Discussion Paper No. 639R

SOME NOTES ON NATURAL CUBIC SPLINE FUNCTIONS NONLINEAR REGRESSION AND THE ELASTICA

by

Dan Trietsch*

January 1985
Revised May 1985

---

*Visiting Assistant Professor, Department of Managerial Economics and Decision Sciences, J. L. Kellogg Graduate School of Management, Northwestern University, 2001 Sheridan Road, Evanston, Illinois 60201.

This work is based on a Ph.D. dissertation written under the supervision of Gabriel Y. Handler in the Faculty of Management, Tel Aviv University, March 1983. The author is very grateful to Gabriel Handler for his help over the years.

The author also acknowledges with pleasure substantial contributions by David Levin, The Computation Center, Tel Aviv University, and by Asher Tishler, Faculty of Management, Tel Aviv University.

"Some Notes on Natural Cubic Spline Functions,
Nonlinear Regression, and the Elastica"

Abstract

A simple shooting forward method is suggested for the computation of natural cubic spline functions through given knots. The method requires O(n) calculations and lends itself to programming easily. The use of such splines for nonlinear regression is discussed, and to that end a search method is applied to the values at the knots--with derivatives of the objective function by such changes in the knots introduced, to enable the use of Quasi-Newton search procedures with derivatives. On the plane, these ideas can be extended to optimal trajectory problems such as the alignment of a highway and solving for the elastica itself. Since the elastica is the "parent" of the natural cubic spline, this is in a sense a very natural and called-for extension, and closes a cycle, so to speak.

Introduction

The Euler-Bernoulli elastica (see [2] for an extensive discussion and references) is the curve defined by a flat spring which is made to pass through n+1 ordered and indexed points or "knots" x_i, y_i; i = 0,1,...,n, on a Euclidean plane in such a manner that it can slide and rotate freely at the knots (but not move them). Since the elastica, by its physical nature, minimizes the spring's potential energy, it must satisfy the following mathematical model (E):

$$(1) \qquad (E) \quad \min_{x(t),y(t)} \int_{t_0}^{t_n} \left[\left(\frac{d^2x}{dt^2}\right)^2 + \left(\frac{d^2y}{dt^2}\right)^2\right]dt$$

s.t.

$$(2) \qquad x(t),\ y(t) \in C^2[t_0,t_n],$$

$$(3) \qquad x(t_i) = x_i,\ y(t_i) = y_i;\ \forall i = 0,1,\ldots,n$$

$$(4) \qquad \left.\frac{d^2x}{dt^2}\right|_{t=t_0} = \left.\frac{d^2y}{dt^2}\right|_{t=t_0} = 0$$

or

$$(4a) \qquad \left.\frac{dx}{dt}\right|_{t=t_0} = \dot{x}_0,\ \left.\frac{dy}{dt}\right|_{t=t_0} = \dot{y}_0$$

$$(5) \qquad \left.\frac{d^2x}{dt^2}\right|_{t=t_n} = \left.\frac{d^2y}{dt^2}\right|_{t=t_n} = 0$$

or

$$(5a) \qquad \left.\frac{dx}{dt}\right|_{t=t_n} = \dot{x}_n,\ \left.\frac{dy}{dt}\right|_{t=t_n} = \dot{y}_n,$$

(6) $$\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 = 1;\ \forall t \in [t_o, t_n]$$

(7) $$0 = t_o < t_1 < \ldots < t_{n-1} < t_n < \infty.$$

(1) is directly proportional to the potential energy of the spring, which is why it is to be minimized; (2) answers continuity up to the second derivative (or curvature); (3) makes the curve pass through the knots; (4) and (5) reflect the fact that knot 0 and knot n can rotate, but if the spring's angle is fixed at any of them, then (4a) or (5a) are used instead to reflect the angle of this fixation; (6) forces the parameter t to be the cumulative distance along the curve from t_0 (which we arbitrarily set to 0). (Without (6), (1) does not minimize the potential energy of the spring.)

If we assume that: (a) $x_0 < x_1 < x_2 \ldots < x_n$, and (b) $y'(x)$ is small for any t, then it can be shown that the following model (NCSF) replaces (E):

(8) $$\text{(NSCF)} \quad \min_{y(x)} \int_{x_0}^{x_n} \left(\frac{d^2y}{dx^2}\right)^2 dx$$

(9) $$\text{s.t.} \quad y(x) \in C^2[x_o, x_n]$$

(10) $$y(x_i) = y_i;\ \forall i = 0,1,\ldots,n$$

(11) $$\left.\frac{d^2y}{dx^2}\right|_{x=x_0} = 0$$

or

(11a) $$\left.\frac{dy}{dx}\right|_{x=x_0} = \dot{y}_0$$

$$\left.\frac{d^2y}{dx^2}\right|_{x=x_n} = 0 \tag{12}$$

or

$$\left.\frac{dy}{dx}\right|_{x=x_n} = \dot{y}_n \tag{12a}$$

where (8), (9), (10), (11), (11a), (12) and (12a) replace (1), (2), (3), (4), (4a), (5) and (5a) respectively, while both (6) and (7) are satisfied under the assumption. We denote this model as NCSF standing for the "natural cubic spline function," though strictly speaking, if (11a) or (12a) are used instead of (11) or (12), we have a cubic spline function which is not necessarily natural (denoted as CSF). This complies with a well known theorem by Holladay (see [1]) who proved that the unique set of cubic polynomials defined between the knots in such a manner that the second derivative at the knots is continuous across the knots (thus satisfying (9)), and such that (10), (11) or (11a) and (12) or (12a) are satisfied, minimizes (8) as required. Thus, the NCSF is a special case of the elastica, but it certainly does not solve the elastica problem. The NCSF can be solved by an efficient method, namely by using B-splines involving the solution of a four band matrix linear equations set [1], however, in this paper, we suggest a new solution which is more straightforward in its approach, is more efficient, and lends itself to some applications better--specifically, it can accommodate some constraints more directly when used for regression. The method can also be extended to include derivatives for such regressions.

