix}, \qquad \mathbf{f} = A^T \mathbf{b} = \begin{pmatrix} 8 \\ 0 \\ -7 \end{pmatrix}.$$

Solving the 3×3 system $K\mathbf{x} = \mathbf{f}$ by Gaussian Elimination, we find

$$\mathbf{x} = K^{-1}\mathbf{f} \approx (\, .4119, .2482, -.9532 \,)^T$$

to be the least squares solution to the system. The least squares error is

$$\| \mathbf{b} - A\mathbf{x}^\star \| \;\approx\; \| (\, -.0917, .0342, .1313, .0701, .0252 \,)^T \| \;\approx\; .1799,$$

which is reasonably small — indicating that the system is, roughly speaking, not too incompatible.

13.2. Data Fitting and Interpolation.

One of the most important applications of the least squares minimization process is to the fitting of data points. Suppose we are running an experiment in which we measure a certain time-dependent physical quantity. At time t_i we make the measurement y_i, and thereby obtain a set of, say, m data points

$$(t_1, y_1), \qquad (t_2, y_2), \qquad \ldots \qquad (t_m, y_m). \tag{13.6}$$

Suppose our theory indicates that the data points are supposed to all lie on a single line

$$y = \alpha + \beta t, \tag{13.7}$$

whose precise form — meaning its coefficients α, β — is to be determined. For example, a police car is interested in clocking the speed of a vehicle by using measurements of its relative distance at several times. Assuming that the vehicle is traveling at constant speed, its position at time t will have the linear form (13.7), with β, the velocity, and α, the initial position, to be determined. Experimental error will almost inevitably make this impossible to achieve exactly, and so the problem is to find the straight line (13.7) that "best fits" the measured data and then use its slope to estimate the vehicle's velocity.

The *error* between the measured value y_i and the sample value predicted by the function (13.7) at $t = t_i$ is

$$e_i = y_i - (\alpha + \beta t_i), \qquad\qquad i = 1, \ldots, m.$$

We can write this system of equations in matrix form as

$$\mathbf{e} = \mathbf{y} - A\mathbf{x},$$

where

$$\mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}, \qquad \text{while} \qquad A = \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{13.8}$$

We call $\mathbf{e}$ the *error vector* and $\mathbf{y}$ the *data vector*. The coefficients α, β of our desired function (13.7) are the unknowns, forming the entries of the column vector $\mathbf{x}$.

If we could fit the data exactly, so $y_i = \alpha + \beta t_i$ for all i, then each $e_i = 0$, and we could solve $A\mathbf{x} = \mathbf{y}$ for the coefficients α, β. In the language of linear algebra, the data points all lie on a straight line if and only if $\mathbf{y} \in \operatorname{rng} A$. If the data points are not collinear, then we seek the straight line that minimizes the total squared error or Euclidean norm

$$\text{Error} = \| \mathbf{e} \| = \sqrt{e_1^2 + \cdots + e_m^2}\,.$$

Pictorially, referring to Figure 13.1, the errors are the vertical distances from the points to the line, and we are seeking to minimize the square root of the sum of the squares of

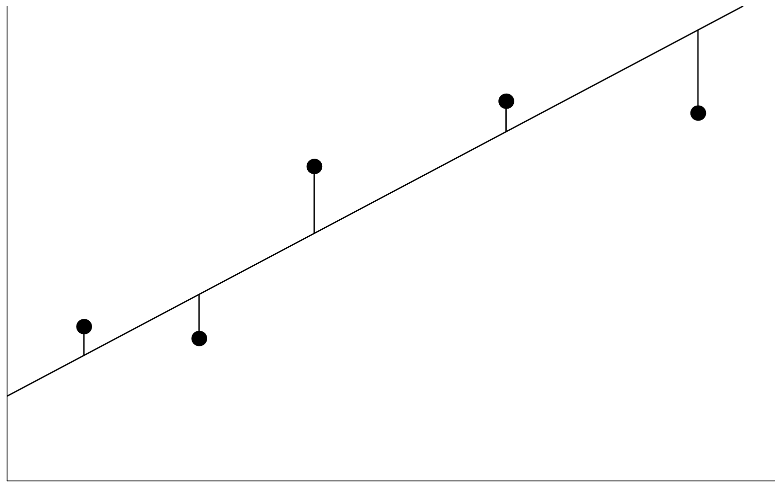


Figure 13.1. Least Squares Approximation of Data by a Straight Line.

the individual errors[†], hence the term *least squares*. In other words, we are looking for the coefficient vector $\mathbf{x} = (\alpha, \beta)^T$ that minimizes the Euclidean norm of the error vector

$$\| \mathbf{e} \| = \| A\mathbf{x} - \mathbf{y} \|. \tag{13.9}$$

Thus, we recover the problem of characterizing the least squares solution to the linear system $A\mathbf{x} = \mathbf{y}$.

Theorem 13.2 prescribes the solution to this least squares minimization problem. We form the normal equations

$$(A^T A)\mathbf{x} = A^T\mathbf{y}, \qquad \text{with solution} \qquad \mathbf{x}^\star = (A^T A)^{-1}A^T\mathbf{y}. \tag{13.10}$$

Invertibility of the Gram matrix $K = A^T A$ relies on the assumption that the matrix A has linearly independent columns. For the particular matrix in (13.8), linear independence of its two columns requires that not all the t_i's be equal, i.e., we must measure the data at at least two distinct times. Note that this restriction does not preclude measuring some of the data at the same time, e.g., by repeating the experiment. However, choosing *all* the t_i's to be the same is a silly data fitting problem. (Why?)

[†] This choice of minimization may strike the reader as a little odd. Why not just minimize the sum of the absolute value of the errors, i.e., the 1 norm $\| \mathbf{e} \|_1 = | e_1 | + \cdots + | e_n |$ of the error vector, or minimize the maximal error, i.e., the ∞ norm $\| \mathbf{e} \|_\infty = \max\{ | e_1 |, \ldots, | e_n | \}$? Or, even better, why minimize the vertical distance to the line? The perpendicular distance from each data point to the line might strike you as a better measure of error. The answer is that, although each of these alternative minimization criteria is interesting and potentially useful, they all lead to *nonlinear* minimization problems, and so are much harder to solve! The least squares minimization problem can be solved by linear algebra, and so, purely on the grounds of simplicity, is the method of choice in most applications. Moreover, one needs to fully understand the linear problem before diving into more treacherous nonlinear waters.

Under this assumption, we then compute

$$A^T A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_m \end{pmatrix} \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix} = \begin{pmatrix} m & \sum t_i \\ \sum t_i & \sum (t_i)^2 \end{pmatrix} = m \begin{pmatrix} 1 & \overline{t} \\ \overline{t} & \overline{t^2} \end{pmatrix},$$

$$A^T \mathbf{y} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_m \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum t_i y_i \end{pmatrix} = m \begin{pmatrix} \overline{y} \\ \overline{t\,y} \end{pmatrix}, \qquad (13.11)$$

where the overbars, namely

$$\overline{t} = \frac{1}{m} \sum_{i=1}^{m} t_i, \qquad \overline{y} = \frac{1}{m} \sum_{i=1}^{m} y_i, \qquad \overline{t^2} = \frac{1}{m} \sum_{i=1}^{m} t_i^2, \qquad \overline{t\,y} = \frac{1}{m} \sum_{i=1}^{m} t_i y_i, \qquad (13.12)$$

denote the *average* sample values of the indicated variables.

