Curve Fitting with a Digital Computer

by C. W. Clenshaw

Summary: Forsythe (1957) has described a method for fitting polynomials to a set of points, using the principle of least squares. The method, designed to exploit the advantages of high-speed computers, uses orthogonal polynomials to overcome the problems of ill-conditioning which are usually associated with this approach. The present paper shows how this powerful method can be modified to save a substantial proportion of the machine storage. This is achieved by representing each polynomial within the machine by the coefficients in its Chebyshev series.

INTRODUCTION

Curve fitting is essentially the process of finding a smooth curve which passes near to each of a number of prescribed points in a plane. In a numerical, as opposed to graphical, approach to this problem, it is customary to use a *polynomial* to provide the curve; polynomials are easy to evaluate, their unknown coefficients occur linearly, and their degree affords a convenient measure of *smoothness*. The *nearness* is then usually achieved by imposing the "least-squares" criterion. By this means the original vague requirement is converted into a definite and tractable problem.

The numerical solution of this problem involves a considerable amount of arithmetic in all but the most trivial cases. The classical method fits a number of function values by a polynomial in the form

$$k_0 + k_1x + k_2x^2 + \ldots + k_nx^n,$$

using the least-squares criterion. It has the grave disadvantage of requiring the solution of a set of simultaneous equations for the coefficients k that may be ill-conditioned, often severely so when n is large. This defect can be removed by using orthogonal polynomials. A method, designed for desk machines, which makes some use of such polynomials has been described by Hayes and Vickers (1951). More recently, a method using essentially the same polynomials, but prepared for use with a digital computer, was discussed by Forsythe (1957) and again by Ascher and Forsythe (1958).

The main purpose of the present paper is to show how this powerful method can be modified to save a substantial proportion of the machine storage.

After a brief outline of Forsythe's proposals, we describe the modified method in detail. Next there is a discussion of the factors which help us decide upon the "best" degree for the approximating polynomial, followed by a description of a curve-fitting program, based on the modified method, which has been prepared for the DEUCE at the National Physical Laboratory. Finally, we show how the method is simplified in cases when the function value is available at *all* points of the range.

FORSYTHE'S METHOD

Let y_r ($r = 0, 1, 2, \ldots m$) be the observed or computed values of a dependent variable y at given values x_r of the independent variable x. Then the polynomial $Y_i(x)$ of degree i which minimizes the residual sum of squares

$$\delta_i^2 = \sum_{r=0}^{m} \{Y_i(x_r) - y_r\}^2 \qquad (1)$$

may be obtained by truncating the series

$$c_0p_0(x) + c_1p_1(x) + c_2p_2(x) + \ldots \qquad (2)$$

after $(i + 1)$ terms. Here $p_i(x)$ is a polynomial of degree i satisfying the orthogonality condition

$$\sum_r p_i(x_r)p_j(x_r) = 0 \quad (i \neq j). \qquad (3)$$

The coefficients c_j in (2) are therefore given by

$$c_j = \sum_r y_rp_j(x_r) / \sum_r \{p_j(x_r)\}^2. \qquad (4)$$

The $p_i(x)$ may be computed successively from the following three-term recurrence formula, as suggested previously by Householder (1953), Stiefel (1955), and Lanczos (1957):

$$p_{i+1}(x) = \lambda_i(x - \alpha_{i+1})p_i(x) - \beta_ip_{i-1}(x). \qquad (5)$$

Here λ_i determines the normalization of the polynomials, $\beta_0 = 0$ and

$$\alpha_{i+1} = \frac{\sum_r x_r\{p_i(x_r)\}^2}{\sum_r\{p_i(x_r)\}^2}, \quad \beta_i = \frac{\lambda_i \sum_r \{p_i(x_r)\}^2}{\lambda_{i-1} \sum_r \{p_{i-1}(x_r)\}^2}. \qquad (6)$$

Forsythe, following the previous writers, chooses $\lambda_i = 1$, so that the coefficient of x^i in the expression for $p_i(x)$ is independent of i, and in fact is unity.

The generation of the polynomials by equation (5), and the computation of the coefficients c_j in

$$Y_i(x) = c_0p_0(x) + c_1p_1(x) + \ldots + c_ip_i(x), \qquad (7)$$

give the approximating polynomial of degree i, and we can allow i to take all values up to m, at which stage the approximating polynomial passes through all the $(m + 1)$ points (x_r, y_r).

