THE COMPUTATION OF SPLINES AND THE SOLUTION OF RELATED EQUATION SYSTEMS

by

G. E. McMaster

A Thesis
Submitted in Partial Fulfilment
of the Requirement of

DOCTOR OF PHILOSOPHY

at

The University of Manitoba
Winnipeg, Manitoba

Department of Computer Science

"THE COMPUTATION OF SPLINES AND THE SOLUTION OF RELATED EQUATION SYSTEMS"

by

G. E. McMaster

A dissertation submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

ACKNOWLEDGEMENTS

I am deeply indebted to my supervisor, Dr. W. D. Hoskins, without whom the production of this thesis would have been impossible. His vast knowledge in the area of mathematics and computation and his infectious enthusiasm proved invaluable. More important, perhaps, were his interest, guidance, encouragement, and availability when they were needed most.

I would like to thank the members of my committee: Dr. R. G. Stanton, Dr. D. W. Trim and Dr. D. D. Cowan.

I wish to thank Shirley Blackwood not only for her careful typing of the thesis but for her constant good nature and her willingness to always type that one extra page. I would also like to thank Miss Eileen Robbins for help in typing the final draft.

Anyone who has ever written a thesis knows what an enormous burden this task imposes on ones family. My wife, Beverly, suffered my constant attention to the thesis with the utmost of goodwill and patience. This thesis is Beverly's, Jeffrey's, Luke's and Jonathan's as much as it is mine.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Goals of the Thesis

The problem of spline interpolation and smoothing falls quite naturally into five main sections:

(a) the formation of the system of linear equations defining the coefficients of the basis functions used in the spline representation,

(b) the examination of properties of the coefficient matrix of this linear system,

(c) the solution of this linear system,

(d) the evaluation of the spline for various values of the argument,

(e) applications that use a spline representation to advantage.

In this thesis, results are obtained primarily in areas (a), (b), (c), and (e).

1.2 The Numerical Evaluation of Splines

A large number of equivalent mathematical descriptions of the polynomial spline are extant in the literature. Representative of these different forms are the works of Greville [1969]; Cox, [1971, 1972, 1973] with the use of B-splines as basis functions for spline interpolation; Ahlberg, Nilson and Walsh [1967] and their use of both divided differences and Hermite Interpolation to derive so-called consistency equations between derivatives; Fyfe [1971] with cardinal spline forms; Späth [1970] with Lidstone polynomials; and Golomb [1968] employing Bernoulli polynomials. Mathematically, all forms give exactly the same spline; computationally there is a wide variation with respect to the properties possessed and the condition number of the resulting set of simultaneous equations. Numerically, it has always been distressing that the forms with the fewest number of parameters, e.g., the representation used by Curtis and Powell [1967], should always be the worst to use from the point of view of rounding errors (c.f. Cox [1971] and Cox [1972]) and that well-conditioned forms should involve too many redundant parameters and large equation systems (Späth [1969]).

The B-spline or basis spline, has been proposed by a number of authors (Anselone and Laurent [1968]; Cox [1971]; Herriot and Reinsch [1971]; Lafata and Rosen [1970]; and Schumaker [1969]) as a convenient basis for problems of interpolation and smoothing. In forming the linear algebraic equations defining the multipliers of the basis functions and in evaluating the subsequent approximating spline, it is necessary to employ an algorithm for evaluating the B-spline. Cox [1972] and de Boor [1973] have obtained,independently, methods for B-spline evaluation that are numerically stable and economical.

1.3 *Outline of the Thesis*

In Chapter 2, background results on the B-spline are presented along with recent theorems which permit in Chapter 3 the formulation of systems of linear equations for both smoothing and interpolating splines. As well, an economical method for the least squares evaluation of a multivariate spline is given. In Chapter 3, algorithms for smoothing and for interpolation are given. The format used for the presentation of algorithms in this thesis closely approximates that of Wilkinson and Reinsch [1971]. Very briefly, the format followed is:

(a) The computer programs that supplement the derived mathematical algorithms are presented in Algol W (Hoare et al [1966]).

(b) The theoretical development giving the mathematical basis for the algorithm is given first. If a competitive published routine exists to solve part of the problem, then it is used, and only the reference is given.

(c) The formal parameter list giving all the input and output parameters for the main procedures is given.

(d) Organizational and notational details explaining unusual features of the algorithm such as storage techniques used or interesting testing procedures are given where necessary.

Error analysis, in general, is not included since, for the methods of solution used, detailed error analyses already exist and the solution methods can be proven stable. In the case of B-spline evaluation, Cox [1972] presents a rigorous error analysis; for the solution of the system of band equations, Wilkinson [1963, 1965] and Wilkinson and Reinsch [1971] give a complete error analysis. In the testing of the smoothing spline in Chapter 3, an interesting forward error analysis (Cody [1973]) is used, and is described in detail.

In Chapter 3, the solution of second-order linear differential equations using cubic splines is examined. In Chapter 4, new decoupling techniques for the rapid solution of systems of band equations resulting from spline representations are presented. The algorithms are competitive in a serial computer system, but are more effective in a particular parallel processing environment. Different concepts relating to parallel processing have been investigated (Flynn [1966, 1972]). Previous direct methods for the solution of tridiagonal linear systems using parallel processing (Stone [1973a], Kogge and Stone [1972]) were directed to SIMD computer systems (single-instruction-stream multiple-data-streams) Traub [1973].

The methods in Chapter 4 adapt well to an MIMD or multiple-instruction-steam multiple-data-stream parallel processing system for which algorithms seem to be difficult to obtain (Stone [1973b]).

In order to determine the effectiveness of the decoupling algorithms in an MIMD environment, an MIMD speed-up coefficient α is defined: let n be the total member of arithmetic operations in the algorithm and m the number that can proceed in parallel, then

$$\alpha = n/(n-m/2)$$

The speed-up factor is evaluated for the variations in the decoupling algorithms for the solution of tridiagonal systems in Chapter 4 and is found in most cases to be approximately 2 for large n . The general polydiagonal decoupling routines employ an extension of the technique used in the tridiagonal case, and similar economies can be expected.

In Chapter 5, properties of some classes of coefficient matrices that arise in the solution of systems of equations determining spline parameters are investigated. The analysis used is then extended to obtain properties of related matrices.

Chapter 2

B-Spline Function

2.1 *Introduction*

The spline function is a piecewise polynomial function that has excellent approximating properties, tends to be smoother and more flexible to use than a polynomial and usually provides better approximating properties (Greville [1969], de Boor [1963]). If the function being approximated is smooth, then spline functions are likely to give better estimates of the low-order derivatives than polynomials (Spӓth [1974]).

In this thesis, the determination and the evaluation of polynomial splines of odd degree $2r+1$ is examined. A spline function of degree $2r+1$ defined on n given knots $x_1 < x_2 < \ldots < x_n$ is a function $S(x) \varepsilon C^{2r}$ such that $S(x) \varepsilon S(x_1, x_2, \ldots, x_n)$, the class of polynomials of degree at most $2r+1$ in each of the intervals in the set $I = \{(-\infty, x_1), (x_1, x_2), \ldots, (x_n, \infty)\}$. In the practical application of spline functions, a finite range $a \leq x \leq b$ is almost always used,and hence $x_1 > a$ and $x_n < b$. In order to determine $S(x)$, first note that $S(x)$ has $n + 2r + 2$ parameters. The $n+1$ polynomials defined on I contain $(2r+2).(n+1)$ undetermined constants; however, $n(2r+1)$ of these constants are determined by the continuity requirements on $S(x)$, i.e. $S(x) \varepsilon C^{2r}$. The additional constants can be determined either by various interpolatory requirements or in a least squares sense.

In this chapter, we consider the spline as a conventional interpolating function; spline interpolation using additional information, such as values of the derivatives at the ends of the interval of interest, is considered in Chapter 3.

There are many ways (cf. Chapter 1) for representing a polynomial spline; however, if the spline is expressed as a linear combination of B-splines, then stable and efficient computational algorithms can be generated (Greville [1972], Cox [1972], de Boor [1973]). The B-spline was first introduced for the uniform partition by Schoenberg [1946] and for the non-uniform partition by Curry and Schoenberg [1966].

In Section 2.2, basic properties of the B-spline are given, along with requirements for the definition of the underlying knot set. An efficient algorithm (Cox [1972]), for B-spline evaluation is outlined which is used in an L_2 algorithm in Chapter 3. An integral result for the product of B-splines on a uniform mesh, useful for smoothing periodic data sets, is obtained in Section 2.3. In Section 2.4, an economical method for determining the coefficients of a multivariate B-spline representation for interpolation or for least squares curve fitting is obtained.

2.2 *The Basis Spline*

Most formulations of spline problems tend to give rise to ill-conditioned systems of linear equations (Greville [1969], Cox [1971]). Problems in solving the system are aggravated when the degree of the spline is increased and when there are many knots in the partition.

For example, it may be readily demonstrated that $S(x)$ is uniquely represented (Greville [1968]) by the two sets of parameters $P(\underline{x}) = (x_1, x_2, \ldots, x_n)$ and $C = (c_1, c_2, \ldots, c_{n+2r+2})$ where

$$S(x) = \sum_{i=1}^{n} c_i \, (x - x_i)_+^{2r+1} + \sum_{i=0}^{2r+1} c_{n+i+1} \, x^i$$

and

$$x_+ = \begin{cases} x & \text{when } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

This representation, although useful for purposes of mathematical analysis, leads to an ill-conditioned system of equations for the determination of the c_i, and is unwieldy to evaluate.

A very desirable representation for the spline is one whose support is finite and whose basis functions require a minimal number of knots in their definition.

The basis spline or B-spline of degree $2r+1$ (order $2r+2$) is non-zero over $2r+2$ consecutive intervals between knots (hence the nomenclature, spline of minimum support); $2r+2$ is the smallest number of intervals over which a spline of degree $2r+1$ can be non-zero. The B-spline is local in the sense that at any point only k B-splines, where k is equal to the order, are non-zero. These properties permit the representation of a spline in terms of B-splines in a stable numerically compact form (Cox [1972]). The forward B-spline (Schoenberg [1973]) $M_{2r+2,i}(x, P(\underline{x}))$ is the spline of degree $2r+1$ specified by the knot set $P(\underline{x}) = \{x_1, x_2, \ldots, x_n\}$. The knot set is specified in the form $P(\underline{x})$ to emphasize that the knots are chosen within the range of the given data by the curve fitter using a general knowledge of the shape of the underlying curve as indicated by the data and by trial and error. In general, more knots are required in those regions where the behaviour of the curve is changing rapidly and fewer knots where it is changing slowly; however, the exact positioning of the knots is often not critical (Cox and Hayes [1973]). With a little experience, satisfactory knot positions can be found after one or two trials.

The B-spline $M_{2r+2,i}(x, P(\underline{x}))$ may be formally defined as follows (de Boor [1973]). Let

$$x_+^{2r+1} = \begin{cases} x^{2r+1} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

and

$$M_{2r+2}(x;y) = (y-x)_+^{2r+1} \quad .$$

Then $M_{2r+2,i}(x, P(\underline{x}))$ is the divided difference of order $2r+2$ of $M_{2r+2}(x;y)$ with respect to the variable y based on the arguments $x_{i-(2r+2)}, x_{i-(2r+1)}, \ldots, x_i$. The evaluation of $M_{2r+2,i}(x, P(\underline{x}))$ through the use of divided differences leads to an unstable evaluation procedure and another technique (de Boor [1973], Cox [1972]) will be used. It is evident from the definition, however, that $M_{2r+2,i}(x, P(\underline{x}))$ is zero everywhere except in the $2r+2$ intervals in the range $R = \{x_{i-(2r+2)} < x < x_i\}$ and is uniquely determined (Cox [1972]), using the $2r+3$ knots defining R, except for a constant multiplier. The sign of the constant multiplier may be chosen to make the B-spline $M_{2r+2,i}(x, P(\underline{x}))$ positive on R. $M_{2r+2,i}(x, P(\underline{x}))$ may be shown to have a single maximum in R, and it and its derivatives up to the $2r$'th are zero at the end points of R, i.e., at $x = x_i$ and $x = x_{i-(2r+2)}$.

Since each B-spline spans $2r+2$ adjacent intervals (the order of the B-spline), then the knot set $P(\underline{x})$ determines $n-(2r+2)$ different

B-splines provided that $n > 2r+2$. The spline representation defined on the knot set $P(\underline{x})$ involves $n+2r+2$ degrees of freedom and requires $n+2r+2$ independent B-splines. It is then necessary to add $2r+2$ artificial knots to augment the given knot set at or outside each end of the given range of interest $[a,b]$ giving a total of $n+2r+2$ knots. For computational convenience, it is possible (Çavasso and Laurent [1969]) to place these extra knots at the appropriate end points, namely,

$$\begin{aligned} &x_{-2r+1} = x_{-2r+2} = \dots = x_0 = a \\ (2.2\text{-}1) \qquad & \\ &x_{n+1} = x_{n+2} = \dots = x_{n+2r+2} = b \end{aligned}$$

giving knots of multiplicity $2r+2$ at both a and b . The discontinuities that this arrangement introduces are at the end points of the range of interest,and so are of no concern. The $n+2r+2$ B-splines are then non-zero only in the range $a < x < b$.

If the spline is to represent a periodic data set $y = (y_v)$ of period r where

$$(2.2\text{-}2) \qquad y_m = y_k \quad \text{if} \quad m \equiv k \bmod (r) ,$$

then the knot set may be extended in an obvious manner using the spacing of the original x_i . This method for extending the knot set is assumed in Chapter 3 in order to obtain a periodic L_2 B-spline for smoothing purposes.

If the spline $S(x)$ of order $2r+2$ with the prescribed knot set $P(\underline{x}) = \{x_1, x_2, \dots, x_n\}$ is to interpolate to the function $f(x)$ at $x = t_1, t_2, \dots, t_p$, then it is assumed that the elements of the given set of nodes and the user-defined knot set $P(\underline{x})$ are strictly ordered, that is:

$$t_1 < t_2 < \ldots < t_p \tag{2.2-3}$$
$$t_1 < x_1 < x_2 < \ldots < x_n < t_p \;.$$

It is usual to assume that $a = t_1$ and $b = t_p$.

The $n+2r+2$ degrees of freedom remaining in $S(x)$ may be reduced by applying the interpolation conditions

$$S(t_i) = f(t_i); \quad i = 1, 2, \ldots, p \;. \tag{2.2-4}$$

To ensure that $S(x)$ be determined uniquely, we require that the number of given nodes, the number of selected knots, and the order of the spline be related by

$$p = n + 2r + 2 \;. \tag{2.2-5}$$

If the number of given nodes p is greater than $n+2r+2$, then the spline $S(x)$ may be determined using the method of least squares.

In order to ensure a unique $S(x)$, the specified knots $P(\underline{x})$ must be chosen to satisfy the Schoenberg-Whitney [1953] conditions

$$\begin{array}{l} t_1 < x_1 < t_{1+2r+2} \\ t_2 < x_2 < t_{2+2r+2} \\ \cdot \quad \cdot \quad \cdot \\ t_n < x_n < t_p \end{array} \tag{2.2-6}$$

which ensures that each B-spline in the representation $S(x)$ has one node in its range of definition.

The evaluation of a B-spline $M_{2r+2,i}\,(x, P(\underline{x}))$ may be effected using a stable recurrence relation given in detail in Cox [1972] or de Boor [1973], namely

$$M_{r,i}(x, P(\underline{x})) = \frac{(x-x_{i-r})\,M_{r-1,i-1}(x, P(\underline{x})) + (x_i-x).M_{r-1,i}(x, P(\underline{x}))}{(x_i - x_{i-r})}$$

(2.2-7)

commencing with

$$M_{1,i}(x, P(\underline{x})) = \begin{cases} 1/(x_i - x_{i-1}), & x_{i-1} \leq x < x_i \\ 0 & \text{otherwise.} \end{cases}$$

The expression on the right-hand side of (2.2-7) is the convex sum of two positive values which gives, in the main, the stability to the B-spline evaluation.

The recurrence relation (2.2-7) is valid for coincident knots, provided that there are no more than $2r+1$ coincidences (the degree of the spline) at any knot. This permits the spline $S(x)$ to have reduced continuity at one or more points in the range of interest $[a,b]$. This form for the spline is called a deficient spline.

For improved numerical stability in the B-spline evaluation, it is preferable (Hayes [1974b]) to use the normalized B-spline (de Boor [1972]) defined as

(2.2-8) $$N_{2r+2,i}(x, P(\underline{x})) = (x_i - x_{i-(2r+2)})\, M_{2r+2,i}(x, P(\underline{x})) \ .$$

The normalized B-spline may be computed from the given recurrence relation for the M's by omitting the final division by $x_i - x_{i-(2r+2)}$.

The spline $S(x)$ on $[a,b]$ may then be expressed uniquely using the $n+2r+2$ normalized B-splines defined on the augmented knot set as

(2.2-9) $$S(x) = \sum_{i=1}^{n+2r+2} c_i . N_{2r+2,i}\,(x,\, P(\underline{x}))\ ,$$

the c_i being constant. In order to determine the c_i, the interpolation condition $S(t_j) = f(t_j)$, where $j = 1, \ldots, p$, may be applied to give the linear system of equations

(2.2-10) $$\sum_{i=1}^{n+2r+2} c_i\, N_{2r+2,i}\,(t_j,\, P(\underline{x})) = f(t_j),$$

where $j = 1, 2, \ldots, p$.

The linear independence of the B-spline functions $N_{2r+2,i}\,(x, P(\underline{x}))$ and the restriction that the user-specified knot set $P(\underline{x})$ satisfy the Schoenberg-Whitney [1953] conditions, ensures a unique solution to the system of linear equations (2.2-10) (Cox [1974]).

If $p > n+2r+2$, then the coefficients c_i in the system of equations (2.2-10) may be obtained in a least squares manner by way of the normal equations. The system of equations to be solved may be represented as

(2.2-11) $$NN^*\underline{C} = \underline{F}$$

where $F = \{f(t_j)\}$, $C^T = \{c_1, c_2, \ldots, c_{n+2r+2}\}$ and the elements n_{ij} of N are given by

(2.2-12) $$n_{ij} = N_{2r+2,i}\,(t_j,\, P(\underline{x}))\ .$$

The system of equations (2.2-11) may then be solved by Gaussian elimination. To ensure a unique solution to (2.2-11), at least one of the p given knots must be in the range of definition of each of the $N_{2r+2,i}\,(x, P(\underline{x}))$ (Hayes and Halliday [1974]).

The least squares solution to (2.2-10) may be obtained more stably by the use of Householder reductions of the matrix NN^* (Bunsinger and Golub [1965]). This is at the cost of nearly doubling the amount of computation.

If the interpolatory spline of degree $2r+1$ defined on the knot set $P(\underline{x})$ is given by some polynomial of degree r or less in each of the intervals $(-\infty, x_1)$, $(x_n, +\infty)$ and the knot set is taken as the given nodes, then a natural spline (Greville [1969]) is obtained. In this circumstance, the coefficient matrix of the linear system to be solved is of strict band style with the non-zero elements appearing on the diagonal band of width $2r+1$. Algorithms for the solution of such linear systems are developed in Chapter 4 to obtain the parameters of this frequently used spline representation.

Finally, we mention Marsden's identity (Marsden [1970]) which permits a polynomial of degree n to be expressed in terms of B-splines. This identity is

$$(2.2\text{-}13) \qquad (u-x)^{k-1} = \sum_i \phi_{i,k}(u) \cdot N_{k,i}(x, P(\underline{x}))$$

where

$$\phi_{i,k}(u) = \prod_{r=1}^{k-1} (u-i-r) \quad .$$

This result is employed in Chapter 3 to inexpensively generate test data to validate a given B-spline representation, since a B-spline of degree m must exactly represent polynomials of degrees $0, 1, 2, \ldots, m$.

2.3 *Integral of the Product of Two B-Splines on a Uniform Mesh*

If the given knot set is assumed to be uniform, then there is no loss in generality in assuming that the B-spline is defined on the integer knots. One formulation for a B-spline of degree N or order $N+1$ is in terms of the backward difference of a truncated power function (Schoenberg and Curry [1966]). This definition gives a forward B-spline (Schoenberg [1973]) and, using the notation of Meek [1974], may be expressed as

(2.3-1) $$Q_{N+1}(x) = \frac{1}{N!} \nabla^{N+1} x_+^N$$

where

$$x_+ = \begin{cases} x & \text{for} \quad x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and where ∇ is the usual backward difference operator defined as

$$\nabla f_x = f_x - f_{x-h} \ .$$

The value h is the interval of differencing and, in this case, is assumed to be 1. Many of the useful computational properties of $Q_{N+1}(x)$ are summarized in Meek [1974]. A further result that enables an L_2 computational technique to be expressed in terms of the general consistency equations obtained by Fyfe [1971] concerns the integral of the product of two forward B-splines.

Theorem 2.3.1

(2.3-2) $$\int_{-\infty}^{\infty} Q_{N+1}(x-j)\, Q_k(x-\ell)\, dx = Q_{k+N+1}(N+1-\ell+j)$$

where j and ℓ are both integers and $N+1 > k$.

Proof:

The left hand side of equation (2.3-2) may be written as

$$L = \int_{-\infty}^{\infty} Q_{N+1}(t)\ Q_k\ (t-s)\ dt$$

with $s = \ell - j$. On using the definition of Equation (2.3-1),

$$L = \sum_{j=0}^{N} \int_{j}^{j+1} Q_{N+1}(t)\ Q_k\ (t-s)\ dt \quad . \tag{2.3-3}$$

Since $Q_{N+1}(t)$ is a polynomial of degree N in $[j, j+1]$,

namely,

$$Q_{N+1}(t) = \frac{1}{N!}\ \nabla^{N+1}\ t_+^N$$

$$= \frac{1}{N!} \sum_{p=0}^{j} (-1)^p \binom{N+1}{p} (t-p)^N \ ,$$

where $t \in [j, j+1]$

it follows that the N^{th} derivative of $Q_{N+1}(t)$ is

$$Q_{N+1}^{(N)}\ (t) = \sum_{p=0}^{j} (-1)^p \binom{N+1}{p} \ , \quad t \in [j, j+1] \quad . \tag{2.3-4}$$

Equation (2.3-3) can be integrated $N+1$ times by parts to give

$$L = (-1)^N \sum_{j=0}^{N} \left[\frac{1}{(N+k)!}\ \nabla_t^k\ (t-s)_+^{N+k}\ Q_{N+1}^{(N)}\ (t) \right]_{t=j}^{t=j+1}$$

where ∇_t is the backward difference operator acting on the variable t . From Equation (2.3-1), it may be seen that $Q_{N+1}^{(N)}\ (t)$ is a constant in the interval $[j, j+1]$; so it is convenient to denote it by $Q_{N+1}^{(N)}\ (j)$ where $Q_{N+1}^{(N)}\ (-1)$ is defined as zero. Then the above expression may be rewritten in the form

$$L = \frac{(-1)^N}{(N+k)!} \sum_{j=0}^{N} Q_{N+1}^{(N)}(j) \left[\nabla_t^k (t-s)_+^{N+k} \right]_{t=j}^{t=j+1}$$

and rearranged as

$$L = \frac{(-1)^N}{(N+k)!} \sum_{j=0}^{N+1} \left[Q_{N+1}^{(N)}(j-1) - Q_{N+1}^{(N)}(j) \right] \nabla_j^k (j-s)_+^{N+k} \quad .$$

However, Equation (2.3-4) gives,on substitution in the above expression,

$$L = \frac{(-1)^N}{(N+k)!} \sum_{j=0}^{N+1} (-1)^{j+1} \binom{N+1}{j} \nabla_j^k (j-s)_+^{N+k}$$

$$= \frac{1}{(N+k)!} \nabla_t^{N+k+1} (t-s)_+^{N+k} \Big|_{t=N+1} \quad .$$

From the definition (2.3-1), it follows that

$$L = Q_{N+k+1}(N+1-s) \quad . \qquad \text{Q.E.D.}$$

2.4 *A Method for Obtaining the Coefficients of a Multivariate Spline*

In this section, a general economical method for solving the system of equations defining a multivariate spline for interpolation or for surface representation is presented. A summary of recent advances in surface representation to which the method applies is given. The derived solution technique possesses definite computational savings over previous methods (Hayes and Halliday [1974], Hayes [1974a, 1974b], Späth [1974], Ahlberg et al [1967]) provided that the system of equations defining the spline parameters is not ill-conditioned or the coefficient matrix is not deficient.

We first examine methods for obtaining a bivariate spline representation where varying assumptions are made concerning the defining knot set. It is assumed that discrete data are given, that they may or may not contain random errors, and that these data are to be smoothed or fitted exactly. If the given data do contain errors, then these errors are assumed to be contained in the dependent variable. The surface representation methods considered here do not deal with the very different question of approximating mathematical functions where the value of the function for any values of the argument can be made available to any desired accuracy. Many publications dealing with cubic splines defined on two variables have appeared; however, these papers have largely concentrated on those interpolation problems in which the given data are known at the nodes of a rectangular mesh. This case is considered initially.

The bivariate spline is defined over a rectangular grid R specified by the partitions $P(\underline{x}) = \{x_1, x_2, \ldots, x_n\}$ and $Q(\underline{y}) = \{y_1, y_2, \ldots, y_m\}$. One of the rectangles R_{ij} in the grid may be defined as

$$(2.4\text{-}1) \qquad R_{ij} = \left\{ \begin{matrix} x_i \le x \le x_{i+1} \\ \\ y_j \le y \le y_{j+1} \end{matrix} \right\}$$

It is usual to treat a finite domain where $a < x < b$ and $c < y < d$.

To compute the coefficients of the bivariate spline, assume that the following data points are given,

$$(2.4\text{-}2) \qquad f(t_i, q_j) \qquad (i = 1, \ldots, p; \; j = 1, \ldots, v)$$

where the t_i are defined in the x direction, the q_j in the y direction.

We require a method for the computation of a surface $s(x,y)$, defined on R, that either interpolates the values $f(t_i, q_j)$ or represents these values in a least squares sense and is such that $s(x,y) \varepsilon C^{2r,2r}$. To represent the general bivariate spline, a set of basis functions is required as in the case when the B-splines are used to represent the one-dimensional case. Such a set for the bivariate spline may be constructed mathematically from the tensor product of two sets of independent B-splines (de Boor [1962]), one in the x-direction, the other in the y-direction. The set of all cross products formed using functions from each set provides the basis functions for the bivariate spline. Thus it is necessary to augment the partition in the y direction as was done in the x direction (2.2-1). The augmented partition for the y variate is

$$\begin{aligned} &y_{-2r+1} = y_{-2r+2} = \dots = y_0 = c \\ (2.4\text{-}3) \qquad &y_{m+1} = y_{m+2} = \dots = y_{m+2r+2} = d \quad . \end{aligned}$$

The given nodes and the user-defined knots must satisfy the Schoenberg-Whitney [1953] conditions in both the x and y directions, that is, each B-spline defined on either the x or the y variate must have a node within its non-zero range of definition. The bivariate spline may then be defined uniquely on R (Hayes [1974b]) as

$$(2.4\text{-}4) \qquad S(x,y) = \sum_{i=1}^{n+2r+2} \sum_{j=1}^{m+2r+2} a_{ij} N_{2r+2,i} (x,P(\underline{x})). N_{2r+2,j} (y,Q(\underline{y})) \ .$$

In the interpolatory case, $p = n+2r+2$ and $v = m+2r+2$ and the coefficients a_{ij} are determined by the $p.v$ equations

$$(2.4\text{-}5) \qquad (t_i, q_j) = f(t_i, q_j) \qquad (i = 1, \dots, p; \ j = 1, \dots, v) \ .$$

For values of p and/or v greater than the upper limits of summation in (2.4-4), the a_{ij} may be determined in a least squares manner.

As in the univariate case (Cox [1973]), the selected knot sets $P(\underline{x})$ or $Q(\underline{y})$ may contain coincident knots in order to permit discontinuities in the calculated surface. For example, if $x_1 = x_2 = \ldots = x_{2r+1}$, then $\frac{\partial^i S}{\partial x^i}(x,y)$, for $i = 1, \ldots, 2r$, would be discontinuous along the entire line $x = x_1$. In order to permit partial discontinuities along a grid line, Hayes [1974b] suggests the use of curved knot sets.