Warning: The average of a product is *not* equal to the product of the averages! In particular,

$$\overline{t^2} \neq (\,\overline{t}\,)^2, \qquad \overline{t\,y} \neq \overline{t}\,\overline{y}.$$

Substituting (13.11) into the normal equations (13.10), and canceling the common factor of m, we find that we have only to solve a pair of linear equations

$$\alpha + \overline{t}\,\beta = \overline{y}, \qquad \overline{t}\,\alpha + \overline{t^2}\,\beta = \overline{t\,y},$$

for the coefficients:

$$\alpha = \overline{y} - \overline{t}\,\beta, \qquad \beta = \frac{\overline{t\,y} - \overline{t}\,\overline{y}}{\overline{t^2} - (\,\overline{t}\,)^2} = \frac{\sum (t_i - \overline{t}\,)\, y_i}{\sum (t_i - \overline{t}\,)^2}\,. \qquad (13.13)$$

Therefore, the best (in the least squares sense) straight line that fits the given data is

$$y = \beta\,(t - \overline{t}\,) + \overline{y}, \qquad (13.14)$$

where the line's slope β is given in (13.13).

Example 13.4. Suppose the data points are given by the table

t_i	0	1	3	6
y_i	2	3	7	12

To find the least squares line, we construct

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \\ 1 & 6 \end{pmatrix}, \qquad A^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 6 \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} 2 \\ 3 \\ 7 \\ 12 \end{pmatrix}.$$

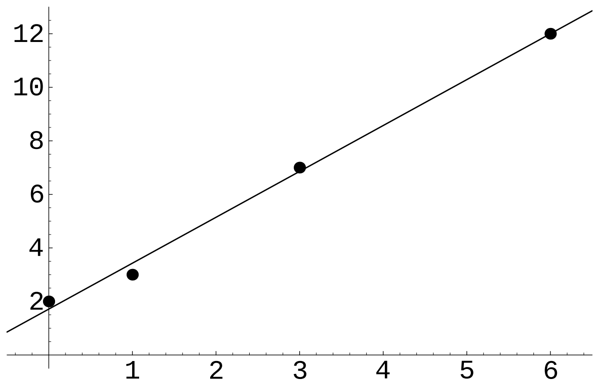


Figure 13.2. Least Squares Line.

Therefore

$$A^T A = \begin{pmatrix} 4 & 10 \\ 10 & 46 \end{pmatrix}, \qquad A^T \mathbf{y} = \begin{pmatrix} 24 \\ 96 \end{pmatrix}.$$

The normal equations (13.10) reduce to

$$4\alpha + 10\beta = 24, \qquad 10\alpha + 46\beta = 96, \qquad \text{so} \qquad \alpha = \tfrac{12}{7}, \qquad \beta = \tfrac{12}{7}.$$

Therefore, the best least squares fit to the data is the straight line

$$y = \tfrac{12}{7} + \tfrac{12}{7}\, t.$$

Alternatively, one can compute this formula directly from (13.13–14). As you can see in Figure 13.2, the least squares line does a fairly good job of approximating the data points.

Example 13.5. Suppose we are given a sample of an unknown radioactive isotope. At time t_i we measure, using a Geiger counter, the amount m_i of radioactive material in the sample. The problem is to determine the initial amount of material along with the isotope's half life. If the measurements were exact, we would have $m(t) = m_0 e^{\beta t}$, where $m_0 = m(0)$ is the initial mass, and $\beta < 0$ the decay rate. The half-life is given by $t^\star = \beta^{-1} \log 2$.

As it stands this is *not* a linear least squares problem. But it can be easily converted to the proper form by taking logarithms:

$$y(t) = \log m(t) = \log m_0 + \beta t = \alpha + \beta t.$$

We can thus do a linear least squares fit on the logarithms $y_i = \log m_i$ of the radioactive mass data at the measurement times t_i to determine the best values for β and $\alpha = \log m_0$.

Polynomial Approximation and Interpolation

The basic least squares philosophy has a variety of different extensions, all interesting and all useful. First, we can replace the straight line (13.7) by a parabola defined by a quadratic function

$$y = \alpha + \beta t + \gamma t^2. \tag{13.15}$$

For example, Newton's theory of gravitation says that (in the absence of air resistance) a falling object obeys the parabolic law (13.15), where $\alpha = h_0$ is the initial height, $\beta = v_0$ is the initial velocity, and $\gamma = -\frac{1}{2} g$ is minus one half the gravitational constant. Suppose we observe a falling body on a new planet, and measure its height y_i at times t_i. Then we can approximate its initial height, initial velocity and gravitational acceleration by finding the parabola (13.15) that best fits the data. Again, we characterize the least squares fit by minimizing the sum of the squares of the individual errors $e_i = y_i - y(t_i)$.

The method can evidently be extended to a completely general polynomial function

$$y(t) = \alpha_0 + \alpha_1 t + \cdots + \alpha_n t^n \tag{13.16}$$

of degree n. The total least squares error between the data and the sample values of the function is equal to

$$\| \mathbf{e} \|^2 = \sum_{i=1}^{m} \big[\, y_i - y(t_i) \, \big]^2 = \| \mathbf{y} - A \mathbf{x} \|^2, \tag{13.17}$$

where

$$A = \begin{pmatrix} 1 & t_1 & t_1^2 & \ldots & t_1^n \\ 1 & t_2 & t_2^2 & \ldots & t_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \ldots & t_m^n \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}. \tag{13.18}$$

The coefficient $m \times (n+1)$ coefficient matrix is known as a *Vandermonde matrix*, named after the eighteenth century French mathematician, scientist and musicologist Alexandre–Théophile Vandermonde — despite the fact that it appears nowhere in his four mathematical papers! In particular, if $m = n + 1$, then A is square, and so, assuming invertibility, we can solve $A\mathbf{x} = \mathbf{y}$ exactly. In other words, there is no error, and the solution is an *interpolating polynomial*, meaning that it fits the data exactly. A proof of the following result can be found at the end of this section.

Lemma 13.6. *If $t_1, \ldots, t_{n+1}$ are distinct, $t_i \neq t_j$, then the $(n+1) \times (n+1)$ Vandermonde interpolation matrix* (13.18) *is nonsingular.*

This result immediately implies the basic existence theorem for interpolating polynomials.

Theorem 13.7. *Let $t_1, \ldots, t_{n+1}$ be distinct sample points. Then, for any prescribed data $y_1, \ldots, y_{n+1}$, there exists a unique interpolating polynomial of degree $\leq n$ with the prescribed sample values $y(t_i) = y_i$ for all $i = 1, \ldots, n+1$.*

Thus, two points will determine a unique interpolating line, three points a unique interpolating parabola, four points an interpolating cubic, and so on; see Figure 13.3.

