Parametric curve fitting

M. Grossman

Centre for Computing and Automation, Imperial College, London, SW7

An iterative method is described for least squares curve fitting, when both the abscissae and ordinates are subject to error. The ordinate and abscissa are fitted by parametric functions, using a constructive non-linear optimisation procedure. The order of the data points is maintained, and hence multi-valued data may be fitted.

(Received August 1969, Revised April 1970)

1. Introduction

The problem of least squares curve fitting when both ordinate and abscissa are subject to error, has been of interest for some time. A recent successful algorithm for solving it is described by O'Neill *et al.* (1969). Given a set of experimental data (x_i,y_i), $1 \leqslant i \leqslant n$, with x_i and y_i subject to error with weights w_{x_i} and w_{y_i}, it is required to calculate points (x'_i,y'_i) that lie on a continuous curve, so as to minimise the sum of squares of weighted perpendicular distances from the points (x_i, y_i) to the curve, i.e. to minimise

$$S = \sum_{i=1}^{n} [w_{x_i}(x_i - x'_i)^2 + w_{y_i}(y_i - y'_i)^2]$$

In O'Neill's work, y' is derived as a polynomial in x', and the discrete values x'_i corresponding to (x_i, y_i) are calculated iteratively. The idea can be generalised by introducing a third variable, u, and functions f_x and f_y, called the parametric functions, such that

$$x' = f_x(u)$$
$$y' = f_y(u)$$

For this work, polynomials, because they are very convenient, have been used as parametric functions, i.e.

$$f_x(u) = \sum_{r=0}^{k_x} a_{x_r} u^r \text{ and } f_y(u) = \sum_{r=0}^{k_y} a_{y_r} u^r$$

Also, because the data sets to be processed had a natural ordering, it was further stipulated that the order of the data points be maintained, which enabled the fitting of multi-valued data sets, and even loops.

The parameters to be calculated are the n-vecto $\bar{u}$ and the polynomial coefficients, i.e. $(n + k_x + k_y + 2)$ unknowns. In this paper it is shown that there are several valid solutions, in the sense that they produce a minimum of S. An iterative technique is described, without a proof, which generally converges to the best solution in least squares sense, without using derivatives. Although polynomials are used throughout the analysis, most of it is general, and can easily be applied to other parametric functions.

2. Theory

2.1. *The objective function*

The function to be minimised is given by

$$S(\bar{a}_x, \bar{a}_y, \bar{u}) = \sum_{i=1}^{n} \left\{ w_{x_i}\left[x_i - \sum_{r=0}^{k_x} a_{x_r} u_i^r\right]^2 + w_{y_i}\left[y_i - \sum_{r=0}^{k_y} a_{y_r} u_i^r\right]^2 \right\} \quad (1)$$

Let $k = \max(k_x, k_y)$.

Then S is a polynomial of order $(2k + 2)$ in $(n + k_x + k_y + 2)$ independent variables. As such, S has at most $k_x k_y$ real minima.

Intuitively, (1) appears to be underdetermined; if $\bar{u}$ is linearly transformed, compensatory changes in $\bar{a}_x$ and $\bar{a}_y$ would still lead to the same minimum solution. By following the method of O'Neill, and applying the Newton-Raphson technique, this property of S is clearly demonstrated; the resulting system of equations has $(n + k_x + k_y + 2)$ unknowns, and only $(n + k_x + k_y)$ equations. The two variables eliminated from the system, were u_1 and u_n, for reasons discussed below. There are algorithms for computing local minima of S, but they are rather inefficient for $n > 20$ (e.g. Powell, 1965).

2.2. *The constraints*

Ordinary least squares fitting takes no account of the order of the independent variable. The reason for fitting x and y in terms of u is that a possibly multi-valued, ordered data set is separated into two independent, single valued sets. This last property ensures that the order is not lost by the least squares fit, if the parametric functions are themselves single valued by nature, e.g. polynomials. To obtain correct separation for all data, some order must be induced in $\bar{u}$. The natural choice was made, that $u_i \leqslant u_{i+1}$, $1 \leqslant i \leqslant n - 1$, with u_1 and u_n fixed. Because of the properties of linear transformation, it is only the distribution of u in the range u_1 to u_n that affects the value of S. The values of u_1 and u_n are arbitrary; they were chosen as $u_1 = -2$ and $u_n = +2$ for reasons discussed in Section 3.1.