The polynomial of "best" fit may be conceived as that which most effectively compromises between smoothness, as represented by the degree of the polynomial, and closeness to the data, measured by δ_i^2 in equation (1). For practical purposes Forsythe regards the "best" value of i as that at which the *mean square of residuals*, defined as $\delta_i^2/(m - i)$, ceases to decrease significantly. This criterion is easy to apply: we have only to examine this quantity for each value of i as the computation proceeds; when its decrease is regarded as insignificant, the desired solution has been reached. Moreover, the quantity δ_i^2 can conveniently be produced as a by-product of the main calculations. For, from (1), (3), (4), and (7), we find

$$\begin{aligned}\delta_i^2 &= \sum_r \{Y_{i-1}(x_r) + c_i p_i(x_r) - y_r\}^2 \\ &= \delta_{i-1}^2 - 2c_i \sum_r p_i(x_r) y_r + c_i^2 \sum_r \{p_i(x_r)\}^2 \\ &= \delta_{i-1}^2 - c_i^2 \sum_r \{p_i(x_r)\}^2. \qquad (8)\end{aligned}$$

In considering the programming of this method, Forsythe suggests that the storage of the polynomials $p_i(x)$ should be effected by the retention of their numerical values at every x_r. Since the $p_i(x)$ are generated by a three-term recurrence relation, we need the values of two such polynomials at any given stage of the calculation. The data x_r and y_r are, of course, also needed throughout the calculation for all r points, so that a main store of $4(m + 1)$ positions is required. We next consider how this requirement might be reduced.

THE MODIFIED METHOD

Essentially the modification consists of a more compact storage procedure. An obvious way to achieve economy of storage would be to store the coefficients of the powers of x in the explicit expression for $p_i(x)$, rather than the values of $p_i(x)$ at every x_r. Similarly, each $Y_i(x)$ could be stored, and punched out, in the same form.

However, in arranging these polynomials as power series, we run the risk of introducing large coefficients, with a possible loss of accuracy when the polynomial is evaluated. This risk becomes serious as i increases.

Similar economy can be gained, however, without incurring this risk, by representing each polynomial within the machine by the coefficients in its *Chebyshev* expansion. When we require, for a given x, the numerical value of a function $f(x)$ which is represented by its Chebyshev series

$$f(x) = \tfrac{1}{2}a_0 + a_1T_1(x) + a_2T_2(x) + \ldots + a_nT_n(x), \quad (9)$$

where $T_s(x) = \cos(s \cos^{-1}x)$ is the Chebyshev polynomial in x of degree s, we may use the method of recurrence given by Clenshaw (1955). Briefly, this entails evaluating successively the quantities $b_n, b_{n-1}, \ldots b_0$, where

$$b_s = 2xb_{s+1} - b_{s+2} + a_s, \text{ with } b_{n+1} = b_{n+2} = 0. \quad (10)$$

The required value is then given by

$$f(x) = \tfrac{1}{2}(b_0 - b_2)$$

as can be verified by substituting for the a_s in (9) their expressions in terms of the b_s from (10), and using the recurrence relation

$$T_{s+1}(x) - 2xT_s(x) + T_{s-1}(x) = 0.$$

It is assumed here, without loss of generality, that the x_r all lie in the range $(-1, 1)$.

In order to be able to calculate $p_i(x)$ at any point, we accordingly store the coefficients $P_j^{(i)}$ in the expression

$$\begin{aligned}p_i(x) = \tfrac{1}{2}P_0^{(i)} + P_1^{(i)}T_1(x) + P_2^{(i)}T_2(x) + \ldots \\ + P_{i-1}^{(i)}T_{i-1}(x) + T_i(x), (i > 0). \quad (11)\end{aligned}$$

The polynomials $p_i(x)$ have been normalized by making the coefficient of $T_i(x)$ in (11), namely $P_i^{(i)}$, equal to unity. The definition of the polynomials is completed by putting $p_0(x) = \frac{1}{2}$, and we set $\lambda_i = 2$ in equations (5) and (6).

The number of storage positions required to represent $p_i(x)$ has thus been reduced from $(m + 1)$ to i, so that for $m \gg i$, the main storage requirement has been nearly halved.