Since this research was conducted originally with the singular purpose of solving highway alignment problems [3], we will use these as examples here.

For the same reason, we focus our attention on CSFs, which are sufficient for that purpose. However, the basic ideas can be applied easily to other spline cases, such as higher (or lower) order splines. We do show, however, that our vertical alignment case generalizes the regular nonlinear regression so our method can serve for it too. We conclude with the elastica itself.

In the next sections we present our basic results in the format of questions (problems) and answers (solutions). Then we show the applications.

Basic Results

(P1) Solve (NCSF) for the equidistant arguments case, i.e., $x_i - x_{i-1} = (x_n - x_0)/n = \Delta$; $\forall i = 1,2,\ldots,n$.

(S1) We first transform the argument x, and for the ith segment, i.e., between x_{i-1} and x_i, we shall define a local argument $z_i \in [0,1]$, such that $z_i = (x - x_{i-1})/\Delta$. Clearly for $i = 1,\ldots,n-1$, if $z_i = 1$, then $z_{i+1} = 0$. For any other argument x, only one local argument exists.

Under these conditions, the NCSF is fully defined by an $n \times 4$ matrix, S, where the ith row gives the parameters of the ith cubic polynomial, defined for z_i, i.e., for the ith row, the polynomial is: $S_{i0} + S_{i1}z_i + S_{i2}z_i^2 + S_{i3}z_i^3$ where the columns of the matrix are 0, 1, 2 and 3 and the rows are $1,2,\ldots,n$. We have to find the 4n values S_{ij}, which conform to the following linear equations:

$$S_{i0} = Y_{i-1}; \quad \forall i = 1,2,\ldots,n \tag{13}$$

$$S_{i1} + S_{i2} + S_{i3} = S_{(i+1)0} - S_{i0}; \quad \forall i = 1,2,\ldots,n-1 \tag{14}$$

$$S_{i1} + 2S_{i2} + 3S_{i3} = S_{(i+1)1}; \quad \forall i = 1,2,\ldots,n-1 \tag{15}$$

(16) $S_{i2} + 3S_{i3} = S_{(i+1)2}$; $\forall i = 1,2,\ldots,n-1$

(17) $S_{n3} = Y_n - S_{n0} - S_{n1} - S_{n2}$

(18) $S_{12} = 0$

(19) $S_{n2} + 3S_{n3} = 0.$

Where (13) makes the NCSF pass through the knots, except for the last one y_n, which is taken care of by (17); (14), (15) and (16) ensure continuity up to the second derivative at the knots; (18) ensures (11); and (19) takes care of (12). ((18) or (19) could be changed easily to accommodate (11a) or (12a), a variation which we will discuss below.)

The Shooting Forward Solution: Clearly, by (13), (14) and (18), the first row of S, $S_{(1)}$ is:

(20) $S_{(1)} = (y_0, \lambda, 0, y_1 - y_0 - \lambda)$

where λ is a bounded, well defined, scalar--this is a direct result of a well-known existence and uniqueness theorem for the NCSF [1]. Now, had we known λ, then by recursive application of (14), (15) and (16) on the rows of S, starting with the known $S_{(1)}$, we would easily obtain S. Since we do not have λ, we may guess a value for it, and try to adjust it later so that (17) and (19) will be satisfied simultaneously (which happens if and only if λ is chosen exactly right). This is the general idea of the shooting forward method.

From here on, we use the term NCSF not just for the natural cubic spline function itself, but also for any (n × 4) matrix such as S, which satisfies (13) through (19). Similarly, such a matrix which satisfies (13) through (17) is a CSF. Since there is a one-to-one relationship between the matrices and the cubic splines, this double usage is defensible.

Clearly if A and B are CSFs for the same values of x_i, but not necessarily the same values y_i (say we have the values y_i^A and y_i^B for them, respectively), then $\alpha A + \beta B$ is a CSF for the knots $(x_i, \alpha y_i^A + \beta y_i^B)$. In other words, the CSFs span a linear space. Specifically, if we take $y_i^A = y_i$ and $y_i^B = 0$; $\forall$ i, then A + wB is a CSF for our original knots!

Now, let A be the CSF defined for our original knots with $\lambda = 0$, and let B fit the knots $(x_i, 0)$, but with $\lambda = 1$, i.e., the first row for B is 0, 1, 0, -1 and the first column contains only zeros. It is easy to verify that both A and B conform to (13) though (18) (and not just through (17)). Therefore, any combination of the form A + wB also conforms to these. It remains to choose w in such a manner that (19) is satisfied as well (and by the existence tneorem, this is clearly possible). For convenience define for CSFs such as A, B, C or D (we shall introduce C and D later), the values α, β, γ, and δ to be the second derivatives at the nth knot, respectively. E.g., $\alpha = a_{n2} + 3a_{n3}$. Now, if we choose

$$w = \lambda = -\alpha/\beta, \tag{21}$$

then

$$NCSF = A + \lambda \cdot B = A - (\alpha/\beta) \cdot B! \tag{22}$$

(By negation, and the existence theorem, $\beta \neq 0$, so (21) is well defined.)