2.3. *Non linear programming*

The constraints change the nature of the problem from a standard minimisation one to a non linear programming problem, with a non-convex objective function and linear constraints. There are many techniques and programs available for the solution of such problems, but none of them take advantage of the relative simplicity of this particular problem.

2.4. *Algorithm*

It occurred to the author that the problem may be split into two simple parts. Given a set of u_i, $1 \leqslant i \leqslant n$, it is a relatively trivial matter to compute the functions f_x and f_y, using least squares; and vice versa—given the functions f_x and f_y, and a valid approximation to $\bar{u}$, then subject to certain conditions (to be discussed below), a new approximation to $\bar{U}$ that results in a reduced value of S, may be computed by local minimisation of S with respect to each u_i in turn. For this to be successful, the components of $\bar{u}$ should be reasonably independent. Loosely defined, this means that the u-values should be ones that arise due to the nature of the data, rather than being forced by the constraints, i.e. the ultimate u-values should be well away from violating the constraints. This condition was found to be sufficient, but not necessary, as explained in more detail in Section 3.2.

The algorithm therefore consists of two processes—each major iteration consists of $(n - 2)$ minor iterations for the new u-values (u_1 and u_n being held fixed), and a least-squares fit to calculate the new polynomials which must, by definition, reduce S further or leave it unaltered. The major iteration is repeated until convergence is reached.

The author is aware that this is a simple-minded approach to a complex problem, but in combination with a good first approximation to $\bar{u}$, it has given rather interesting results.

3. Computational techniques

3.1. *Least squares derivation of polynomials*

The method used was that of orthogonal polynomials (Forsythe, 1957). The parametric functions are given by,

$$f_x(u) = \sum_{r=0}^{k_x} C_{x_r}\Phi_r(u)$$

$$f_y(u) = \sum_{r=0}^{k_y} C_{y_r}\Phi_r(u)$$

If the functions Φ are orthogonal over the set of points, then it can be shown, without stating the nature of Φ, that

$$C_{x_r} = \frac{\sum_{i=1}^{n} x_i\Phi_r(u_i)}{\sum_{i=1}^{n} \Phi_r^2(u_i)} \text{ and } C_{y_r} = \frac{\sum_{i=1}^{n} y_i\Phi_r(u_i)}{\sum_{i=1}^{n} \Phi_r^2(u_i)}$$

If the orthogonal functions are polynomials, they can be generated by the three term recurrence formula

$$\Phi_{r+1}(u) = \lambda_r(u-\alpha_r)\,\Phi_r(u)-\beta_{r-1}\,\Phi_{r-1}(u)$$

where the values of α and β can be shown to be

$$\alpha_r = \frac{\sum_{i=1}^{n} u_i\Phi_r^2(u_i)}{\sum_{i=1}^{n} \Phi_r^2(u_i)}$$

and $$\beta_{r-1} = \frac{\sum_{i=1}^{n} \Phi_r^2(u_i)}{\sum_{i=1}^{n} \Phi_{r-1}^2(u_i)}$$

with $\beta_0 = 0$.

λ_r is an arbitrary multiplier, which governs the value of the leading coefficient in the orthogonal polynomials, and thus the optimum range of u. With $\lambda = 1$ throughout, it can be shown that the highest degree of computational precision is obtained with $-2 \leqslant u \leqslant 2$ with fixed point arithmetic, and has been found to be so, in practice, with floating point arithmetic.

Φ_0, too, is an arbitrary function, usually taking the value 1; however, the fact that Φ_0 is a factor of all the higher polynomials, is useful for the imposition of end conditions (Section 3.5).

The polynomials, f_x and f_y, are ultimately generated by summing up the coefficients of the powers of u, multiplied by their respective C_x and C_y values.