Substituting (11) in the equation (5) with $\lambda_i = 2$ and comparing coefficients of $T_j(x)$, we obtain

$$P_j^{(i+1)} = P_{j+1}^{(i)} + P_{|j-1|}^{(i)} - 2\alpha_{i+1}P_j^{(i)} - \beta_i P_j^{(i-1)}, \quad (12)$$

where $P_i^{(i)} = 1$ for $i \geqslant 0$. Equation (12) is valid for all i and j.

From these equations we calculate the coefficients in the Chebyshev series for $p_{i+1}(x)$ from those for $p_i(x)$ and $p_{i-1}(x)$.

Similarly, the approximating polynomial $Y_i(x)$ can be represented within the machine by the coefficients $A_j^{(i)}$, say, in its Chebyshev series

$$\begin{aligned}Y_i(x) = \tfrac{1}{2}A_0^{(i)} + A_1^{(i)}T_1(x) \\ + A_2^{(i)}T_2(x) + \ldots + A_i^{(i)}T_i(x). \quad (13)\end{aligned}$$

As each coefficient c_i in the expression (7) is found from (4), the coefficients $A_j^{(i)}$ are obtained from their predecessors by means of the relation

$$A_j^{(i)} = A_j^{(i-1)} + c_iP_j^{(i)}, \quad (14)$$

which may be derived by comparing the coefficients of $T_j(x)$ in the equation

$$Y_i(x) = Y_{i-1}(x) + c_ip_i(x). \quad (15)$$

While the modification which we have introduced certainly saves much storage, a loss in speed may sometimes result, since the computation of $p_i(x)$ from its Chebyshev series for each x_r would take longer than the extraction of each $p_i(x)$ from its store. On the other hand, the modified method requires the application of the recurrence relation (12) only $(i + 2)$ times to define $p_i(x)$, whereas in the original method of Forsythe, equation (5) is used $(m + 1)$ times. The final decision as to which method is faster will depend upon the number of data points, the highest degree of polynomial required, and the characteristics of the computer; in particular the original method is faster whenever $4(m + 1)$ locations are available in the immediate-access store.

CHOICE OF "BEST" DEGREE

The main output of the program consists of the coefficients $A_j^{(i)}$, which can be punched out at the conclusion of each i-cycle. These coefficients provide another indication of the "best" value of i. If we were dealing with exact values of a well-behaved mathematical function, we would expect the $A_j^{(i)}$ to decrease with increasing j, for fixed large values of i. In practice, however, the readings y_r invariably contain rounding and observational errors, which affect the behaviour of the $A_j^{(i)}$. For values of j exceeding a certain value k, say, the coefficients will fluctuate about zero in an apparently random manner, and as i increases this behaviour persists with little change in k. The polynomial $Y_k(x)$ may then be accepted as the desired solution. Although it may sometimes be difficult to alight confidently on a definite k, the choice in such a case is not critical, since the difference between $Y_k(x)$ and each of its immediate neighbours $Y_{k-1}(x)$ and $Y_{k+1}(x)$ is then small for all x in the range, and any one of these three polynomials may be regarded as a satisfactory solution.

The values of all the coefficients $A_j^{(i)}$ will, of course, all be changed during every i-cycle. When the "best" degree k has been passed, however, the changes in the $A_j^{(i)}$ will merely be of the order of the unwanted "noise" present in the data, and the smallness of these changes is a most valuable check on the arithmetic.

Another criterion for the choice of k depends on the behaviour of the successive sets of residuals $Y_i(x_r) - y_r$. Examination of the complete set would usually be excessively laborious; in practice it is often sufficient to inspect the extreme values. In the DEUCE program compiled in the Mathematics Division of the National Physical Laboratory, the largest positive residual P_i and the numerically largest negative residual N_i are punched out, together with the corresponding values of x, ξ_i and η_i, say.

Like δ_i^2, the numerically larger of P_i and N_i will decrease appreciably as i increases until k is reached, after which it will usually decrease only slowly.

The quantities P_i and N_i serve another and more useful purpose, however. If one reading y_r has an outstandingly large error, there will be a tendency for P_i (or N_i) to occur at the corresponding x_r, for different values i. Therefore if ξ_i (or η_i) remains unchanged for several values of i, it may be desirable to examine the reading at this point, or to repeat the calculation with this reading omitted.