The basic ideas of interpolation and least squares fitting of data can be applied to approximate complicated mathematical functions by much simpler polynomials. Such approximation schemes are used in all numerical computations. Your computer or calculator is only able to add, subtract, multiply and divide. Thus, when you ask it to compute $\sqrt{t}$ or

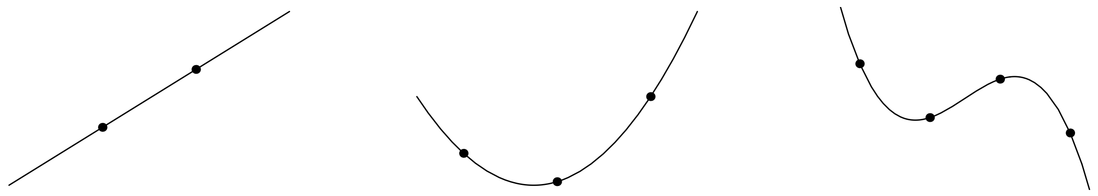


Figure 13.3. Interpolating Polynomials.

e^t or $\cos t$ or any other non-rational function, the program must rely on an approximation scheme based on polynomials[†]. In the "dark ages" before computers, one would consult precomputed tables of values of the function at particular data points. If one needed a value at a nontabulated point, then some form of polynomial interpolation would be used to accurately approximate the intermediate value.

Example 13.8. Suppose that we would like to compute reasonably accurate values for the exponential function e^t for values of t lying in the interval $0 \le t \le 1$ by approximating it by a quadratic polynomial

$$p(t) = \alpha + \beta t + \gamma t^2. \tag{13.19}$$

If we choose 3 points, say $t_1 = 0, t_2 = .5, t_3 = 1$, then there is a unique quadratic polynomial (13.19) that interpolates e^t at the data points, i.e.,

$$p(t_i) = e^{t_i} \qquad \text{for} \qquad i = 1, 2, 3.$$

In this case, the coefficient matrix (13.18), namely

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & .5 & .25 \\ 1 & 1 & 1 \end{pmatrix},$$

is nonsingular. Therefore, we can exactly solve the interpolation equations

$$A\mathbf{x} = \mathbf{y}, \qquad \text{where} \qquad \mathbf{y} = \begin{pmatrix} e^{t_1} \\ e^{t_2} \\ e^{t_3} \end{pmatrix} = \begin{pmatrix} 1. \\ 1.64872 \\ 2.71828 \end{pmatrix}$$

is the data vector, which we assume we already know. The solution

$$\mathbf{x} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} 1. \\ .876603 \\ .841679 \end{pmatrix}$$

[†] Actually, one could also allow interpolation and approximation by rational functions, a subject known as *Padé approximation theory*, [**3**].

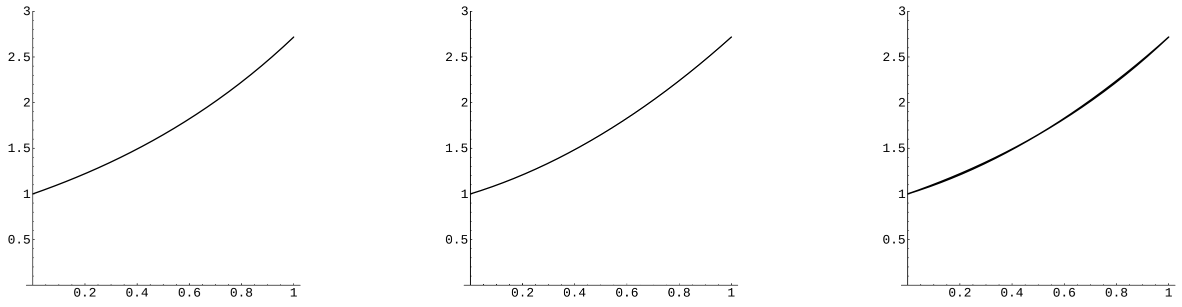


Figure 13.4. Quadratic Interpolating Polynomial for e^t.

yields the interpolating polynomial

$$p(t) = 1 + .876603\,t + .841679\,t^2. \tag{13.20}$$

It is the unique quadratic polynomial that agrees with e^t at the three specified data points. See Figure 13.4 for a comparison of the graphs; the first graph shows e^t, the second $p(t)$, and the third lays the two graphs on top of each other. Even with such a primitive interpolation scheme, the two functions are quite close. The maximum error or L^∞ norm of the difference is

$$\| \, e^t - p(t) \, \|_\infty = \max \left\{ \, | \, e^t - p(t) \, | \; \middle| \; 0 \leq t \leq 1 \, \right\} \approx .01442,$$

with the largest deviation occurring at $t \approx .796$.

There is, in fact, an explicit formula for the interpolating polynomial that is named after the influential eighteenth century Italo–French mathematician Joseph–Louis Lagrange. Suppose we know the solutions $\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}$ to the particular interpolation systems

$$A\,\mathbf{x}_k = \mathbf{e}_k, \qquad k = 1, \ldots, n+1, \tag{13.21}$$

where $\mathbf{e}_1, \ldots, \mathbf{e}_{n+1}$ are the standard basis vectors of $\mathbb{R}^{n+1}$. Then the solution to

$$A\,\mathbf{x} = \mathbf{y} = y_1\,\mathbf{e}_1 + \;\cdots\; + y_{n+1}\,\mathbf{e}_{n+1}$$

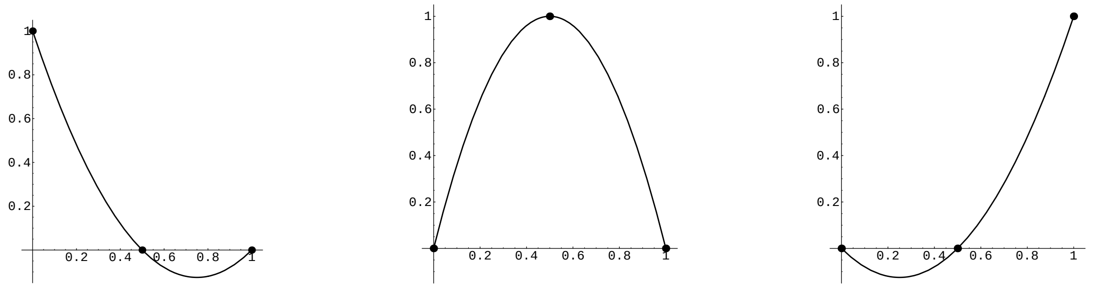
is given by the superposition formula

$$\mathbf{x} = y_1\,\mathbf{x}_1 + \;\cdots\; + y_{n+1}\,\mathbf{x}_{n+1}.$$

The particular interpolation equation (13.21) corresponds to the interpolation data $\mathbf{y} = \mathbf{e}_k$, meaning that $y_k = 1$, while $y_i = 0$ at all points t_i with $i \neq k$. If we can find the $n+1$ particular interpolating polynomials that realize this very special data, we can use superposition to construct the general interpolating polynomial.