If a curved knot set is desired in the x direction, then a set of m+2r+2 single-valued functions of y denoted by $P(\underline{y})$ may be defined which intersects the planes $y = y_j$ $(j = 1, \ldots, m)$ defined by the knot set $Q(\underline{y})$ at the points of a curvilinear grid. In this case, (2.4-3) may be expressed as

$$(2.4\text{-}6) \qquad S(x,y) = \sum_{i=1}^{n+2r+2} \sum_{j=1}^{m+2r+2} a_{ij} \cdot N_{2r+2,i}(x,\underline{P}(y)) \cdot N_{2r+2,j}(y,Q(\underline{y})) .$$

If $S(x,y) \in C^{2r,2r}$, then it is necessary that each of the functions $P_k(y)$ in $\underline{P}(y)$ have continuity of order $2r$. This could readily be assured, for example, if splines of order $2r+1$ are fitted to the specified knot set.

If curved knot sets (defined by single-valued functions) are permitted in both the x and the y directions, then the bivariate function $S(x,y)$ may be expressed as (Hayes [1974b])

$$(2.4\text{-}7) \qquad S(x,y) = \sum_{i=1}^{n+2r+2} \sum_{j=1}^{m+2r+2} a_{ij} N_{2r+2,i}(x,\underline{P}(y)) \cdot N_{2r+2,j}(y,\underline{Q}(x)) .$$

The Schoenberg-Whitney conditions are satisfied if at least one of the

$f(t_i, q_j)$ appears in each of the curvilinear rectangles defined by $\underline{P}(y)$ and $\underline{Q}(x)$; we then have a unique solution to (2.4-7). If this is not the case, then the resulting system of equations is deficient. The single-valued functions in the sets $\underline{P}(y)$ and $\underline{Q}(x)$ may be any functions; however, an effective algorithm from a continuity standpoint is to use a B-spline representation, where the B-splines are C^{2r} . It is evident from Equation (2.4-7) that, once values for x and y are specified, then a set of interpolations may be performed using the functions in the sets $\underline{P}(y)$ and $\underline{Q}(x)$ to obtain the knot sets defining the B-splines on both the x and the y variates. Once the curvilinear rectangles are determined, then the knot set they define may be augmented as was done in the case of a rectangular grid.

The a_{ij} are then obtained by solving the linear system of equations

$$(2.4\text{-}8) \qquad f(t_\ell, q_w) = \sum_{i=1}^{n+2r+2} \sum_{j=1}^{m+2r+2} a_{ij} \, N_{2r+2,i}(t_\ell, \underline{P}(x)) . N_{2r+2,j}(q_w, \underline{Q}(y))$$

where

$$\ell = 1, 2, \ldots, p \quad ,$$

$$w = 1, 2, \ldots, v \quad .$$

Apart from the interpolation required to determine the knot sets in (2.4-8), this defining system of equations is similar to that obtained in (2.4-7) and (2.4-4), and we can consider the solution of the simpler case (2.4-4) without loss of generality.

It is usual to obtain the parameters a_{ij} in (2.4-4) in a manner (Hayes [1974a], Hayes and Halliday [1972]) other than by solving (2.4-4) as it stands. The given data values f_k

of a dependent variable f are given at the points (x_k, y_k), where $k = 1, 2, \ldots, pv$; we again assume pv data values. The system of equations analogous to (2.4-4) is then

$$\sum_{i=1}^{n+2r+2} \sum_{j=1}^{m+2r+2} a_{ij} . N_{2r+2,i}\,(x_k, P(\underline{x})) . N_{2r+2,j}\,(y_k, P(\underline{y})) = f_k , \tag{2.4-9}$$

where $k = 1, 2, \ldots, pv$.

If the more difficult case $pv > (n+2r+2) \times (m+2r+2)$ is considered, then a least squares solution to (2.4-9) may be attempted, i.e. we find the a_{ij} which minimize the sum

$$\sum_{i=1}^{pv} \{S(x_i, y_i) - f_i\}^2 \quad . \tag{2.4-10}$$

The system of equations corresponding to (2.4-9) is then written in matrix form (Hayes [1974a], Hayes and Halliday [1972]) as

$$NA = F \quad . \tag{2.4-11}$$

The matrix N has pv rows and $(n+2r+2) \times (m+2r+2)$ columns,and the k'th row consists of the values at the point (x_k, y_k) of all the basis functions $N_{2r+2,i}\,(x_k) . N_{2r+2,j}\,(y_k)$. The vector A consists of the unknown coefficients a_{ij} ordered according to the columns of N and vector F contains the pv data values f_r . The least squares solution to (2.4-10) may be obtained from

$$N^*NA = N^*F, \quad N^* \text{ being the transpose of } N. \tag{2.4-12}$$

It is then evident that the dimension of the resulting system of equations is large; in fact, it is $(n+2r+2)(m+2r+2)$ by $(n+2r+2)(m+2r+2)$ and is computationally expensive to solve.

It is possible, however, to obtain the matrix representation for (2.4-4) in a different manner. Consider the simpler interpolatory case first, where

$$u = n+2r+2 \quad \text{and} \quad v = m+2r+2 \quad .$$

If the matrix F contains the function values and the matrix A the coefficients, then (2.4-4) may be represented as

$$G_{u\times u} \cdot A_{u\times v} \cdot M_{v\times v} = F_{u\times v} \quad , \tag{2.4-13}$$

where the elements in G and M are defined as

$$g_{ij} = N_j\ (x_i, P(\underline{x}))$$

(2.4-14) and

$$m_{ij} = N_i\ (y_j, Q(\underline{y})) \quad .$$

The solution matrix A may be obtained as

$$A_{u\times v} = G^{-1}_{u\times u} \cdot F_{u\times v} \cdot M^{-1}_{v\times v} \quad .$$

If $u > n+2r+2 = k$ and/or $v > m+2r+2 = \ell$, then the entries a_{ij} in A may be determined in a least squares manner. Thus, it is necessary to find the least squares solution to the over-determined system of equations

$$G_{u\times k}\ A_{k\times \ell}\ M_{\ell\times v} = F_{u\times v} \tag{2.4-15}$$

where the $g_{ij} \varepsilon G$ and $m_{ij} \varepsilon M$ are defined as in (2.4-14). This may be effected by proceeding as follows. First define the functional J as

$$J(A) = \text{trace}\ \{(GAM-F)^* \ (GAM-F)\} \quad . \tag{2.4-16}$$

If the matrix A is perturbed by $E \neq 0$, then

$$J(A+E) - J(A) = tr\ \{[G(A+E)M-F]^* [G(A+E)M-F]\} - tr\ \{[GAM-F]^* [GAM-F]\}$$

$$= tr\ \{[G(A+E)M-F]^* [G(A+E)M-F] - [GAM-F]^* [GAM-F]\}$$

$$= tr\ \{[GAM-F+Q]^* [GAM-F+Q] - [GAM-F]^* [GAM-F]\}$$

where we have set $Q = GEM$.

Hence

$$J(A+E) - J(A) = tr\ \{Q^* (GAM-F) + (GAM-F)^* Q+Q^*Q\} .$$

Now, $tr\ (Q^*Q)$ and $G, M, E \neq 0$. Hence

$$J(A+E) - J(A) \geq tr\ \{M^*E^*G^* (GAM-F) + (GAM-F)^* GEM\}$$

$$= tr\ \{E^*G^* (GAM-F)M^* + [E^*G^* (GAM-F)M^*]^*\}$$

$$\geq tr\ (Z+Z^*) ,$$

where $Z = E^*G^* (GAM-F)M^*$.

Thus

$$J(A+E) - J(A) \geq 0$$

if

$$G^*GAMM^* = G^*FM^* . \tag{2.4-17}$$

The solution A to the system (2.4-17) is then the least squares solution to (2.4-9). It is obvious that this solution may be obtained in two stages by solving two sets of equations smaller than in the usual case where the system (2.4-11) is used to define the normal equations (Hayes [1974a], Hayes [1974b], Hayes [1973], Hayes and Halliday [1972]).

Let

$$T = GA ;$$

then solve

$$M^*T^* = F^* ,$$

and finally obtain the solution A by solving

$$GA = T .$$

The above algorithm for determining the coefficients of the bivariate spline may be readily extended to the general multivariate case.

Chapter 3

The Computation and Use of Spline Functions

3.1 Introduction

In this chapter, we consider the formation of systems of linear equations for both spline interpolation and smoothing using the B-Spline results of Chapter 2. Standard methods for solving equations (cf. Wilkinson [1965] , Wilkinson and Reinsch [1971], Hoskins and McMaster [1973]) are used to solve the resulting system of equations. Algol W procedures are presented for both interpolation and smoothing to compute accurately the values of the basis functions and their derivatives at various points. In addition, an application of the cubic spline to the solution of a class of differential equations is considered.

In Section 3.2, an explicit method of obtaining the formulae for the derivatives of the spline of order $2r+1$ at two boundaries $x = x_0$ and $x = x_n$ in terms of known function values and any of the computed derivatives of the spline is presented. These formulae are then used to derive an algorithm that permits interpolation using odd-order polynomial splines with equidistant knots and arbitrary linear boundary conditions.

In Section 3.3, a fast economical method is developed for smoothing periodic uniformly spaced data sets using polynomial splines. The approximation is obtained in an L_2 sense for a series of increasingly high order functions and is developed in such a way that applications of this technique to smoothing contours in many variables are readily made. An Algol W procedure for the smoothing of periodic data sets is given.

In Section 3.4, some relations between cubic spline solutions to the integral equation corresponding to a second order differential equation and a finite difference simulation are given. Using the multipoint boundary conditions for spline interpolation developed in Section 3.2, more general boundary conditions are permitted in the approximate solution of the integral equation.

3.2 Polynomial Spline Interpolation on a Uniform Set of Knots

3.2.1 Introduction

The present results arose from the study of relations between the cubic spline solution to a particular Fredholm integral equation of the second kind and the corresponding quintic spline solution to the related second order differential equation (Section 3.4). Both developments led quite naturally to a five-term difference approximation. Two additional boundary equations relating a given derivative at the boundary with the function values and other derivatives at internal knots of the spline were required in order to obtain a set of simultaneous equations with a coefficient matrix of band width five.

Some of the desired expansions appear to be in common use; cf. Hoskins [1970], where the determination of a quintic spline on uniform knots, subject to given first and second derivatives at the boundaries, entails the use of one of the special equations. Ahlberg, Nilson, and Walsh [1967] develop two further equations for the quintic spline.

For a spline $S(x)$ of order $2r+1$ defined on a uniform partition $\pi:\{x_0 < x_1 < \ldots < x_n \; ; \; x_k = x_0 + kh\}$ of $[x_0, x_n]$, these equations take the form

$$h^m S_0^{(m)} = a_1 S_0 + a_2 S_1 + \dots + a_{2r} S_{2r-1} +$$

(3.2-1)

$$+ h^n (a_{2r+1} S_0^{(n)} + a_{2r+2} S_1^{(n)} + \dots + a_{4r} S_{2r-1}^{(n)}) ,$$

where $m, n = 1, 2, \dots, 2r$ $(m \neq n)$, and the corresponding equation at the other boundary $x = x_n$ is obtained by symmetry. These equations, together with those of Theorem 2.3.1, enable us to construct a general algorithm for polynomial spline interpolation on a uniform set of knots; it is faster and requires less storage than the methods proposed by Späth [1970], Albasiny and Hoskins [1971], and Meek and Hoskins [1971], and is similar in requirements to the algorithm given in Herriot and Reinsch [1974] for equally-spaced knots.

3.2.2 Defining Equations for the Polynomial Spline

If the p^{th} derivative of $S(x)$ at the point x_i is denoted by $S_i^{(p)}$, then these equations are given in their most general form in Meek [1974]; for the odd order polynomial spline of degree $2r+1$, they are

(3.2-2)
$$\sum_{i=0}^{2r} C_{i,2r+1}^{(0)} S_{m+i}^{(p)} = \sum_{i=0}^{2r} C_{i,2r+1}^{(p)} S_{m+i}$$

$$p = 1, 2, \dots, 2r$$

$$m = 0, 1, 2, \dots, n-2r$$

with the quantities $C_{i,2r+1}^{(0)}$, $C_{i,2r+1}^{(p)}$ given by

(3.2-3) $$C^{(0)}_{i,2r+1} = \frac{h^{2r+1}}{(2r+1)!} \nabla^{2r+2} k_+^{2r+1}$$

and

(3.2-4) $$C^{(p)}_{i,2r+1} = \frac{h^{2r+1-p}}{(2r+1-p)!} \nabla^{2r+2} k_+^{2r+1-p}$$

where $k = 2r + 1 - i$ and $z_+ = z$ for $z > 0$

$= 0$ for $z \leq 0$.

Determination of the quantities $S^{(p)}_{m+i}$ is straightforward in the case where the required function is periodic, for then Equation (3.2-2) applies for $m = 1, 2, \ldots, n$, and leads to n equations in the n unknowns $S_1^{(p)}, S_2^{(p)}, \ldots, S_n^{(p)}$. The determination of the interpolating spline then requires determination of the set of derivatives $\{S_i^{(q)} ;\ q = 0, 1, 2, \ldots, 2r+1\}$ at a point x_i and the use of Taylor series interpolation in the interval $[x_i, x_{i+1}]$. Algorithms for determining such periodic polynomial splines appear in Andres, Hoskins, and King [1972] and are stable with respect to rounding errors; they also appear in a recent work by Ford [1975].

In the non-periodic case, the additional $2r$ boundary conditions can be in their most general form

(3.2-5) $$\sum_{p=1}^{2r} \alpha_{pi} S_0^{(p)} = \gamma_i \qquad i = 1, 2, \ldots, r \ .$$

If the conditions are evenly divided between the two boundaries, then

(3.2-6) $$\sum_{p=1}^{2r} \beta_{pi} S_n^{(p)} = \delta_i \qquad i = 1, 2, \ldots, r \ .$$

The $n+1$ interpolation conditions $\{(x_0, y_0), (x_1, y_1) \ldots, (x_n, y_n)\}$ used in conjunction with the Equations (3.2-5) and (3.2-6) demonstrate that the spline

$$S(x) = p_{2r}(x) + \sum_{t=0}^{n-1} d_t (x-x_t)_+^{2r+1} \quad , \tag{3.2-7}$$

($p_{2r}(x)$ a polynomial of degree $2r$) is uniquely determined if the $n+1$ equations (3.2-5), (3.2-6), and (3.2-2) have a solution. A solution is assured when the boundary conditions satisfy the Polya conditions (Polya [1931]).

3.2.3 Multipoint Boundary Equations

For the quintic spline, it is possible to derive the Equations (3.2-1) by manipulation and use of existing continuity equations in the manner used by Hoskins [1971]; these derivations are now given.

If the continuity of third and fourth derivatives is required, then the following two relationships between the first and second derivative values of the spline apply:

$$\delta^2 y_j - \frac{2h}{5}(S_{j+1}^{(1)} - S_{j-1}^{(1)}) + \frac{h^2}{20}\delta^2 S_j^{(2)} = \frac{h^2}{5} S_j^{(2)} \tag{3.2-8}$$

$$y_{j+1} - y_{j-1} - \frac{7h}{15}\delta^2 S_j^{(1)} + \frac{h^2}{15}(S_{j+1}^{(2)} - S_{j-1}^{(2)}) = 2h\, S_j^{(1)} \; . \tag{3.2-9}$$

Equation (3.2-8) was first given in the literature by Späth [1969]. Manipulation of these two relations produces the following two expansions

$$12hS_0^{(1)} = -37S_0 + 54S_1 - 9S_2 - 8S_3 \tag{3.2-10}$$
$$+ \frac{h^2}{120}(-138S_0^{(2)} + 2124S_1^{(2)} + 1206S_2^{(2)} + 48S_3^{(2)})$$

and

$$16h^2S_0^{(2)} = -235S_0 + 65S_1 + 155S_2 + 15S_3 \tag{3.2-11}$$
$$- 111hS_0^{(1)} - 227hS_1^{(1)} - 79hS_2^{(1)} - 3hS_3^{(1)} \quad .$$

Two additional expansions may be obtained in a similar manner if the quintic spline is expressed in terms of the second and fourth derivatives. Then the continuity of first and third derivatives at interior knots leads to the two relations

$$\delta^2 S_j^{(2)} = h^2 \left[1 + \frac{\delta^2}{6}\right] S_j^{(4)} \tag{3.2-12}$$

and

$$\delta^2 S_j = h^2 \left[1 + \frac{\delta^2}{12}\right] S_j^{(2)} - \frac{h^4}{180}\,\delta^2 S_j^{(4)} \tag{3.2-13}$$

where δ is defined as $\delta f_j = f_{j+1/2} - f_{j-1/2}$ (an integer knot set is assumed).

Combining Equations (3.2-12) and (3.2-13) immediately produces two additional expansions

$$2h^4S_0^{(4)} = -240S_0 + 420S_1 - 120S_2 - 60S_3 \tag{3.2-14}$$
$$+ 24h^2S_0^{(2)} + 195h^2S_1^{(2)} + 78h^2S_2^{(2)} + 3h^2S_3^{(2)}$$

and

$$h^2S_0^{(2)} = 2S_0 - 5S_1 + 4S_2 - S_3$$

(3.2-15)

$$+ \frac{h^4}{120}(18S_0^{(4)} + 65S_1^{(4)} + 26S_2^{(4)} + S_3^{(4)}) \quad .$$

Equations (3.2-8) and (3.2-9) may be combined to give

$$h^2S_1^{(2)} = \frac{h^2S_0^{(2)}}{3}$$

(3.2-16)

$$+ \frac{1}{6}(-hS_2^{(1)} + 16hS_1^{(1)} + 15hS_0^{(1)} + 5S_2 - 40S_1 + 35S_0) \quad ,$$

a similar expression holding for $h^2S_2^{(2)}$.

Substituting for the second derivatives in Equation (3.2-14) using the forms (3.2-15) and (3.2-16) gives the additional expansion,

$$4h^4S_0^{(4)} = -765S_0 - 345S_1 + 1005S_2 + 105S_3$$

(3.2-17)

$$- h(249S_0^{(1)} + 1173S_1^{(1)} + 537S_2^{(1)} + 21S_3^{(1)}) \quad .$$

Two additional equations were obtained by Hoskins [1971], where all derivatives up to order $2r-1$ are obtained in terms of $2r$'th derivatives. They are

(3.2-18) $$h^3S_0^{(3)} = \Delta^3S_0 - \frac{h^4}{120}(59S_0^{(4)} + 93S_1^{(4)} + 27S_2^{(4)} + S_3^{(4)})$$

$$12hS_0^{(1)} = -22S_0 + 36S_1 - 18S_2 + 4S_3$$

(3.2-19)

$$- \frac{h^4}{120}(4S_3^{(4)} + 102S_2^{(4)} + 216S_1^{(4)} + 38S_0^{(4)}) \quad ,$$

the expansion of $S_0^{(2)}$ in terms of fourth derivatives having already been obtained as Equation (3.2-15).

The derivatives at x_0 in terms of function values and third derivatives may be obtained by first applying the continuity equations obtained by Gryte and Hoskins [1971]. They are

$$h(S_0^{(1)} - 2S_1^{(1)} + S_2^{(1)}) = \frac{h^3}{12}(S_0^{(3)} + 10S_1^{(3)} + S_2^{(3)}) \tag{3.2-20}$$

and

$$h(S_0^{(1)} + 4S_1^{(1)} + S_2^{(1)}) = 3S_2 - 3S_0 + \frac{h^3}{30}(S_0^{(3)} - 2S_1^{(3)} + S_2^{(3)}) \; . \tag{3.2-21}$$

Shifting (3.2-20) and (3.2-21) to the right gives two additional equations. These four equations may be combined linearly to give the required relation

$$\begin{aligned} hS_0^{(1)} &= -S_0 + \frac{S_1}{2} + S_2 - \frac{S_3}{2} \\ &\quad + h^3\left[\frac{S_0^{(3)}}{15} + \frac{13}{24}S_1^{(3)} + \frac{13}{60}S_2^{(3)} + \frac{S_3^{(3)}}{120}\right] \; . \end{aligned} \tag{3.2-22}$$

Using Taylor Series, S_1, S_1' and S_1''' may be obtained as expansions in terms of the function values and odd derivatives at the boundary point x_0. These expressions may be combined to produce

$$\frac{h^4 S_0^{(iv)}}{60} = S_0 - S_1 + \frac{h}{2}(S_1' + S_0') - \frac{7h^3}{120}S_0''' - \frac{h^3}{40}S_1''' \quad . \tag{3.2-23}$$

Now (3.2-20) and (3.2-21) may be linearly combined to give

$$hS_0^{(1)} = \frac{1}{2}(S_2 - S_0) - \frac{h^3}{120}(S_0^{(3)} + 18S_1^{(3)} + S_2^{(3)}) \; . \tag{3.2-24}$$

Equations (3.2-22) and (3.2-24), when substituted into (3.2-23), give the additional relation

$$h^4 S_0^{(iv)} = 15S_0 - 45S_1 + 45S_2 - 15S_3$$

(3.2-25)

$$+ \frac{h^3}{4} (- 7S_0^{(3)} + 41S_1^{(3)} + 25S_2^{(3)} + S_3^{(3)}) .$$

Again, repeated application of Taylor Series expansions produces

(3.2-26) $$h^2 S_0'' = 5(S_1 - S_0) - \frac{3h}{2} S_1' - \frac{7h}{2} S_0' + \frac{h^3}{24} S_1^{(3)} - \frac{h^3}{8} S_0^{(3)}$$

which, when combined with Equations (3.2-22) and (3.2-24), gives

$$h^2 S_0^{(2)} = \frac{1}{4} (- 3S_0 + 13S_1 - 17S_2 + 7S_3)$$

(3.2-27)

$$- \frac{h^3}{240} (83S_0^{(3)} + 391S_1^{(3)} + 179S_2^{(3)} + 7S_3^{(3)}) .$$

Now, in order to obtain $S_0^{(3)}$ in terms of function values and second derivatives, Equations (3.2-12) and (3.2-13) may be combined to produce

(3.2-28) $$\frac{h^4 S_2^{(3)}}{30} = S_3 - 2S_2 + S_1 - \frac{h^2}{20} (S_3^{(2)} + 18S_2^{(2)} + S_1^{(2)})$$

and a similar expression for $S_1^{(4)}$. These two relations may then be combined with the forms (3.2-14) and (3.2-18) to give

$$h^3 S_0^{(3)} = 35S_0 - 60S_1 + 15S_2 + 10S_3$$

(3.2-29)

$$- \frac{h^2}{12} (57S_0^{(2)} + 324S_1^{(2)} + 153S_2^{(2)} + 6S_3^{(2)}) .$$

Finally, the expansion of $S_0^{(3)}$ in terms of function values and first derivatives is produced by combining the expressions (3.2-8) and (3.2-9) into

$$h^2S_1^{(2)} = \frac{h^2S_0^{(2)}}{3}$$

(3.2-30)

$$+ \frac{1}{6}(5S_2 - 40S_1 + 35S_0 - hS_2^{(1)} + 16hS_1^{(1)} + 15hS_0^{(1)})$$

and applying this result, with Equations (3.2-10) and (3.2-11) substituted in (3.2-29), to produce

$$6h^3S_0^{(3)} = 450S_0 + 45S_1 - 450S_2 - 45S_3$$

(3.2-31)

$$+ 162hS_0^{(1)} + 585hS_1^{(1)} + 234hS_2^{(1)} + 9hS_3^{(1)} .$$

It is possible, however, to compute the above multipoint expansions numerically. Under the assumption that the coefficients in Equation (3.2-1) exist and are unique, this equation should be satisfied by all polynomials up to order $2r+1$ and similarly by any polynomial spline of order $2r+1$ defined on the same partition π of $[x_0, x_n]$. Therefore, if $4r$ independent splines are used successively to replace the appropriate function values and derivatives in (3.2-1), then the resulting set of $4r$ equations constitutes a determining set for the $4r$ quantities $a_1, a_2, \ldots, a_{4r}$. As the $4r$ independent splines of order $2r+1$, we choose the functions $1, x, x^2, \ldots, x^{2r+1}, (x-1)_+^{2r+1}$, $(x-2)_+^{2r+1}, \ldots, (x-2r+2)_+^{2r+1}$, and determine the quantities a_j on the integer knots. For $n = 2r$, $m = 1$, and $h = 1$, in Equation (3.2-1), the set of equations is

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & 2 & 3 & \cdots & (2r-1) & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & 4 & 9 & \cdots & (2r-1)^2 & 0 & 0 & 0 & \cdots & 0 \\
 & & & \vdots & & & & & & & \\
0 & 1 & 2^{2r} & 3^{2r} & \cdots & (2r-1)^{2r} & 2r! & 2r! & 2r! & \cdots & 2r! \\
0 & 1 & 2^{2r+1} & 3^{2r+1} & \cdots & (2r-1)^{2r+1} & 0 & 1.(2r+1)! & 2.(2r+1)! & \cdots & (2r-1)(2r+1)! \\
0 & 0 & 1 & 2^{2r+1} & \cdots & (2r-2)^{2r+1} & 0 & 0 & 1.(2r+1)! & \cdots & (2r-2)(2r+1)! \\
 & & & \vdots & & & & & \vdots & & \\
0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & (2r+1)!
\end{bmatrix}
\begin{bmatrix}
a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{2r-1} \\ a_{2r} \\ a_{2r+1} \\ \vdots \\ a_{4r}
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 1 \\ 0 \\ \vdots \\ \vdots \\ \\ \\ \\ 0
\end{bmatrix}
$$

Figure 3.2.1

It is evident that the constant vector on the right-hand side has only one non-zero element for any m in the range 1 to $2r$. Hence, finding the inverse of the coefficient matrix for a particular value of n immediately produces $(2r-1)$ formulae. Therefore, for any spline of order $2r+1$, an integer inverse routine (Gabel [1973]) need only be applied $2r$ times. The coefficient matrix resembles the ill-conditioned Vandermonde matrix involved in the determination of the coefficients of an interpolation polynomial; however, an exact solution may be obtained using integer arithmetic.

Now if formulae in the form

$$
\begin{aligned}
h^m S_0^{(m)} &= a_1 S_0 + a_2 S_1 + a_3 S_2 + a_4 S_3 \\
&\quad + h^2 \, [a_5 S_0^{(2)} + a_6 S_1^{(2)} + a_7 S_2^{(2)} + a_8 S_3^{(2)}]
\end{aligned}
$$

are required for $m = 1, 3, 4$, then the system of equations becomes

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 9 & 2 & 2 & 2 & 2 \\ 0 & 1 & 8 & 27 & 0 & 6 & 12 & 18 \\ 0 & 1 & 16 & 81 & 0 & 12 & 48 & 108 \\ 0 & 1 & 32 & 243 & 0 & 20 & 160 & 540 \\ 0 & 0 & 1 & 32 & 0 & 0 & 20 & 160 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 20 \end{bmatrix} \cdot A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 24 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Figure 3.2.2

The solution of these equations gives

$$hS_0^{(1)} = -\frac{37}{12} S_0 + \frac{9}{2} S_1 - \frac{3}{4} S_2 - \frac{2}{3} S_3$$

$$+ h^2 \left[-\frac{23}{240} S''_0 + \frac{59}{40} S''_1 + \frac{67}{80} S''_2 + \frac{1}{30} S''_3\right]$$

$$h^3 S_0^{(3)} = 35S_0 - 60S_1 + 15S_2 + 10S_3$$

$$- \frac{h^2}{12} [57S''_0 + 324S''_1 + 153S''_2 + 6S''_3]$$

$$h^4 S_0^{(4)} = -120S_0 + 210S_1 - 60S_2 - 30S_3$$

$$+ \frac{h^2}{2} [24S''_0 + 195S''_1 + 78S''_2 + 3S''_3] .$$

Using extended precision integer arithmetic, the following tables have been constructed giving the boundary formulae exactly for $r = 2, 3$, and 4.