It may be observed that P_i and N_i would be determined more rapidly if $y_r - Y_{i-1}(x_r)$ were stored in place of y_r. This replacement would not affect the calculation of c_i, since $\sum\limits_r Y_{i-1}(x_r)p_i(x_r) = 0$.

DETAILED DESCRIPTION OF PROGRAM

To clarify the procedure, we now consider the arithmetic involved in one cycle. Let us suppose that at the beginning of the ith cycle the machine store holds the following quantities:

(A) x_r and y_r for $r = 0, 1, 2, \ldots m$.
(B) $P_j^{(i+1)}$ for $j = 0, 1, 2, \ldots i$;
$P_j^{(i)}$ for $j = 0, 1, 2, \ldots (i-1)$.
(C) $A_j^{(i)}$ for $j = 0, 1, 2, \ldots i$.
(D) P_{i-1}, N_{i-1}, ξ_{i-1}, η_{i-1} and δ_i^2.
(E) Other numbers, not required for output.

Of these stores, (A), (B) and (C) must have $2(m+1)$, $(2i+1)$ and $(i+1)$ positions respectively, while (D) and (E) each need only a few positions. The contents of (C) and (D) comprise the output at the end of the previous cycle.

The first step is the calculation of those quantities which depend directly on the initial data. Each pair (x_r, y_r) is extracted in turn from store (A), and $p_{i+1}(x_r)$ calculated by summation of its Chebyshev series (cf. (10)), using the coefficients stored in (B). Then $\{p_{i+1}(x_r)\}^2$ is formed and added to the partial sum $\Sigma\{p_{i+1}(x_r)\}^2$ stored in (E). Similarly the sums $\Sigma\, x_r\{p_{i+1}(x_r)\}^2$ and $\Sigma\, y_r p_{i+1}(x_r)$ are accumulated. In conjunction with this computation it is convenient to calculate the function $Y_i(x_r)$, whose Chebyshev coefficients are in store (C), and thence the residual $\{Y_i(x_r) - y_r\}$ for each x_r. If this exceeds numerically the largest so far obtained in this cycle we send it to the P_i (or N_i) position, and store the corresponding value of ξ_i (or η_i).

When the complete sums have been formed, we calculate α_{i+2} and β_{i+1} from (6) with $\lambda_{i+1} = 2$, c_{i+1} from (4) and hence δ_{i+1}^2 from (8).

Now we can evaluate from (14) the coefficients $A_j^{(i+1)}$ which overwrite the contents of store (C). Likewise equation (12) enables us to compute the $P_j^{(i+2)}$, which can replace the $P_j^{(i+1)}$ after the latter have been transferred to the $P_j^{(i)}$ position. The cycle is now complete; we can punch out the contents of stores (C) and (D), and start a new cycle.

The main program is entered at the beginning of the above cycle, with $i = -1$. The stores (C), (D) and (E) then contain zeros, while (B) holds only $P_0^{(0)} = 1$. For this first cycle only, we can arrange to omit that part of the program which evaluates the residuals $\{Y_i(x_r) - y_r\}$, and to insert the value $\beta_0 = 0$.

If m is less than 30, the program may be allowed to find the best least-squares polynomial of degree m. The corresponding P_m and N_m would be zero if there were no rounding errors, so their actual size affords a valuable check on the build-up of such errors.

In general, the results required from a specific problem will not be simply the Chebyshev coefficients $A_j^{(i)}$, and it will usually be necessary to operate on these subsequently with an auxiliary program. A few such programs will cover most of the common requirements, and might include, for example, one which calculates all the residuals δ_i for a given k, one to form a table of $Y_k(x)$ at equal intervals of x, and one which rearranges $Y_k(x)$ into a power series (where such rearrangement is possible without loss of accuracy).

SPEED AND STORAGE

In order to give an indication of the rate of operation of the program, which uses fixed-point arithmetic throughout, we present in Table 1 examples of the time taken by DEUCE, which has an addition time of 0·062 msec, and a multiplication and division time of 2 msec approximately. The table shows the number of minutes taken to read in the program and data, and calculate and punch all the least-squares polynomials $Y_0(x)$, $Y_1(x)$, . . . $Y_i(x)$, using $(m + 1)$ points, for various values of i and m.