Theorem 13.9. *Given distinct sample points $t_1, \ldots, t_{n+1}$, the* k^{th} Lagrange interpolating polynomial *is given by*

$$L_k(t) = \frac{(t - t_1)\;\cdots\;(t - t_{k-1})(t - t_{k+1})\;\cdots\;(t - t_{n+1})}{(t_k - t_1)\;\cdots\;(t_k - t_{k-1})(t_k - t_{k+1})\;\cdots\;(t_k - t_{n+1})}\,, \qquad k = 1, \ldots, n+1. \tag{13.22}$$



$L_1(t)$ $\qquad$ $L_2(t)$ $\qquad$ $L_3(t)$

Figure 13.5. Lagrange Interpolating Polynomials for the Points $0, .5, 1$.

It is the unique polynomial of degree n that satisfies

$$L_k(t_i) = \begin{cases} 1, & i = k, \\ 0, & i \neq k, \end{cases} \qquad i, k = 1, \ldots, n+1. \tag{13.23}$$

Proof: The uniqueness of the Lagrange interpolating polynomial is an immediate consequence of Theorem 13.7. To show that (13.22) is the correct formula, we note that when $t = t_i$ for any $i \neq k$, the factor $(t - t_i)$ in the numerator of $L_k(t)$ vanishes, while the denominator is not zero since the points are distinct. On the other hand, when $t = t_k$, the numerator and denominator are equal, and so $L_k(t_k) = 1$. *Q.E.D.*

Theorem 13.10. *If $t_1, \ldots, t_{n+1}$ are distinct, then the polynomial of degree $\leq n$ that interpolates the associated data $y_1, \ldots, y_{n+1}$ is*

$$p(t) = y_1 \, L_1(t) + \ \cdots \ + y_{n+1} \, L_{n+1}(t). \tag{13.24}$$

Proof: We merely compute

$$p(t_k) = y_1 \, L_1(t_k) + \ \cdots \ + y_k \, L_k(t) + \ \cdots \ + y_{n+1} \, L_{n+1}(t_k) = y_k,$$

where, according to (13.23), every summand except the kth is zero. *Q.E.D.*

Example 13.11. For example, the three quadratic Lagrange interpolating polynomials for the values $t_1 = 0, t_2 = \frac{1}{2}, t_3 = 1$ used to interpolate e^t in Example 13.8 are

$$\begin{aligned} L_1(t) &= \frac{\left(t - \frac{1}{2}\right)(t-1)}{\left(0 - \frac{1}{2}\right)(0-1)} = 2t^2 - 3t + 1, \\ L_2(t) &= \frac{(t-0)(t-1)}{\left(\frac{1}{2} - 0\right)\left(\frac{1}{2} - 1\right)} = -4t^2 + 4t, \\ L_3(t) &= \frac{(t-0)\left(t - \frac{1}{2}\right)}{(1-0)\left(1 - \frac{1}{2}\right)} = 2t^2 - t. \end{aligned} \tag{13.25}$$

Thus, we can rewrite the quadratic interpolant (13.20) to e^t as

$$\begin{aligned} y(t) &= L_1(t) + e^{1/2} \, L_2(t) + e \, L_3(t) \\ &= (2t^2 - 3t + 1) + 1.64872(-4t^2 + 4t) + 2.71828(2t^2 - t). \end{aligned}$$

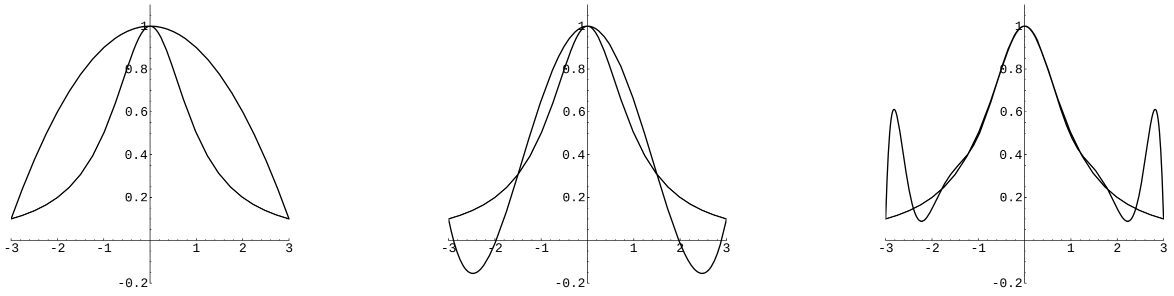


Figure 13.6. Degree 2, 4 and 10 Interpolating Polynomials for $1/(1+t^2)$.

We stress that this is the *same* interpolating polynomial — we have merely rewritten it in the more transparent Lagrange form.

You might expect that the higher the degree, the more accurate the interpolating polynomial. This expectation turns out, unfortunately, not to be uniformly valid. While low degree interpolating polynomials are usually reasonable approximants to functions, high degree interpolants are not only more expensive to compute, but can be rather badly behaved, particularly near the ends of the interval. For example, Figure 13.6 displays the degree $2, 4$ and 10 interpolating polynomials for the function $1/(1+t^2)$ on the interval $-3 \le t \le 3$ using equally spaced data points. Note the rather poor approximation of the function near the ends of the interval. Higher degree interpolants fare even worse, although the bad behavior becomes more and more concentrated near the endpoints. As a consequence, high degree polynomial interpolation tends not to be used in practical applications. Better alternatives rely on least squares approximants by low degree polynomials, to be described next, and interpolation by piecewise cubic splines, a topic that will be discussed in depth later.

If we have $m > n+1$ data points, then, usually, there is no degree n polynomial that fits all the data, and so we must switch over to a least squares approximation. The first requirement is that the associated $m \times (n+1)$ interpolation matrix (13.18) has rank $n+1$; this follows from Lemma 13.6, provided that at least $n+1$ of the values $t_1, \ldots, t_m$ are distinct. Thus, given data at $m \ge n+1$ different sample points $t_1, \ldots, t_m$, we can uniquely determine the best least squares polynomial of degree n that fits the data by solving the normal equations (13.10).

Example 13.12. Let us return to the problem of approximating the exponential function e^t. If we use more than three data points, but still require a quadratic polynomial, then we can no longer interpolate exactly, and must devise a least squares approximant. For instance, using five equally spaced sample points $t_1 = 0$, $t_2 = .25$, $t_3 = .5$, $t_4 = .75$, $t_5 = 1$, the coefficient matrix and sampled data vector (13.18) are

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & .25 & .0625 \\ 1 & .5 & .25 \\ 1 & .75 & .5625 \\ 1 & 1 & 1 \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} 1. \\ 1.28403 \\ 1.64872 \\ 2.11700 \\ 2.71828 \end{pmatrix}.$$

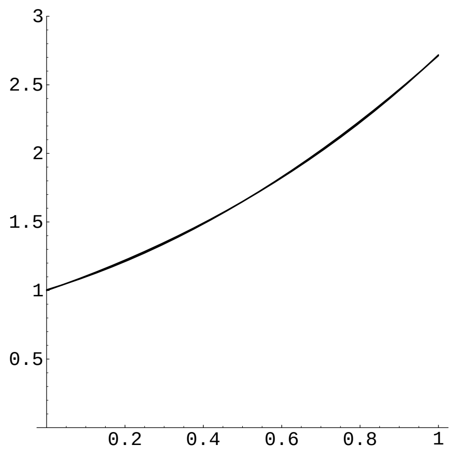
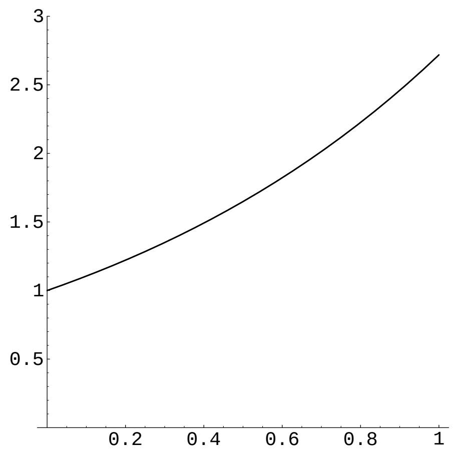



Figure 13.7. Quadratic Approximant and Quartic Interpolant for e^t.