N= 1 r = 2

M				
2	$\frac{-235}{16}$	$\frac{65}{16}$	$\frac{155}{16}$	$\frac{15}{16}$
	$\frac{-111}{16}$	$\frac{-227}{16}$	$\frac{-79}{16}$	$\frac{-3}{16}$
3	$\frac{75}{1}$	$\frac{15}{2}$	$\frac{-75}{1}$	$\frac{-15}{2}$
	$\frac{27}{1}$	$\frac{195}{2}$	$\frac{39}{1}$	$\frac{3}{2}$
4	$\frac{-765}{4}$	$\frac{-345}{4}$	$\frac{1005}{4}$	$\frac{105}{4}$
	$\frac{-249}{4}$	$\frac{-1173}{4}$	$\frac{-537}{4}$	$\frac{-21}{4}$

N= 2 r = 2

M				
1	$\frac{-37}{12}$	$\frac{9}{2}$	$\frac{-3}{4}$	$\frac{-2}{3}$
	$\frac{-23}{240}$	$\frac{59}{40}$	$\frac{67}{80}$	$\frac{1}{30}$
3	$\frac{35}{1}$	$\frac{-60}{1}$	$\frac{15}{1}$	$\frac{10}{1}$
	$\frac{-19}{4}$	$\frac{-27}{1}$	$\frac{-51}{4}$	$\frac{-1}{2}$
4	$\frac{-120}{1}$	$\frac{210}{1}$	$\frac{-60}{1}$	$\frac{-30}{1}$
	$\frac{12}{1}$	$\frac{195}{2}$	$\frac{39}{1}$	$\frac{3}{2}$

N= 3 r = 2

M				
1	$\frac{-1}{1}$	$\frac{1}{2}$	$\frac{1}{1}$	$\frac{-1}{2}$
	$\frac{1}{15}$	$\frac{13}{24}$	$\frac{13}{60}$	$\frac{1}{120}$
2	$\frac{-3}{4}$	$\frac{13}{4}$	$\frac{-17}{4}$	$\frac{7}{4}$
	$\frac{-83}{240}$	$\frac{-391}{240}$	$\frac{-179}{240}$	$\frac{-7}{240}$
4	$\frac{15}{1}$	$\frac{-45}{1}$	$\frac{45}{1}$	$\frac{-15}{1}$
	$\frac{-7}{4}$	$\frac{41}{4}$	$\frac{25}{4}$	$\frac{1}{4}$

N= 4 r = 2

M				
1	-11/6	3/1	-3/2	1/3
	-19/720	-3/20	-17/240	-1/360
2	2/1	-5/1	4/1	-1/1
	3/20	13/24	13/60	1/120
3	-1/1	3/1	-3/1	1/1
	-59/120	-31/40	-9/40	-1/120

N= 1 r = 3

M						
2	-545083/20808	-228403/3468	55895/10404	366835/5202	110159/6936	2947/10404
	-202717/20808	-34831/612	-263465/2601	-126241/2601	-5947/1224	-421/10404
3	14259/68	13874/17	-427/17	-13818/17	-12551/68	-56/17
	4077/68	10094/17	39251/34	9558/17	3841/68	8/17
4	-885542/867	-2910551/578	-57904/867	4305371/867	329762/289	35231/1734
	-230546/867	-340451/102	-6044852/867	-2993998/867	-17762/51	-5033/1734
5	57330/17	323260/17	17080/17	-321930/17	-74410/17	-1330/17
	14310/17	203140/17	447140/17	225090/17	22790/17	190/17
6	-2046905/289	-12465075/289	-1057630/289	12583970/289	2933175/289	52465/289
	-500495/289	-446725/17	17322080/289	-8843500/289	-52865/17	-7495/289

N= 2 r = 3

M						
1	-227/80	4985/768	-389/96	-123/128	125/96	221/3840
	-111/1120	160931/161280	-25811/20160	-37837/26880	-3277/20160	-221/161280
3	469/8	-6181/128	-959/16	1981/64	287/16	91/128
	-307/48	-40589/768	-6115/96	-8695/384	-197/96	-13/768
4	-6279/16	-301/2	8911/8	-441/2	-5383/16	-14/1
	939/32	5047/12	39251/48	1593/4	3841/96	1/3
5	2961/2	61635/32	-25095/4	14805/16	7455/4	2499/32
	-381/4	-111991/64	-33689/8	-69909/32	-1783/8	-119/64
6	-53235/16	-12705/2	138075/8	-4725/2	-80115/16	-210/1
	6375/32	16625/4	178405/16	23445/4	19175/32	5/1

N= 3 r = 3

M						
1	-39/4	154/9	25/9	-18/1	251/36	8/9
	-11/840	-95/27	-35681/3780	-521/105	-767/1512	-4/945
2	58075/1836	-17821/306	-9899/918	30509/459	-5261/204	-3019/918
	-19037/385560	151873/11340	1686071/48195	884623/48195	8513/4536	3019/192780
4	-72380/153	45115/51	20720/153	-150430/153	6580/17	7525/153
	-2870/459	-11669/54	-243692/459	-126358/459	-758/27	-215/918
5	2310/1	-13160/3	-1400/3	4620/1	-5530/3	-700/3
	23/1	9604/9	22970/9	1310/1	1199/9	10/9
6	-287630/51	184870/17	33740/51	-552580/51	74550/17	28210/51
	-7781/153	-23581/9	-938480/153	-476488/153	-2843/9	-403/153

N= 4 r = 3

M						
1	-6/5	73/120	26/15	-27/20	1/15	17/120
	-29/4200	-29833/100800	-6191/12600	-3533/16800	-32/1575	-17/100800
2	-3/1	206/15	-314/15	57/5	1/15	-19/15
	3/56	1451/900	22357/6300	2501/1400	2279/12600	19/12600
3	15/1	-227/4	76/1	-75/2	-1/1	17/4
	-15/56	-13693/3360	-4583/420	-659/112	-509/840	-17/3360
5	-182/1	679/1	-896/1	434/1	14/1	-49/1
	-167/60	4463/120	1831/15	4049/60	419/60	7/120
6	630/1	-2352/1	3108/1	-1512/1	-42/1	168/1
	21/4	-616/5	-4237/10	-1161/5	-479/20	-1/5

N= 5 r = 3

M						
1	-7/4	7/3	1/3	-2/1	17/12	-1/3
	1/480	11/135	4237/15120	43/280	479/30240	1/7560
2	181/102	-201/68	-98/51	689/102	-163/34	233/204
	-4103/257040	-11507/30240	-63947/64260	-68003/128520	-821/15120	-233/514080
3	-3/2	4/1	-2/1	-3/1	7/2	-1/1
	139/1680	167/180	3011/2520	139/280	241/5040	1/2520
4	61/17	-288/17	542/17	-508/17	237/17	-44/17
	-14219/42840	-733/630	4859/5355	5641/5355	307/2520	11/10710
6	-630/17	3150/17	-6300/17	6300/17	-3150/17	630/17
	-137/68	9/4	-336/17	-268/17	-7/4	-1/68

N= 6 r = 3

M						
1	-137 60	5 1	-5 1	10 3	-5 4	1 5
	-283 302400	-1513 50400	-1069 12600	-6967 151200	-479 100800	-1 25200
2	15 4	-77 6	107 6	-13 1	61 12	-5 6
	17 2240	343 2160	2297 6048	131 672	1199 60480	1 6048
3	-17 4	71 4	-59 2	49 2	-41 4	7 4
	-823 20160	-9689 20160	-281 315	-151 360	-841 20160	-1 2880
4	3 1	-14 1	26 1	-24 1	11 1	-2 1
	93 560	167 180	3011 2520	139 280	241 5040	1 2520
5	-1 1	5 1	-10 1	10 1	-5 1	1 1
	-2519 5040	-4919 5040	-233 315	-82 315	-121 5040	-1 5040

= 1 r = 8

M								
2	-895082537 21895424	-9080036441 21895424	-24119893737 21895424	530324527 21895424	24381266173 21895424	8655639837 21895424	525645589 21895424	2136589 21895424
	-2471553065 197058816	-36432938329 197058816	-21784144483 21895424	-337743313859 197058816	-189175599463 197058816	-3470414975 21895424	-1072612373 197058816	-2136589 197058816
3	1226108 2759	135839345 22072	183540853 11036	-4487523 22072	-46113958 2759	-131319389 22072	-3989453 11036	-32433 22072
	2616644 24831	170348411 66216	162461805 11036	5100281023 198648	238947731 16554	52664007 22072	8140877 99324	10811 66216
4	-8702178387 2736928	-142576623411 2736928	-395173367595 2736928	1290191781 2736928	395430614751 2736928	141118069023 2736928	8578422687 2736928	34871151 2736928
	-5659549553 8210784	-172134983185 8210784	-343775898185 2736928	-1818949097003 8210784	-1026186615199 8210784	-56606418709 2736928	-5835145781 8210784	-11623717 8210784
5	95231865 5518	6855125685 22072	4860230775 5518	85515345 22072	-4849964565 5518	-6938925585 22072	-105498075 5518	-1715445 22072
	59554675 16554	2697520205 22072	4170757345 5518	89095720105 66216	4201536115 5518	2783957795 22072	71762185 16554	190605 22072
6	-98361656175 1368464	-1865345776635 1368464	-5382343827495 1368464	-56309279115 1368464	5363118654315 1368464	1921847210775 1368464	116919372075 1368464	475302255 1368464
	-60375452965 4105392	-2173427827625 4105392	-4573766067165 1368464	-24599400447595 4105392	-13955014281515 4105392	-771190452045 1368464	-79532249785 4105392	-158434085 4105392
7	607086270 2759	23701036905 5518	34596811620 2759	1031317245 5518	-34451555670 2759	-24726237165 5518	-752342220 2759	-6116985 5518
	123075470 2759	9136706145 5518	29208637920 2759	105249376855 5518	29908819710 2759	9923311335 5518	170590700 2759	679665 5518
8	-152100846135 342116	-3015959324175 342116	-8869924835415 342116	-159077229255 342116	8830946373795 342116	3172249236555 342116	193081679595 342116	784945035 342116
	-30694211135 342116	-1157824910935 342116	-7458158482845 342116	-13480674373565 342116	-7671527853745 342116	-1273227794265 342116	-43780845395 342116	-87216115 342116

2 r = 4

N								
1	-853397701 249943680	-28329121 17853120	-108782791 11902080	41647633 1785312	13490159 7141248	-58280587 5951040	-45413203 35706240	-21118 1952685
	-1396095419 17995944960	-20083102759 8997972480	-51945513817 5998648320	17179776557 899797248	44678411215 3599188992	-6491502349 2999324160	1355884357 17995944960	10559 70296660
3	11127277 99184	27911215 49592	14855115 99184	-40434851 24796	8075243 99184	31919439 49592	7953037 99184	4214 6199
	-61116013 7141248	-564708847 3570624	-1943245249 2380416	-2583076081 1785312	-5907663203 7141248	-163905157 1190208	-33843653 7141248	-2107 223164
4	-6541660 6199	-45581409 6199	-5962056 6199	117557321 6199	-7959084 6199	-45796599 6199	-5668480 6199	-48033 6199
	6321859 111582	252661451 148776	180530879 18597	7554490639 446328	353893465 37194	233986757 148776	3014144 55791	5337 49592
5	314101533 49592	1320555531 24796	376965243 49592	-1672090485 12398	437614395 49592	1297277091 24796	321292893 49592	340329 6199
	-354644191 1190208	-6545837209 595104	-26866567067 396736	-35571957925 297552	-80180748305 1190208	-2210188139 198368	-455594527 1190208	-113443 148776
6	-171261840 6199	-1607320395 6199	-311742090 6199	4128673155 6199	-238462380 6199	-1599636285 6199	-198567210 6199	-1682955 6199
	22757390 18597	2486889395 49592	8052379485 24796	86977609495 148776	4116081935 12398	2730424605 49592	140803655 74388	186995 49592
7	1072996005 12398	5366830035 6199	2625762195 12398	-14058508170 6199	1434141555 12398	5442459435 6199	1354059525 12398	5739060 6199
	-1107496375 297552	-23971163185 148776	-106905386675 99184	-146840868565 74388	-335851791545 297552	-9305133875 49592	-1920552535 297552	-478255 37194
8	-1096845120 6199	-11378014515 6199	-3218445090 6199	30288459915 6199	-1409286060 6199	-11714254965 6199	-1459243170 6199	-12370995 6199
	46476920 6199	16570045275 49592	56532164745 24796	209629532245 49592	30105345375 12398	20049712725 49592	344983625 24796	1374555 49592

3 r = 4

N								
1	-203801 30465	-14768357 243720	15706193 121860	356489 16248	-844789 6093	9346553 243720	2004791 121860	24823 81240
	8798 1919295	-108014327 40944960	-116705083 2924640	-2124519599 24566976	-35556037 682416	-51521407 5849280	-18685847 61417440	-24823 40944960
2	670829501 30221280	1382403721 6044256	-4867046137 10073760	-476769095 6044256	3128097787 6044256	-1448162387 10073760	-371867333 6044256	-34529329 30221280
	-1505102659 15231525120	-168183976813 15231525120	109665292571 725310720	-988635974035 3046305024	-594092590391 3046305024	-167257097953 5077175040	-17328476057 15231525120	34529329 15231525120
4	-127750007 251844	-866316647 251844	661844323 83948	252956137 251844	-2097323333 251844	198801113 83948	251823859 251844	4673683 251844
	-127046609 18132768	-4818268513 18132768	-15915803273 6044256	-97496064827 18132768	-57760096063 18132768	-3237840037 6044256	-335090549 18132768	-667669 18132768
5	2532236 677	11239249 677	-28834232 577	-2414475 677	29940260 677	-8757301 677	-3638264 677	-67473 677
	456017 12186	31159117 16248	31016224 2031	1440929705 48744	69961655 4062	46809379 16248	604753 6093	1071 5416
6	-716426473 41974	-1757844445 41974	5647994583 41974	26059355 41974	-5654310235 41974	1739251353 41974	702269645 41974	13006217 41974
	-465001999 3022128	-25793714747 3022128	-54087765943 1007376	-289343252545 3022128	-163802631905 3022128	-9045792167 1007376	-932722123 3022128	-1858031 3022128
7	36444240 677	36377460 677	-194276880 677	24810660 677	181223280 677	-60755940 677	-23390640 677	-432180 677
	316030 677	35687715 1354	89647870 677	285591605 1354	77180690 677	25158085 1354	430570 677	1715 1354
8	-2303992530 20987	-92148210 20987	8436464190 20987	-2728743570 20987	-6968838870 20987	2680402410 20987	959202090 20987	17654490 20987
	-78728215 83948	-4445266055 83948	-18515466245 83948	-25791970365 83948	-13035408985 83948	-2070131785 83948	-70384235 83948	-140115 83948

N= 4 r = 8

M								
1	-3739747/5760	450367/576	1010989/384	-1556645/288	2719655/1152	882737/960	-705959/1152	-4019/144
	3687907/17418240	177896273/1741824	875020355/1161216	1225086077/870912	2816692393/3483648	390817663/2903040	16138919/3483648	4019/435456
2	243551/60	-390529/80	-165193/10	1626703/48	-59195/4	-461303/80	57632/15	13999/80
	-239231/181440	-154929151/241920	-47609969/10080	-1279980607/145152	-61316453/12096	-68064179/80640	-658771/22680	-13999/241920
3	-761051/60	3658343/240	516669/10	-5086225/48	555199/12	1442463/80	-360391/30	-131311/240
	730811/181440	1453688857/725760	148847557/10080	4001846593/145152	575138033/36288	212813659/80640	8239051/90720	131311/725760
5	1967637/8	-1183245/4	-8008245/8	4107495/2	-7174965/8	-1397781/4	1862805/8	10605/1
	-99457/1152	-22384855/576	-109942405/384	-153920755/288	-353912975/1152	-9821413/192	-2027905/1152	-505/144
6	-1558116/1	1875258/1	6336918/1	-13005090/1	5680080/1	2212686/1	-1474578/1	-67158/1
	2141/4	1969759/8	43524107/24	27078165/8	5836655/3	7774519/24	89181/8	533/24
7	11249343/2	-6773508/1	-45723447/2	46930905/1	-40999455/2	-7984494/1	10643031/2	242361/1
	-184321/96	-10666621/12	-209466957/32	-586362215/48	-674044175/96	-9352317/8	-3862057/96	-3847/48
8	-12262320/1	14773185/1	49818510/1	-102287745/1	44683380/1	17402175/1	-11598930/1	-526255/1
	4175/1	31006635/16	114155005/8	426036005/16	61215895/4	40769125/16	701485/8	2795/16

N= 5 r = 8

M								
1	-761/396	4751/1584	-175/396	-101/48	901/396	-1451/1584	35/396	1/48
	2803/5987520	638789/7983360	373361/1995840	-314231/23950080	-57611/665280	-150989/7983360	-4127/5987520	-1/725760
2	-75013/24552	1560215/98208	-395525/16368	429281/98208	237643/12276	-517181/32736	143677/49104	46751/98208
	-263251/53032320	-939214889/1484904960	-824866601/247484160	-7862031073/1484904960	-1055877659/371226240	-228791069/494968320	-2347585/148490496	-46751/1484904960
3	5123/132	-80879/528	6049/33	277/16	-26143/132	73883/528	-794/33	-65/16
	71651/1995840	7618843/2661120	2320931/110880	45858401/1140480	2224507/95040	3472319/887040	67289/498960	13/48384
4	-2089189/16368	8097635/16368	-3117127/5456	-1569637/16368	10961273/16368	-2514581/5456	1277633/16368	217409/16368
	-51586021/247484160	-1820089421/247484160	-5237567029/82494720	-31720362319/247484160	-18795621227/247484160	-1054151873/82494720	-109112497/247484160	-217409/247484160
6	652435/341	-5064535/682	2941890/341	878675/682	-3366125/341	4652655/682	-394520/341	-134155/682
	-570607/147312	27775139/294624	2876245/3069	560227687/294624	165812767/147312	18588223/98208	60116/9207	3833/294624
7	-102900/11	401205/11	-472920/11	-4725/1	514500/11	-359625/11	61320/11	945/1
	4507/396	-236785/528	-151271/33	-14595077/1584	-715799/132	-480391/528	-6211/198	-1/16
8	7798035/341	-30543135/341	36518895/341	2621325/341	-37742775/341	26699715/341	-4578315/341	-773745/341
	-1104229/49104	53044489/49104	183468751/16368	1096082663/49104	641235349/49104	35779853/16368	3698627/49104	7369/49104

N= 6 r = 8

M								
1	-10459/15120	-3911/945	82697/5040	-33769/1512	43559/3024	-4013/1260	-13219/15120	3107/7560
	-125621/914457600	-3551741/114307200	-141487457/304819200	-90046151/91445760	-107343647/182891520	-7531417/76204800	-3118637/914457600	-3107/457228800
2	-53/18	4301/168	-1516/21	47833/504	-2549/42	2243/168	233/63	-97/56
	1349/1088640	1696531/10160640	2511037/1270080	126219791/30481920	6276173/2540160	4231189/10160640	27383/1905120	97/3386880
3	33/8	-243/8	667/8	-889/8	583/8	-133/8	-35/8	17/8
	-271/32256	-26909/53760	-283583/96768	-368729/69120	-1485871/483840	-7081/13824	-1219/69120	-17/483840
4	38/1	-429/2	494/1	-1163/2	346/1	-139/2	-22/1	19/2
	1493/30240	35759/40320	-206761/30240	-2452813/120960	-79465/6048	-274541/120960	-2383/30240	-19/120960
5	-971/6	5561/6	-4327/2	15475/6	-9325/6	637/2	589/6	-259/6
	-89749/362880	-241937/362880	4429447/120960	6892129/72576	4364993/72576	178469/17280	129947/362880	37/51840
7	1940/1	-11120/1	25980/1	-31000/1	18700/1	-3840/1	-1180/1	520/1
	-9169/3024	-3011/756	-454537/1008	-1736009/1512	-2192831/3024	-15679/126	-13045/3024	-13/1512
8	-6720/1	38520/1	-90000/1	107400/1	-64800/1	13320/1	4080/1	-1800/1
	55/9	3253/168	131081/84	2004521/504	35155/14	72371/168	3763/252	5/168

N= 7 r = 8

M								
1	-67/30	269/60	-59/20	-11/12	23/6	-67/20	27/20	-13/60
	29/777600	22759/3628800	766207/10886400	71215/435456	27593/272160	188399/10886400	6523/10886400	13/10886400
2	324691/89280	-201409/17856	324053/29760	42471/17856	251803/17856	385723/29760	-95695/17856	78061/89280
	-6103469/16198963200	633155857/16198963200	-1864857733/5399654400	2270506759/3239792640	-1348359431/3239792640	378414677/5399654400	-39176513/16198963200	-78061/16198963200
3	-9/2	141/8	-47/2	41/8	37/2	-169/8	19/2	-13/8
	37/13440	74591/483840	171461/181440	730777/483840	293939/362880	190907/1451520	17/3780	13/1451520
4	847/186	-2050/93	2581/62	-3337/93	1621/186	244/31	-1139/186	119/93
	-80231/4821120	-239419/527310	-18584297/11249280	-30383473/16873920	-25154239/33747840	-75451/703080	-23953/6749568	-17/2410560
5	-2/1	19/2	-16/1	15/2	10/1	-31/2	8/1	-3/2
	7559/90720	120289/120960	18467/10080	18847/10368	701/864	4981/40320	377/90720	1/120960
6	-883/124	6305/124	-19287/124	32765/124	-33385/124	20403/124	-6925/124	1007/124
	-7500403/22498560	-30440339/22498560	-3043697/2499840	-18728141/4499712	-15291481/4499712	-1585333/2499840	-504631/22498560	-1007/22498560
8	2835/31	-19845/31	59535/31	-99225/31	99225/31	-59535/31	19845/31	-2835/31
	-3967/1984	8437/1984	6171/1984	82063/1984	74127/1984	14107/1984	501/1984	1/1984

N= 8   r = 4

M								
1	$\frac{-363}{140}$	$\frac{7}{1}$	$\frac{-21}{2}$	$\frac{35}{3}$	$\frac{-35}{4}$	$\frac{21}{5}$	$\frac{-7}{6}$	$\frac{1}{7}$
	$\frac{-299}{16934400}$	$\frac{-13589}{5080320}$	$\frac{-2995}{112896}$	$\frac{-854599}{15240960}$	$\frac{-343331}{10160640}$	$\frac{-36311}{6350400}$	$\frac{-3011}{15240960}$	$\frac{-1}{2540160}$
2	$\frac{469}{90}$	$\frac{-223}{10}$	$\frac{879}{20}$	$\frac{-949}{18}$	$\frac{41}{1}$	$\frac{-201}{10}$	$\frac{1019}{180}$	$\frac{-7}{10}$
	$\frac{6011}{32659200}$	$\frac{193099}{10886400}$	$\frac{3191623}{21772800}$	$\frac{133847}{466560}$	$\frac{2609}{15552}$	$\frac{38731}{1360800}$	$\frac{63241}{65318400}$	$\frac{1}{518400}$
3	$\frac{-967}{120}$	$\frac{638}{15}$	$\frac{-3929}{40}$	$\frac{389}{3}$	$\frac{-2545}{24}$	$\frac{268}{5}$	$\frac{-1849}{120}$	$\frac{29}{15}$
	$\frac{-59513}{43545600}$	$\frac{-1630847}{21772800}$	$\frac{-6985331}{14515200}$	$\frac{-918509}{1088640}$	$\frac{-4113707}{8709120}$	$\frac{-565427}{7257600}$	$\frac{-116471}{43545600}$	$\frac{-29}{5443200}$
4	$\frac{28}{3}$	$\frac{-111}{2}$	$\frac{142}{1}$	$\frac{-1219}{6}$	$\frac{176}{1}$	$\frac{-185}{2}$	$\frac{82}{3}$	$\frac{-7}{2}$
	$\frac{323}{38880}$	$\frac{172031}{725760}$	$\frac{398323}{362880}$	$\frac{3618319}{2177280}$	$\frac{17693}{20160}$	$\frac{20585}{145152}$	$\frac{5273}{1088640}$	$\frac{1}{103680}$
5	$\frac{-23}{3}$	$\frac{295}{6}$	$\frac{-135}{1}$	$\frac{1235}{6}$	$\frac{-565}{3}$	$\frac{207}{2}$	$\frac{-95}{3}$	$\frac{25}{6}$
	$\frac{-45337}{1088640}$	$\frac{-249383}{435456}$	$\frac{-15947}{9072}$	$\frac{-960139}{435456}$	$\frac{-235243}{217728}$	$\frac{-123407}{725760}$	$\frac{-157}{27216}$	$\frac{-5}{435456}$
6	$\frac{4}{1}$	$\frac{-27}{1}$	$\frac{78}{1}$	$\frac{-125}{1}$	$\frac{120}{1}$	$\frac{-69}{1}$	$\frac{22}{1}$	$\frac{-3}{1}$
	$\frac{15119}{90720}$	$\frac{120289}{120960}$	$\frac{18467}{10080}$	$\frac{18847}{10368}$	$\frac{701}{864}$	$\frac{4981}{40320}$	$\frac{377}{90720}$	$\frac{1}{120960}$
7	$\frac{-1}{1}$	$\frac{7}{1}$	$\frac{-21}{1}$	$\frac{35}{1}$	$\frac{-35}{1}$	$\frac{21}{1}$	$\frac{-7}{1}$	$\frac{1}{1}$
	$\frac{-181439}{362880}$	$\frac{-362377}{362880}$	$\frac{-38641}{40320}$	$\frac{-51907}{72576}$	$\frac{-20669}{72576}$	$\frac{-1679}{40320}$	$\frac{-503}{362880}$	$\frac{-1}{362880}$

3.2.4 *An Algorithm for the Rapid Calculation of Odd-Order Polynomial Splines with Equidistant Knots and Arbitrary Linear Boundary Conditions*

A good computational method of obtaining the coefficients $C_{i,k}^{(p)}$ in Equation (3.2-2) is to make use of the recurrence relations given in Fyfe [1971], Albasiny and Hoskins [1972], and Meek [1974]. In particular, these relations may be expressed as

$$(3.2\text{-}32) \qquad C_{i,2r+2}^{(k)} = (2r+2-i)\, C_{i-1,2r+1}^{(k)} + (i+1)\, C_{i,2r+1}^{(k)} ,$$

where $k = 0, 1, \ldots, 2r$;

the coefficients $C_{i,\ell}^{(k)}$ are symmetric in the sense that

$$(3.2\text{-}33) \qquad C_{i,2r+1}^{(k)} = (-1)^k\, C_{2r-i,2r+1}^{(k)} .$$

The spline $s(x)$ must satisfy the given $2r$ linearly independent boundary conditions which may be assumed to be in the general form given by (3.2-5) and (3.2-6). For numerical purposes, it is better if Equations (3.2-5) and (3.2-6) are strongly independent; however, a failure label is provided in procedure bandet 1 (Wilkinson and Reinsch [1971]) in the event that they should appear dependent numerically. These boundary equations are transformed into equations involving only p'th derivatives by use of the multipoint expansions given in Section (3.2.3). For convenience, an algorithm is included here which generates these expansions and uses them to re-express the boundary conditions in the required form. However, these expansions are tabulated exactly and extensively in Section 3.2.3 if the user wishes to perform this step himself.

It can be seen that the $n+1-2r$ equations (3.2-2) and the $2r$ boundary conditions (3.2-5) and (3.2-6) suffice to determine the $n+1$ unknown p'th derivatives $s_j^{(p)}$ of the interpolating polynomial spline $s(x)$. The transformed versions of Equations (3.2-4) and (3.2-5) do not give, in conjunction with the Equations (3.2-2), a set of linear simultaneous equations with a coefficient matrix of band width precisely $2r+1$; however, a small amount of initial elimination converts this complete set of equations into strict band form. In this form, solution of the system is both rapid and economical (Wilkinson and Reinsch [1971]). Although the coefficient matrix of this set of equations is not in general diagonally dominant or symmetric, existence of the spline is assured when the boundary equations satisfy the Polya conditions (Polya [1931]).

It is difficult to estimate, for general boundary conditions, the conditioning of the coefficient matrix A involved in the determination of the parameters of the spline. However, the condition number of the coefficient matrix B, for periodic polynomial spline interpolation, is well known (Albasiny and Hoskins [1972]). Thus, in the ensuing analysis, the matrix A is written as

$$A = B - C \;;$$

C is a matrix depending directly on the boundary conditions. It can be concluded that

$$\|A^{-1}\| \le \frac{\|B^{-1}\|}{1 - \|B^{-1}C\|}$$

provided $\|B^{-1}C\| < 1$. Therefore, the conditioning of the periodic and non-periodic cases will be approximately similar if $||B^{-1}C||$ is small and $||A||$ is comparable in size with $||B||$. A method for economically evaluating the infinity norm of a specific band matrix is given in Chapter five.

The p'th derivatives of the spline can now be considered available at the knots, and the calculation of the other derivatives of the spline at any particular element of the partition is easily performed using the expansion (3.2-1). Choice of the parameter p is at the user's convenience, since there appears to be no general rule for determining an optimal value of this quantity.

The algorithm involves the use of five procedures in addition to the routines <u>bandet 1</u> and <u>bansol 1</u> for solving band equations (Wilkinson and Reinsch [1971]), and a standard Gaussian elimination procedure . Specifically, the procedures are named

<u>multipoint</u>

<u>consistency</u>

<u>coefficient</u>

<u>elimination</u>

<u>interpolate</u> .

The two procedures central to the determination of the spline parameters and the performing of interpolation are:

(a) the procedure <u>multipoint</u> which solves for the p'th derivatives of the spline at the interior knots, and

(b) the procedure <u>interpolate</u> which accepts as input these derivatives and performs interpolation.