3.2. *Iteration for u-vector*

The contribution at the i^{th} point to the sum of squares of deviations is given by

$$s(u_i) = w_{x_i}[x_i-f_x(u_i)]^2+w_{y_i}[y_i-f_y(u_i)]^2$$

It is a $(2k)^{\text{th}}$ order polynomial in u_i. A new value u_i' is to be found, such that $s(u_i') \leqslant s(u_i)$, subject to $u_{i-1} \leqslant u_i' \leqslant u_{i+1}$. **Figs. 1 to 5** show typical s/u behaviour, in order of decreasing frequency of occurrence. The program is written so that the local minimisation starts from u_2. Hence, at the i^{th} point, in the j^{th} iteration, the value of u_{i-1} is from the current iteration, and u_{i+1} is the value from the $(j - 1)^{\text{th}}$ iteration. **Fig. 5** depicts the problem that was encountered by O'Neill, but is less likely to occur, since it depends on the turning points of two functions.

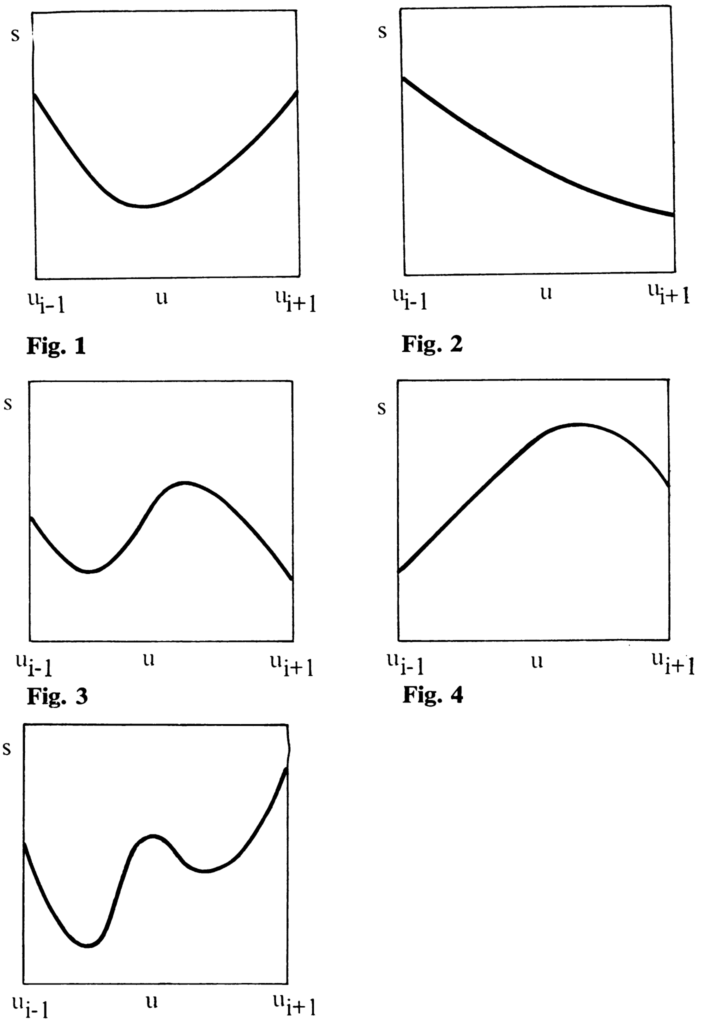

Fig. 1

Fig. 2

Fig. 3

Fig. 4

Fig. 5

Figs. 1 to 5. Typical behaviour of the i^{th} contribution to the sum of squares of deviations, with varying u_i

Behaviour as in Figs. 3 and 4 is more common, because it can be caused by inflexions in f_x and f_y, and by certain combinations of derivatives of f_x and f_y. Fig. 1 is the most desirable behaviour; it is the geometrical representation of independence between successive u values. Fig. 2 is a case that occurs frequently in the early iterations; as the u values are tackled in order, cases such as Fig. 2, that would generally converge to Fig. 1 cannot do so because of the constraints.

There are many ways of locating the minimum of s, of varying sophistication. The method finally adopted, a simple constant step-size search, was found to be more than adequate. It was a little inefficient in the first iteration, but this fact was more than offset by the simplicity of the program. The approximate position of the minimum is sought in steps of $(u_{i+1} - u_{i-1})/20$, starting from u_i (reversing direction if necessary). The position is refined by parabolic interpolation, and the process repeated with a step-size 1/100 of the original. The average number of calls to the function that evaluates the polynomial $s(u_i)$, was found to be 6 to 8 per point.