Note now that B is independent of the data! That means that it can be stored in memory (say, 30 rows, where we use as many as we need, and add more if necessary). The first five rows of B are as follows:

$$B = \begin{pmatrix} 0, & 1, & 0, & -1 \\ 0, & -2, & -3, & 5 \\ 0, & 7, & 12, & -19 \\ 0, & -26, & -45, & 71 \\ 0, & 97, & 168, & -265 \end{pmatrix}$$

The readers may notice that the numbers grow exponentially. This means that for numerous knots some computational problems may occur. We will discuss a possible solution to this problem below. A more immediate conclusion is that double precision should be used if available, as was done in [2] and in [3], for instance. Note that it would not suffice to merely reduce b_{11}, since the ratio between the extreme values would not be affected by it.

The Computational Complexity of the Model: For better performance we prefer up to three additions over one multiplication, so we would perform S + S + S instead of 3S, etc. Using this convention we calculate the number of additions and of multiplications separately.

In order to compute A, we must use (14), (15) and (16) n - 1 times, each costing seven additions or subtractions. Hence A requires about 7n additions/subtractions. B requires about 5n additions/subtractions, plus one sign change. However, we can store the last three columns of B, as mentioned above (clearly we do not need to store the zero column of B). The computations required are illustrated in the following table:

Operation	Number of Additions or Subtractions	Cumulative Number
(a) let $C = S_{i1} + S_{i2}$	1	1
(b) $S_{i3} = S_{(i+1)0} - S_{i0} - C$	2	3
(c) $S_{(i+1)2} = S_{i2} + S_{i3} + S_{i3} + S_{i3}$	3	6
(d) $S_{(i+1)1} = S_{(i+1)2} + C$	1	7

Where for B, (b) can be executed by just inverting the sign of $C--S_{(i+1)0}$ and S_{i0} being zeros in this case.

Finally, in order to utilize (22) we need 3n multiplications and additions, so our total is 10n additions and 3n multiplications, if B is ready (or 15n additions if it is not). Compare this to the B-splines method which requires 13n additions and 15n multiplications (assuming that the three band matrix linear equations system used there is solved by shooting forward, which would require 5n additions/subtractions and 3n multiplications). The shooting forward method is clearly more efficient here both in the total number of operations required and in the mix between the types of the operations. □

This concludes our preliminary discussion for the equidistant arguments case, and we now allow different distances, as follows:

(P2) Same as (P1), but $x_i - x_{i-1} \neq x_j - x_{j-1}$ for some i,j.

(S2) Let $\Delta_i = x_i - x_{i-1}$; $i = 1,2,\ldots,n$ and let $\nu_i = \Delta_{i+1}/\Delta_i$; $i = 1,2,\ldots,n-1$. Now, (13), (14), (17), (18) and (19) remain valid for our new case; for (15), we have to multiply the right side by ν_i and for (16), likewise, by ν_i^2.

This time it is not expedient to prepare B in advance, and we need some more multiplications too, i.e., we will require about 16n additions/subtractions and 9n multiplications. However, the B-splines method is much worse off here, since we have to recompute the basis for the given set $\{x_i\}$, a task which would be at least three times as complex as the shooting forward method. []

If we return, for convenience, to the equidistant case, we can show now how the NCSF can be used as an interpolating method for the minimization of functionals, utilizing quasi-Newton search methods:

(P3) Let $J(f) = J = \int_a^b F[f(x),g(x)]dx$, be a functional where F(f,g) and g(x) are given functions, and we look for f(x) to minimize J. f, f' and f'' may have to conform to some constraints, which may be incorporated into the functional by proper penalty functions.

(S3) f can be represented by an "equidistant" NCSF with n + 1 knots, and if n is large enough, the approximation converges to any function in $C^2[a,b]$ superlinearly. (See the "best approximation" theorem in [1] and also a detailed discussion there of the merit of choosing an NCSF for approximation in such cases.)

Obviously the problem of searching for the optimal f has now been transformed to the problem of searching for the best n + 1 y_i values. For this purpose, we may want to utilize Newton or quasi-Newton search methods which make use of the n + 1 partial derivatives of J(NCSF), $\frac{\delta J}{\delta y_i}$. However, it may be easier to obtain the partial derivatives of J by the 4n elements of the NCSF matrix, which means that we have to do some transformations to obtain the result we need. Hence we assume that matrix E is given where the elements are the partial derivatives of J by the 4n corresponding values of the NCSF, i.e.,

$$E = \left\{ e_{ij} = \frac{\delta J}{\delta S_{ij}} \right\}, \quad \begin{matrix} i = 1,\ldots,n \\ j = 0,1,2,3. \end{matrix}$$

(E can be found numerically or analytically, depending on the case.)