The parameter lists of these two procedures will be described in detail.

(i) Formal Parameter List

(a) Input to *__procedure__* *__multipoint:__*

p is the order of the spline derivatives calculated.

r the odd order spline has order 2r+1 .

n there are n+1 equally spaced knots in the partition y(0), y(1), ..., y(n) .

b is a two dimensional array of order 2r by 2r+1 . The coefficients of the left boundary conditions appear in the first r rows, those of the right in the second r rows. The constant multiples of the first derivatives in the boundary equations appear in column one, the constant multiples of the second derivatives appears in column two, and so on; in column 2r , we have the constant multiples of the 2r'th derivatives. The corresponding constants on the right hand side of the boundary equations appear in column 2r+1 .

y a vector of length n+1 containing the n+1 given function values at each of the n+1 knots, 0, 1, 2, ..., n .

Output of *__procedure__* *__multipoint:__*

bb a matrix of size n+1 by 1 returning the p'th derivatives at each of the n+1 knots. A matrix rather than a vector is used to facilitate input to the published __procedure__ __bansol 1__ (Wilkinson and Reinsch [1971]).

(b) *Input to procedure interpolate:*

p the order of the spline derivatives that have been determined by the procedure multipoint.

r the spline is of order $2r+1$.

n the $n+1$ knots $0, 1, \ldots, n$ define the partition.

inter the number of values at which interpolation is to take place.

ider the order of the interpolated derivative; ider ranges from 0 to $2r$.

y is a vector of length $n+1$ $(y(0), y(1), \ldots, y(n+1))$ containing the given functional values.

dp a vector of length $n+1$ $(dp(0), \ldots, dp(n+1)$ containing the calculated p'th derivatives at the knots of the partition. These values may be obtained from the matrix bb as output from procedure multipoint.

ti a vector of length inter containing the values of the independent argument at which interpolation is to be performed.

Output from procedure interpolate:

ti a vector of length inter containing the required interpolated values.

(ii) Algol W Programs

```
procedure multipoint (integer value p, r, n; real array b(*,*);
                      real array bb (*,*); real array y(*));
comment:    the procedure multipoint solves for the p'th derivatives
            s_i^(p)/p! at the n - 1 interior knots of the partition.
            This is accomplished by converting the 4r given
            boundary conditions to a form involving only the
            p'th derivatives.  The partition is assumed to be,
            without loss of generality, the integers 0, 1, ..., n.
            The boundary equations and the set of spline consistency
            equations (Fyfe [1971]) are placed in band form,
            and the published procedures bandet 1 and bansol 1
            (Wilkinson and Reinsch [1971]) are called to solve
            for the required p'th derivatives ;
begin integer i,j,tr,fr,trl,ℓ,d2,k,rl;
    real    sum,suml,dl;
    real array c(1::4*r;1::2*r+1), f(0::2*r), aa(1::n+1;-2*r::2*r),
    m(1::n+1;1::2*r);
    integer array int(1::n+1);
    tr:=2*r;trl:=tr+1;fr:=tr+tr;rl:=r+1;
    coef(p,r,c);
    d2:=1;
    for j:=1 step 1 until p do
       d2:=d2*j;
    for j:=1 step 1 until tr do
    for i:=trl step 1 until fr do
       c(i,j):=c(i,j)*d2;
```

```
for i:=1 step 1 until tr do
begin
     sum1:=0.;
     for j:=1 step 1 until tr do
        sum1:=sum1+c(j,i)*y(j-1);
     c(i,tr1):=sum1
end;
for i:=1 step 1 until r do
begin
     for j:=1 step 1 until tr do
        f(j):=b(i,j);
     sum1:=0.;
     for k:=1 step 1 until tr do
     begin sum:=0.;l:=tr+k;
          for j:=1 step 1 until tr do
             sum:=sum+f(j)*c(l,j);
          b(i,k):=sum;
          sum1:=sum1+f(k)*c(k,tr1)
     end;
     b(i,tr1):=b(i,tr1)-sum1
end;
l:=n+1;
for i:=1 step 1 until tr do
begin
     sum1:=0.;
     for j:=1 step 1 until tr do
        sum1:=sum1+c(j,i)*y(l-j);
     c(tr+i,tr1):=(-1)**i*sum1;
```

```
        if p-i-(p-i) div 2*2¬= 0 then
        begin for j:=trl step 1 until fr do
                 c(j,i)=-c(j,i)
        end
    end;
    for i:=rl step 1 until tr do
    begin
        for j:=1 step 1 until tr do
           f(j):=b(i,j);
        l:=0;
        for k:=fr step -1 until trl do
        begin sum:=0.;l:=l+1;
              for j:=1 step 1 until tr do
                 sum:=sum+f(j)*c(k,j);
              b(i,l):=sum
        end;
        sum:=0.;
        for k:=1 step 1 until tr do
             sum:=sum+f(k)*c(tr+k,trl);
        b(i,trl):=b(i,trl)-sum
    end;
    comment: adjust the boundary equations to fit into band form.;
    elimination(1,r,b);
    elimination(-1,r,b);
    consistency(0,trl,f);
    d2:=1; l:=tr+2;
    for i:=1 step 1 until p do
       d2:=(l-i)*d2 div i;
```

```
l:=n+1-r;
for i:=1 step 1 until r do
for j:=-r step 1 until r do
   aa(i,j):=aa(l+i,j):=0.;
for i:=1 step 1 until r do
begin
      k:=r+i;
      for j:=1 step 1 until k do
          aa(i,j-i):=b(i,j);
      for j:=1 step 1 until tr1-i do
          aa(l+i,j-r1):=b(k,i+j-1)
end;
for i:=0 step 1 until tr do
begin
      sum:=f(i);
      for k:=r1 step 1 until l do
          aa(k,-r+i):=sum
end;
for i:=1 step 1 until r do
begin
     bb(i,1):=b(i,tr1);
     bb(n+2-i,1):=b(tr1-i,tr1)
end;
     consistency(p,tr1,f);
     for i:=r1 step 1 until l do
     begin
```

```
        sum:=0.;
        for k:=0 step 1 until tr do
           sum:=sum+f(k)*y(k+i-r1);
        bb(i,1):=sum*d2
     end;
     bandet1(n+1,r,r,1,0,0,aa,d1,d2,m,int,fail);
     bansol1(n+1,r,r,0,1,aa,m,int,bb)
end;
```

```
procedure  interpolate(integer value p,r,n,inter,ider;
        real array Y(*);real array dp(*);real array ti(*));
comment  the procedure interpolate accepts as input the p'th
         derivatives divided by p! and determines the remaining
         2r derivatives at an element of the position, a unit
         distance from which interpolation is required.  Inter-
         polation is then carried out for a function value  or any
         derivative using a finite Taylor series expansion;
begin integer tr,tr1,l1,l2,i,q,sign,m,pf,mf,l3;
      real h2,sum,sum1,d2r,sign2,t;
      real array c(1::4*r,1::2*r);
      tr:=2*r;tr1:=tr+1;
      coef(p,r,c);
      pf:=1;
      for i:=1 step 1 until p do
         pf:=pf*i;
      for m:=1 step 1 until inter do
```

```
begin
    t:=ti(m);
    l1:=entier(t);
    if t > = l1+0.5 then
    begin l2:=l1;l1:=l1+1
    end
    else l2:=l1+1;
    if l1>n+1-tr then
    begin sign:=-1;sign2:=(-1)**p
    end
    else
        begin
            sign2:=1;sign:=1
        end;
    der(0):=y(l1);
    mf:=1;
    for q:=1 step 1 until tr do
    begin mf:=mf*q;
        if q ¬ = p then
        begin sum:=sum1:=0.;
            for i:=1 step 1 until tr do
            begin
                l3:=l1+sign*(i-1);
                sum:=sum+c(i,q)*y(l3);
                sum1:=sum1+c(tr+i,q)*dp(l3)
            end;
```

```
            der(q):=(sum+sum1*sign2*pf)/(mf*(sign)**q)
        end
        else der(q):=sign2*dp(l1)/ sign **q
    end;
    if l2>n+1-tr then
    begin sign:=-1;sign2:=(-1)**p
    end
    else
        begin
            sign2:=1;sign:=1
        end;
    sum:=sum1:=0.;
    for i:=1 step 1 until tr do
    begin
        l3:=l2+sign*(i-1);
        sum:=sum+c(i,tr)*y(l3);
        sum1:=sum1+c(tr+i,tr)*dp(l3)
    end;
    d2r:=(sum+sum1*sign2*pf)/(mf* sign **tr);
    der(tr1):=(der(tr)-d2r)/(mf*tr1);
    if l2>l1 then der(tr1):=-der(tr1);
    h2:=t-l1;
    sum1:=1;sum:=0;l1:=tr1-ider;
    for i:=1 step 1 until l1 do
      begin
```

```
                    suml:=1.;ℓ3:=trl-i+2;
                    for q:=1 step 1 until ider do
                       suml:=suml*(ℓ3-q);
                    sum:=(sum+der(ℓ3-1)*suml)*h2
               end;
          sum:=sum+der(ider)*suml/(ider+1);
   ti(m):=sum
   end
end;
```

```
procedure consistency (integer value s,n; real array m(*));
comment  the procedure consistency generates the coefficients in
         the set of equations (3.2-2) using the relations(3.2-32)
         and (3.2-33)and has been written as a procedure since it
         can be usefully employed in generating the non-zero ordinates
         of a B-spline defined on a uniformly distributed set of
         knots (Cox [1973]).
         s  is the order of the derivative in (3.2-2).
         n  is the order of the spline.
         m  is a vector of length n+ 1 (m(0), m(1), ..., m(n))
            containing the coefficients required in the consistency
            equations (3.2-2);
begin integer i,j,mid,cent,sign,nl,k;
     nl:=n-1;
     for i:=1 step 1 until nl do m(i):=0.;
```

```
m(0):=1.;
if s¬= 0 then
begin
      sign:=(if s rem 2 = 0 then 1 else -1);
      for k:=1 step 1 until s do
      begin for i:=k step -1 until 1 do
                m(i):=m(i-1)-m(i);
             m(0):=-m(0)
      end
end
else
      begin
           m(1):=1;
           s:=sign:=1
      end;
if s<n1 then
begin for j:=s+2 step 1 until n do
      begin cent:=mid:=(j-1) div 2;
              for i:=mid step -1 until 1 do
                m(i):=(j-1)*m(i-1)+(i+1)*m(i);
              if j rem 2 ¬= 0 then mid:=mid-1;
              for i:=1 step 1 until mid do
                m(cent+i):=sign*m(mid-i+1)
      end
end;
```

```
    m(nl):=1
end;
```

```
procedure elimination (integer value sign,r;real array b(*,*));
comment procedure elimination takes the matrix system b of
        2r rows and 2r+1 columns and uses Gaussian elimination
        with partial pivoting to reduce a triangular
        portion of size  r-1 by r-1 to zero (upper
        triangular if sign is +1, lower triangular
        if sign is -1).  This puts the coefficient matrix
        in strict band form;
begin integer j2,j3,j6,i,l,j,kl,imodbl,j5,jl;
    real big;
    jl:=(if sign = 1 then r else r+1);
    j2:=jl+sign*(2-r);
    j3:=j2-sign;
    j5:=2*r+1;
    for i:=jl step -sign until j2 do
    begin
        j6:=i+sign*r;
        l:=i;
        imodbl:=i-sign;
        big:=abs(b(i,j6));
        for j:= imodbl step -sign until j3 do
        if abs(b(j,j6))>big then
            begin big:=abs(b(j,j6));l:=j
```

```
            end;
        if l¬=j then
        begin for j:=j5 step -1 until 1 do
            begin big :=b(l,j);b(l,j):=b(i,j);b(i,j):=big
            end
        end;
        if b(i,j6)¬=0 then
        begin
            l:=j6-sign;
            for j:=imodbl step -sign until j3 do
            begin big :=b(j,j6)/b(i,j6);
                for kl:=j3 step sign until l do
                 b(j,kl):=b(j,kl)-big*b(i,kl);
                b(j,j5):=b(j, j5)-big *b(i, j5)
            end
        end
    end
end;
```

```
procedure coef  (integer value p,r; real array b(*,*));
comment  the procedure coef determines the constants involved in
         the multipoint expansions (3.2-1) using a system of equations
         generated by requiring that Equation (3.2-1) be satisfied
         by the 4r independent splines given by the polynomials
         1, x, ..., x^2r and the 2r - 1 cardinal splines
         x_+^(2r+1), (x-1)_+^(2r+1), ..., (x-2r+2)_+^(2r+1) .
```

A standard Gaussian elimination algorithm supplied by the user is used to determine the quantities $a_1, a_2, \ldots, a_{4r}$.

p is the order of the spline derivatives that are determined.

r the spline is of order 2r+1

b a two-dimensional array where 'coef' stores the 4r multi-point expansions. The user-supplied procedure Gauss (r,b,c) solves the matrix system cx=b and returns the solution x in b ;

```
begin integer fr,tr,tr1,tr2,tr3,ifac,i,j,i2,i1;
    real array c(1::4*r,1::4*r);
    fr:=4*r;tr:=2*r;tr1:=tr+1;tr2:=tr+2;tr3:=tr+3;
    for j:=1 step 1 until tr do
    begin i1:=tr+j;c(j,tr1):=c(i1,tr1):=c(i1,1):=
        c(j,1):=c(1,i1):=0;c(1,j):=1;
        for i:=1 step 1 until j do
        begin b(i,j):=b(j,i):=0;
            b(i1,i):=b(tr+i,j):=0
        end
    end;
    ifac:=1;
    for j:=1 step 1 until tr do
    begin i1:=j+1;b(i1,j):=fac;ifac:=ifac*i1
    end;
    ifac:=1;i1:=p-1;
    for i:=1 step 1 until i1 do
```

```
ifac:=i*ifac;
for i:=2 step 1 until tr2 do
begin i1:=i-1;i2=i1-p;
     if p <= i1 then
     begin ifac:=i1*ifac;
            if i2>0 then ifac:=ifac div i2;
          for j:=2 step 1 until tr do
          begin if i2<0 then
               c(i,tr+j):=0.
               else c(i,tr+j):=ifac*(j-1)**i2;
               c(i,j):=(j-1)**(i-1)
          end;
          if i2=0 then
          c(i,tr1):=ifac
     end
     else for j:=tr1 step 1 until fr do
     begin c(i,j-tr):=(j-1-tr)**(i-1),c(i,j):=0.
     end
end;
i1:=2;tr2:=tr1-p;
for i:=tr3 step 1 until fr do
begin for j:=1 step 1 until i1 do
    c(i,tr+j):=c(i,j):=0.;
    i2:=i1+1;
    for j:=i2 step 1 until tr do
    begin c(i,tr+j):=ifac*(j-i1)**tr2;
```

```
            c(i,j):=(j-il)**trl
            end;
            il:=il+1
    end;
    gauss(r,b,c)
end;
```

(iii) Organizational and Notational Details

The coefficient matrix of the system of linear equations for the spline parameters is stored in a matrix of size $n+1$ by $2r+1$ using the transformation $a(i,j) = a(i,j-i)$. The constant vector on the right-hand side of the system of equations is placed in a matrix of size $(n+1)$ by 1 to facilitate input to the published routine <u>bansol 1</u>.

If the boundary conditions are symmetric, then the resulting coefficient matrix is centro-symmetric and efficient routines are developed in Chapter four to solve this type of system. For the non-symmetric case, if processing is performed in an MIMD environment, then the resulting band matrices with the same number of non-zero off-diagonal elements can be solved effectively by the decoupled algorithm (Chapter 4) for polydiagonal systems.

3.3 Smoothing of Periodic Data Sets

3.3.1 Introduction

The fitting of least-squares polynomial splines has been investigated by a number of workers; we particularly cite the work of Ahlberg, Nilson and Walsh [1967], Powell [1967], and Lyche and Schumaker [1971a, 1971b].

We assume the partition $\Pi: x_0 < x_1 < \ldots < x_n$, and we require the periodic function $s(x) \in K^m(x_0, x_n)$ to be determined such that the functional

$$J(s) = \int_{x_0}^{x_n} \left[s^{(m)}(x)\right]^2 dx \tag{3.3-1}$$

be extremal for a function $s(x)$ with $n-1$ derivatives at x_0 and x_n. This leads to the condition that $s(x)$ be a periodic m'th order spline interpolating to the set $\{f_i;\ i = 0, 1, 2, \ldots, n\}$. This result is known as the minimum norm property for the polynomial spline. The proof of this property for the cubic spline is due to Holladay [1957]; the general result is due to Ahlberg et al [1967]. The work by Lyche and Schumaker on the determination of smoothing splines [1971a, 1971b] considers the consequences of using relations between equations (3.3-1) and the weighted deviations

$$E(s) = \sum_{i=0}^{n} w_i\,(s_i - f_i)^2 \tag{3.3-2}$$

to determine the parameters of the m'th order spline $s(x)$. An alternative technique proposed by Powell [1967] relies on finding the function $s(x)$ which makes the integral

$$I(s) = \int_{x_0}^{x_n} \Omega(x)\,\{s(x) - f(x)\}^2\,dx \tag{3.3-3}$$

stationary, and we shall later consider a variant of this procedure. We shall restrict ourselves to the case when the partition Π of $[x_0, x_n]$ is uniform and given by $\{x_i = x_0 + ih;\ i = 0, \pm1, \pm2, \ldots;\ h = (x_n - x_0)/n\}$.

3.3.2 A Smoothing Method

Consider the least squares technique proposed by Powell [1967]. The parameters of $s(x)$ defined on a uniform partition of $[a, b]$ are to be chosen so as to minimize

$$I = \int_a^b \Omega(x)\ [s(x) - y(x)]^2\ dx\ ,$$

where $\Omega(x)$ is non-negative and $y(x)$ is some given function. Then in the case where $\Omega(x) \equiv 1$ and the given function is periodic, more detailed results can be obtained.

Let us consider the integral

$$I = \int_{-\infty}^{\infty} [s(x) - y(x)]^2\ dx\ . \tag{3.3-4}$$

Since $s(x)$ is defined on a uniform partition of $[a, b]$, the partition of $[a, b]$ can be transformed into the integer sequence Π': $\{i,\ i = -[m/2],\ 1 + [m/2],\ \ldots,\ n + [m/2]\}$, and the range on the integral extended. If the sequence $\{y_i\}$ is assumed to define a periodic interpolating polynomial spline $s(x, k-1)$ of odd degree $k-1$ on Π' , it follows from the representation theorem of Schoenberg and Curry [1966] that

$$s(x, k-1) = \sum_{j=-[(k-1)/2]}^{n+[(k-1)/2]} c_j^{(k)}\ Q_k\ (x + k/2 - j) \tag{3.3-5}$$

where the $n+k-1$ quantities $c_j^{(k)}$ are determined by the interpolation condition and, because of periodicity, $c_j^{(k)} = c_p^{(k)}$ when $j \equiv p \bmod n$.

Let the required least squares polynomial spline $s(x)$ be of odd degree $m > k-1$. Then

$$(3.3\text{-}6) \qquad s(x, m) = \sum_{j=-[m/2]}^{n+[m/2]} c_j^{(m+1)} Q_{m+1} (x + (m+1)/2 - j) \quad .$$

The integral (3.3-4) can be written, using Equations (3.3-5) and (3.3-6), as

$$I = \int_{-\infty}^{\infty} [s(x, m) - s(x, k-1)]^2 \, dx$$

and the $n+m$ quantities $c_j^{(m+1)}$ of Equation (3.3-6) can be determined by the condition that $\partial I/\partial c_j^{(m+1)}$ be stationary. Since

$$\frac{\partial I}{\partial c_j^{(m+1)}} = 2 \int_{-\infty}^{\infty} \left\{ Q_{m+1} (x + (m+1)/2 - j) \sum_p c_p^{(m+1)} Q_{m+1} (x + (m+1)/2 - p) \right\} dx$$

$$-2 \int_{-\infty}^{\infty} Q_{m+1} (x + (m+1)/2 - j) \, s(x, k-1) \, dx \ , \ (j = 1, 2, \ldots, n),$$

the stationary condition becomes

$$\sum_p c_p^{(m+1)} \int_{-\infty}^{\infty} Q_{m+1} (x + (m+1)/2 - j) \, Q_{m+1} (x + (m+1)/2 - p) \, dx =$$

$$\int_{-\infty}^{\infty} Q_{m+1} (x + (m+1)/2 - j) \, s(x, k-1) \, dx \ .$$

$$(j = 1, 2, \ldots, n) \ .$$

Now, on substitution from (3.3-5) into the right-hand side of this equation, and use of Theorem 2.3.1, yields

$$(3.3\text{-}7) \qquad \sum_p c_p^{(m+1)} Q_{2m+2} (m+1+j-p) = \sum_p c_p^{(k)} Q_{m+k+1} ((m+1+k)/2+j-p)$$

or, in matrix notation,

$$A_{(2m+2)}\,\underline{c}^{(m+1)} = A_{(m+k+1)}\,\underline{c}^{(k)} \tag{3.3-8}$$

where $A_{(t+1)}$ is the $n\times n$ symmetric circulant $(d_0^{(t+1)}, \ldots, d_{n-1}^{(t+1)})$ with elements given by

$$\begin{aligned} d_i^{(t+1)} &= Q_{t+1}\,((t+1)/2 - i) \qquad (i = 0, 1, \ldots, (\tfrac{t+1}{2}) - 1) \\ &\text{and} \\ d_{n-i}^{(t+1)} &= d_i^{(t+1)} \qquad (i = 1, 2, \ldots, (\tfrac{t+1}{2}) - 1)\,. \end{aligned} \tag{3.3-9}$$

The method thus consists of the following steps.

(i) The order k of an initial interpolating periodic polynomial spline $s(x, k-1)$ of degree $k-1$ is selected.

(ii) The parameters of the spline $s(x, k-1)$ are determined so that $s(x; k-1)$ interpolates to $\{y_i,\ i = 0, 1, \ldots, n\}$ on the partition Π' .

(iii) For given $m \geq k-1$, the polynomial spline $s(x; m)$ of degree m approximating $s(x; k-1)$ in the least squares sense is determined from Equations (3.3-8).

(iv) Step (iii) may be repeated with m replacing k and a new choice for m .

Lemma 3.3.1

Any component of the Fourier Series expansion of $s(x, k-1)$, for example, $q_t\,e^{itx}$, can be expressed (if evaluated at the elements of Π')

in the form

$$q_t\, e^{it\underline{x}} = q_t \sum_{j=1}^{n} \alpha_{j\,t}\, u_j \;,$$

where $u_1, \ldots, u_n$ are the eigenvectors of the general $n \times n$ circulant, $\alpha_{j\,t}$ are unique scalars, and $e^{it\underline{x}}$ is the vector with components $e^{it.0}, e^{it.1}, e^{it.2}, \ldots, e^{it.(n-1)}$.

Proof:

The n eigenvectors are a basis for E_n .

Lemma 3.3.2

The eigenvectors u_j $(j = 2, 3, \ldots n)$ of a circulant can be paired for odd n so that

$$u_j + u_{n-j} = 2 \cos \left[\frac{2\Pi j}{n} (\underline{x})\right]$$

$$u_j - u_{n-j} = 2i \sin \left[\frac{2\Pi j}{n} (\underline{x})\right], \quad j = 2, \ldots, [n/2]$$

and so that, for even n

$$u_{[n/2]} = \cos \Pi\underline{x} \;.$$

Proof:

The result follows from the form of the eigenvectors of a circulant.

Lemma 3.3.3

The periodic polynomial spline $s(x, t)$ defined on an n - point uniform partition can be written as

$$s(\underline{x}, k-1) = \sum_{j=0}^{[n/2]} \left(\alpha_j \left[\cos \frac{2\pi j}{n} \underline{x}\right] + \beta_j \sin \left[\frac{2\pi j}{n} \underline{x}\right]\right)$$

with α_j and β_j scalars.

Proof:

The result follows from Lemmas 3.3.1 and 3.3.2.

The polynomial spline $s(x, m)$ of order $m+1$ defined on an n-point uniform partition as approximating a k'th order polynomial spline $s(x, k-1)$ in the least squares sense smoothes the function $s(x, k-1)$ according to Theorem 3.3.1.

Theorem 3.3.1

If $s(\underline{x}, k-1)$ is given by

$$s(\underline{x}, k-1) = \sum_{j=0}^{[n/2]} \left\{\alpha_j \cos \left(\frac{2\Pi j}{n} \underline{x}\right) + \beta_j \sin \left(\frac{2\Pi j}{n} \underline{x}\right)\right\}$$

and $s(x, m)$ is the least squares approximation obtained from equation (3.3-4), then

$$s(x, m) = \sum_{j=0}^{[n/2]} w_j \left\{\alpha_j \cos \left(\frac{2\Pi j}{n} \underline{x}\right) + \beta_j \sin \left(\frac{2\Pi j}{n} \underline{x}\right)\right\}$$

with the sequence $w_0, w_1, \ldots, w_{[n/2]}$ being a sequence of positive weights such that

$$w_k = \frac{\lambda_k(A_{(m+1)})\, \lambda_k(A_{(m+k+1)})}{\lambda_k(A_{(2m+2)})}$$

where $\lambda_p\, (A_{(\ell)})$ is the p'th eigenvalue of the matrix $A_{(\ell)}$.

Proof:

The result follows from Lemmas 3.3.1 and 3.3.2 and the positive definiteness of $A_{(\ell)}$ (Golomb [1968]).

Corollary

For $m = 3$ and $k = 2$, the sequence $w_0, w_1, \ldots, w_{[n/4]}$ is monotone decreasing whilst the other weights $w_{[n/4]+1}, \ldots, w_{[n/2]}$ are all less than or equal to $w_{[n/4]}$.

A graph of w_k for $m = 3$ and $k = 2$ appears in Figure 3.3.1.

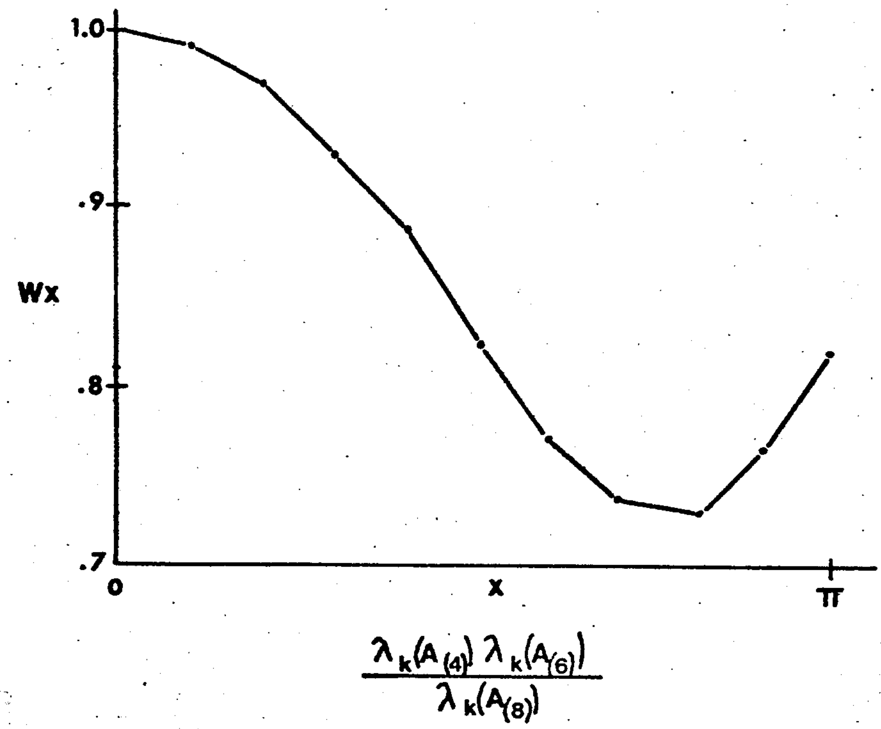


Figure 3.3.1

If $k = 2$, it follows from Equations (3.3-5) and (2.3-1) that $s(x, k-1)$ is a piecewise linear function interpolating to the sequence $\{y_i\}$ at the corresponding elements of Π'. Hence $c_j^{(k)} = y_j$ for all j, and Equation (3.3-8) can be written

(3.3-10) $$A_{(2m+2)} \underline{c}^{(m+1)} = A_{(m+3)} \underline{y}$$

with $\underline{y} = \{y_1, y_2, y_3, \ldots, y_n\}^T$.