No satisfactory solution has been found for problematic s/u curves. Generally speaking, successive u values tended to move in the same direction, that is when $u_{i-1} \rightarrow u_i$, u_i tended to increase too. Cases such as Figs. 3 to 5 were rarely found, and then there would not be more than k such points.

One possible improvement to the program, is when a pair of u values persistently tends to violate their mutual constraint, the pair might be moved together. Such a case is a theoretical possibility, but has not been found in practice in any of the data experimented with.

3.3. *The initial approximation to u*

There are two 'natural' choices for the initial values of u_i, $2 \leqslant i \leqslant n-1$, as follows:

(*a*) Equally spaced, i.e. $u_i = u_{i-1} + h$, where

$$h = \frac{u_n - u_1}{n-1}$$

(*b*) Linear approximation to arc-length, i.e. joining the data points by straight line segments, and relating u to the lengths of these segments as

$$u_i = u_{i-1} + m\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

where m is a constant, such as to make $u_n = 2$. (*a*) is the simpler of the two, but (*b*) may be justified qualitatively, as follows: Given the functions $f_x(u)$ and $f_y(u)$, and their derivatives, $f_x'(u)$ and $f_y'(u)$, the arc length σ is related to u by

$$\frac{d\sigma}{du} = [f_x'^2(u) + f_y'^2(u)]^{\frac{1}{2}}$$

i.e.

$$\Delta\sigma \approx \Delta u[f_x'^2(u) + f_y'^2(u)]^{\frac{1}{2}}$$

and

$$\Delta u \approx \Delta\sigma[f_x'^2(u) + f_y'^2(u)]^{-\frac{1}{2}} \qquad (2)$$

(any text on elementary Differential Geometry, e.g. Eisenhart, 1960).

If the functions are a good approximation to the data points, the constant m can be thought of as an *a priori* approximation to the mean (over n points) of the bracketed expression in (2). In words, the concentration of u is directly related to the concentration of the data points.

In practice, it has been found that starting from either (*a*) or (*b*) led to the same solution, but, as hoped, (*b*) *always* gave a much better starting point.

In an attempt to locate all the minima of S, other initial approximations have been tried for several sets of data. These approximations were logarithmic, with values of u concentrated at either end of the range. The minima obtained in these cases were different from the ones found using (*a*) and (*b*); they were in every instance worse, both in starting point and final value of objective function.

Method (*b*) was finally adopted as the best first approximation. Results obtained led the author to believe that it would normally lead to the global minimum.

3.4. *Convergence*

The question of convergence is rarely a simple one. The obvious way is to compute the derivatives of the objective function with respect to the variables, and stop iterating when their values are sufficiently small (except when variables violate their mutual constraints). It was decided, however, that calculating derivatives would detract from the simplicity of the program, and would slow it down considerably. Because of the high speed of the program, it was permitted to iterate until the limit of precision of the machine is reached (about $8\frac{1}{2}$ d.p. in single precision). The initial sum of squares of deviations, S_0, is calculated; iteration stops when the reduction in S in two successive iterations is less than $S_0 \times 10^{-8}$, or when a predetermined number of iterations (read in as data) have been completed, whichever occurs earlier.

3.5. *Weighting and imposition of end-conditions*

Discrete weighting of data points is a standard feature of least-squares fitting, and is included simply in the derivation of the orthogonal polynomials. It was made optional in the program.

The other form of 'weighting' experimented with, is the imposition of an exact fit at the end-points. The method used consists of splitting each of the functions f_x and f_y into three parts, as follows (Clenshaw and Hayes, 1965):

$$f_x(u) = \mu(u)\, p_x(u) + \nu_x(u)$$
$$f_y(u) = \mu(u)\, p_y(u) + \nu_y(u)$$

where $\mu(u)$ is called the 'zeroising' function; it causes $\mu(u)$ $p(u) = 0$ at the end points, and $\nu(u)$ supplies the desired values of the functions at the end points.