In order to transform the matrix E with its 4n dependent elements to our required gradient with n + 1 independent partial derivatives, we make use of cardinal NCSFs, such as C^k which is an NCSF for $y_k = 1$, and $y_i = 0$; $\forall\ i \neq k$. If we have C^k, we can use the chain rule to obtain:

$$\frac{\delta J}{\delta y_k} = \sum_{i=1}^{n} \sum_{j=0}^{3} e_{ij} c_{ij}^k. \tag{23}$$

Hence, we only need C^k for k = 0,1,...,n to complete our transformation.

Solving for C^k is an instance of (P1) for each k, but we discuss it in some further detail (for a reason which will become obvious soon).

Let D^k be the analog of A, i.e., its first column is filled with zeros except for the k^{th} element which is one, and $d_{11}^k = 0$. There is no reason not to prepare D^k in advance, and as an example we list here the six D^ks for n = 5 (which we can use for any lower number as well):

$$D^0 = \begin{pmatrix} 1, & 0, & 0, & -1 \\ 0, & -3, & -3, & 6 \\ 0, & 9, & 15, & -24 \\ 0, & -33, & -57, & 90 \\ 0, & 123, & 213, & -336 \end{pmatrix}$$

$$D^1 = \begin{pmatrix} 0, & 0, & 0, & 1 \\ 1, & 3, & 3, & -7 \\ 0, & -12, & -18, & 30 \\ 0, & 42, & 72, & -114 \\ 0, & -156, & -270 & 426 \end{pmatrix}$$

$$D^2 = \begin{bmatrix} 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 1 \\ 1, & 3, & 3, & -7 \\ 0, & -12, & -18, & 30 \\ 0, & 42, & 72, & -114 \end{bmatrix}$$

$$D^3 = \begin{bmatrix} 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 1 \\ 1, & 3, & 3, & -7 \\ 0, & -12, & -18, & 30 \end{bmatrix}$$

$$D^4 = \begin{bmatrix} 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 1 \\ 1, & 3, & 3, & -7 \end{bmatrix}$$

$$D^5 = \begin{bmatrix} 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 1 \end{bmatrix}$$

But the sharp-eyed readers must have already noticed that given D^1, we actually have D^k for any $k > 1$; so we just need D^0 and D^1! Now by (22) we obtain

$$C^k = D^k - (\delta^k/\beta)\cdot B \tag{24}$$

which completes (S3). □

Incidentally, since in effect we now have the cardinal basis, we can use it for the direct computation of any equidistant NCSF. Or, put in another way:

$$\text{NCSF} = \sum_{k=0}^{n} y_k C^k, \tag{25}$$

and we only need to store B, D^0 and D^1 for an almost straightforward application of (25). Also, we do not really have to store the first column of D^0 or D^1, since we know where the "one" is. It follows that for n rows we merely need $9n_0$ RAM, which (today) is negligible indeed for $n_0 \leq 50$ or so. However, even if we had all the C^k ready, we would still need $0(n^2)$ additions and multiplications to use this method, so it is not efficient.

Example: We take C^2 for 5 knots, i.e., n = 4 as an example. Since C^2 is an NCSF, this may serve as an example for (S1) as well. Here $\beta = -45 + 3 \times 71 = 168$ and $\delta^2 = -18 + 3 \times 30 = 72$, hence $-\delta^2/\beta = -3/7$ and $C^2 = D^2 - (3/7) \cdot B$ or

$$C^2 = \begin{pmatrix} 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & ,1 \\ 1, & 3, & 3, & -7 \\ 0, & -12, & -18, & 30 \end{pmatrix} - (3/7) \cdot \begin{pmatrix} 0, & 1, & 0, & -1 \\ 0, & -2, & -3, & 5 \\ 0, & 7, & 12, & -19 \\ 0, & -26, & -45, & 71 \end{pmatrix} = \begin{pmatrix} 0, & -3/7, & 0, & 3/7 \\ 0, & 6/7, & 9/7, & -8/7 \\ 1, & 0, & -15/7, & 8/7 \\ 0, & -6/7, & 9/7, & -3/7 \end{pmatrix}$$

Before presenting some applications, we address the issue of computing CSFs with a large number of knots. The problem here is that the elements of A and of B tend to grow exponentially in absolute value. We will deal explicitly with the equidistant arguments NCSF case—the other being similar.

It turns out that in order to solve this problem we have to solve the general CSF problem first (for a small number of knots). Clearly there are

two degrees of freedom associated with any CSF passing through given knots, and in (S1) we used them to adjust the second derivative at the endpoints to zero, while the first derivatives are determined by the algorithm. In general, out of the four derivatives at the endpoints we may wish to set any two to predetermined values. We now proceed to describe a general method designed to achieve that.

(P4) For the equidistant arguments case, solve for the CSF such that:

(i) $f'(0) = f'_0$, $f''(0) = f''_0$; or

(ii) $f'(n) = f'_n$, $f''(n) = f''_n$); or

(iii) $f'(0) = f'_0$, $f'(n) = f'_n$; or

(iv) $f''(0) = f''_0$, $f''(n) = f''_n$; or

(v) $f'(0) = f'_0$, $f''(n) = f''_n$; or, finally,

(vi) $f''(0) = f''_0$, $f'(n) = f'_n$.