The following consistency equation appears in Meek [1974]

(3.3-10b) $$\sum_{k=0}^{m-1} Q_{m+1}^{(r)} (m-k) S_k = \sum_{k=0}^{m-1} Q_{m+1} (m-k) S_k^{(r)} ,$$

and can be rewritten in the form

(3.3-11) $$A_{(2m+2)} \underline{s}^{(r)} = A_{(2m+2)}^{(r)} \underline{y}$$

where $\underline{s}^{(r)} = \left\{s_1^{(r)}, s_2^{(r)}, s_3^{(r)}, \ldots, s_n^{(r)}\right\}^T$ and $A_{(2m+2)}^{(r)}$ denotes the r'th derivative of each element of $A_{(2m+2)}$. Comparison of Equations (3.3-10) and (3.3-11) indicates the similarity for the polynomial spline between the combination of interpolation and least squares technique and direct interpolation. Consider the case $r = m-1$. From (2.3-1) we have that

$$Q_{n+1}(x) = \nabla Q_n(x) .$$

Equations (3.3-10) and (3.3-11) give, since $A_{(2m+2)}$ is nonsingular (Ahlberg et al [1967], Golomb [1968]),

(3.3-12) $$\underline{s}^{(m-1)} = B_{(m-1)} \underline{c}^{(m+1)}$$

where $B_{(m-1)}$ is the singular circulant $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ and

$$\alpha_k = (-1)^{(\frac{m-1}{2})} \binom{m-1}{(m-1)/2+k} \text{ and } \quad \alpha_k = \alpha_{n-k} \quad .$$

Thus the (m-1)th derivatives of the interpolating spline of degree 2m+1 are simply connected with the parameters $\underline{c}^{(m+1)}$ of the least squares polynomial spline of degree m+1 approximating the piecewise linear spline interpolating the given sequence $\{y_i; i = 1, 2, \ldots, n\}$.

An example illustrating the effectiveness of the smoothing appears in Figure 3.3.2, where the data are taken from the function at unit spacing in the range [0, 10] . Both the exact function and the smoothing spline are displayed in this figure,and comparison with the work of Whiten [1972] on smoothing of periodic data sets for non-uniformly spaced points leads to the conclusion that some of the efficiency and economy of the present algorithm could be extended to the case when the number of data points is the same as the number of knots of the spline.

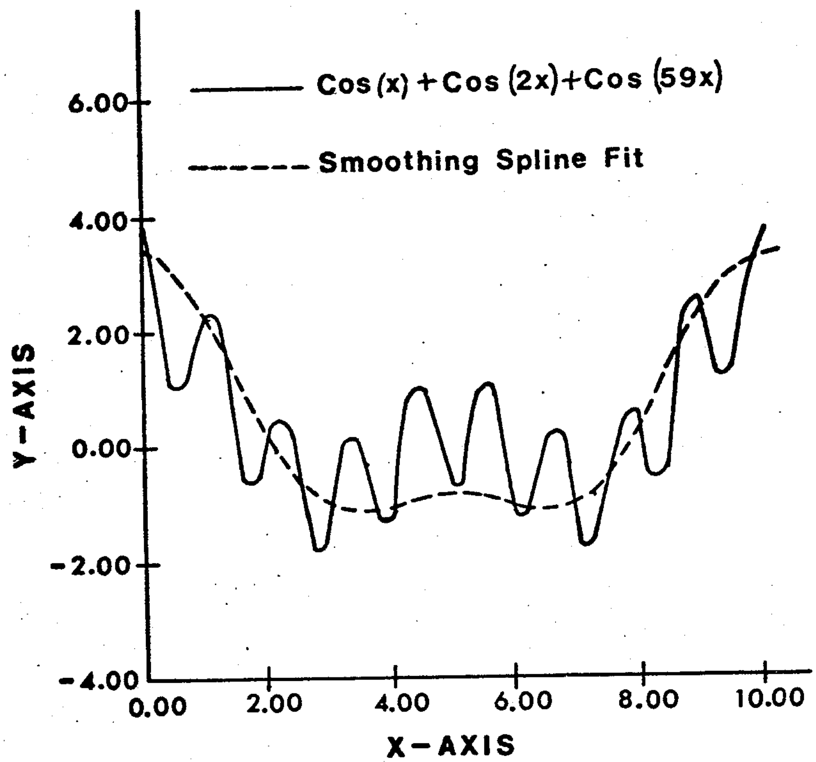


Figure 3.3.2

Smoothed spline fit to the function $\cos x + \cos 2x + \cos 59x$ *with 10 points generated by algorithm smooth given in Section 3.3.3.*

A second example that illustrates the effectiveness of the smoothing method is the application of the technique to an ordered sequence of periodic data points in two-space where a parametric polynomial spline is fitted on each of the co-ordinate directions. The straight line segments give the original figure, the dotted lines the smoothed result.

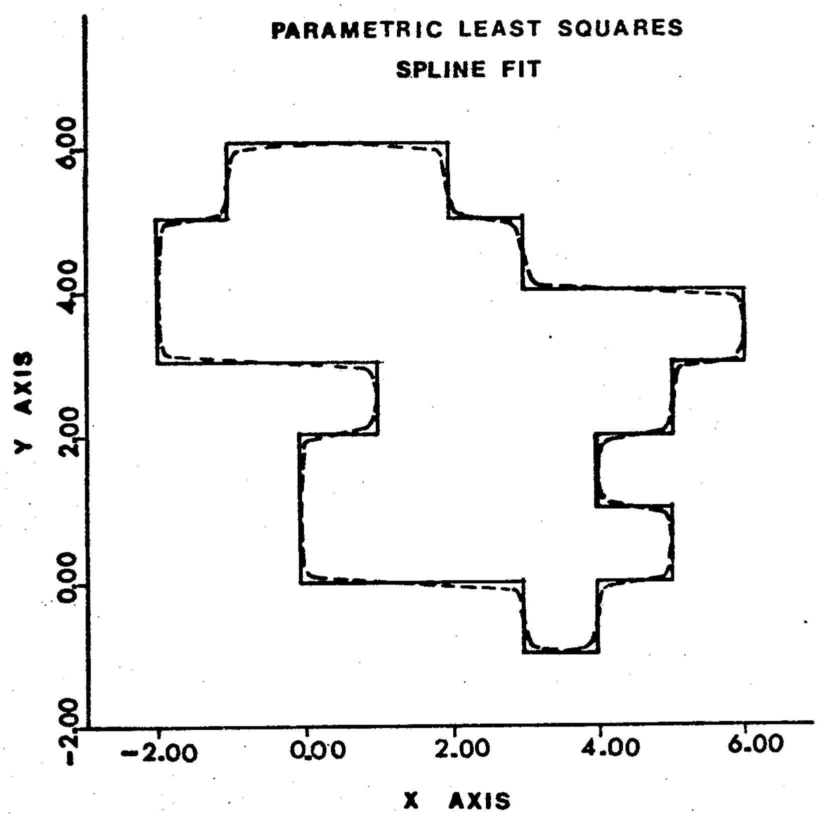


Figure 3.3.3

3.3.3 *An Algorithm for L_2 Polynomial Spline Fitting on Uniformly Spaced Periodic Data Sets.*

The set of equations to be solved for the coefficients of the smoothing polynomial spline (3.3-6) is given by (3.3-8) and (3.3-9).

Attention will be restricted to splines for which t is odd since,for t even and n odd,the inverse of the coefficient matrix does not exist (Ahlberg et al [1967]). The matrix $A_{(2m+2)}$ is positive definite (Albasiny and Hoskins [1972]), and an explicit formula is available for $||A^{-1}_{(2m+2)}||_\infty$.

The positive definiteness of $A_{(2m+2)}$ ensures that Gaussian elimination with no pivotal strategy is stable with respect to the propagation of rounding errors (Wilkinson [1965]). Because the coefficient matrix is band-circulant, an extremely economical method for solving Equations (3.3-8) is to use an extension of the algorithm of Andres et al [1972] given in Hoskins and McMaster [1973].

The algorithm is easily extended to deal with a least squares fit to an ordered sequence of periodic data points in ℓ-space. If the sequence is $z_0, z_1, \ldots, z_n$, and $z_k = \{u_{1,k}, u_{2,k}, \ldots, u_{\ell,k}\}$, then each of the co-ordinate directions can be fitted with a parametric polynomial spline so that the integrals

$$J_\nu = \int_0^n \{u_\nu (t, m) - u_\nu (t, k-1)\}^2 \, dt$$

$$\nu = 1, 2, \ldots, \ell$$

are extremal for all ν , where $u_\nu (t, k-1)$ is the (k-1)th degree parametric polynomial spline fit to the ordered sequence of νth co-ordinate values. Equation (3.3-8), however, is now replaced by the more general form

$$A_{(2m+2)}\, C^{(m)} = A_{(m+k+1)}\, C^{(k)}\,, \tag{3.3-13}$$

where $C^{(j)}$ is the matrix whose columns are $\underline{c}_{\nu}^{(j)}$ $(\nu = 1, 2, \ldots, \ell)$, and the elements of this column vector are the parameters of the polynomial spline $u_{\nu}(t, j)$.

Having determined the parameters $\underline{c}^{(m)}$ of $s(x; m)$ by solving Equations (3.3-8), interpolation is performed using the process due to deBoor [1973] for determining the B-splines in a manner stable with respect to rounding errors. Precise details of this technique are given in Cox [1972] and deBoor [1973]; thus detail sufficient only to describe the practical use of the algorithm is given with the procedure named B-spline.

(i) Formal Parameter List

(a) *Input to procedure smooth:*

n — the number of knots in the partition is n+1 .

parm — the number of dependent variables. This permits smoothing in many variables.

p1 — the degree of the smoothing spline.

p2 — the degree of the original spline fit.

ni — the number of values at which interpolation is performed.

interp — a vector containing the values at which interpolation is to be performed. It has length ni * parm and the first ni values are filled on input.

xy — contains the values for the dependent variables.

t — a vector containing the knots in the partition, t(0), t(1), ..., t(n) .

(b) *Output for procedure smooth:*

interp — a vector containing the smoothed values. The first ni values contain the smoothed values in direction 1 , the second ni values the smoothed values in direction 2 and so on to parm directions.

(ii) Algol W Procedures

```
procedure smooth (integer value n, parm, p1, p2, mo, ni, der; real
            array interp (*);
            real array xy (*,*); real array t(*));
comment:  procedure smooth defines the linear system (3.3-13) using
          the procedure setup to calculate the coefficient matrices
          where m=p1, the order of the smoothing
          spline, and k = p2, the order of the original fit.  To
          effect this, the procedure consistency defined in Section
          3.2 generates for a known value x_t the quantities Q_k (x_t).
          The procedure symmetric circulant is then called by procedure
          setup to solve the linear system (3.3-13).  Finally,
          procedure smooth uses the solution vectors returned by the
          procedure symmetric circulant and further calls of the
          procedure Bspline using the technique of deBoor [1972]
          to return a vector, interp, of ni × parm elements which
          represent interpolated values of the der^th derivative
          evaluated at the ni initial contents of the first
          ni elements of the vector, interp ;
begin integer k,j,kk,ℓ,q,r,m,tr,i; real array temp (1::n);
      real sum,f; real array c(o::2*pi+1); real array z(1::n,1::parm);
comment do a spline fit of degree p2 to the data;
      r:=p2 div 2; m:=2* r+1; ℓ:=(n*(n+1)-(n-m)*(n-m+1) div 2;
      q:=p2; setup (r,q,ℓ,xy);
comment; scale the values in xy by the required factorial.;
      kk:=1;
```

```
for i: = 1 step 1 until p2 do
   kk:=kk*i;
for i:=1 step 1 until n do
   for k:=1 step 1 until parm do
      begin xy(i,k):=xy(i,k)*kk
      end;
comment redefine the values of p1 and p2 so that these values
      correspond to the degree of the splines in the smoothing
      equation rather than the original spline orders ;
   i:=2*p1+2;p2:=p1+p2+2;p1:=i;q:=p2-1;
   consistency (0, q, c);
comment use the cyclic property of the matrix on the right-hand
      side of (3.3-13) to define this vector without doing a full
      matrix multiplication ;
   q:= q div 2;
   for i:=1 step 1 until q do
      begin temp(i):=c(q+i-1); temp (n-i+1):=c(q-i)
      end;
   temp(q+1):=c(2+q);
   for i:=q+2 step 1 until n-q do
      temp(i):=0.;
   for i:=1 step 1 until n do
   begin for j:= 1 step 1 until parm do
         begin sum:=0.;
               for k:= 1 step 1 until n do
                  sum:=sum + xy(k,j)*temp (k);
               z(i,j):=sum
         end;
```

```
            sum:=temp(n);
            for k:=n step - 1 until 2 do
               temp (k): = temp (k-1); temp (1):=sum
        end;
        r:=(p1-1) div 2; m:=2*r+1;
        ℓ:=(n*(n+1)-(n-m)*(n-m+1))div 2;
        setup (r,m,ℓ,z);
    comment: scale the result.;
       kk:=1;
       for i:=p2 step 1 until p1-1 do
          kk:=kk*i;
       for i:=1 step 1 until n do
          for k:= 1 step 1 until parm do
             z(i,k):=z(i,k)*kk;
       mo:=mo+1;
       for i:=1 step 1 until ni do
       begin sum:=interp(i);
           for k:=1 step 1 until parm do
           begin for j:= 1 step 1 until n do
                    temp (j):=z(j,k);
                 bspline (t,n,mo,der,sum,f,temp);
                 interp ((k-1)*ni+i):=f
           end
       end;
```

```
procedure setup (integer value r,q,ℓ; real array z(*,*));
comment the procedure setup defines the system of equations (3.3-13)
     and calls procedure symmetric circulant to solve the
     equation system.
```

r is the dimension of the non-zero triangular corner piece that is in the upper right and the lower left of the circulant matrix.

q is the degree of the spline used.

ℓ is the number of memory locations required by the vector in procedure symmetriccirculant to store the non-zero elements in the coefficient matrix.

z the right-hand sides of the equation systems are passed in z.;

```
begin integer ml; real array v(1::ℓ); real array c(0::q);
    if q > 1 then
    begin consistency (0,q,c);
        tr:=q-1;kk:=1;
        for i:=1 step 1 until n-q do
        begin for j:=r step 1 until tr do
              begin v(kk):=c(j); kk:=kk+1 end;
              for j:= 1 step 1 until i-1 do
              if j < = r then
              begin v(kk):=0.; kk:=kk+1 end;
              for j:=0 step 1 until r-i do
              begin v(kk):=c(j);kk:=kk+1 end
        end;
        for k:=1 step 1 until q do
        begin j:=1;
            while ((j <= (r+1)) and (j < = (q+1-k)))do
            begin v(kk):=c(r+j-1);kk:=kk+1;j:=j+1 end;
            for j:=r step 1 until tr-k do
            begin v(kk):=0.;kk:=kk+1 end
        end;
```

```
            symmetriccirculant (n, r, r, parm, z, v);
        end
end;
procedure bspline (real array t(*)); integer value n, splord, derord;
                  real tt; real intval; real array c(*));
comment performs interpolation for functional values or derivatives
        using the algorithm described in deBoor [1972] for determining
        the B-splines in a manner stable with respect to rounding
        errors.
        n         the number of bspline  coefficients c(1), c(2), ...,
                  c(n).
        splord    the order of the spline used in the interpolation.
        derord    the order of the derivative of the function that
                  is being estimated.
        tt        the point at which interpolation is performed.
        intval    the interpolated value corresponding to tt.
        c         contains the bspline ordinates.
        t         the vector of integer knots, dimensioned as t(-splord
                  + 1::n+splord-1);
begin real array aa(1::splord,a::splord,0::derord);
    real array nn(1::splord-derord, 1::splord-derord);
    integer ℓ, loc, shift, kk, spldeg;
    procedure genn (integer value loc, ss);
    comment loc is the integer value for which t(loc) ≤ tt ≤ t(loc+1).
```

```
begin real array dn(1::ss); real array dp (1::ss);
    integer s, r, ℓ; real m; nn(1,1):=0.;
    for s:=1 step 1 until ss-1 do
    begin dp(s):=t(loc+s)-tt;dm(s):=tt-t(loc+1-s);
        nn(1,s+1):=0.0;
        for r:=1 step 1 until s do
        begin m:=nn(r,s)/(dp(r)+dm(s+1-r));
            nn(r,s+1):=nn(r,s+1)+dp(r)*m;
            nn(r+1,s+1):=dm(s+1-r)*m
        end
    end
end;
spldeg:=splord-1;
comment since the function is periodic, then it is straight-forward
    to define the extended partition.;
for ℓ:=-1 step -1 until -splord +1 do
    t(ℓ):=t(ℓ+1)-t(n+ℓ+1)+t(n+1);
for ℓ:=t(n+ℓ-1)+t(ℓ)-t(ℓ-1);
    t(n+l):= t(n+l-1) + t(l) -t (ℓ-1);
loc:=0;
while (loc < (n+1)) and ((tt-t(loc)) > = 0) do loc:=loc + 1;
loc:=loc-1;
comment map the requisite number (splord) and corresponding B-spline
    ordinates onto the array aa. Using our notation, the B-spline
    is centred over the requisite coefficient and the use of shift
    takes care of the centering ;
    shift:=loc - (splord div 2);
```

```
for kk:=1 step 1 until splord do
begin ℓ:=shift+kk;
comment perform the mapping, use the fact that the function is
        periodic,;
        if ℓ < = 0 then ℓ:=ℓ+n;
        if ℓ > n then ℓ:=ℓ-n;
        aa(kk,0):=c(ℓ)
    end;
comment generate the requisite number of elements in the array aa,
        this number depends on the order of the derivative ;
        shift:=splord-deord;
    for kk:=1 step 1 until derord do
        for ℓ:=1 step 1 until splord-kk do
          aa(splord-ℓ+1,kk):=(aa(splord-ℓ+1,kk-1)-aa(splord-ℓ,kk-1)
          /(t(loc+splord-ℓ-kk+1)-t(loc+1-ℓ));
    genn(loc,shift);intval:=0.0;
    for ℓ:=derord + 1 step 1 until splord do
      intval:=intval + aa (ℓ,derord)*nn (1-derord, shift);
    for ℓ:=1 step 1 until derord do
       intval:=intval*(splord-ℓ)
end;
```

(iii) Testing of the Algorithm Smooth

Testing of this algorithm is most easily done by separating the complete process into two distinct parts, i.e.,

a) the computation of the matrix $c^{(m)}$ in Equation (3.3-8),

b) performing the interpolation for the der^{th} derivative at the n required points.

a) The defining equations for an interpolating parametric polynomial spline of degree $k-1$ on a uniform partition are, in the notation of Equation (3.3-8),

$$A_{(k)} C^{(k-1)} = U$$

with $U = \{\underline{u}_1, \underline{u}_2, \underline{u}_3, \ldots, \underline{u}_n\}$ and $\underline{u}_r = \{u_{r,1}, u_{r,2}, \ldots, u_{r,n}\}^T$. Thus, Equation (3.3-8) can be written in the expanded form

(3.3-14) $$A_{(2m+2)} C^{(m)} = A_{(m+k+1)} A^{-1}_{(k)} U$$

where matrices $A_{(2m+2)}$, $A_{(m+k+1)}$ and $A_{(k)}$ are circulant. However, if $w_k = \exp(2\pi ki/n)$, then the n eigenvectors v_k $(k = 1, 2, \ldots, n)$ of $A_{(s)}$ (s any integer equal to $2r+1$; $r = 1, 2, \ldots$) are given by

(3.3-15) $$v_k = (1, w_k, w_k^2, \ldots, w_k^{n-1})^T$$

and form a basis for E_n. The natural method of establishing that $C^{(m)}$ is computed correctly is to choose the data set U to be the n

eigenvectors v_k , in which case the k'th column in the solution matrix of (3.3-8) is given simply by the product of the k'th eigenvalue of $(A^{-1}_{(2m+2)} A_{(m+k+1)} A^{-1}_{(k)})$ with the k^{th} eigenvector v_k . The eigenvalues can be calculated by use of the formulae given in Golomb [1968] . However, it is sufficient to establish that

(i) the columns of U are given by (3.3-15) and the computed columns of $C^{(m)}$ are simple numerical multiples of the corresponding columns of U;

(ii) the elements in the matrix $A_{(s)}$ are as defined. The latter of these two conditions reduces to establishing that a single row of $A_{(s)}$ is circulant (thus,if one row is correct, then they are all correct). Calculation of any row of $A_{(s)}$ is performed by the <u>procedure consistency</u> (Section 3.2). The method for establishing that the <u>procedure B-spline</u> is correct is a by-product of validating step (b).

(b) Performing interpolation for the der^{th} derivative of s(x, t) is done using

$$\text{(3.3-16)} \qquad \overset{(der)}{s(x,\ t)} \leq \sum_{r=-[\frac{t-1}{2}]}^{n+[\frac{t-1}{2}]} c_r^{(t)}\ Q_{t+1}^{(der)}\ (x + \frac{t+1}{2} - r)$$

(t odd)

and the method described by deBoor [1972]. Verification of the <u>procedure</u> B-spline is most effectively done by using Marsden's identity

$$(u - x)^{k-1} = \sum_i \phi_{i,k}(u)\ Q_k\ (x + \frac{k+1}{2} - i)$$

with

$$\phi_{i,k}(u) = \sum_{r=1}^{k-1} (u - i - r) \quad (\text{all } i) \quad ,$$

and checking for a variety of values of x, u, n, der, and k that Equation (3.3-16) is as exact as rounding errors allow.

If this check is made as complete as possible, no further numerical checks are necessary, and the requirement of step (a) is automatically met.

The numerical values used to produce Figures 3.3.2 and 3.3.3 were obtained using the procedure smooth.

3.4 Cubic Spline Solution to a Class of Second Order Differential Equations

3.4.1 Introduction

It has been remarked by Albasiny and Hoskins [1972] that, for the differential equation

(3.4-1) $$y''(x) + g(x)\, y(x) = r(x)$$

subject to the boundary conditions

(3.4-2)
$$y(0) = y_0 = a$$
and
$$y(1) = y_n = b \quad ,$$

and the corresponding Fredholm integral equation of the second kind

$$y(x) = \int_0^1 k(x, t)\, g(t)\, y(t)\, dt - F(x) , \tag{3.4-3}$$

where

$$k(x, t) = (1 - x)t \qquad 0 \le t \le x$$
$$= (1 - t)x \qquad x < t \le 1$$

and

$$F(x) = \int_0^1 k(x, t)\, r(t)\, dt - (1 - x)\, y_0 - xy_n ,$$

the cubic spline solution to the integral equation is more accurate than the cubic spline approximation to the differential equation (Fyfe [1969], Albasiny and Hoskins [1969]). Albasiny and Hoskins established that, to accuracy $O(h^6)$ (where $h = 1/n$) , the cubic spline approximation to the solution of the integral equation (3.4-3) was identical with a three term difference equation known as the Numerov formula.

We shall establish that the cubic spline approximating $y(x)$ in (3.4-3) is identical in certain cases with the quintic spline solution to the related differential equation (3.4-1) and, in general, this cubic spline is such that its parameters are given by the solution of a set of linear simultaneous equations with a coefficient matrix of band form with bandwidth five. As well, we make an application to general boundary conditions using the multipoint boundary expansions of Section 3.2.

3.4.2 Special Case $g(x) = \alpha$, $r(x) = 0$

In the special case when $g(x)$ is a non-zero constant α , the analysis is particularly straightforward and illustrates the underlying connection with the quintic spline.

The integral equation is

$$y(x) = \alpha \int_0^1 k(x, t)\, y(t)\, dt \tag{3.4-4}$$

with related differential equation

$$y''(x) = -\alpha\, y(x) \quad , \tag{3.4-5}$$

and it is easily seen that this is the example used by Ahlberg, Nilson, and Walsh [1967].

Let the cubic spline $s(x)$ be taken as the approximation to $y(x)$, and further let $x_j = jh$. Then, from Equation (3.4-4),

$$s_j = \alpha \int_0^{x_j} k(x_j, t)\, s(t)\, dt + \alpha \int_{x_j}^1 k(x_j, t)\, s(t)\, dt$$

and hence

$$\delta^2 s_j = \alpha \int_{x_j}^{x_{j+1}} (t - x_{j+1})\, s(t)\, dt + \alpha \int_{x_{j-1}}^{x_j} (x_{j-1} - t)\, s(t)\, dt \quad .$$

This can be expressed as

$$\delta^2 s_j = -\alpha \int_0^h [ts(x_{j+1} - t) + ts(x_{j-1} + t)]\, dt \quad ,$$

and integration by parts produces

$$\begin{aligned} \delta^2 s_j = -\alpha \Big(& \frac{h^2}{2}[s_{j+} + s_{j-}] + \frac{h^3}{6}[s'_{j+} - s'_{j-}] \\ & + \frac{h^4}{24}[s''_{j+} + s''_{j-}] + \frac{h^5}{120}[s'''_{j+} - s'''_{j-}]\Big) + 0(h^6\, s^{(iv)}) \quad . \end{aligned} \tag{3.4-6}$$

For the cubic spline, $s^{(iv)}(x)$ is zero and the first derivatives are continuous; so the above expansion is precisely

$$(3.4\text{-}7) \qquad \delta^2 s_j = -\alpha \left[h^2 s_j + \frac{h^4}{12} s_j'' + \frac{h^4}{120} \delta^2 s_j'' \right]$$

with no truncation of terms introduced by the spline approximation.

One form for the continuity equation of the cubic spline is

$$(3.4\text{-}8) \qquad \delta^2 s_j = h^2 \left(1 + \frac{\delta^2}{6}\right) s_j'' , \quad \text{for } j = 1, 2, 3, \ldots, n-1$$

(cf. Ahlberg et al [1967]).

Equation (3.4-7) can be rearranged in the form

$$\alpha \frac{h^4}{12} s_j'' + \alpha \frac{h^4}{120} \delta^2 s_j'' = -\delta^2 s_j - \alpha h^2 s_j \quad .$$

On using relation (3.4-8), we obtain

$$s_j'' = \frac{-30}{h^4 \alpha} \left(h^2 \alpha + \delta^2 + h^2 \alpha \frac{\delta^2}{20} \right) s_j \quad .$$

This equation allows us to replace equations (3.4-7) and (3.4-8) by the single five-term recurrence relation

$$-\frac{h^4 \alpha}{30} \delta^2 s_j = h^2 \left[1 + \frac{\delta^2}{6} \right] \left[h^2 \alpha + \delta^2 + h^2 \alpha \frac{\delta^2}{20} \right] s_j$$

or, after rearrangement, by

$$(3.4\text{-}9) \qquad \begin{aligned} & s_{j-2} + 2s_{j-1} - 6s_j + 2s_{j+1} + s_{j+2} = \\ & \qquad -\frac{h^2 \alpha}{20} [s_{j-2} + 26s_{j-1} + 66s_j + 26s_{j+1} + s_{j+2}] , \\ & \qquad \qquad \text{for } j = 2, 3, \ldots, n-2 \quad . \end{aligned}$$

This is easily identified as the second continuity equation for the quintic spline

$$s_{j-2} + 2s_{j-1} - 6s_j + 2s_{j+1} + s_{j+2}$$

$$(3.4\text{-}10) \qquad = \frac{h^2}{20}\left[s''_{j+2} + 26s''_{j+1} + 66s''_j + 26s''_{j-1} + s''_{j-2}\right]$$

(cf. Hall [1969]).

This is the example treated by Ahlberg et al [1967] and illustrated with a numerical solution. However, their analysis does not derive Equations (3.4-8) and (3.4-9) or illustrate the simple connection between the cubic spline approximation to (3.4-4) and the quintic spline. The later work of Albasiny and Hoskins [1972], although it does relate the cubic spline solution to Equation (3.4-4) with the Numerov formula, missed deriving the simpler relation (3.4-9) and, instead of using a direct method of determining the quantities s_j, employed an iterative technique on Equation (3.4-7) to obtain both s''_j and s_j $(j = 1, 2, \ldots, n-1)$.

Both previous methods of solving the integral equation (3.4-4) did note that two additional boundary conditions, over and above those of Equation (3.4-2), were needed to determine $s(x)$ uniquely. This can be seen by observing that Equations (3.4-7) and (3.4-8) contain $2n+2$ unknowns, but only give $2n-2$ equations for them. The usual boundary conditions (3.4-2) account for two more conditions, but an extra two must be provided. The same remark applies to Equations (3.4-9); they give $n-3$ equations for the $n+1$ unknowns $s_0, s_1, \ldots, s_n$. The most obvious way of obtaining two further conditions is to use both second derivatives at the boundaries.