The solution to the normal equations (13.3), with

$$K = A^T A = \begin{pmatrix} 5. & 2.5 & 1.875 \\ 2.5 & 1.875 & 1.5625 \\ 1.875 & 1.5625 & 1.38281 \end{pmatrix}, \qquad \mathbf{f} = A^T \mathbf{y} = \begin{pmatrix} 8.76803 \\ 5.45140 \\ 4.40153 \end{pmatrix},$$

is

$$\mathbf{x} = K^{-1}\mathbf{f} = (\, 1.00514, .864277, .843538 \,)^T .$$

This leads to the quadratic least squares approximant

$$p_2(t) = 1.00514 + .864277\, t + .843538\, t^2.$$

On the other hand, the quartic interpolating polynomial

$$p_4(t) = 1 + .998803\, t + .509787\, t^2 + .140276\, t^3 + .069416\, t^4$$

is found directly from the data values as above. The quadratic polynomial has a maximal error of $\approx .011$ over the interval $[0,1]$ — slightly better than the quadratic interpolant — while the quartic has a significantly smaller maximal error: $\approx .0000527$. (In this case, high degree interpolants are not ill behaved.) See Figure 13.7 for a comparison of the graphs.

Proof of Lemma 13.6: We will establish the rather striking LU factorization of the transposed Vandermonde matrix $V = A^T$, which will immediately prove that, when $t_1, \ldots, t_{n+1}$ are distinct, both V and A are nonsingular matrices. The 4×4 case is instructive for understanding the general pattern. Applying regular Gaussian Elimination, we find the explicit LU factorization

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ t_1 & t_2 & t_3 & t_4 \\ t_1^2 & t_2^2 & t_3^2 & t_4^2 \\ t_1^3 & t_2^3 & t_3^3 & t_4^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_1 & 1 & 0 & 0 \\ t_1^2 & t_1 + t_2 & 1 & 0 \\ t_1^3 & t_1^2 + t_1 t_2 + t_2^2 & t_1 + t_2 + t_3 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & t_2 - t_1 & t_3 - t_1 & t_4 - t_1 \\ 0 & 0 & (t_3 - t_1)(t_3 - t_2) & (t_4 - t_1)(t_4 - t_2) \\ 0 & 0 & 0 & (t_4 - t_1)(t_4 - t_2)(t_4 - t_3) \end{pmatrix}.$$

In the general $(n+1)\times(n+1)$ case, the individual entries of the matrices appearing in factorization $V = LU$ are

$$v_{ij} = t_j^{i-1}, \qquad i,j = 1,\ldots,n+1, \tag{13.26}$$

$$\ell_{ij} = \sum_{1 \le k_1 \le \cdots \le k_{i-j} \le j} t_{k_1} t_{k_2} \cdots t_{k_{i-j}}, \quad 1 \le j < i \le n+1, \qquad \begin{aligned} \ell_{ii} &= 1, \quad i = 1,\ldots,n+1, \\ \ell_{ij} &= 0, \quad 1 \le i < j \le n+1, \end{aligned}$$

$$u_{ij} = \prod_{k=1}^{i} (t_j - t_k), \qquad 1 < i \le j \le n+1, \qquad \begin{aligned} u_{1j} &= 1, \quad j = 1,\ldots,n+1, \\ u_{ij} &= 0, \quad 1 \le j < i \le n+1. \end{aligned}$$

Full details of the proof that $V = LU$ can be found in [**21, 45**]. (Surprisingly, as far as we know, these are the first places this factorization appears in the literature.) The entries of L lying below the diagonal are known as the *complete monomial polynomials* since ℓ_{ij} is obtained by summing, with unit coefficients, all monomials of degree $i-j$ in the j variables $t_1, \ldots, t_j$. The entries of U appearing on or above the diagonal are known as the *Newton difference polynomials*. In particular, if $t_1, \ldots, t_n$ are distinct, so $t_i \ne t_j$ for $i \ne j$, all entries of U lying on or above the diagonal are nonzero. In this case, V has all nonzero pivots, and is a regular, hence nonsingular matrix. *Q.E.D.*

Approximation and Interpolation by General Functions

There is nothing special about polynomial functions in the preceding approximation scheme. For example, suppose we were interested in finding the best trigonometric approximation

$$y = \alpha_1 \cos t + \alpha_2 \sin t$$

to a given set of data. Again, the least squares error takes the same form $\| \mathbf{y} - A\mathbf{x} \|^2$ as in (13.17), where

$$A = \begin{pmatrix} \cos t_1 & \sin t_1 \\ \cos t_2 & \sin t_2 \\ \vdots & \vdots \\ \cos t_m & \sin t_m \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}.$$

Thus, the columns of A are the sampled values of the functions $\cos t, \sin t$. The key is that the unspecified parameters — in this case α_1, α_2 — occur *linearly* in the approximating function. Thus, the most general case is to approximate the data (13.6) by a linear combination

$$y(t) = \alpha_1 h_1(t) + \alpha_2 h_2(t) + \cdots + \alpha_n h_n(t)$$

of prescribed functions $h_1(x), \ldots, h_n(x)$. The least squares error is, as always, given by

$$\text{Error} = \sqrt{\sum_{i=1}^{m} \big(y_i - y(t_i) \big)^2} = \| \mathbf{y} - A\,\mathbf{x} \|,$$

where the sample matrix A, the vector of unknown coefficients $\mathbf{x}$, and the data vector $\mathbf{y}$ are

$$A = \begin{pmatrix} h_1(t_1) & h_2(t_1) & \ldots & h_n(t_1) \\ h_1(t_2) & h_2(t_2) & \ldots & h_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(t_m) & h_2(t_m) & \ldots & h_n(t_m) \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}. \tag{13.27}$$

If A is square and nonsingular, then we can find an interpolating function of the prescribed form by solving the linear system

$$A\,\mathbf{x} = \mathbf{y}. \tag{13.28}$$

A particularly important case is provided by the $2n+1$ trigonometric functions

$$1, \qquad \cos x, \qquad \sin x, \qquad \cos 2x, \qquad \sin 2x, \qquad \ldots \qquad \cos nx, \qquad \sin nx.$$

Interpolation on $2n+1$ equally spaced data points on the interval $[0, 2\pi]$ leads to the Discrete Fourier Transform, used in signal processing, data transmission, and compression.