$p_x(u)$ and $p_y(u)$ are generated, using the orthogonal polynomials method, after separating $\mu(u)$ and $\nu(u)$ from the data.

In practice, only $\nu(u)$ need be subtracted from the data, and $\mu(u)p(u)$ can be computed directly, by putting $\Phi_0 = \mu(u)$ (Section 3.1).

4. Results

4.1. *Data*

A typical set of data for which this program was developed, is shown in **Fig. 6**, with the solution superimposed. Many data sets of this kind were successfully tackled by the program, using cubics and quartics for parametric polynomials. Typical x/u and y/u curves generated are shown in **Figs. 7 and 8**.

As an experiment, the program was applied to data that had been generated by parametric quartics. The computed quartics were found to be identical to the generated ones—**Fig. 9**, within the precision of the computer.

Theoretically, the technique should be able to tackle a loop in the data. Data was generated using the equation

$$x^2 = y^2\left(\frac{0{\cdot}5 + y}{0{\cdot}5 - y}\right)$$

and was fitted with parametric quartics—**Fig. 10.**

These examples illustrate the great flexibility of the algorithm; given enough degrees of freedom in the polynomials, most shapes could be fitted. Unwanted oscillations between the points were found to be much less severe than with a y/x fit, probably because low order parametric polynomials permit a quality of fit only attainable by very high order polynomials of y/x.

4.2. *Effects of weighting*

As expected, neither discrete weighting nor the imposition of end-conditions (Fig. 10), affected the efficacy of the technique. For some data (e.g. Fig. 6) the free fit was so close, that imposed end-condition had no material effect on the result.

4.3. *Timing*

The computing time was found to be approximately proportional to the number of points, the first iteration usually taking longer than the others. The times on the IBM 7094 II, were typically 2·3 msec/point/iteration for cubics, and 2·7 msec/point/iteration for quartics.

The number of iterations required was found to be unpredictable. In some cases, both initial and final fit were bad, resulting in extremely slow convergence. In other cases, practically complete convergence was attained within 3 to 4 iterations.

4.4. *Validity of results*

No mathematical proof has been found that a program based on this algorithm should converge to a valid solution, global or otherwise. In practice, however, no data has been found for which it failed. As a check on the results and the convergence criterion, the problem was set up as a non-linear programming problem, and solved by a general problem solving package. The package is based on a variant of the Method of Feasible

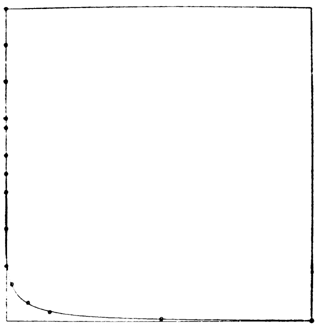

Fig. 6. Ship cross-sectional curve (quartic fit)

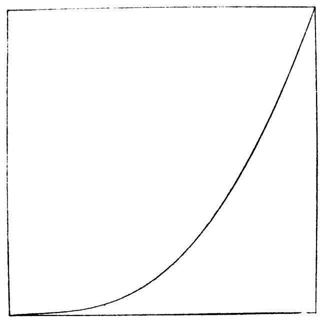

Fig. 7. x/u **curve for data of Fig. 6 (quartic)**

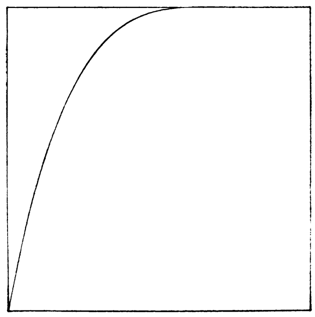

Fig. 8. y/u **curve for data of Fig. 6 (quartic)**

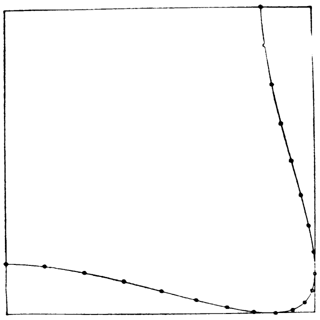

Fig. 9. Symmetric parametric quartics (quartic fit) converged to the generating functions