3.4.3 A More General Case

The functions $g(x)$ and $r(x) \varepsilon C^3 [0, 1]$ and the cubic spline approximation to Equation (3.4-3) can be written as the solution of the set of equations

$$\left[1 + \frac{h^2}{12} g_{j+1}\right] s_{j+1} - \left[2 - \frac{5}{6} h^2 g_j\right] s_j + \left[1 + \frac{h^2}{12} g_{j-1}\right] s_{j-1} - \frac{h^4}{180} g_j \delta^2 s_j''$$

$$= \frac{h^2}{12}\left[r_{j+1} + 10r_j + r_{j-1}\right] \tag{3.4-11}$$

with a local truncation error of $0(h^6)$ (cf. Albasiny and Hoskins [1972]).

The term in $\delta^2 s_j''$ can be eliminated by using the continuity equation (3.4-8), and the resulting expression is

$$h^2 g_j s_j'' = k_j \tag{3.4-12}$$

where

$$k_j = \left[\frac{h^2}{12}\left(r_{j+1} + 10r_j + r_{j-1}\right) - \left(1 + \frac{h^2}{12} g_{j+1}\right) + \left(2 - \frac{5}{6} h^2 g_j\right) s_j\right.$$

$$\left. - \left(1 + \frac{h^2}{12} g_{j-1}\right) s_{j-1} + \frac{h^2}{30} g_j \ \delta^2 s_j\right] \Big/ \frac{h^2}{30} \ . \tag{3.4-13}$$

From Equation (3.4-8)

$$\delta^2 s_j = \frac{h^2}{6} s_{j+1}'' + \frac{2}{3} h^2 s_j'' + \frac{h^2}{6} s_{j-1}'' \ .$$

Thus

$$g_{j+1}\, g_j\, g_{j-1}\, \delta^2 s_j = \frac{g_j g_{j-1}}{6}\left[h^2 g_{j+1}\, s_{j+1}''\right] + \frac{2}{3} g_{j+1}\, g_{j-1}\left[h^2 g_j s_j''\right]$$

$$+ \frac{g_{j+1} g_j}{6}\left[h^2 g_{j-1}\, s_{j-1}''\right] \ . \tag{3.4-14}$$

On substitution for the terms $h^2 g_{j+1} s''_{j+1}$, $h^2 g_j s''_j$, $h^2 g_{j-1} s''_{j-1}$ from expressions (3.4-12) and (3.4-13), we get the final five term form

$$s_{j-2}\left[g_{j+1}\, g_j\left(1+\frac{h^2}{12}\, g_{j-2}-\frac{h^2}{30}\, g_{j-1}\right)\right]$$

$$+\ s_{j-1}\left[\frac{29}{30}\, h^2\, g_{j+1}\, g_j\, g_{j-1}+4g_{j-1}\, g_{j+1}-2g_{j+1}\, g_j+\frac{h^2}{3}\, g_{j-1}^2\, g_{j+1}\right]$$

$$+\ s_j\left[\frac{47}{15}\, h^2 g_{j+1} g_j g_{j-1}-8g_{j-1} g_{j+1}+g_{j-1} g_j+g_{j+1} g_j+\frac{h^2}{12}\left(g_{j-1} g_j^2+g_{j+1} g_j^2\right)\right]$$

$$+\ s_{j+1}\left[\frac{29}{30}\, h^2\, g_{j-1}\, g_j\, g_{j+1}+4g_{j-1}\, g_{j+1}-2g_{j-1}\, g_j+\frac{h^2}{3}\, g_{j-1}\, g_{j+1}^2\right]$$

$$+\ s_{j+2}\left[g_j\, g_{j-1}\left(1+\frac{h^2}{12}\, g_{j+2}-\frac{h^2}{30}\, g_{j+1}\right)\right]$$

$$=\frac{h^2}{12}\Big(g_{j-1}\, g_j\, r_{j+2}+\left[4g_{j+1}+10g_{j-1}\, g_j\right] r_j+\left[40g_{j-1}\, g_{j+1}\right.$$

$$\left.+\ g_{j-1}\, g_j+g_{j+1}\, g_j\right] r_j+\left[10g_{j+1}\, g_j+4g_{j-1}\, g_{j+1}\right] r_{j-1}+g_j\, g_{j+1}\, r_{j-2}\Big)\ ,$$

(3.4-15) for $j = 2, 3, 4, \ldots, n-2$.

Equations (3.4-15) are an exact representation of the cubic spline approximation to the integral equation if $r^{iv}(x) = 0$, and can be made so if the integral appearing in the definition of $F(x)$ (see Equation (3.4-3)) can be written in closed form.

The form of Equations (3.4-15) leads to a set of simultaneous equations with a coefficient matrix of band width five if all four of the boundary conditions are of a similar form.

3.4.4 Boundary Conditions

So far, the only boundary conditions considered are those given in Equation (3.4-2) and supplemented in Section (3.4.2) by using the differential equation.

When the boundary conditions are of the more general form

$$\begin{aligned} &\alpha y_0 + \beta y_0' = \gamma \\ &\nu y_n + \varepsilon y_n' = \omega \ , \end{aligned} \tag{3.4-16}$$

it is important to observe that, if we use the usual equations for the derivatives of the cubic spline, that is,

$$\begin{cases} s'(x_j+) = \dfrac{s_{j+1} - s_j}{h} - \dfrac{h}{3} s_j'' - \dfrac{h}{6} s_{j+1}'' \\ \text{and} \\ s'(x_j-) = \dfrac{s_j - s_{j-1}}{h} + \dfrac{h}{3} s_j'' + \dfrac{h}{6} s_{j-1}'' \ , \end{cases} \tag{3.4-17}$$

to replace y_0' and y_n' in equations (3.4-16), then the differential equation (3.4-1) no longer produces the most accurate approximate solution. This is easily seen from Equation (3.4-11), since the set of simultaneous equations has a local truncation error of $0(h^6)$ and the two formulae (3.4-17) have a truncation error of $0(h^3)$ (Hoskins [1970]).

Although the effect of errors in the boundary conditions of a polynomial spline can be proved to diminish geometrically away from the boundaries (deBoor [1968]), from the point of view of consistency it is preferable to use approximations to (3.4-16) which are of comparable truncation error to that of Equations (3.4-15).

One method for obtaining the equations for the first derivatives is to use the equations for the first derivative of a quintic spline at a boundary point, where the derivative is expressed as a linear combination of second derivatives and function values at or close to the boundary. Thus, from (3.2-10), we have

$$12hs_0' = -37s_0 + 54s_1 - 9s_2 - 8s_3$$

$$+ \frac{h^2}{120}[-138s_0'' + 2124s_1'' + 1206s_2'' + 48s_3'']$$

(3.4-18) and

$$12hs_n' = 8s_{n-3} + 9s_{n-2} - 54s_{n-1} + 37s_n$$

$$+ \frac{h^2}{120}[-48s_{n-3}'' - 1206s_{n-2}'' - 2124s_{n-1}'' + 138s_n''] \quad .$$

The second derivatives are then replaced by use of the differential equation, and two four-term recurrence relations in the quantities s_j are obtained by substitution in Equation (3.4-16).

Two supplementary equations are needed and there are a number of different possible ways of obtaining them. If the boundary conditions (3.4-16) are of the form (3.4-2), then the differential equation yields explicitly the second derivatives at the boundaries, and Equations (3.4-14) and (3.4-12) give the supplementary equations. The set of equations thus determines the quantities $s_1, s_2, \ldots, s_{n-1}$.

When the boundary conditions are of the more general form (3.4-16), then the two supplementary equations are obtained from (3.4-14), where we use Equation (3.4-1) to replace s''_0 and s''_n and Equation (3.4-12) to replace the remaining second derivatives. We obtain $n+1$ equations in the unknowns $s_0, s_1, \ldots, s_n$.

3.4.5 A Numerical Example

For illustration, we solve numerically the example considered by Ahlberg et al [1967], namely,

$$y(x) = -100 \int_0^1 k(x, t)\, y(t)\, dt \, .$$

In Table (3.4.1), we give the numerical solution and the exact solution $y = \cosh \frac{10(x - .5)}{\cosh 5}$.

x	Exact Solution	Cubic Spline Solution
0	1.000000	1.000000
.1	0.367986	0.368003
.2	0.135664	0.135677
.3	0.050697	0.050703
.4	0.020793	0.020797
.5	0.013475	0.013478

Table 3.4.1

Comparison of the cubic spline approximation and the exact solution of the integral equation ($n = 20$) .

The more general integral equation

$$(3.4\text{-}19) \qquad y(x) = -(x-1) + x\sqrt{e} + \int_0^1 k(x, t)\,(-t^2 - 1)\,y(t)\,dt ,$$

where

$$k(x, t) = (1 - x)t \qquad 0 \le t \le x$$
$$= (1 - t)x \qquad x \le t \le 1 ,$$

related to the differential equation

$$y'' = (x^2 + 1)y$$

with boundary conditions

$$y(0) = 1 ; \quad y(1) = \sqrt{e} ,$$

was solved numerically.

The exact solution is

$$y(x) = \exp\,(x^2/2) ,$$

and is tabulated with the corresponding spline approximation in Table 3.4.2.

x	Spline Solution	Error in Spline Solution
0	1.00000	0.0000000
0.2	1.02020	0.0000005
0.4	1.08329	0.0000008
0.6	1.19722	0.0000010
0.8	1.37713	0.0000008
1.0	1.64872	0.0000000

Table 3.4.2

Comparison of the cubic spline approximation with the exact solution of the integral equation (n = 10) .

Chapter 4

The Solution of Systems of Band Linear Equations Arising from Spline Computation

4.1 Introduction

Gaussian elimination for the solution of linear equations occurs in many algorithmic variants; however, they are all algebraically the same. The methods used differ essentially only in how the matrices are represented to conserve core storage (Wilkinson and Reinsch [1971], Brandon [1973], Pooch and Nieder [1973], Hoskins and McMaster [1974]), the order in which elimination is performed, and the precautions taken against the build-up of large rounding errors in the solution (Wilkinson [1963]). Special variants have been developed for systems with symmetric positive definite matrices in which storage is halved (Forsythe and Moler [1967]).

In subsequent paragraphs, a variant form called a decoupling technique is defined for certain classes of band equations.

This decoupling method enables a reduction in the storage required for the coefficient matrices, significantly reduces the amount of computation in the reduction phase, diminishes the effect of round-off error in the back-substitution process, and increases the speed of computation by a factor approaching two in a MIMD parallel processing environment.

In section 4.2, the decoupling technique is defined for a system of linear equations with a general tridiagonal coefficient matrix. The method is expressed in a modified LU (Forsythe and Moler [1967]) decomposition form and the validity of the decomposition is proved. In Section 4.3, the decoupling technique is applied to a special class of linear equations where the tridiagonal coefficient matrix is symmetric and centrosymmetric. Other current techniques in the literature for the solution of similar special forms of tridiagonal systems are examined and the relative merits are discussed. Also, algorithms for the special tridiagonal systems of equations that arise from cubic spline interpolation with boundary conditions are examined. In Section 4.4, the decoupling technique is extended to general polydiagonal systems of linear equations. Several difficulties arise in the polydiagonal case that are concealed in the simpler tridiagonal case, and several solutions are presented. The decoupling technique is applied to an algorithm of Herriot and Reinsch [1973], in order to demonstrate the economical and practical aspects of the new method.

4.2 General Tridiagonal Systems of Linear Equations and the Decoupling Method

4.2.1 Introduction

A tridiagonal system of n equations, $A\underline{x} = \underline{d}$,

or

$$
\begin{bmatrix}
a_1 & b_1 & & & & & \\
c_1 & a_2 & b_2 & & & & \\
 & c_2 & a_3 & b_3 & & & \\
 & & \cdot & \cdot & \cdot & & \\
 & & & \cdot & \cdot & \cdot & \\
 & & & & \cdot & \cdot & \cdot \\
 & & & & c_{n-2} & a_{n-1} & b_{n-1} \\
 & & & & & c_{n-1} & a_n
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_{n-1} \\ x_n \end{bmatrix}
=
\begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ d_{n-1} \\ d_n \end{bmatrix}
\tag{4.2-1}
$$

occurs rather frequently in the solution of ordinary and partial differential equations (Peaceman and Rachford [1955], Fox [1962], Smith [1965], Buzbee et al [1970]) and cubic spline approximations (Greville [1967], Hoskins [1970], Herriot and Reinsch [1973], Cline [1974], Spath [1974], and merits the production of special algorithms.

4.2.2 Decoupling Method

Generally, the system (4.2-1) is such that $|a_i| \geq |b_i| + |c_{i-1}|$, $i = 1(1)n$, with inequality for at least one i, and $c_o = b_n = 0$; consequently, the coefficient matrix is known to have a bounded inverse (Wilkinson [1965]). The general LU decomposition for A, where L is a lower triangular and U an upper triangular matrix, may be written instead in a decoupled LU decomposition form $L_c U_c$. The forms for the matrices L_c and U_c differ slightly, depending on whether n is odd or even and on the position in which the decoupling is to be effected. For n even, and k some integer in the range $1 < k < n$, the $L_c U_c$ decomposition for A is the product of the matrices L_c and U_c given in (4.2-2).

$$
\begin{pmatrix}
1 &&&&&&&&&&& \\
\ell_1 & 1 &&&&&&&&&& \\
& \cdot & \cdot &&&&&&&&& \\
&& \cdot & \cdot &&&&&&&& \\
&&& \ell_{k-2} & 1 &&&&&&& \\
&&&& \ell_{k-1} & 1 & \ell_k &&&&& \\
&&&&&& 1 & \ell_{k+1} &&&& \\
&&&&&&& \cdot & \cdot &&& \\
&&&&&&&& \cdot & \cdot && \\
&&&&&&&&& 1 & \ell_{n-2} & \\
&&&&&&&&&& 1 & \ell_{n-1} \\
&&&&&&&&&&& 1
\end{pmatrix}
$$

(4.2-2)

$$
\begin{pmatrix}
u_1 & r_1 &&&&&&&&&& \\
& u_2 & r_2 &&&&&&&&& \\
&& \cdot & \cdot &&&&&&&& \\
&&& \cdot & r_{k-2} &&&&&&& \\
&&&& u_{k-1} & r_{k-1} &&&&&& \\
&&&&& u_k &&&&&& \\
&&&&& r_k & u_{k+1} &&&&& \\
&&&&&& r_{k+1} & u_{k+2} &&&& \\
&&&&&&& \cdot & \cdot &&& \\
&&&&&&&& \cdot & \cdot && \\
&&&&&&&&& r_{n-2} & u_{n-1} & \\
&&&&&&&&&& r_{n-1} & u_n
\end{pmatrix}
$$

For n odd, A is the product of the matrices L_c and U_c given in (4.2-3).

$$
\begin{bmatrix}
1 & & & & & & & & & & \\
\ell_1 & 1 & & & & & & & & & \\
 & \cdot & \cdot & & & & & & & & \\
 & & \cdot & \cdot & & & & & & & \\
 & & & \ell_{k-1} & 1 & \ell_k & & & & & \\
 & & & & & 1 & \ell_{k+1} & & & & \\
 & & & & & & 1 & \ell_{k+2} & & & \\
 & & & & & & & \cdot & \cdot & & \\
 & & & & & & & & \cdot & \cdot & \\
 & & & & & & & & & 1 & \ell_{n-1} \\
 & & & & & & & & & & 1
\end{bmatrix}
$$

(4.2-3)

$$
\begin{bmatrix}
u_1 & r_1 & & & & & & & & & \\
 & u_2 & r_2 & & & & & & & & \\
 & & \cdot & \cdot & & & & & & & \\
 & & & \cdot & r_{k-1} & & & & & & \\
 & & & & u_k & & & & & & \\
 & & & & r_k & u_{k+1} & & & & & \\
 & & & & & r_{k+1} & u_{k+2} & & & & \\
 & & & & & & \cdot & \cdot & & & \\
 & & & & & & & \cdot & \cdot & & \\
 & & & & & & & & r_{n-2} & u_{n-1} & \\
 & & & & & & & & & r_{n-1} & u_n
\end{bmatrix}
$$

Definition 4.2-1

The L_c and U_c matrices in (4.2-2) and (4.2-3) will be called tridiagonal *decoupled* matrices since the two elimination patterns decouple at column k and row k .

Definition 4.2-2

The *decoupling parameter* is the value k used to define the tridiagonal decoupled matrices.

The algebraic basis for this variation in the Gaussian elimination method is the following theorem:

Theorem 4.2.1

Assume that a non-singular square tridiagonal matrix A of order n is given. Let A_j denote the principal minor created from the first j rows and first j columns and assume that $\det(A_j) \neq 0$ for $j = 1, 2, \ldots n$. Then given a decoupling parameter k , there exists a unique decoupled tridiagonal matrix L_c and a unique decoupled tridiagonal matrix U_c (as specified in 4.2-2 and 4.2-3) such that $L_c U_c = A$. Moreover, $\det(A) = u_1 u_2 \ldots u_n$.

Proof:

To prove the theorem, we show that is is possible to uniquely determine the elements of L_c and U_c . Consider the case when n is odd which corresponds to (4.2-3). Partition A into submatrices in the same manner that the principal minors do, and in the same manner, partition L_c and U_c . We then proceed down the diagonal with the counter j . If $j = 1$, then clearly $a_1 = 1.u_1$ and u_1 may be determined uniquely. Assume that the elements of L_c and U_c can be determined uniquely for $j = 1$ to $k-1$. For $j = k$, the submatrices may be partitioned in the following manner

$$\left[\begin{array}{cccc|c} 1 & & & & \\ \ell_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & \ell_{k-2} & 1 & \\ \hline & & & & \ell_{k-1} \end{array}\right] \left[\begin{array}{cccc|c} u_1 & r_1 & & & \\ & u_2 & r_2 & & \\ & & \ddots & \ddots & \\ & & & \cdot & r_{k-2} \\ & & & u_{k-1} & r_{k-1} \\ \hline & & & & u_k \end{array}\right] = \left[\begin{array}{cccc|c} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & & \ddots & & \\ & & & b_{k-2} & \\ & & c_{k-2} & a_{k-1} & b_{k-1} \\ \hline & & & c_{k-1} & a_k \end{array}\right]$$

or represented more simply as,

$$\begin{bmatrix} L_{k-1} & 0 \\ m & 1 \end{bmatrix} \begin{bmatrix} U_{k-1} & w \\ 0 & u_k \end{bmatrix} = \begin{bmatrix} A_{k-1,\, k-1} & c \\ r & a_k \end{bmatrix}$$

where m, w, r and c are the appropriate vectors containing k-1 components. The left hand side may be written as $L_k U_k$,

$$\begin{bmatrix} L_{k-1} \cdot U_{k-1} & L_{k-1} \cdot w \\ m \cdot U_{k-1} & mw + u_k \end{bmatrix}$$

By the induction hypothesis, L_{k-1} and U_{k-1} are uniquely determined and $L_{k-1}\, U_{k-1} = A_{k-1}$. Moreover, neither L_{k-1} and U_{k-1} is singular (or else, $A_{k-1,k-1}$ would be singular, contrary to the hypothesis). The requirement $L_k U_k = A$ is equivalent to $L_{k-1}\, w = c$ and $m\, U_{k-1} = r$ and $mw + u_k = a_k$. Thus w, m, and u_k can be determined uniquely in that order and L_k and U_k are determined uniquely. As well,

$$\begin{aligned} \det(A_{k,k}) &= \det(L_k)\, \det\,(U_k) \\ &= 1\, .\, \det(U_{k-1}) .\, u_k \\ &= u_1 .\, u_2 \ldots u_k \end{aligned}$$

If this argument is applied by partitioning the matrices $L_c U_c$ and A from the bottom right as the principal minors do, then a similar argument shows that the ℓ_i, i = n-1(-1)k ; the r_i, i = n-1(-1)k; the u_i, i = n(-1) k are uniquely determined and in particular, the u_i are all non-zero.

This leads to two independent and parallel processes where the u_i, ℓ_i, and r_i are obtained in the following order:

```
u[1]:=a[1]
ℓ[1]:=c[1]/u[1]
for i:=1 step 1 until k-2 do
   begin r[i]:=b[i]
         u[i+1]:=a[i+1]-ℓ[i].r[i]
         ℓ[i+1]:=c[i+1]/u[i+1]
   end
```

```
u[n]:=a[n]
ℓ[n-1]:=b[n-1]/u[n]
for i:=n-2 step -1 until k-1 do
   begin r[i+1]:=c[i+1]
         u[i+1]:=a[i+1]-ℓ[i+1].r[i+1]
         ℓ[i]:=b[i]/u[i+1]
   end
```

The elements of L_c and U_c are thus uniquely determined. In a similar way, the product of the matrices in (4.2-3) is compared element by element with (4.2-1). This again leads to two parallel processes for obtaining the u_i, ℓ_i, and r_i where, if we let m be $[n/2]$, then we have

```
u[1]:=a[1]
ℓ[1]:=c[1]/u[1]
r[1]:=b[1]
for i:= 2 step 1 until m do
   begin
         u[i]:=a[i]-ℓ[i-1].r[i-1]
         ℓ[i]:=c[i]/u[i]
         r[i]:=b[i]
   end
```

```
u[n]:=a[n]
ℓ[n-1]:=b[n-1]/u[n]
r[n-1]:=c[n-1]
for i:=n-1 step -1 until m+2 do
   begin
         u[i]:=a[i]-ℓ[i].r[i]
         ℓ[i-1]:=b[i-1]/u[i]
         r[i-1]:=c[i-1]
   end
```

Lastly u[m+1]:= a[m+1] - ℓ[m].r[m] - ℓ[m+1].r[m+1] .

The proof in the even case proceeds in an analogous manner.

In order to demonstrate the manner in which the elements in $L_c U_c$ decomposition are obtained in practice, we give the defining equations. To obtain the ℓ_i, u_i, and r_i, solve for these elements, successively down each column from the first column to the (k-1)th column. Also, the solution for these elements must proceed from the lower right from the n'th column to the k'th column.

If the determinant of A is represented as $\det A$, then

$$\det A = \det L_c U_c$$

$$= \det L_c . \det U_c .$$

Trivially, $\det L_c = 1$, and $\det U_c = u_1 u_2 \ldots u_n$.

Q.E.D.

Once A has been factorized into the decoupled product $L_c U_c$, then the system $A\underline{x} = \underline{d}$ can be written as

$$L_c U_c \underline{x} = \underline{d} .$$

This represents two tridiagonal decoupled systems

$$L_c \underline{y} = \underline{d}$$

and

$$U_c \underline{x} = \underline{y} ,$$

and these can be readily solved. The components of the intermediate solution $\underline{y}$ can be obtained for n even by two simultaneous back-substitution processes: y_1 and y_n are obtained first, then y_2 and y_{n-1}, and so on to y_{k-1} and y_k.

The components of $\underline{x}$ can be similarly obtained from the second system in the order x_{k-1}, x_k, then x_{k-2}, x_{k+1}, and so on. In the case when n is odd, the components of y are obtained in the order y_1, y_n; y_2, y_{n-1}; and so on; y_k is obtained last. The components of $\underline{x}$ are then obtained in the order x_k, then x_{k-1}, x_{k+1}; x_{k-2}, x_{k+2}; and so on.

Definition 4.2-3

When n is odd or even, it is evident that the operations performed in determining L_c and U_c and in solving for $\underline{y}$ and $\underline{x}$ have been decoupled into two distinct parallel processes; this explains the nomenclature in describing this variation in the Gaussian elimination process as the *decoupling method.*

In the more usual case, when the value of the decoupling parameter k is $[(n+1)/2]$, the decomposition of A into the product L_cU_c and the subsequent solution for $\underline{x}$ involves two parallel processes which could be placed economically in the same program loop, or better still, used in a two-processor MIMD environment to decrease the time required for computation by the MIMD speed-up factor of $(8n + 25)/(4n + 15)$. This value is actually within 1% of 2 for n as small as 60.

Stone [1973b] presents an algorithm for the solution of a general tridiagonal system of linear equations; however, the technique is applicable only to an SIMD system. Additionally, Stone admits to a lack of error analysis for the method.

A further advantage of the decoupling technique is that the solution value x_k is found first; then if desired, values adjacent to it are found. This procedure is especially advantageous if we only require particular solution values as an index of approximation accuracy and for comparison as to consistency with a previous solution of smaller order. Such a requirement might arise in an interactive graphics environment where the effect of a variation in a set of parameters must be examined economically under user control.

The question of rounding errors and their stable propagation in the diagonal entries in the coefficient matrix has already been given in detail in Wilkinson [1963], since the decoupling method may be regarded as the usual Gaussian elimination method obtained by generating a sequence of elementary transformations with an elimination step interspersed with a row-column permutation of the coefficient matrix. The same error analysis applies in the back-substitution phase; however, since the decoupling of the system of equations occurs in the k'th position, the accumulation of round-off error in the back-substitution process in the worst case is over $\max(k, n-k)$ terms rather than the usual n terms.

4.2.3 *Algol-W Procedure 'Generaltrcple' for the Solution of a System of Linear Equations with a General Tridiagonal Coefficient Matrix*

(i) Formal Parameter List

(a) *Input to procedure GENERALTRCPLE*

n	the order of the equation system.
k	the decoupling parameter.
a(1::n)	a vector containing the diagonal elements of the matrix A .
b(1::n-1)	a vector containing the elements on the superdiagonal.
c(1::n-1)	a vector containing the elements on the subdiagonal.
d(1::n)	a vector containing the constant values on the right-hand side of the equation system.
sw	acts as a switch, and if set to 2, returns all the solution values; any other value returns only the k'th value.

(b) *Output of procedure GENERALTRCPLE*

d(1::n)	contains the solution value in d(k) if sw = 2 or the complete n solution values if sw ≠ 2 . The original contents of vectors a, c, and d are destroyed.

(ii) Algol W programme

```
procedure  generaltrcple (integer n,k; real array a(*);real array b(*);
           real array c(*); real array d(*); integer value sw);
comment    solves Ax=d, where A is a general tridiagonal band matrix
           (4.2-1) using a decoupled form for the LU decomposition.;
begin  integer r,kp1,km1,kp2,j;
```

```
real save, savel;
c(1):=c(1)/a(1);kpl:=k+1;kml:=k-1;kp2:=k+2;
save:=d(k);b(n-1):=b(n-1)/a(n);j:=kpl;savel:=d(kpl);
for i:=2 step 1 until k-(n-n  div  2*2) do
  begin a(i):=a(i)-c(i-1)*b(i-1);
        c(i):=c(i)/a(i);
        d(i):=d(i)-c(i-1)*d(i-1)
  end;
  if n div 2 * 2 = n then
    begin a(kpl):=a(kpl)-b(kpl)*c(kpl);b(k):=b(k)/a(kpl);
        d(k):=save;save:=d(kpl);j:=kp2;
        d(kpl):=(c(k)*(d(k)-c(kml)*d(kml))-d(kpl)  + b(kpl)*d(kp2))/
                ((c(k)*b(k)-1.)*a(kpl));
        d(k):=(d(k)-c(kml)*d(kml)-b(k)*(save-b(kpl)*d(kp2)))/
                ((1.-c(k)  *  b(k))*a(k))
    end else
  begin a(k):=a(k)-c(kml)*b(kml)-b(k)*c(k);
        d(k):=(d(k)-d(kml)*c(kml)-d(kpl)*b(k))/a(k)
  end;
comment begin the back substitution;
if sw=2 then
begin for i:=k-1 step -1 until 1 do
        d(i):=(d(i)-b(i)*d(i+1))/a(i);
      for i:=j step 1 until n do
        d(i):=(d(i)-c(i-1)*d(i-1))/a(i)
      end
end;
```

4.3 The Solution of Tridiagonal Systems of Equations with Special Symmetries

4.3.1 Introduction

In the solution of the cubic spline interpolation problem on a uniform partition with suitable boundary conditions (Späth [1974]), a coefficient matrix A in the special form

$$(4.3\text{-}1) \qquad \begin{pmatrix} a & b & & & & & \\ b & a & b & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & b & a & b \\ & & & & & b & a \end{pmatrix}$$

with $a = 4$, $b = 1$, is obtained. As well, this form for A (with a and b other than 4 and 1 and $|a| > 2|b|$) occurs in the numerical solution of certain elliptic partial difference equations over regions involving Dirichlet boundary conditions (Peaceman and Rachford [1955]) . Often it is necessary to solve such tridiagonal systems repeatedly if an iterative process is used.

4.3.2 Examination of Existing Methods

In this section, the solution of systems of equations with coefficient matrices of the form (4.3-1),and those where the coefficient matrix (4.3-1) has the elements $a(1, 1)$ and $a(n, n)$ modified, will be examined.

Recently, several new methods have appeared, notably those of Evans and Forrington [1962], Rose [1969], Atkinson and Evans [1970], Evans [1973], and Malcolm and Palmer [1974] to solve this specialized system of equations. The latter two will be examined in some detail, since the operation counts are lowest of the methods mentioned.