If there are more than n data points, then we cannot, in general, interpolate exactly, and must content ourselves with a least squares approximation that minimizes the error at the sample points as best it can. The least squares solution to the interpolation equations (13.28) is found by solving the associated normal equations $K\,\mathbf{x} = \mathbf{f}$, where the (i,j) entry of $K = A^T A$ is m times the average sample value of the product of $h_i(t)$ and $h_j(t)$, namely

$$k_{ij} = m\,\overline{h_i(t)\,h_j(t)} = \sum_{\kappa=1}^{m} h_i(t_\kappa)\,h_j(t_\kappa), \tag{13.29}$$

whereas the i^{th} entry of $\mathbf{f} = A^T \mathbf{y}$ is

$$f_i = m\,\overline{h_i(t)\,y} = \sum_{\kappa=1}^{m} h_i(t_\kappa)\,y_\kappa. \tag{13.30}$$

The one issue is whether the columns of the sample matrix A are linearly independent. This is more subtle than the polynomial case covered by Lemma 13.6. Linear independence of the sampled function vectors is, in general, more restrictive than merely requiring the functions themselves to be linearly independent.

If the parameters do not occur linearly in the functional formula, then we cannot use linear algebra to effect a least squares approximation. For example, one cannot determine the frequency ω, the amplitude r, *and* the phase shift δ of the general trigonometric approximation

$$y = c_1 \cos \omega t + c_2 \sin \omega t = r \cos(\omega t + \delta)$$

that minimizes the least squares error at the sample points. Approximating data by such a function constitutes a *nonlinear* minimization problem.

Weighted Least Squares

Another extension to the basic least squares method is to introduce weights in the measurement of the error. Suppose some of the data is known to be more reliable or more significant than others. For example, measurements at an earlier time may be more accurate, or more critical to the data fitting problem, than later measurements. In that situation, we should penalize any errors in the earlier measurements and downplay errors in the later data.

In general, this requires the introduction of a positive weight $c_i > 0$ associated to each data point (t_i, y_i); the larger the weight, the more vital the error. For a straight line approximation $y = \alpha + \beta t$, the *weighted least squares error* is defined as

$$\text{Error} = \sqrt{\sum_{i=1}^{m} c_i e_i^2} = \sqrt{\sum_{i=1}^{m} c_i \left[y_i - (\alpha + \beta t_i) \right]^2}\ .$$

Let us rewrite this formula in matrix form. Let $C = \operatorname{diag}(c_1, \ldots, c_m)$ denote the diagonal *weight matrix*. Note that $C > 0$ is positive definite, since all the weights are positive. The least squares error,

$$\text{Error} = \sqrt{\mathbf{e}^T C \mathbf{e}} = \| \mathbf{e} \|,$$

is then the norm of the error vector $\mathbf{e}$ with respect to the weighted inner product $\langle \mathbf{v} ; \mathbf{w} \rangle = \mathbf{v}^T C \mathbf{w}$. Since $\mathbf{e} = \mathbf{y} - A\mathbf{x}$,

$$\begin{aligned} \| \mathbf{e} \|^2 = \| A\mathbf{x} - \mathbf{y} \|^2 &= (A\mathbf{x} - \mathbf{y})^T C (A\mathbf{x} - \mathbf{y}) \\ &= \mathbf{x}^T A^T C A \mathbf{x} - 2\mathbf{x}^T A^T C \mathbf{y} + \mathbf{y}^T C \mathbf{y} = \mathbf{x}^T K \mathbf{x} - 2\mathbf{x}^T \mathbf{f} + c, \end{aligned} \tag{13.31}$$

where

$$K = A^T C A, \qquad \mathbf{f} = A^T C \mathbf{y}, \qquad c = \mathbf{y}^T C \mathbf{y} = \| \mathbf{y} \|^2.$$

Note that K is the weighted Gram matrix derived in (12.10), and so is positive definite provided A has linearly independent columns or, equivalently, has rank n.

Theorem 13.13. *Suppose A is an $m \times n$ matrix with linearly independent columns. Suppose $C > 0$ is any positive definite $m \times m$ matrix. Then, the quadratic function* (13.31) *giving the weighted least squares error has a unique minimizer, which is the solution to the* weighted normal equations

$$A^T C A \mathbf{x} = A^T C \mathbf{y}, \qquad \textit{so that} \qquad \mathbf{x} = (A^T C A)^{-1} A^T C \mathbf{y}. \tag{13.32}$$

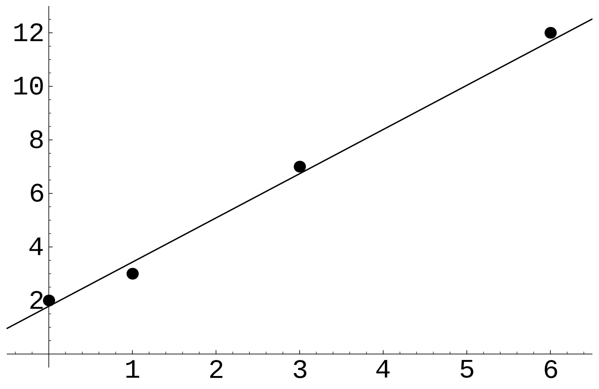


Figure 13.8. Weighted Least Squares Line.

In brief, the weighted least squares solution is obtained by multiplying both sides of the original system $A\mathbf{x} = \mathbf{y}$ by the matrix $A^T C$. The derivation of this result allows $C > 0$ to be *any* positive definite matrix. In applications, the off-diagonal entries of C can be used to weight cross-correlation terms in the data, although this extra freedom is rarely used in practice.

Example 13.14. In Example 13.4, we fit the following data

t_i	0	1	3	6
y_i	2	3	7	12
c_i	3	2	$\frac{1}{2}$	$\frac{1}{4}$

with an unweighted least squares line. Now we shall assign the weights listed in the last row of the table for the error at each sample point. Thus, errors in the first two data values carry more weight than the latter two. To find the weighted least squares line $y = \alpha + \beta t$ that best fits the data, we compute

$$A^T C A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 6 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \\ 1 & 6 \end{pmatrix} = \begin{pmatrix} \frac{23}{4} & 5 \\ 5 & \frac{31}{2} \end{pmatrix},$$

$$A^T C \mathbf{y} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 6 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 2 \\ 3 \\ 7 \\ 12 \end{pmatrix} = \begin{pmatrix} \frac{37}{2} \\ \frac{69}{2} \end{pmatrix}.$$

Thus, the weighted normal equations (13.32) reduce to

$$\tfrac{23}{4}\alpha + 5\beta = \tfrac{37}{2}, \qquad 5\alpha + \tfrac{31}{2}\beta = \tfrac{69}{2}, \qquad \text{so} \qquad \alpha = 1.7817, \qquad \beta = 1.6511.$$

Therefore, the least squares fit to the data under the given weights is

$$y = 1.7817 + 1.6511\,t,$$

as plotted in Figure 13.8.