(S4) We proceed to describe a unified solution for all six cases. This solution is not necessarily the most efficient for each case separately, though. The general idea is like (S1)'s, but with two B-type CSF matrices, namely, B itself and G, which is designed to "adjust" $f''(0)$ as follows:

$$G = \begin{pmatrix} 0, & 0, & 1, & -1 \\ 0, & -1, & -2, & 3 \\ 0, & 4, & 7, & -11 \\ 0, & -15, & -26, & 41 \\ 0, & 56, & 97, & -153 \end{pmatrix}$$

Note that adding $\lambda \cdot G$ to a CSF adds λ to $f''(0)$, and leaves $f'(0)$ unaltered. Since A is designed with $f'(0) = f''(0) = 0$, it follows that a CSF with $f'(0) = f'_0$ and $f''(0) = f''_0$ is simply $A + f'_0 \cdot B + (f''_0/2) \cdot G$--which solves case (i) immediately. (A more efficient method here would be to start with

the first row of the CSF, namely $[y_0, f_0', f_0''/2, y_1 - y_0 - f_0' - f_0''/2]$, and proceed with recursive application of (14), (15) and (16).)

In general, for the n + 1 equidistant arguments case, if CSF = A + λ•B + μ•G then exactly two of the following equations must hold:

(26a) . $\lambda = f_0'$;

(26b) $\mu = f_0''/2$;

(26c) $a_{n1} + 2a_{n2} + 3a_{n3} + \lambda(b_{n1} + 2b_{n2} + 3b_{n3}) + \mu(g_{n1} + 2g_{n2} + 3g_{n3}) = f_n'$;

(26d) $2a_{n2} + 6a_{n3} + \lambda(2b_{n2} + 6b_{n3}) + \mu(2g_{n2} + 6g_{n3}) = f_n''$.

Since λ and μ are the only unknowns, we can obtain them by solving the appropriate two (independent) linear equations. E.g., in case (ii) we use (26c) and (26d); case (iii) implies the use of (26a) and (26c); case (iv) implies using (26b) and (26d); etc. (Note that if we set $f_0'' = f_n'' = 0$, as in (P1), then (26b) implies $\mu = 0$ and (26d) implies $\lambda = -\alpha/\beta$ as in (S1).) □

We are now prepared to tackle cases with numerous knots, as follows:

(P5) Solve (P1) for large values of n without numerical problems due to the exponential growth of the elements of A, B (or G).

(S5) It is always possible to partition the knots to, say, m parts each with a small enough number of knots, say $k_j \leq 30$, $\forall$ j = 1,2,...m, s.t. the last endpoint of the j-th part is the first endpoint of the (j + 1)-th part: $\forall$ j = 1,2,...,m - 1. Now, for each part, say j, we can construct an A-type CSF matrix $A^j = \{a^j_{rs}\}_{r=1,\ldots,k_j,s=0,\ldots,3}$. We also have B and G. Therefore,

for any pair of derivatives at the endpoints of part j we can solve the partial CSF $A^j + \lambda_j B + \mu_j G$. It only remains to choose values for the pairs λ_j, μ_j so that the resulting CSFs will comprise the complete NCSF we are looking for (or, similarly, any other type of CSF as in (P4)). To that end solve the following system of 2m (five-band-matrix) linear equations:

(27a) $\mu_1 = 0,$

(27b) $a^j_{k_j 1} + 2a^j_{k_j 2} + 3a^j_{k_j 3} + \lambda_j(b_{k_j 1} + 2b_{k_j 2} + 3b_{k_j 3})$

$+ \mu_j(g_{k_j 1} + 2g_{k_j 2} + 3g_{k_j 3}) = a^{j+1}_{11} + \lambda_{j+1} b_{11} + \mu_{j+1} g_{11};\ \forall\ j = 1,2,\ldots,m-1,$

(27c) $a^j_{k_j 2} + 3a^j_{k_j 3} + \lambda_j(b_{k_j 2} + 3b_{k_j 3}) + \mu_j(g_{k_j 2} + 3g_{k_j 3}) =$

$a^{j+1}_{12} + \lambda_{j+1} b_{12} + \mu_{j+1} g_{12};\ \forall\ j = 1,2,\ldots,m-1,$

(27d) $a^m_{k_m 2} + 3a^m_{k_m 3} + \lambda_m(b_{k_m 2} + 3b_{k_m 3}) + \mu_m(g_{k_m 2} + 3g_{k_m 3}) = 0.$

Clearly, if m is small, our problem is solved, though the complexity is slightly higher than in (S1)--chiefly due to the fact that we need two "correction matrices" (B and G) instead of one. If m is large, note that (27) can be solved by a similar shooting forward method, and therefore we can partition it again! In theory, for <u>huge</u> values of n we might need O(log n) partitions, so our algorithm would become O(n log n). []

<u>Application 1</u>: At this stage we are ready to discuss the highway vertical alignment problem as an application of (P3). This will also serve as a nonlinear regression example with any monotone "penalty" function to be

minimized.