The usual LU decomposition of A requires n-1 divisions, n-1 multiplications, and n-1 additions. The solution of the system LU $\underline{x} = \underline{d}$ requires an additional n divisions, 2n-2 multiplications, and 2n-2 additions for a total of 8n-7 floating point operations required to solve the linear system (Forsythe and Moler [1967]).

The Retrife algorithm (Evans [1973]) represents an improvement over previously published algorithms, and requires the order of 4n additions and 5n multiplications, with only 4n multiplications required if the method is subsequently applied. Close examination of the algorithm (after the statement: <u>array</u> x[1:n]; is corrected to read <u>array</u> x[1:n+1];), however, indicates a serious affect in the implementation of the algorithm. Five major program loops are required with indices running at least from 1 to n-1 .

In each loop pass, the loop index must be incremented and tested, and the testing itself requires time comparable to several floating point additions. It is easily seen that the number of program loops may be cut down to four if the loop containing the statement

sum1: = d[n-1] + alpha × sum1;

is amalgamated with the loop containing the statement

sum2: = d[i+1] + alpha × sum2;

that is, we write

```
sum1: = d[n]; sum2:=d[1];
for i: = 1 step 1 until n-1 do
begin
    sum1: = d[n-1] + alpha × sum1;
    sum2: = d[i+1] + alpha × sum2;
end;
sum1:=alpha ↑ (n+1) × sum1;
```

The remaining statements accompanying the loops may now be included. As well, the statement

```
sum1:=sum1+i × d[i] × (-1)↑ (n-i);
```

sets a sign using the expression $(-1)^{n-i}$. This sign should be set using a switch giving one operation per loop pass instead of n-i . Memory requirement for the vectors in the Retrife algorithm is 2n+1 words of storage.

The Malcolm-Palmer [1974] algorithm requires fewer operations and less storage than the Evans algorithm, and a brief description is given.

The matrix A, after a suitable scaling (each equation is divided by b), may be written in the form :

$$
(4.3\text{-}2)\qquad
\begin{bmatrix}
1 & & & & & \\
\ell_1 & 1 & & & & \\
 & \ell_2 & 1 & & & \\
 & & \cdot & \cdot & & \\
 & & & \cdot & \cdot & \\
 & & & & \cdot & \cdot \\
 & & & & \ell_{n-1} & 1
\end{bmatrix}
\begin{bmatrix}
u_1 & 1 & & & & \\
 & u_2 & 1 & & & \\
 & & \cdot & \cdot & & \\
 & & & \cdot & \cdot & \\
 & & & & \cdot & \cdot \\
 & & & & u_{n-1} & 1 \\
 & & & & & u_n
\end{bmatrix}
$$

The factoring of A into this LU product may be effected using the recurrence relations: $u_1 = a/b$, and

$$\ell_{i-1} = 1/u_{i-1},\quad u_i = a/b - \ell_{i-1}\ ,\qquad i = 1(1)n\ .$$

Under the assumption $|a| > 2|b|$, the ℓ_i and u_i convergewithin machine accuracy and $\ell_k = \ell_{k+1} = \dots = \ell_n = \ell$, $u_{k+1} = \dots = u_n = u$, for some k (a table of values for k are given in Malcolm-Palmer[1974]). However, through an easy manipulation, it may be shown that one need only compute the values ℓ_i , $i = 1, 2, \dots, k$. The LU decompositon may then be written as

$$
(4.3\text{-}3)\qquad
\begin{bmatrix}
1 & & & & & & & \\
\ell_1 & 1 & & & & & & \\
 & \cdot & \cdot & & & & & \\
 & & \cdot & \cdot & & & & \\
 & & & \ell_{k-1} & 1 & & & \\
 & & & & \ell & 1 & & \\
 & & & & & \ell & \cdot & \\
 & & & & & & \ell & \cdot \\
 & & & & & & & \ell \quad 1
\end{bmatrix}
\begin{bmatrix}
u_1 & 1 & & & & & \\
 & u_2 & 1 & & & & \\
 & & \cdot & \cdot & & & \\
 & & & \cdot & \cdot & & \\
 & & & & u_{k-1} & 1 & \\
 & & & & & u & 1 \\
 & & & & & & \cdot \quad \cdot \\
 & & & & & & \cdot \quad 1 \\
 & & & & & & u
\end{bmatrix}
$$

The solution vector $\underline{x}$ for the system of equations $A\underline{x} = \underline{d}$ may then be computed in the usual manner. The implementation of this algorithm requires only three major program loops, reduces the total number of floating point operations to $5n + 2k - 3$, where k is generally much less than n, requires k words of storage for the entire LU decomposition, and saves considerably on array subscripting through the convergence of the ℓ_i to ℓ.

A common variation in the boundary conditions of the spline interpolation problem (Späth [1974]), however, can alter the elements $a(1, 1)$ and $a(n, n)$; in this case, the Evans technique is no longer valid, and the Malcolm-Palmer technique requires that the convergence parameter be recalculated for each set of boundary values. Generally, these modifiedcoefficient matrices possess symmetry and centrosymmetry.

We first consider a decoupled algorithm for the symmetric and centrosymmetric case, and treat the case $a(1, 1) \neq a(n, n)$ separately.

The proposed decoupling method is valid for variations in the spline boundary conditions as well, and has arithmetic operation counts and memory requirements closely approximating the Malcolm-Palmer technique.

Assume that the coefficient matrix takes the form,

$$\text{(4.3-4)} \qquad \begin{bmatrix} a_1 & b_1 & & & & & \\ b_1 & a_2 & b_2 & & & & \\ & b_2 & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & b_2 & \\ & & & & b_2 & a_2 & b_1 \\ & & & & & b_1 & a_1 \end{bmatrix}$$

This includes the special matrices of the form (4.3-1). After suitable scaling, the decoupled L_cU_c decomposition for n even, corresponding to (4.2-2), may be illustrated by the following example where $n = 6$ and the decoupling parameter has a value of 4.

$$
(4.3\text{-}5)\qquad
\begin{bmatrix}
1 & & & & & \\
\ell_1 & 1 & & & & \\
 & \ell_2 & 1 & & & \\
 & & \ell_3 & 1 & \ell_2 & \\
 & & & & 1 & \ell_1 \\
 & & & & & 1
\end{bmatrix}
\begin{bmatrix}
u_1 & 1 & & & & \\
 & u_2 & 1 & & & \\
 & & u_3 & 1 & & \\
 & & & u_3 & & \\
 & & & 1 & u_2 & \\
 & & & & 1 & u_1
\end{bmatrix} .
$$

For the case n odd, corresponding to (4.2-3), and equal to 7, and splitting parameter again 4, the L_cU_c decomposition is:

$$
(4.3\text{-}6)\qquad
\begin{bmatrix}
1 & & & & & & \\
\ell_1 & 1 & & & & & \\
 & \ell_2 & 1 & & & & \\
 & & \ell_3 & 1 & \ell_3 & & \\
 & & & & 1 & \ell_2 & \\
 & & & & & 1 & \ell_1 \\
 & & & & & & 1
\end{bmatrix}
\begin{bmatrix}
u_1 & 1 & & & & & \\
 & u_2 & 1 & & & & \\
 & & u_3 & 1 & & & \\
 & & & u_4 & & & \\
 & & & 1 & u_3 & & \\
 & & & & 1 & u_2 & \\
 & & & & & 1 & u_1
\end{bmatrix} .
$$

Using the centrosymmetric property in both (4.3-5) and (4.3-6), it is obvious that only half the arithmetic operations are required to obtain the L_cU_c decomposition with the decoupling technique. In obtaining the L_cU_c decomposition, there is an obvious dependency of the ℓ_i and u_i so that at most $\ell_1, \ell_2, \ldots, \ell_{[(n+1)/2]}$ need be stored for the entire process. As well, the diagonal elements of L_c and the off-diagonal elements of U_c need not be stored since they are all known to be 1's ; so only $[(n+1)/2]$ words of additional storage are required for the entire L_cU_c decomposition. The total number of floating point operations for the decoupling technique is of the order $6n$. For special matrices of the form (4.3-1) and for $k << n/2$, a comparison of the operation counts for the decoupling and Malcolm-Palmer techniques indicates that the latter is, in this special case, more advantageous.

The decoupling method for the special matrices of the form (4.3-1) appears in Andres, Hoskins, and McMaster [1974]. To illustrate the L_cU_c decomposition for the symmetric and centrosymmetric case (4.3-4) corresponding to (4.2-2) and n even, let $n = 6$ and the decoupling parameter be 4 . We write

$$(4.3\text{-}7)\quad \begin{pmatrix} 1 & & & & & \\ \ell_1 & 1 & & & & \\ & \ell_2 & 1 & \ell_3 & & \\ & & \ell_3 & 1 & \ell_2 & \\ & & & & 1 & \ell_1 \\ & & & & & 1 \end{pmatrix} \begin{pmatrix} u_1 & b_1 & & & & \\ & u_2 & b_2 & & & \\ & & u_3 & b_3 & & \\ & & & u_3 & & \\ & & & b_2 & u_2 & \\ & & & & b_1 & u_1 \end{pmatrix} .$$

Corresponding to (4.2-3) and n odd, let $n = 7$ and the decoupling parameter again be 4 ; then we have

$$(4.3\text{-}8)\qquad \begin{bmatrix} 1 & & & & & & \\ \ell_1 & 1 & & & & & \\ & \ell_2 & 1 & & & & \\ & & \ell_3 & 1 & \ell_3 & & \\ & & & & 1 & \ell_2 & \\ & & & & & 1 & \ell_1 \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} u_1 & b_1 & & & & & \\ & u_2 & b_2 & & & & \\ & & u_3 & b_3 & & & \\ & & & u_4 & & & \\ & & & b_3 & u_3 & & \\ & & & & b_2 & u_2 & \\ & & & & & b_1 & u_1 \end{bmatrix}$$

The Algol-W procedure follows.

4.3.3 *Algol-W Procedure, TRIDIAGDUALSYM for Solving a Tridiagonal System of Linear Equations Possessing Symmetry and Centrosymmetry in the Coefficient Matrix*

In the algorithmic implementation of the decoupled solution using a symmetric and centrosymmetric matrix, n words of computer storage are required for the decomposition.

(i) Formal Parameter List

(a) Input to procedure TRIDIAGDUALSYM

n	the order of the equation system.
a(1::(n+1)<u>div</u>2)	contains the diagonal elements of A .
b(1::(n+1)<u>div</u>2)	contains the off-diagonal elements of A .
d(1::n)	contains the constant values on the right-hand side of the system of equations.

(b) Output of procedure TRIDIAGDUALSYM

d(1::n) contains the solution values. The original contents of the vectors a and d are destroyed.

(ii) Algol-W Programme for TRIDIAGDUALSYM

```
procedure tridiagdualsym (integer value n; real array a(*);
        real array b(*); real array d(*));
comment solves ax=d where a is a tridiagonal, symmetric and
        centrosymmetric band matrix,using a decoupled ℓu
        decomposition. the vector a contains the diagonal elements,
        the vector b, the off diagonal elements ;
begin integer r,k,m; real temp, templ, save; m:=n div 2;
    for r:=1 step 1 until m-2+(n-m+2) do
    begin k:=r+1; temp:=b(r)/a(r);
        a(k):=a(k)-temp*b(r);
        d(k):=d(k) - temp * d(r); k:=n-r;
        d(k):=d(k) - temp * d(k+1);
    end;
    k:=m+1;
    if m+m=n then
    begin temp:=b(m-1)/a(m-1); a(m):=a(m)-temp*b(m-1);
        templ:=b(m)/a(m); save:=d(m);
        d(m):=(d(m)-temp* d(m-1)-templ*(d(k)-temp*d(k+1)))/
            ((1.-templ**2)*a(m));
        d(k):=(d(k)-temp*d(k+1)-templ*(save-temp*d(m-1)))/
            ((1.-templ * * 2) * a(m))
    end else
```

```
begin temp:=b(m)/a(m);a(k):=a(k)-2.*temp*b(m);
        d(k):=(d(k)-temp*(d(m)+d(k+1)))/a(k)
    end
comment complete the back substitution ;
    for r:=n-k step -1 until 1 do
    begin d(r):=(d(r)-b(r) * d(r+1))/a(r);m:=n+1-r;
        d(m):=(d(m)-b(r)*d(m-1))/a(r)
    end
end;
```

In a two-processor MIMD system, the MIMD speed-up factor would be $(13n + 46) / (8n + 26)$, which is within 1% of 1.625 for n larger than 26 .

4.3.4 *A Combined Decoupled and Malcolm-Palmer Algorithm for Solving a Specialized Tridiagonal System of Linear Equations, Mpcoupled*

We assume that the system of equations to be solved has the form $A\underline{x} = \underline{d}$,where A is an $n \times n$ tridiagonal matrix of the form (4.3-1) with superdiagonal and subdiagonal entries equal to one. If the decoupled L_cU_c decomposition is used, then a set of m values ℓ_i must be determined, where $m = (n+1)/2$. However, if the Malcolm-Palmer technique is combined with the decoupled algorithm, then only k values of the ℓ_i (k the convergence factor given in Malcolm-Palmer) are required, and the entire L_cU_c decomposition and back-substitution can proceed as two nearly independent processes. The procedure MPCOUPLED follows.

(i) Formal Parameter List

(a) *Input to procedure MPCOUPLED*

n	the order of the matrix A .
k	the convergence factor specifying the number of iterations required for the ℓ_i to converge to machine precision.
a	the value on the diagonal of the coefficient matrix A .
d	the vector of length n containing the values on the right-hand side of the system of equations.

(b) *Output of procedure MPCOUPLED*

d	contains the n solution values.

(ii) The Algol-W procedure mpcoupled

```
procedure mpcoupled (integer value n,k; real a; real array d(*));
begin integer r,j,m;
real array ℓ(1::k-1); real lim;
ℓ(1):=a; m:=n div 2;
for r:=1 step 1 until k-2 do
begin j:=r+1; ℓ(j):=a-1./ℓ(r);
      d(j):=d(j)-d(r)/ℓ(r);j:=n-r;
      d(j):=d(j)-d(j+1)/ℓ(r)
end;
lim:=a-1./ℓ(k-1);
for r:=k-1 step 1 until m-1 do
   begin j:=r+1; d(j):= d(j)- d(r)/lim;
         j:=n-r; d(j):=d(j)-d(j+1)/lim
   end;
comment begin the back substitution phase.;
j:=m+1; if m+m=n then
begin d(j):=(d(j)*lim-d(m))/(lim*lim-1);
      d(m):=(d(m)-d(j))/lim
end
else d(j):=(d(j)*lim-d(m)-d(j+1))/(a*lim-2);
```

```
    for r:=n-j step -1 until k do
    begin d(r):=(d(r)-d(r+1))/ℓ(r);m:=n+1-r;
          d(m):=(d(m)-d(m-1))/lim
    end;
    for r:=k-1 step -1 until 1 do
    begin d(r):=(d(r) - d(r+1))/ℓ(r);m:=n+1-r;
          d(m):=(d(m)-d(m-1))/ℓ(r);
    end
end;
```

In a uniprocessor system, the procedure mpcoupled combines the better features of both the decoupled technique and the Malcolm-Palmer method. In an MIMD parallel processing system, the MIMD speed-up ratio is $(6n - 6k + 1)/(3n + 2k - 7)$ which closely approximates two for a typical k. As well, the subscripting savings that the Malcolm-Palmer technique provides when using a single processor, would be available on both MIMD processors.

4.3.5 *Tridiagonal Systems of Equations with Symmetry and Near Centrosymmetry in the Coefficient Matrix*

When a cubic spline with prescribed derivatives at the boundaries is fitted to a set of data points, a linear system $A\underline{x} = \underline{d}$ with coefficient matrix A of the form

$$(4.3\text{-}9) \qquad \begin{bmatrix} z & b & & & & & \\ b & a & b & & & & \\ & b & a & b & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & b & a & b \\ & & & & & b & w \end{bmatrix}_{n,n}$$

must be solved (Späth [1974]). This system is symmetric and, save for the elements in position (1, 1) and (n, n), is centrosymmetric.

The Malcolm-Palmer algorithm for the LU decomposition can be shown to converge for most boundary conditions; however, for each set of boundary conditions, a value for the convergence factor c must be determined. The decoupled procedure generaltrcple and a slight modification of tridiagdualsym would effectively solve this equation system; however, full advantage would not be taken of the symmetries present in A.

The *Madison* algorithm, proposed here, exploits several of the economies of the Malcolm-Palmer technique; for sufficiently large n (greater than the convergence factor k), the convergence is independent of the boundary conditions imposed on the spline problem. Algorithm *Madison* utilizes the symmetry and near centrosymmetry of A to conserve storage, and uses the near centrosymmetry to decouple the equation system at some specified value m where $c \le m \le n - c$ (c the convergence factor), and to separate the work into two nearly parallel processes. Approximately 2c+1 words of storage are required for the entire decomposition. A description of the algorithm follows.

A multiple of the m'th row of the system of equations may be added to the remaining rows to attempt to change the coefficient matrix A to upper triangular form for those rows below the m'th row and lower triangular form for those rows above the m'th row. However, this elimination step propagates a non-zero entry in column m+1 in those rows above the m'th row and column m-1 in those rows below the m'th row of the coefficient matrix. The resulting form follows.

$$
\text{(4.3-10)} \qquad
\begin{bmatrix}
s^* &&&&&&&&&&&&&&&&&&&& \\
1 & u &&&&&&&&&&&&&&&&&&& \\
& \cdot & \cdot &&&&&&&&&&&&&&&&&& \\
&& \cdot & \cdot &&&&&&&&&&&&&&&&& \\
&&& 1 & u &&&&&&&&&&&&&&&& \\
&&&& 1 & u_c &&&&&& s_c &&&&&&&&& \\
&&&&& 1 & \cdot &&&&& \cdot &&&&&&&&& \\
&&&&&& \cdot & \cdot &&&& \cdot &&&&&&&&& \\
&&&&&&& 1 & u_2 &&& s_2 &&&&&&&&& \\
&&&&&&&& 1 & u_1 & 0 & s_1 &&&&&&&&& \\
&&&&&&&&& 1 & a & 1 &&&&&&&&& \\
&&&&&&&&& s_1 & 0 & u_1 & 1 &&&&&&&& \\
&&&&&&&&& s_2 &&& u_2 & 1 &&&&&&& \\
&&&&&&&&& \cdot &&&& \cdot & \cdot &&&&&& \\
&&&&&&&&& \cdot &&&&& \cdot & \cdot &&&&& \\
&&&&&&&&& s_c &&&&&& u_c & 1 &&&& \\
&&&&&&&&&&&&&&&& u & 1 &&& \\
&&&&&&&&&&&&&&&&& \cdot & \cdot && \\
&&&&&&&&&&&&&&&&&& \cdot & \cdot & \\
&&&&&&&&&&&&&&&&&&& u & \\
&&&&&&&&&&&&&&&&&&&& w^*
\end{bmatrix}
$$

Notice that the u_i are shown as converging to u (Malcolm and Palmer [1974]) ; simultaneously, the s_i given by

$$s_i = (-1)^i \, s_{i-1}/u_{i-1}$$

are shown as converging to zero after some predetermined value $i = c$. The values $x_1, x_2, \ldots, x_{m-c-1}$; $x_n, x_{n-1}, \ldots, x_{m+c+1}$ may be obtained immediately by back-substitution. The values x_{m-c-1} and x_{m+c+1} may be subtracted from the $(m-c)^{th}$ and $(m+c)^{th}$ equations respectively to leave a smaller system of equations with coefficient matrix in the form

$$(4.3\text{-}11) \qquad \begin{bmatrix}
u_c & & & & & & & s_c & & & & & \\
1 & u_{c-1} & & & & & & \cdot & & & & & \\
 & \cdot & \cdot & & & & & \cdot & & & & & \\
 & & \cdot & \cdot & & & & \cdot & & & & & \\
 & & & 1 & u_2 & & & s_2 & & & & & \\
 & & & & 1 & u_1 & 0 & s_1 & & & & & \\
 & & & & & 1 & a & 1 & & & & & \\
 & & & & & s_1 & 0 & u_1 & 1 & & & & \\
 & & & & & s_2 & & & u_2 & 1 & & & \\
 & & & & & \cdot & & & & \cdot & \cdot & & \\
 & & & & & \cdot & & & & & \cdot & \cdot & \\
 & & & & & \cdot & & & & & & u_{c-1} & 1 \\
 & & & & & s_c & & & & & & & u_c
\end{bmatrix}$$

Using an additional elimination step, we have

$$(4.3\text{-}12)\qquad \begin{pmatrix}
u_c & & & & & & s_c & & & & \\
 & u^*_{c-1} & & & & & s^*_{c-1} & & & & \\
 & & \cdot & & & & \cdot & & & & \\
 & & & \cdot & & & \cdot & & & & \\
 & & & & u^*_1 & 0 & s^*_1 & & & & \\
 & & & & 1 & a & 1 & & & & \\
 & & & & s^*_1 & 0 & u^*_1 & & & & \\
 & & & & \cdot & & & \cdot & & & \\
 & & & & \cdot & & & & \cdot & & \\
 & & & & s^*_{c-1} & & & & & u^*_{c-1} & \\
 & & & & s_c & & & & & & u_c
\end{pmatrix}$$

The two-by-two system with coefficient matrix

$$(4.3\text{-}13)\qquad \begin{pmatrix} u^*_1 & s^*_1 \\ s^*_1 & u^*_1 \end{pmatrix}$$

at the centre is solved first, and the remaining values are obtained by back-substitution.

In a manner analogous to that of Malcolm and Palmer [1974], it is possible to determine an upper bound C after which the u_i have converged to u and the s_i are zero (within machine precision).

For seven figures of accuracy, an upper bound is

$$C = [8 \ln (10)/(\ln (a^2 - 4)^{1/2}) - \ln (2))] .$$

a	C	c
3.	19	17
3.5	16	14
4.0	14	12
5.0	12	12
7.0	10	10
8.0	9	9
9.0	8	8
20.0	6	6
25.0	5	5

Table 4.3.1

Upper bound C and observed values c on the IBM 360.

4.3.6 *Algol-W Procedure Madison for the Solution of the Tridiagonal Systems of Equations Arising from Cubic Spline Interpolation with Specified Boundary Conditions*

(i) Formal Parameter List

(a) *Input to the procedure Madison*

n the order of the equation system to be solved.

k the value after which the u_i in the LU decomposition are equal to u and the s_i are assumed to be zero (from Table 4.3.1).

m the value at which decoupling takes place (generally (n+1) div 2); it must be $\geq k$ or $\leq n-k$.

d the vector of constant values on the right-hand side of the system of equations.

a the value on the main diagonal of the coefficient matrix, except for positions (1, 1) and (n, n) .

z the value in position (1, 1) of the coefficient matrix.

w the value in position (n, n) of the coefficient matrix.

(b) Output from procedure Madison

d the vector of solution values. The original contents of d are lost.

(ii) Algol-W programme for procedure Madison

```
procedure Madison (integer n,k,m; real array d(*); real a,z,w);
comment madison solves a tridiagonal linear system of equations
      αx=d where α has ones on the off-diagonals and the values
      (z,a,...,a,w) on the diagonal. The effect of the boundary
      values z, w,is minimized in the solution process since these
      values are not used until the final stage in the
      elimination process.;
begin real array s(0::n); real array u(0::n);
      real uℓ, alpha, temp, t; integer i,j;
      s(0):=1.; u(0):=a;
      for i:=1 step 1 until k do
      begin s(i):=-s(i-1)/u(i-1);
            u(i):= a-1./u(i-1);
            d(m-i):=d(m-i)-d(m-i+1)/u(i-1);
            d(m+i):=d(m+i)-d(m+i-1)/u(i-1)
      end;
```

```
uℓ:=u(k);
for i:=m-k-1 step -1 until 2 do
    d(i):=d(i)-d(i+1)/uℓ;
for i:=m+k+1 step 1 until n-1 do
    d(i):=d(i)-d(i-1)/uℓ;
d(1):=(d(1)-d(2)/uℓ)/(z-1./uℓ);
d(n):=(d(n)-d(n-1)/uℓ)/(w-1./uℓ);
for i:= 2 step 1 until m-k-1 do
    d(i):=(d(i)-d(i-1))/uℓ;
for i:=n-1 step -1 until m+k+1 do
    d(i):=(d(i)-d(i+1))/uℓ;
d(m-k):=d(m-k)-d(m-k-1);d(m+k):=d(m+k)-d(m+k+1);
for i:=k-1 step -1 until 1 do
begin s(i):=s(i)-s(i+1)/u(i+1);
        d(m-i):=d(m-i)-d(m-i-1)/u(i+1);
        d(m+i):=d(m+i)-d(m+i+1)/u(i+1)
end;
alpha:=-1./(u(1) + s(1)); temp:=(u(1)**2-s(1)**2);
d(m):=(d(m)+alpha*(d(m-1)+d(m+1)))/a;
t:=(d(m-1)*u(1)-s(1)*d(m+1))/temp;
d(m+1):=(u(1)*d(m+1)-s(1)*d(m-1))/temp;
d(m-1):=t; temp:=d(m+1);
for i:=2 step 1 until k do
begin d(m-i):=(d(m-i)-s(i)*temp)/u(i);
        d(m+i):=(d(m+i)-s(i)*t)/u(i)
end
end;
```

The potential application of MIMD parallel processing in procedure Madison is very evident. If the number of iterations after which both the u_i have converged to u and the s_i may be assumed to be zero is some fraction α of the total number of equations, then the MIMD speed-up factor is $(4 + 11\alpha + 27/n)/(2 + 8\alpha + 15/n)$; this value closely approximates two for large n .

4.4 The General Polydiagonal System of Linear Equations

4.4.1 Introduction

Polydiagonal systems with band-width greater than three can be related to higher-order differential equations (Fox [1957]), and polynomial splines (Ahlberg et al. [1967], Greville [1969], Reinsch [1967], Hoskins and McMaster [1974]). The high level of interest in solving systems with this type of coefficient matrix is demonstrated by the extensive literature on the subject (Wilkinson and Reinsch, Vol. II [1971], Herriot and Reinsch [1973], Reid [1970]).

However, the common high speed algorithms for polydiagonal systems of equations that possess special symmetries in the coefficient matrices invariably do not take advantage of centrosymmetry. The decoupling technique,when extended to the polydiagonal case, takes advantage of this additional symmetry to reduce both the amount of computer storage and the number of operations required in the solution.

For general polydiagonal systems, the relaxing of any symmetry requirements leads to a much simplified decoupling algorithm and thus it will not be given here. Such an algorithm however, would be very effective in a parallel processing environment.

In this section, we concentrate on polydiagonal systems where the coefficient matrices are centrosymmetric or Toeplitz and centrosymmetric; in this case, algorithms may be defined that are effective for MIMD parallel processing and are also competitive in a uniprocessing environment.

4.4.2 Extension of the Decoupling Method to the Polydiagonal Case

The application of the decoupling method to the special tridiagonal case conceals some of the difficulties involved when the band-width of the coefficient matrix increases in size. The decoupling technique, in the elimination stage, may be viewed in the following manner: Gaussian elimination is applied commencing at the upper left of the band-matrix, and transforms the upper portion of the band matrix to upper triangular form; simultaneously, Gaussian elimination is applied at the lower right and transforms the lower half of the band matrix to lower triangular form. Since the matrix is centrosymmetric, the two processes are identical to a point. Thus, if the two elimination processes were continued to the centre of the matrix as in the tridiagonal case, then the elements that were manipulated by the two processes would overlap, and the symmetry of the over-all elimination would disappear. If the two elimination processes are discontinued before this interference occurs, then only one elimination process need be performed; this process may be called a *real* elimination, and the other one may be considered a *virtual* elimination. The effect of the virtual elimination must, however, modify the elements on the right-hand side of the system of equations.

Definition 4.4.1

The matrix resulting from the application of two simultaneous elimination processes changing the upper and lower halves of a band matrix A to upper triangular and lower triangular form, respectively, will be referred to as *bitriangular*. A bitriangular form with band width $2r+1$ may be defined in the following manner:

$U = (u_{ij})$ is an $n \times n$ bitriangular matrix such that $u_{ij} = 0$ for $i > j$ and $j \leq [n/2]-r$ or for $i < j$ and $j \geq [n/2] + r$ or for $|i-j| > r$.

The elimination portion of the algorithm for the polynomial case must be divided into two distinct phases: the decoupling section, which reduces the coefficient matrix to bitriangular form; and a *double solve* section which reduces a non-sparse matrix equation to a form permitting the back-substitution to be made. The *double solve* method in the centrosymmetric case utilizes only those elements in the section of the matrix above and including the middle row of the matrix. The decoupling method is unique mathematically; however, the *double solve* technique gives rise to a number of possibilities.