13.3. Splines.

Polynomials are but one of the options for interpolating data points by smooth functions. In pre–CAD (computer aided design) draftsmanship, a *spline* was a long, thin, flexible strip of wood that was used to draw a smooth curve through prescribed points. The points were marked by small pegs, and the spline rested on the pegs. The mathematical theory of splines was first developed in the 1940's by the Romanian mathematician Isaac Schoenberg as an attractive alternative to polynomial interpolation and approximation. Splines have since become ubiquitous in numerical analysis, in geometric modeling, in design and manufacturing, in computer graphics and animation, and in many other applications.

We suppose that the spline coincides with the graph of a function $y = u(x)$. The pegs are fixed at the prescribed data points $(x_0, y_0), \ldots, (x_n, y_n)$, and this requires $u(x)$ to satisfy the interpolation conditions

$$u(x_j) = y_j, \qquad\qquad j = 0, \ldots, n. \tag{13.33}$$

The *mesh points* $x_0 < x_1 < x_2 < \cdots < x_n$ are distinct and labeled in increasing order. The spline is modeled as an elastic beam, [**42**], and so

$$u(x) = a_j + b_j (x - x_j) + c_j (x - x_j)^2 + d_j (x - x_j)^3, \qquad \begin{array}{c} x_j \le x \le x_{j+1}, \\ j = 0, \ldots, n-1, \end{array} \tag{13.34}$$

is a piecewise cubic function — meaning that, between successive mesh points, it is a cubic polynomial, but not necessarily the same cubic on each subinterval. The fact that we write the formula (13.34) in terms of $x - x_j$ is merely for computational convenience.

Our problem is to determine the coefficients

$$a_j, \qquad b_j, \qquad c_j, \qquad d_j, \qquad\qquad j = 0, \ldots, n-1.$$

Since there are n subintervals, there are a total of $4n$ coefficients, and so we require $4n$ equations to uniquely prescribe them. First, we need the spline to satisfy the interpolation conditions (13.33). Since it is defined by a different formula on each side of the mesh point, this results in a total of $2n$ conditions:

$$\begin{aligned} u(x_j^+) &= a_j = y_j, \\ u(x_{j+1}^-) &= a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 = y_{j+1}, \end{aligned} \qquad j = 0, \ldots, n-1, \tag{13.35}$$

where we abbreviate the length of the j^{th} subinterval by

$$h_j = x_{j+1} - x_j.$$

The next step is to require that the spline be as smooth as possible. The interpolation conditions (13.35) guarantee that $u(x)$ is continuous. The condition $u(x) \in \mathrm{C}^1$ be continuously differentiable requires that $u'(x)$ be continuous at the interior mesh points $x_1, \ldots, x_{n-1}$, which imposes the $n - 1$ additional conditions

$$b_j + 2 c_j h_j + 3 d_j h_j^2 = u'(x_{j+1}^-) = u'(x_{j+1}^+) = b_{j+1}, \qquad j = 0, \ldots, n-2. \tag{13.36}$$

To make $u \in \mathrm{C}^2$, we impose $n-1$ further conditions

$$2\,c_j + 6\,d_j\,h_j = u''(x_{j+1}^-) = u''(x_{j+1}^+) = 2\,c_{j+1}, \qquad j = 0, \ldots, n-2, \tag{13.37}$$

to ensure that u'' is continuous at the mesh points. We have now imposed a total of $4n-2$ conditions, namely (13.35–37), on the $4n$ coefficients. The two missing constraints will come from boundary conditions at the two endpoints, namely x_0 and x_n. There are three common types:

(i) *Natural boundary conditions*: $u''(x_0) = u''(x_n) = 0$, whereby

$$c_0 = 0, \qquad c_{n-1} + 3\,d_{n-1}\,h_{n-1} = 0. \tag{13.38}$$

Physically, this models a simply supported spline that rests freely on the first and last pegs.

(ii) *Clamped boundary conditions*: $u'(x_0) = \alpha,\ \ u'(x_n) = \beta$, where α, β, which could be 0, are fixed by the user. This requires

$$b_0 = \alpha, \qquad b_{n-1} + 2\,c_{n-1}\,h_{n-1} + 3\,d_{n-1}\,h_{n-1}^2 = \beta. \tag{13.39}$$

This corresponds to clamping the spline at prescribed angles at each end.

(iii) *Periodic boundary conditions*: $u'(x_0) = u'(x_n),\ \ u''(x_0) = u''(x_n)$, so that

$$b_0 = b_{n-1} + 2\,c_{n-1}\,h_{n-1} + 3\,d_{n-1}\,h_{n-1}^2, \qquad c_0 = c_{n-1} + 3\,d_{n-1}\,h_{n-1}. \tag{13.40}$$

If we also require that the end interpolation values agree,

$$u(x_0) = y_0 = y_n = u(x_n), \tag{13.41}$$

then the resulting spline will be a periodic C^2 function, so $u(x+p) = u(x)$ with $p = x_n - x_0$ for all x. The periodic case is used to draw smooth closed curves; see below.

Theorem 13.15. *Suppose we are given mesh points $a = x_0 < x_1 < \cdots < x_n = b$, and corresponding data values $y_0, y_1, \ldots, y_n$, along with one of the three kinds of boundary conditions* (13.38), (13.39), *or* (13.40). *Then there exists a unique piecewise cubic spline function $u(x) \in \mathrm{C}^2[a,b]$ that interpolates the data, $u(x_0) = y_0, \ldots, u(x_n) = y_n$, and satisfies the boundary conditions.*

Proof: We first discuss the natural case. The clamped case is left as an exercise for the reader, while the slightly harder periodic case will be treated at the end of the section. The first set of equations in (13.35) says that

$$a_j = y_j, \qquad j = 0, \ldots, n-1. \tag{13.42}$$

Next, (13.37–38) imply that

$$d_j = \frac{c_{j+1} - c_j}{3\,h_j}\,. \tag{13.43}$$

This equation also holds for $j = n-1$, provided that we make the convention that[†]

$$c_n = 0.$$

[†] This is merely for convenience; there is no c_n used in the formula for the spline.

We now substitute (13.42–43) into the second set of equations in (13.35), and then solve the resulting equation for

$$b_j = \frac{y_{j+1} - y_j}{h_j} - \frac{(2\,c_j + c_{j+1})\,h_j}{3}\,. \tag{13.44}$$

Substituting this result and (13.43) back into (13.36), and simplifying, we find

$$h_j\,c_j + 2\,(h_j + h_{j+1})\,c_{j+1} + h_{j+1}\,c_{j+2} = 3\left[\,\frac{y_{j+2} - y_{j+1}}{h_{j+1}} - \frac{y_{j+1} - y_j}{h_j}\,\right] = z_{j+1}, \tag{13.45}$$

where we introduce z_{j+1} as a shorthand for the quantity on the right hand side.