TABLE 1

$i \diagdown m$	20	100	300
2	1	$1\frac{1}{2}$	$3\frac{1}{2}$
4	1	2	5
6	1	3	$7\frac{1}{2}$
8	$1\frac{1}{2}$	4	$10\frac{1}{2}$
10	2	5	14
15	$3\frac{1}{2}$	$9\frac{1}{2}$	25
20	5	15	39
25	—	22	58

The DEUCE has a high-speed store of 400 words and a magnetic-drum store of 8,192 words, each word consisting of 32 binary digits. The program described above allows for values of $(m + 1)$ up to 3,840, and for polynomials of degree not exceeding 29. Its practical application has so far been restricted to data with m not exceeding 100; the greatest value of i required has been 12. The numbers in Table 1 which refer to larger problems have been obtained from data specially prepared to test the present program.

FITTING TO A COMPLETE CURVE

The general procedure may be simplified when the y_r are given at the points $x_r = \cos \pi r/m$, as has been shown by Lanczos (N.B.S., 1952). We can make direct use of this solution in problems where we are given a formula or graph from which the value of y_r corresponding to any chosen x_r may be readily obtained.

We first suppose y to be a polynomial of degree less than or equal to m, given by

$$y = \tfrac{1}{2}a_0 + a_1T_1(x) + a_2T_2(x) + \dots + a_{m-1}T_{m-1}(x) + \tfrac{1}{2}a_mT_m(x) \qquad (16)$$

$$= \sum_{i=0}^{m}{}'' a_iT_i(x), \text{ say,}$$

where Σ'' denotes a summation in which the first and last terms are halved. Then, as Lanczos shows, the coefficients in (16) are given by

$$a_i = \frac{2}{m} \sum_{r=0}^{m}{}'' \; y_r \cos \frac{\pi i r}{m}. \qquad (17)$$

In general, of course, the function y is *not* a polynomial; but if it is continuous and of bounded variation, its expansion in Chebyshev polynomials converges, often very rapidly. By choosing a sufficiently large value of m, we can approximate the function to any desired degree of accuracy by a polynomial of the form (16), and find its coefficients from (17).

The solution of this problem is a special case of that of the general problem considered earlier, since if $x_r = \cos \dfrac{\pi r}{m}$, the polynomials $p_i(x)$ become the Chebyshev polynomials $T_i(x)$.

CONCLUSION

Considerable economy of storage space and a more concise primary output have been achieved by using Chebyshev expansions in programming Forsythe's method of curve fitting. Some time may also be saved, particularly on machines whose high-speed store cannot accommodate all the data. It is possible, however, that Forsythe's original method may be faster on machines having ample high-speed storage, when polynomials of high degree are required.

The work described above has been carried out as part of the research programme of the National Physical Laboratory, and this paper is published by permission of the Director of the Laboratory.

REFERENCES

ASCHER, M., and FORSYTHE, G. E. (1958). "SWAC Experiments on the Use of Orthogonal Polynomials for Data Fitting," *J. Ass. Comp. Mach.*, Vol. 5, p. 9.

CLENSHAW, C. W. (1955). "A Note on the Summation of Chebyshev Series," *Math. Tab., Wash.*, Vol. 9, p. 118.

FORSYTHE, G. E. (1957). "Generation and Use of Orthogonal Polynomials for Data Fitting with a Digital Computer," *J. Soc. Indust. Appl. Math.*, Vol. 5, p. 74.

HAYES, J. G., and VICKERS, T. (1951). "The Fitting of Polynomials to Unequally-Spaced Data," *Phil. Mag.*, Ser. 7, Vol. 42, p. 1387.

HOUSEHOLDER, A. S. (1953). *Principles of Numerical Analysis*, New York–Toronto–London: McGraw-Hill.

LANCZOS, C. (1957). *Applied Analysis*, London: Pitman.

National Bureau of Standards Appl. Math. Series No. 9 (1952). *Tables of Chebyshev Polynomials $S_n(x)$ and $C_n(x)$*, Washington: Government Printing Office.

STIEFEL, E. L. (1955). "Kernel Polynomials in Linear Algebra and their Numerical Applications," multilithed report, National Bureau of Standards, Washington. (Subsequently published in National Bureau of Standards Appl. Math. Series No. 49, 1958.) *Further Contributions to the Solution of Simultaneous Linear Equations and the Determination of Eigenvalues*, (Washington: Government Printing Office).