In [3,5] it was shown that if we want to optimize the vertical alignment of a highway--that is, plan its elevation along the given horizontal alignment in such a manner that the earth moving costs together with the capitalized projected users costs will be minimized (other costs, such as pavement, land use, etc., are not sensitive to the vertical alignment)--then what we have to do is to minimize a functional defined along the horizontal trajectory as follows:

$$(28) \qquad J(f) = \int_{t_0}^{t_n} [EMC(f(t) - g(t)) + UC(f'(t)) + P(f''(t))]dt,$$

where EMC is a convex function of the elevation f minus the ground level g (EMC is also influenced by the ground side slope, so it varies along the trajectory, but it is still convex as a function of f - g); UC is another convex function reflecting the users' costs in time and fuel combined (see [4] for a detailed proof that fuel alone is not sufficient for convexity); and P is a convex penalty function of the second derivative of the alignment, designed to make sure that the vertical curvature of the highway will not be excessive. In [3,5] it is also demonstrated that the convexity of EMC, UC and P makes our functional (strictly) convex too, in the sense that only one local minimum exists, which is also the global minimum. Furthermore, it is easy to show that if we restrict f to be an NCSF (with equidistant knots or any other fixed arguments case), then we still have a strictly convex functional on our hands. Add to that the convergence properties of the NCSF, and it follows that by specifying enough knots, we can approximate the optimal value of f as closely as we may wish by an NCSF. (In the particular case of the highway alignment problem the NCSF is very attractive for another reason too, namely,

that in a sense it minimizes the squared curvature, and the curvature is a constrained variable for a highway. In other words, for a given set of knots, the NCSF tends to satisfy the curvature constraints [if they can be satisfied], and it does that along the whole length of the highway.)

It follows that what we have to do is use a search procedure to determine the optimal values of the knots, using the NCSF to interpolate between them, and calculating (28) along the NCSF. Now, this could be done by the existing NCSF procedures (such as B-splines, as discussed in [1]), as well, but our method has an advantage here, since it allows additional constraints (not reflected in (28)), on the maximal value of $|f - g|$, or similar max or min type constraints on the value of f at the knots, to be incorporated into the search almost without any extra effort, and such constraints are imposed sometimes for this problem. In contrast, dealing with this issue using B-splines, would be cumbersome, to say the least. The use of derivatives is also important but it could be adapted to the B-spline method as well.

When we say that the vertical alignment problem is an instance of nonlinear regression, what we mean is that we actually sample (although not necessarily randomly) values of g (the ground level along the highway), and we can calculate J in (28) numerically as the sum of the values we obtain, properly weighted. If we sample at equal distances along the highway (but much more closely than the n + 1 knots!), then we can use the sum of the values we obtain for the integrand in (28), multiplied by the distance between our data points, as a numerical approximation for J. In this paper we will assume this manner of numerical integration. In a nonlinear regression context, we would also have more data points than knots, and we would generally want to minimize the sum of the "penalties" associated with each data point as a monotone function of the vertical distance between the point

and f. Thus, the analogy between the cases is obvious. (However, in [3] a more sophisticated numerical integration was utilized where if each of our segments is subdivided to m parts, thus obtaining m + 1 integrand values, then an NCSF is passed through these values and integrated analytically. This is easily done since it is a set of polynomials which can be readily integrated. Finally, the sum of the n integrals thus obtained is used for J.)

To recapitulate, we have n + 1 equidistant arguments, (substantially) more than n + 1 data points, each of which "belongs" to one of the segments (if it happens to coincide with the argument of a knot between segments, it can be assigned arbitrarily to any one of them), and we wish to minimize the sum of nonlinear monotone penalties associated with the distances of the data points from an NCSF passing through n + 1 knots which are our decision variables (and thus, under our control, possibly with some constraints). We shall also assume convexity. (But if we do not have it, we will still obtain a minimum, though not necessarily the global one.)

Any search method which does not use derivatives can now be utilized to locate the optimal values of the knots within a prescribed tolerance. However, in the highway vertical alignment instance, and conceivably many others, the penalty functions associated with the data points are differentiable, and this will enable us to use (S3) and obtain the gradient of J by the n + 1 knots y_i. In order to show that, we just need the first "link" in our "chain rule" derivatives of (23), namely, the partial derivatives of J by the 4n elements of S, s_{ij}. These are easily obtained as the sum of the contributions of all the data points associated with each row (segment). For instance, if for segment i, ℓ data points are given with transformed arguments $z_i^1, z_i^2, \ldots z_i^\ell$, then for any of these, say z_i^j, we obtain the values of f_j, f_j' and f_j'' (where the index is as for z_i^j):

$$(29) \qquad f_j = \sum_{m=0}^{3} s_{im}(z_i^j)^m,$$

$$(30) \qquad f_j' = (1/\Delta_i) \sum_{m=1}^{3} m s_{im}(z_i^j)^{m-1},$$

$$(31) \qquad f_j'' = (1/\Delta_i^2) \sum_{m=2}^{3} m(m-1) s_{im}(z_i^j)^{m-2}.$$

Recalling that g is given (see (28)), we now have all the necessary arguments for EMC, UC and P (which we are using in this example). Since all these are differentiable, we have the three respective first derivatives of EMC, UC and P as well. Using the chain rule again we finally obtain:

$$(32) \qquad e_{i0} = \frac{\delta J}{\delta S_{i0}} = \sum_{j=1}^{\ell} \frac{dEMC}{df_j} \frac{\delta f_j}{\delta S_{i0}} = \sum_{j=1}^{\ell} \frac{dEMC}{df_j}$$

$$(33) \qquad e_{i1} = \frac{\delta J}{\delta S_{i1}} = \sum_{j=1}^{\ell} \left[\frac{dEMC}{df_i} \frac{\delta f_j'}{\delta f_j'} + \frac{dUC}{df_j'} \frac{\delta f_j'}{\delta S_{i1}}\right]$$