In the case of natural boundary conditions, we have

$$c_0 = 0, \qquad\qquad c_n = 0,$$

and so (13.45) constitutes a tridiagonal linear system

$$A\,\mathbf{c} = \mathbf{z}, \tag{13.46}$$

for the unknown coefficients $\mathbf{c} = \left(c_1, c_2, \ldots, c_{n-1} \right)^T$, with coefficient matrix

$$A = \begin{pmatrix} 2\,(h_0+h_1) & h_1 & & & & \\ h_1 & 2\,(h_1+h_2) & h_2 & & & \\ & h_2 & 2\,(h_2+h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & 2\,(h_{n-3}+h_{n-2}) & h_{n-2} \\ & & & & h_{n-2} & 2\,(h_{n-2}+h_{n-1}) \end{pmatrix} \tag{13.47}$$

and right hand side $\mathbf{z} = \left(z_1, z_2, \ldots, z_{n-1} \right)^T$. Once (13.47) has been solved, we will then use (13.42–44) to reconstruct the other spline coefficients a_j, b_j, d_j.

The key observation is that the coefficient matrix A is *strictly diagonally dominant*, cf. Definition 6.25, because all the $h_j > 0$, and so

$$2\,(h_{j-1} + h_j) > h_{j-1} + h_j.$$

Theorem 6.26 implies that A is nonsingular, and hence the tridiagonal linear system has a unique solution $\mathbf{c}$. This suffices to prove the theorem in the case of natural boundary conditions. *Q.E.D.*

To actually solve the linear system (13.46), we can apply our tridiagonal solution algorithm (4.47). Let us specialize to the most important case, when the mesh points are equally spaced in the interval $[a,b]$, so that

$$x_j = a + j\,h, \qquad \text{where} \qquad h = h_j = \frac{b-a}{n}\,, \qquad j = 0, \ldots, n-1.$$

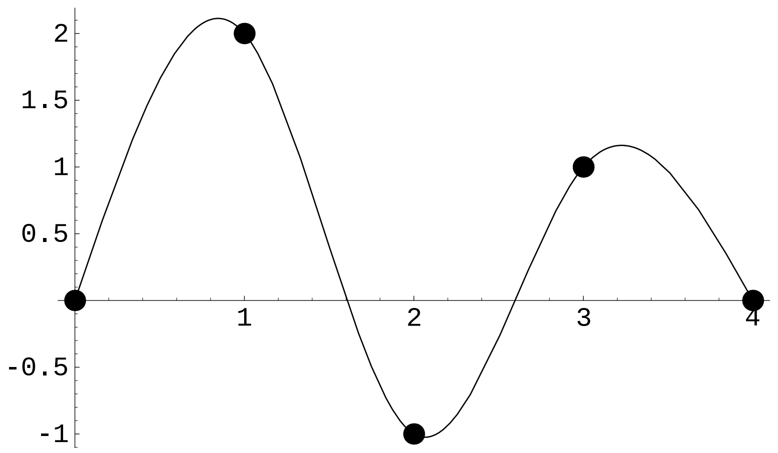


Figure 13.9. A Cubic Spline.

In this case, the coefficient matrix $A = hB$ is equal to h times the tridiagonal matrix

$$B = \begin{pmatrix} 4 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & 1 & 4 & 1 & \\ & & & 1 & 4 & 1 \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}$$

that first appeared in Example 4.26. Its LU factorization takes on an especially simple form, since most of the entries of L and U are essentially the same decimal numbers. This makes the implementation of the Forward and Back Substitution procedures almost trivial.

Figure 13.9 shows a particular example — a natural spline passing through the data points $(0,0)$, $(1,2)$, $(2,-1)$, $(3,1)$, $(4,0)$. As with the Green's function for the beam, the human eye is unable to discern the discontinuities in its third derivatives, and so the graph appears completely smooth, even though it is, in fact, only C^2.

In the periodic case, we set

$$a_{n+k} = a_n, \qquad b_{n+k} = b_n, \qquad c_{n+k} = c_n, \qquad d_{n+k} = d_n, \qquad z_{n+k} = z_n.$$

With this convention, the basic equations (13.42–45) are the same. In this case, the coefficient matrix for the linear system

$$A\,\mathbf{c} = \mathbf{z}, \qquad \text{with} \qquad \mathbf{c} = \left(c_0, c_1, \ldots, c_{n-1} \right)^T, \qquad \mathbf{z} = \left(z_0, z_1, \ldots, z_{n-1} \right)^T,$$

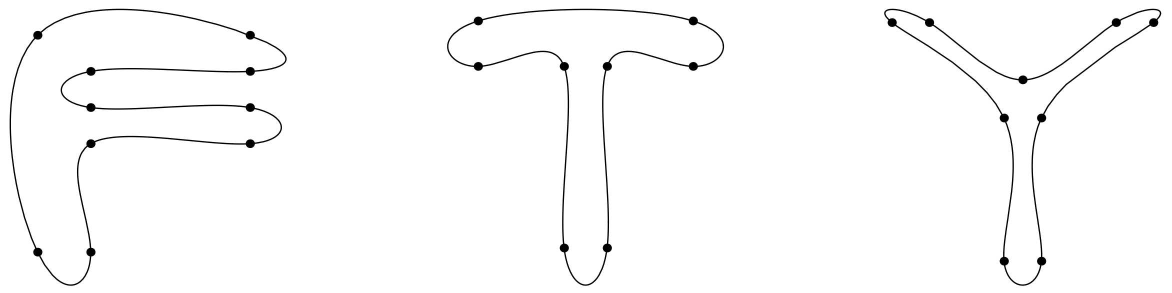


Figure 13.10. Three Sample Spline Letters.

is of *circulant tridiagonal* form:

$$A = \begin{pmatrix} 2\,(h_{n-1}+h_0) & h_0 & & & & h_{n-1} \\ h_0 & 2\,(h_0+h_1) & h_1 & & & \\ & h_1 & 2\,(h_1+h_2) & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & 2\,(h_{n-3}+h_{n-2}) & h_{n-2} \\ h_{n-1} & & & & h_{n-2} & 2\,(h_{n-2}+h_{n-1}) \end{pmatrix}. \tag{13.48}$$

Again A is strictly diagonally dominant, and so there is a unique solution $\mathbf{c}$, from which one reconstructs the spline, proving Theorem 13.15 in the periodic case.

One immediate application of splines is curve fitting in computer aided design and graphics. The basic problem is to draw a smooth parametrized curve $\mathbf{u}(t) = (\,u(t), v(t)\,)^T$ that passes through a set of prescribed data points $\mathbf{x}_k = (\,x_k, y_k\,)^T$ in the plane. We have the freedom to choose the parameter value $t = t_k$ when the curve passes through the k^{th} point; the simplest and most common choice is to set $t_k = k$. We then construct the functions $x = u(t)$ and $y = v(t)$ as cubic splines interpolating the x and y coordinates of the data points, so $u(t_k) = x_k$, $v(t_k) = y_k$. For smooth closed curves, we require that both splines be periodic; for curves with ends, either natural or clamped boundary conditions are used.

Most computer graphics packages include one or more implementations of parametrized spline curves. The same idea also underlies modern font design for laser printing and typography (including the fonts used in this book). The great advantage of spline fonts over their bitmapped counterparts is that they can be readily scaled. Some sample letter shapes parametrized by periodic splines passing through the indicated data points are plotted in Figure 13.10. Better fits can be easily obtained by increasing the number of data points. Various extensions of the basic spline algorithms to space curves and surfaces are an essential component of modern computer graphics, design, and animation, [**15**, **49**].