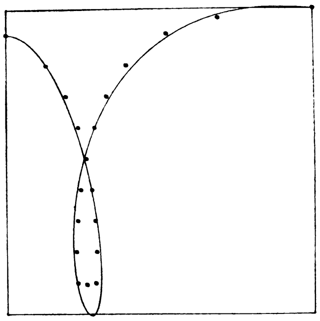

Fig. 10. Example of a loop, or multi-valued data, generated from $x^2 = y^2(0{\cdot}5 + y)/(0{\cdot}5 - y)$ **for** $-1{\cdot}2 \leqslant x \leqslant 3{\cdot}36$ **(quartic fit), with exact fit at end points.**

Directions (Benbow and Whitehead, 1968), and results from the author's program were used as the initial feasible solution vector. Several sets of data were tried, and in no case was there a further reduction in the objective function. This, combined with the relative slowness of the general problem solver, confirmed the validity of the author's approach.

4.5. *Artificially found minima*

As mentioned in Section 3.3, various first approximations to $\bar{u}$ were tried, in an attempt to locate all the constrained minima of S. In some cases, up to three additional minima were found (verified by the general problem solver, as in Section 4.4). As expected, these additional solutions were always worse than the one obtained using the arc-lengths as first approximations. It has not been possible to prove that it would be so in every case.

4.6. *A possible improvement*

It is possible to include the parametric functions in the local minimisation iterations. This means performing a least squares fit each time a value u_i is moved, so that the functions remain true least squares; this clearly eliminates the least squares fit at the end of the major iteration. At first sight it seems impractical, but on further scrutiny it turns out that a few intermediate sums would have to be stored in the fitting program; only these sums need be updated, and a few divisions result in the new polynomials. This is somewhat similar to O'Neill's method with constraints.

Intuitively, the same results should be obtained as with the basic method, but in fewer iterations. Some tests showed, however, that the increase in computation time per iteration roughly offset the reduction in the number of iterations, and the program required was more complex, so the idea was abandoned.

5. Conclusions

Two questions arise immediately from the results.

(*a*) Can convergence to a minimum be guaranteed? and

(*b*) If so, can it be proved to be the global minimum in every case?

No direct mathematical proof has been found for (*a*), but the results can be validated in every case by a technique that is known to converge to a minimum, and no data set has been found, which caused the program to fail. The answer to (*b*) is much more difficult. Such a proof exists for very few non-linear programming techniques. It can therefore only be stated that, based on experience, the program tends to converge to the global minimum. If the above statements are untrue, it should be possible to construct data to disprove them; the author would appreciate such data.

Where does this leave us? We have a stable and efficient algorithm, with a rather frail mathematical basis. It is also very flexible, and can be tailored to meet particular demands.

6. Acknowledgements

The author wishes to thank J. Benbow and P. Whitehead for their helpful comments, and for their invaluable assistance with the non linear programming part of the work; and also the Minister of Technology for permission to publish this paper. The work was done under a Ministry of Technology contract entitled 'The Application of Computers to Ship Design'.

References

Benbow, J., and Whitehead, P. (1968). Method of Feasible Directions in Non-Linear Programming, to be published as Ph.D. Thesis at Imperial College.

Clenshaw, C. W., and Hayes, J. G. (1965). Curve and Surface Fitting, *J. Inst. Maths. Applics.*, Vol. I, pp. 164-183.

Eisenhart, L. P. (1960). *A Treatise on the Differential Geometry of Curves and Surfaces*, Dover Publications.

Forsythe, G. E. (1957). Generation and Use of Orthogonal Polynomials for Data Fitting with a Digital Computer, *J. Soc. Indust. Appl. Math.*, Vol. 5, pp. 74-88.

O'Neill, M., Sinclair, I. G., and Smith, F. J. (1969). Polynomial Curve Fitting when Abscissas and Ordinates are Both Subject to Error *The Computer Journal*, Vol. 12, No. 1, pp. 52-56.

Powell, M. J. D. (1965). A Method for Minimising the Sum of Squares of Non-Linear Functions without Calculating Derivatives, *The Computer Journal*, Vol. 7, pp. 303-307.