$$= \sum_{j=1}^{\ell} \left[\frac{dEMC}{df_j} z_i^j + \frac{dUC}{df_j'}(1/\Delta_i)\right],$$

$$(34) \qquad e_{i2} = \frac{\delta J}{\delta S_{i2}} = \sum_{j=1}^{\ell} \left[\frac{dEMC}{df_j} \frac{\delta f_j}{\delta S_{i2}} + \frac{dUC}{df_j'} \frac{\delta f_j'}{\delta S_{i2}} + \frac{dP}{df_j''} \frac{df_j''}{\delta S_{i2}}\right] =$$

$$= \sum_{j=1}^{\ell} \left[\frac{dEMC}{df_i}(z_i^j)^2 + 2\frac{dUC}{df_j'}(1/\Delta_i) z_i^j + 2\frac{dP}{df_j''}(1/\Delta_i^2)\right],$$

and

$$(35) \qquad e_{i3} = \frac{\delta J}{\delta S_{i3}} = \sum_{j=1}^{\ell} \left[\frac{dEMC}{df_j} \frac{\delta f_j}{\delta S_{i3}} + \frac{dUC}{df_j'} \frac{\delta f_j'}{\delta S_{i3}} + \frac{dP}{df_j''} \frac{\delta f_j''}{\delta S_{i3}}\right] =$$

$$= \sum_{j=1}^{\ell} \left[\frac{dEMC}{df_j}(z_i^j)^3 + 2\frac{dUC}{df_j'}(1/\Delta_i)(z_i^j)^2 + 6\frac{dP}{df_j''}(1/\Delta_i^2) z_i^j\right].$$

This, we recall, is "chained" again as per (23) to obtain $\delta J/\delta y_k$, as required.

This scheme was indeed used to find the optimal vertical alignment of a "simulated" highway, using a rather complex function for EMC (see [5]). For P, the following function was used, where $\rho \in (0,1)$ is a small user specified parameter, and μ is the constraint for f'':

$$(36) \qquad P_\rho(f'') = \begin{cases} 0 & ; \quad (f''/\mu)^2 \leq 1 - \rho \\ [(f''/\mu)^2 - 1 + \rho]^2/\rho^2 & ; \quad (f''/\mu)^2 \in (1 - \rho, 1) \\ [(f''/\mu)^2 + \rho/2]/\rho & ; \quad (f''/\mu)^2 \geq 1 \end{cases}$$

Both EMC and P_ρ (and likewise UC) are continuously differentiable. P_ρ is based on an idea developed by Zang in [6] for the representation of a step function by a continuously differentiable approximate function. The smaller we take ρ, the sharper the penalty. P_ρ has the following advantages over penalty functions such as $(f''/\mu)^{2N}$ with a large N (used by some practitioners): (a) no penalty when we are well within the bound; (b) virtually no risk of overflow for large deviations from the bound. Also, by tightening ρ between iterations the bound may be forced as closely as we may wish, without serious numerical problems. In practice we used $\rho = 0.01$ which was tight enough to begin with (since the real bound is rather arbitrary anyway), and no numerical problems were encountered. We may note here that UC may include a "built in" bound on f', but, as discussed in [3,5], this bound can be much higher than those practiced by highway engineers today, since UC takes care of the users' interest, instead of a tighter bound.

Before we conclude the discussion, note that for a regression situation the data may be scattered unevenly, and this may motivate us to use variable distances between the arguments, as in (P2). This can be done without any difficulty (even though it may be cumbersome). However, if we do that we have to decide where to place the arguments in addition to the decisions concerning the y_i values, thus it would be wasteful in terms of degrees of freedom. Rather than do that, we can double the number of the knots! Therefore this refinement will not be beneficial in most cases. If we take into account the complexity of (S2) relative to (S1), and the additional programming effort, we might be justified in omitting this extension. However, other situations may exist where we will have to use (S2). For instance, it is possible to obtain a fair approximation of the elastica itself by parametric NCSFs, but in this case we cannot maintain the distances equal, and have to resort to (S2).

Before discussing the elastica case, however, we describe an application to a curve fitting problem where practically the equidistant knots proved successful, even though at first glance it seems they would not do.

Application 2: Having solved the highway vertical alignment problem we now tackle the horizontal alignment as well. In this case we are looking for a curve which may not be approximable by any function, due to the possibility of backwards bends. In addition, both the convexity result we had for the vertical alignment, and the analytic derivatives we had there, cannot be generalized for the horizontal case (which is the reason we chose to decompose the alignment problem in the first place).

Since we cannot use any regular function (though it has been done in practice by some), we choose to use a parametric function. Specifically, we choose to use parametric NCSFs (see [1]) for x(t) and y(t), where for t we use an approximation of the cumulative distance along the chosen alignment.

(Below when discussing the elastica again, we elaborate on the reason t cannot be taken as the exact cumulative distance, though we would like that to happen.)

Usually, when we look for the exact (or nearly exact) horizontal alignment, we do so after having found an approximate piecewise linear alignment (see [4] for an extensive discussion of this part of the problem). This piecewise linear solution is actually a set of knots $p_i = (x_i, y_i)$; $i = 0,1,\ldots,n$ (which we may later add to, delete from, shift around, etc.). In order to pass a parametric NCSF through these knots, we initialize a vector $\underset{\sim}{t} \in R^{n+1}$ where $t_0 = 0$, $t_i = t_{i-1} + d(p_{i-1}, p_i)$; $i = 1,2,\ldots,n$ and $d(p_i, p_j)$ is the Euclidean distance between knot p_i and knot p_j. With this $\underset{\sim}{t}$, there is no problem in finding NCSFs for x(t) and for y(t) (here t is taken as a continuous parameter, having the values t_i at the knots). Now we may correct the value of t by taking a numerical integral of $[(\frac{dx}{dt})^2 + (\frac{dy}{dt})^2]^{1/2}$ along it (NCSFs may be useful for this numerical integration too). In practice t converges very fast--one or two iterations. We do not offer any theoretical results for that issue here, however.

Note that for this procedure we must use (S2), since the distances between the knots may not be equal. We may continue this way, and use a search method to relocate the knots until our objective function--calculated by solving (28) for the given knots--is minimized. (We do not discuss stopping criteria here.)

However, the original knots have no special importance to us. Hence we may redistribute them along the smooth trajectory obtained after the first iteration, in such a manner that they will be equidistant! Also, when searching for better locations for them, shifts perpendicular to the present trajectory are practically sufficient. However, every once in a while we may

have to readjust to maintain the equidistance property to a sufficient degree. We may also wish to redefine the perpendicular search directions on these occasions

This scheme was adopted, satisfactorily, in [3].

The last case we discuss is the case of the elastica itself. Since parametric splines can approximate any curve in C^2, and since they do stem from the elastica, the temptation to apply them is too strong to resist.

Application 3: Through n given knots, pass the elastica. I.e., find a curve which satisfies (1) through (7).

Pseudo-Theorem: If the parametric spline (as discussed in Application 2 for the given knots) converges, it is the elastica for the same knots.

Pseudo-Proof: Convergence implies that the cumulative length of the "spring" along the knots is the same for x(t) and for y(t) since the knots t_i are the same for them. (As an aside, this makes it worth our while to store the "B type" matrix of (S2), to use on both x(t)'s and y(t)'s NCSFs approximations.) Now, by Holladay's theorem (see in [1]), NCSF(x) minimizes

$$\int_{t_0}^{t_n} \left(\frac{d^2x}{dt^2}\right)^2 dt. \tag{37}$$

Similarly NCSF(y) minimizes

$$\int_{t_0}^{t_n} \left(\frac{d^2y}{dt^2}\right) dt. \tag{38}$$

So obviously their sum minimizes (1). Also, NCSF(x), NCSF(y) is certainly a curve in C^2, as required by (2), and it is easy to see that it satisfies (3), (4) and (5). (6) and (7) are taken care of by the stipulated convergence of t

to the cumulative distance at the knots. □

Note that the convergence requirement is not really restrictive, since the elastica is indeed not guaranteed to converge, unless the length is bounded!

It turns out that the pseudo-proof would be a valid one except for the very last sentence. Indeed, the parameter t can be made to match the length at the knots themselves, but along the trajectory (6) is not satisfied! This is especially significant if the curve changes its direction considerably between some consecutive pairs of knots, while else it is practically not an issue.

The pseudo-theorem does indicate that the parametric spline can serve as an approximation to the elastica—but this was to be expected anyway, since the elastica is in C^2. However, the really important point is that if dx/dy remains fairly constant between each pair of knots, then the parametric spline converges to the elastica. This is a simple generalization of the claim (mentioned above, and see [1]), that if y' is small enough, the NCSF converges to the elastica.

What we can do, then, is to add a sufficient number of "search-knots" between mal-behaved pairs of "fixed" knots, pass parametric splines through all knots, and search for the best location of the search knots so as to minimize (1).

Furthermore, since NCSFs are differentiable, extending (S3) to (P2) will enable us to use analytic derivatives for this search. Note that between pairs the search knots can be distributed equidistantly, if we so wish, which would make some of the v_i (see S2) equal to 1, with adjustments necessary only at the fixed knots.

Also note that using this method it is very easy to force the total

length to be any specified value, so long as it is long enough to connect the fixed knots with $\varepsilon > 0$ to spare (else (1) would be unbounded). This version may be of some interest to engineers and physicists.

In conclusion of this example, the cycle elastica-NCSF-elastica has been closed, for any set of knots. (In contrast Malcolm [2] specifies $x_0 < x_1 < \ldots < x_n$ or similarly for y.) However, it remains to test the efficiency of the algorithm as compared to other methods in practice.

References

[1] Ahlberg, J. H., E. N. Nielson and J. L. Walsh, The Theory of Splines and Their Applications, Academic Press, 1967.

[2] Malcolm, M. A., "On the Computation of Nonlinear Spline Functions," SIAM J. Numer. Anal., 14, 1977, 254-282.

[3] Trietsch, D., "On the Transportation Network Design Problem," Ph.D. dissertation, 1983 (in Hebrew).

[4] Trietsch, D., "A Family of Methods for Preliminary Design of Highways" Northwestern University, Center for Mathematical Studies in Economics and Management Science, Discussion Paper No. 645, March 1985.

[5] Trietsch, D. and G. Y. Handler, "On Highway Fuel and Time Expenditures," to appear in Transportation Science, August 1985.

[6] Zang, I., "Discontinuous Optimization by Smoothing," Mathematics of Operations Research, 6, Vol. 1, 1981, 140-152.