Assume that the given set of equations is $A\underline{x} = \underline{d}$ with

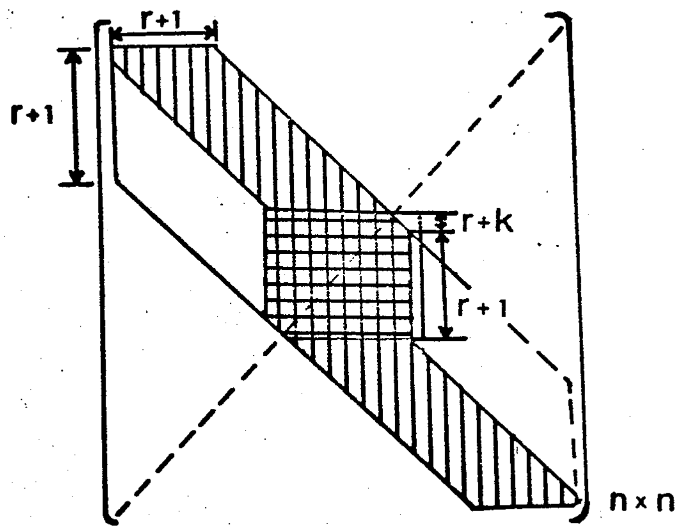
$$a_{ij} = 0 \quad \text{for} \quad |i-j| \geq r+1, \; \underline{x} = \{x_1, x_2, \ldots, x_n\}^T ,$$

and

$$\underline{d} = \{d_1, d_2, \ldots, d_n\}^T .$$

The memory requirements necessary for the decoupling technique when A possesses various symmetry properties will be examined. Application of the decoupling method produces the equivalent matrix equation $\tilde{A}\underline{x} = \underline{d}$, where $\tilde{A}$ is bitriangular and of the form

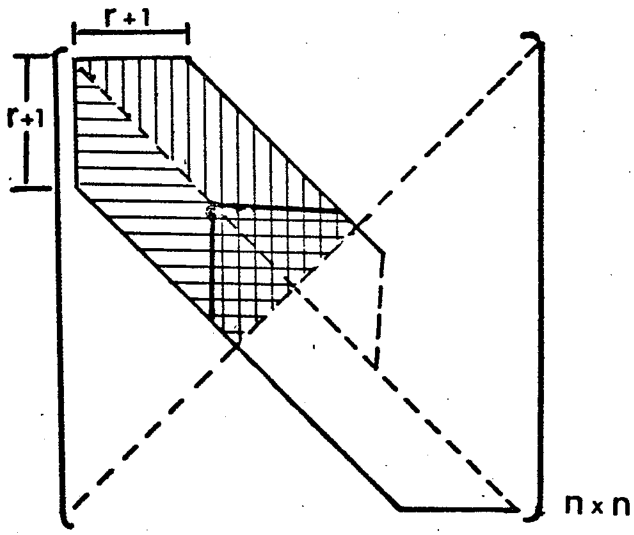
(4.4-1)



Here, $k = n-2\times[n/2]-1$; the shaded area indicates the only non-zero entries in $\tilde{A}$ and the cross-hatched areas contain those elements that remain to be manipulated by *double solve*. The algorithm for this general case is straightforward, and will not be included here.

If A is symmetric, only the elements above and including the main diagonal would be required for Gaussian elimination (Forsythe and Moler [1967], Herriot and Reinsch [1974]) to give the form (4.4-1). If the matrix A were symmetric about the skew diagonal only, then the real and virtual eliminations performed by the splitting technique would give $\tilde{A}$ in the form

(4.4-2)

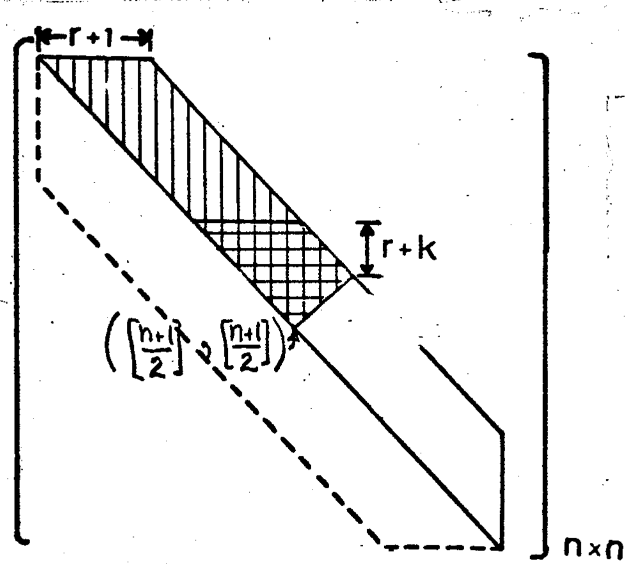


where the horizontal and vertical lines denote the matrix elements used by the method, and the cross-hatched and vertical markings denote the form after the method is applied.

The elimination steps performed below the skew diagonal would be virtual. The cross-hatched area indicates those elements that remain to be manipulated by *double solve* to complete the elimination process.

If A is centro-symmetric (that is, $a_{i,j} = a_{n+1-i,n+1-j}$) and symmetric then the real and virtual eliminations can be performed by considering those elements that lie in the shaded area of (4.4.-3) .

(4.4-3)



Again, the non-sparse section denoted by the cross-hatching can be manipulated by *double solve*. The square coefficient matrix could be obtained by reflecting the cross-hatched area of (4.4-3) about the main diagonal and then by reflecting both parts about the skew diagonal. The decoupling technique leaves this centre portion with the original matrix symmetries; hence, in this case, it is symmetric about both diagonals. Notice that only approximately one-quarter of the elements in A need be stored and manipulated. When the *double solve* elimination is carried out, the centrosymmetry of A must be retained at each stage in the elimination process to ensure the memory saving.

4.4.3 Solution of Linear Systems with Centrosymmetric and Symmetric Centrosymmetric Coefficient Matrices

If we assume that the coefficient matrix of the general polydiagonal system has been reduced to the bitriangular form (4.4-1), then it is still necessary to solve the remaining $m = 2r + k + 1$ equations with coefficient matrix indicated by the cross-hatched pattern. If the original coefficient matrix is centrosymmetric, then the decoupling reduction preserves this symmetry. If as well, the matrix is centrosymmetric and symmetric then this reduction preserves these symmetries. Several techniques which will systematically reduce the order of the equation system further and still preserve centrosymmetry in the remaining portion follow.

The usual Gaussian elimination step adds a multiple of one equation to another to eliminate a variable in the second equation. It is possible, however, to proceed with the following elimination step. Add a linear combination of the first and last equations in the linear system to the middle equations to obtain a square matrix of the form illustrated in the case for 7 equations,

$$
(4.4\text{-}4) \qquad \begin{bmatrix}
x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & & \\
 & x & x & x & x & x & & \\
 & x & x & x & x & x & & \\
 & x & x & x & x & x & & \\
 & x & x & x & x & x & & \\
x & x & x & x & x & x & x &
\end{bmatrix}
$$

This elimination operation preserves centrosymmetry and will be called a centrosymmetric elimination. Such a process may then be repeated on the middle $m = 2$ equations.

If centrosymmetry is used to advantage, only $(m-1)/2$ multiples of the first and last (which may be obtained from the first through centrosymmetry) equations need be determined. An example that illustrates a centrosymmetric elimination step follows.

Assume that a matrix C is of size m by m and assume that elements c_{21} and c_{2m} are to be eliminated (through centrosymmetry $c_{m-1,1}$ and $c_{m-1,n}$ could be assumed to be zeroed).

We first determine an α and a β that satisfies the two by two linear system

$$\alpha\, c_{11} + \beta\, c_{m1} + c_{21} = 0$$

$$\alpha\, c_{1m} + \beta\, c_{mm} + c_{2m} = 0$$

For this equation system to have a solution, the leading principal minors of C must be non-zero since this operation corresponds to Gaussian elimination without pivoting (Forsythe and Moler [1967]). Notice as well that $c_{11} = c_{mm}$ and $c_{1m} = c_{m1}$ through centrosymmetry. Then α times row 1 plus β times row m is added to row 2 and c_{21} and c_{2m} are then zeroed. One step in the elimination process may be expressed in terms of a premultiplication by an elementary operation matrix of the form

$$\begin{bmatrix} 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ \alpha & 1 & 0 & & & & & \beta \\ \cdot & \cdot & \cdot & \cdot & & & & \cdot \\ \cdot & & \cdot & \cdot & \cdot & & & \cdot \\ \cdot & & & \cdot & \cdot & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot & \cdot & \cdot \\ \beta & & & & & \cdot & 1 & \alpha \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & 0 & 1 \end{bmatrix}_{m\times m}$$

where it is assumed that α and β have been determined so that the first and the last elements in rows 2 and $m-1$ will be zeros. This elementary row operation matrix is centrosymmetric, and we prove that, when it premultiplies another centrosymmetric matrix, then the centrosymmetric property is preserved.

Lemma 4.4.1

The product of two centrosymmetric matrices E and G is centrosymmetric.

Proof:

Since E and G are centrosymmetric, then they may be expressed in the form

$$\begin{bmatrix} C & DP \\ PP & PCP \end{bmatrix} \text{ for } n \text{ even, } \begin{bmatrix} A & Pu & BP \\ -v'P & \beta & v' \\ -PB & -u & A \end{bmatrix} \text{ for } n \text{ odd and}$$

where $P^2 = I$ (Andrews [1973]).

It is a simple exercise to show that the product of two matrices of these forms is centrosymmetric.

Q.E.D.

The centrosymmetric reduction process on a matrix A may be viewed as the application of r such elementary row operation matrices

$$E_r, E_{r-1}, \ldots, E_1$$

such that

$$E_r \, E_{r-1} \ldots E_1 A = A^{(r)}$$

where the resultant matrix $A^{(r)}$ is some desired target form. Using this elimination technique, one possible form for the target matrix is an hourglass (illustrated for $m = 7$)

$$\begin{bmatrix} x & x & x & x & x & x & x \\ & x & x & x & x & x & \\ & & x & x & x & & \\ & & & x & & & \\ & & x & x & x & & \\ & x & x & x & x & x & \\ x & x & x & x & x & x & x \end{bmatrix} \qquad (4.4\text{-}5)$$

The back-substitution step in solving the equations can then proceed outwards from the centre by solving sets of two-by-two linear equations (after the centre solution value is obtained).

In the implementation of this algorithm for the reduction of A to hourglass form, only those elements in A that are enclosed by the dotted lines in Figure (4.4-5) are required before the procedure to convert A to hourglass form begins.

In the even case (for example, $m = 6$), the hourglass form has a two-by-two block in the centre:

$$\begin{pmatrix} x & x & x & x & x & x \\ & x & x & x & x & \\ & & x & x & & \\ & & x & x & & \\ & x & x & x & x & \\ x & x & x & x & x & x \end{pmatrix} \tag{4.4-6}$$

These hourglass forms for solving a centrosymmetric system cannot be used to advantage by a MIMD processor since back-substitution could not proceed as two independent processes out from the centre.

A different target form for $\tilde{A}$ may be obtained by varying the elimination pattern used to obtain the hourglass form. If the elimination of the last element in the first row and the first element in the last row as the first elimination in each successive inner square block is effected, then this results in the following form illustrated by the case $m = 7$:

$$
\begin{bmatrix}
x & x & x & x & x & x & \\
 & x & x & x & x & & \\
 & & x & x & & & \\
 & & & x & & & \\
 & & & x & x & & \\
 & & x & x & x & x & \\
 & x & x & x & x & x & x
\end{bmatrix}
\tag{4.4-7}
$$

This form will be called the *extended hourglass* form.

The back substitution process may proceed out from the centre of this system without requiring the repeated solution of two-by-two linear systems as in the hourglass form. However, the back-substitution process is not amenable to an MIMD processing system. The case m even leads to the form illustrated for $m = 6$

$$
\begin{bmatrix}
x & x & x & x & x & \\
 & x & x & x & & \\
 & & x & x & & \\
 & & x & x & & \\
 & & x & x & x & \\
 & x & x & x & x & x
\end{bmatrix}
\tag{4.4-8}
$$

where an extra elimination is required before the back-substitution phase begins.

An extension of the hourglass and extended hourglass methods produces a target form that adapts well to an MIMD processing environment. The elimination sequence that resulted in (4.4-4) is carried further at each stage to produce a matrix of the form

$$(4.4\text{-}9) \qquad \begin{bmatrix} x & x & x & x & & & \\ & x & x & x & & & \\ & & x & x & & & \\ & & & x & & & \\ & & & x & x & & \\ & & & x & x & x & \\ & & & x & x & x & x \end{bmatrix}$$

for m = 7 and to the form

$$(4.4\text{-}10) \qquad \begin{bmatrix} x & x & x & & & \\ & x & x & & & \\ & & x & & & \\ & & & x & & \\ & & & x & x & \\ & & & x & x & x \end{bmatrix}$$

for m = 6 .

This form for the target matrix will be termed a *decoupled bitriangular form*.

Reduction to this form requires three distinct elimination steps as each inner square block is encountered in the elimination process:

(i) reduce the upper right element and lower left element in any inner block to zero.

(ii) reduce the remaining elements in the first column and the remaining elements in the last column to zero by use of a multiple of the first and last rows (a centrosymmetric elimination),

(iii) reduce the remaining elements in the outer block that are in the first and last columns of the block currently being examined.

Step (iii) is not required in the first block. The advantage of the forms (4.4-9) and (4.4-10) is that the back-substitution process may proceed in two independent steps out from the centre, and this method of solution may be implemented to advantage in a two-processor MIMD system. The total operation counts (algebraic) required to solve m equations $(m > 1)$ possessing symmetry and centrosymmetry using the hourglass method and the decoupled bitrangular method follow.

	decoupled bitriangular	hourglass
elimination	$(18m^3+51m^2+10m-129)/24$	$(2m^4+11m^3-13m^2-11m+11)/16$
back substitution	$(m^2+2m-1)/2$	$(2m^2+7m-7)/2$
total	$(18m^3+63m^2+34m-141)/24$	$(2m^4+11m^3+3m^2+45m-45)/16$

Figure 4.4.1 gives a plot of the logarithm to the base ten of the total number of operations, for m in the range 2 to 10, for both the decoupled bitriangular and hourglass methods.

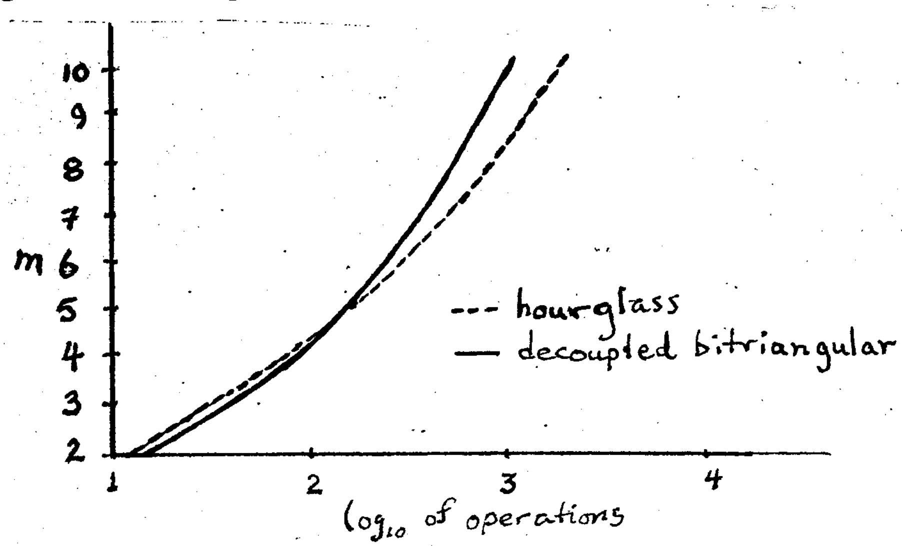


Figure 4.4.1

Normally , m is small since it has approximately the same value as the bandwidth. For $m > 4$, the decoupled bitriangular form is much more advantageous if one only considers operation counts; however, the implementation of the hourglass technique requires less computer memory. Hence, for the solution of the polydiagonal systems resulting from a spline approximation where the bandwidth is normally small (≤ 5) (Cox [1974]), the hourglass method is competitive with the decoupled bitriangular method.

If the coefficient matrix of the polydiagonal system is centrosymmetric only, then the decoupling method (4.4-2) can reduce this matrix to bitriangular form by storing and manipulating approximately one-half of the elements in the coefficient matrix. Once the matrix is in the bitriangular form, then the remaining non-sparse system of equations with coefficient matrix indicated by the cross-hatching may be solved by the hourglass, extended hourglass, and decoupled bitriangular methods. However, if the size of this system is large, then a more economical algorithm exists to solve the system at the centre of (4.4-2).

The Andrew [1973] method requires one-half the computer storage and approximately one-quarter the time required by standard solution methods, and a brief description of the technique follows. Assume that the system to be solved is $R\underline{x} = \underline{y}$, where R is centrosymmetric. An $m \times m$ matrix is centrosymmetric if and only if it can be partitioned into one of the forms

$$(4.4\text{-}11) \qquad \begin{bmatrix} A & BK \\ KB & KAK \end{bmatrix}, \quad \begin{bmatrix} A & Ku & BK \\ v'K & \beta & v' \\ KB & u & KAK \end{bmatrix}$$

depending on whether m is even or odd. A and B are

$[m/2, m/2]$ matrices, u and v' are column and row vectors respectively, β is a scalar, and K is the $m/2$ by $m/2$ skew identity matrix (note that $K^2 = I$). The equations to be solved are then

$$\begin{bmatrix} A & BK \\ KB & KAK \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{4.4-12}$$

or

$$\begin{bmatrix} A & Ku & BK \\ v'K & \beta & v' \\ KB & u & KAK \end{bmatrix} \begin{bmatrix} x_1 \\ \alpha \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ \gamma \\ y_2 \end{bmatrix} \tag{4.4-13}$$

In order to solve (4.4-12), solve

$$(A + B)z_1 = y_1 + Ky_2$$

$$(A - B)z_2 = y_1 - Ky_2$$

and then set

$$x_1 = (z_1 + z_2)/2$$

and

$$x_2 = K(z_1 - z_2)/2 \ .$$

In order to solve (4.4-13), solve

$$(A - B)z_3 = y_1 - Ky_2$$

and

$$\begin{bmatrix} A+B & 2Ku/c \\ cv'K & \beta \end{bmatrix} \begin{bmatrix} z_4 \\ \delta \end{bmatrix} = \begin{bmatrix} y_1+Ky_2 \\ c\gamma \end{bmatrix}$$

where c is a non-zero constant (generally, $\sqrt{2}$ is chosen). The solution is then

$$x_1 = (z_3 + z_4)/2$$

$$x_2 = K(z_4 - z_3)/2 \quad \text{and} \quad \alpha = \delta/c \ .$$

The Andrew algorithm offers a further advantage in an MIMD system in that the solution process has been divided into two independent parts. The back-substitution corresponding to the bitriangular portion of the system can also be accomplished in two independent steps.

If the Andrew algorithm were applied to the complete polydiagonal system, then approximately $(n-2)r(2r+3) + 2r(r+1)+2r(n-2r)+2(n-2)+2+2n$ operations would be required. This is more than twice the number of operations required if the system is first reduced to bitriangular form by the decoupled step and then solved by the Andrew method; in this latter case, only $(n/2-r)r(2r+3) + 4r^3/3 + 3r^2 - r/3-2$ operations are needed.

We now extend the Andrew method to cover a different class of coefficient matrices.

Theorem 4.4.1

If R is an $m \times m$ skew-symmetric and skew-centrosymmetric matrix, then the solution of the system $R\underline{x} = \underline{y}$ may be obtained by solving two systems of size $[m/2]$ by $[m/2]$; if m is an odd integer, the solution of an extra equation is needed.

Proof:

When m is an even integer, R may be partitioned into the form

$$\begin{bmatrix} A & BP \\ -PB & A \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \qquad (4.4\text{-}14)$$

where A is skew-symmetric and skew-centrosymmetric and B is symmetric, and where $P^2 = I$ (P is generally taken to be the skew identity matrix).

Since $PAP = -A$, then (4.4-14) may be written (in order to apply the Andrew method) as

$$\begin{pmatrix} A & BP \\ PB & PAP \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ -y_2 \end{pmatrix} \quad . \tag{4.4-15}$$

The solution is then

$$x_1 = \frac{z_1 + z_2}{2} \quad , \quad x_2 = P(z_1 - z_2)/2$$

where z_1 and z_2 must satisfy

$$(A + B)z_1 = y_1 - Py_2$$
$$(A - B)z_2 = y_1 + Py_2 \quad .$$

Leaf blank to correct numbering.

4.4.4 *Algol-W Procedure for the Solution of a Symmetric and Centrosymmetric Polydiagonal System of Equations*

The first centrosymmetric reduction method outlined in Section 4.4.3, i.e., the hourglass form, was implemented in the *double solve* portion of the algorithm. The algorithm divides naturally into three sections: procedure CSPSOLVE which performs the reduction of A to quasi-bitriangular form, procedure DSPS which solves the non-sparse matrix equation corresponding to the checkered portion of (4.4-3), and procedure COUPLEDPOLY which calls CSPSOLVE and DSPS and then completes the back-substitution process stored in DSPS.

The construction of the algorithm is a non-trivial task because of the limited number of matrix elements accessed, the merging of the real and virtual elimination patterns, and the requirement of maintaining centrosymmetry in the inner matrices. The algorithm for DSPS which solves the equations at the centre of the hourglass form was written for a matrix of size $m \times m$; then the parameters in this portion of the algorithm were transformed so as to manipulate the required elements in the cross-hatched portion of (4.3-3).

(i) Formal Parameter List

(a) Input to procedure COUPLEDPOLY

a the coefficient matrix of dimension [1::(n+1)/2, 1::r+1] .

b the vector of length n on the right-hand side of the system of equations.

n the order of the system of equations.

r   the band width of A is 2r+1 elements.

diml   the integer portion of n+1 divided by 2 .

(b) Output of procedure COUPLEDPOLY

b   the vector of length n containing the solution values.

(ii) Algol-W program

```
procedure coupledpoly (real array a(*,*); real array b(*); integer
        value n, r, diml);
comment  coupledpoly invokes cspsolve to reduce the coefficient
        matrix of the equation system ax=b to quasi-bitriangular
        form, invokes dsps to solve the system at the
        centre of the quasi-bitriangular form, and then completes
        the back-substitution started by dsps.;
begin integer ni, edge; edge:=n div 2-r;
    cspsolve (a, b, n, r, edge);
    dsps (a, b, 2*r+n-2* (n div 2), edge, diml, r+1);
    comment complete the back substitution;
    for i:=edge step -1 until 1 do
    begin ni:=n+1-i;
        for j:=i+1 step 1 until i+r do
        begin b(i):=b(i)-a(i,j-i+1)*b(j);
            b(ni):= b(ni)-a(i,j-i+1)*b(n+1-j)
        end;
```

```
        b(i):=b(i)/a(i,1);
        b(ni):=b(ni)/a(i,1)
      end
   end;
```

```
 procedure dsps (real array a(*,*); real array b(*); integer value n,q,
          dim1, dim2);
 comment this procedure solves the non-sparse system of equations
          that has,as coefficient matrix, the matrix in the cross-
          hatched area in (4.4-3).  The partial solutions to  ax=b
          are returned in the vector  b.;
 begin real z1, z2, z3, s, z, t1, t2, t4, t5;
      integer nq, nq1, ndiv2;
      nq:=n+2*(q+1);nq1:=nq-1;ndiv2:=n div 2;
 comment begin the elimination phase to the hour-glass form.;
 for i:=1+q step 1 until ndiv2+q do
* begin if (nq-2*i) > dim2 then t1:=0 else t1:=a(i,nq-2*i);
     z1:=a(i,1)*a(i,1)-t1*t1;
     for j:=i+1 step 1 until n-ndiv2+q do
*    begin if (nq-i-j) > dim2 then t2:=0 else t2:=a(i,nq-i-j);
          z2:=a(i,1) * a(i,j-i+1) - t1 * t2;
          z3:=-t1*a(i,j-i+1)+a(i,1)*t2;
          z2:=z2/z1;z3:=z3/z1;
          if (nq1-j) ¬= j then b(j):=b(j)-z2*b(i)-z3*b(nq1-i);
          b(nq1-j):=b(nq1-j)-z3*b(i)-z2*b(nq1-i);
```

```
        for k:=i+1 step 1 until n-i + 2*q do
        begin if (k-j+1) < = dim2 then
            begin t3:=a(j,k-j+1);
*               if (k-i+1) > dim2 then t4:=0 else t4:=a(i,k-i+1);
*               if (nq-k-i) > dim2 then t5:=0 else t5:=a(i,nq-k-i);
                a(j,k-j+1):=t3-z2*t4-z3*t5
            end
        end
    end
 end;
 comment begin the back substitution,;
 if ndiv2*2 < n then b(ndiv2+1 + q):=b(ndiv2+1+q)/a(ndiv2+1+q,1);
 for i:=ndiv2+q step -1 until 1+q do
 begin for j:=i+1 step 1 until n-i+2*q do
     begin t3:=0;t4:=0;
*        if (j-i+1) < = dim2 then t4:=a(i,j-i+1);
         b(i):=b(i)-t4*b(j);
*        if (n-i-j+2*(q+1)) < = dim2 then t3:=a(i,n-i-j+2*(q+1)) ;
         b(n+1-i+2*q):=b(n+1-i+2*q)-t3*b(j)
     end;
     nq:=n+2*(q+1-i);
*    if nq > dim2 then t1:=0 else t1:=a(i,nq);
     t2:=a(i,1); z:=t2*t2-t1*t1;
     s:=t2*b(i)-t1*b(nq1-i);b(i):=s/z;
     b(nq1-i):= (t2*b(nq1-i)-t1*b(i))/z
   end
 end;
```

```
procedure cspssolve (real array a(*,*); real array b(*);
                          integer value n,r, edge);
comment this procedure reduces a set of n simultaneous equations
        of the form ax=b so that the coefficient matrix is in the form
        (4.4-3).  a is a band matrix, band width 2r+1, and is
        symmetric and centrosymmetric.  Taking advantage of these
        symmetries permits the coefficient matrix a to be stored in
        int ((n+1)/2) rows and r+1 columns.;
begin integer m; real z;
    for i:=1 step 1 until edge do
    begin m:=i+r; m:=if m < n+1 then m else n;
        for k:=i+1 step 1 until m do
        if k < n+1 then
        begin z:=a(i,k-i+1)/a(i,1);
        b(k):=b(k)-z*b(i);b(n+1-k):=b(n+1-k)-z*b(n+1-i);
        for j:=k step 1 until m do
        a(k,j-k+1):=a(k,j-k+1)-z*a(i,j-i+1)
        end
      end
end;
```

(iii) Organizational and Notational Details

The coefficient matrix A is stored in a matrix of size $[(n+1)/2]$ rows by $r+1$ columns using the transformation

$$a(i, j-i+1) \leftarrow a(i, j)$$

(Forsythe and Moler [1967]). Procedure DSPS solves the equations corresponding to the square centre portion of the bitriangular matrix (4.4-3). The general routine to reduce a matrix to hourglass form accesses elements outside the band width of the original matrix and the allocated memory; hence, when this occurs the element accessed is set to zero. This modification simplifies the procedure loops. To solve a general centrosymmetric system, these simple tests (the lines in which they appear in DSPS are marked with an asterisk) would be removed.

4.4.5 Algol-W Procedure for the Solution of a Centrosymmetric Polydiagonal System of Equations

In this section, a procedure cscoupledpoly is defined which solves a centrosymmetric system of equations $A\underline{x} = \underline{b}$ by reducing A in the manner described in (4.4-2). Also, we extend <u>cscoupledpoly</u> to solve the spline equations arising when a spline of order $2r+1$ is fitted at equally spaced knots and symmetric boundary conditions are imposed at the end points (Chapter 3). This leads to r boundary equations of width $2r$ that apply to the upper and lower part of the coefficient matrix (Hoskins and McMaster [1974]).

(i) Formal Parameter List

(a) Input to procedure CSCOUPLEDPOLY

a matrix containing the elements of the coefficient matrix A . They must be input as described in (iii), which gives organizational and notational details.

b vector on the right-hand side of the system of equation.

n the order of the coefficient matrix a .

r the band width of the equation system is 2r+1 .

sw is a switch indicating the form of the system; if sw = 1 then cscoupledpoly invokes a procedure belim to system to strict band form, otherwise the system is assumed to be in band form initially. .

eps a user supplied value; if, at any time during the elimination process with sw = 1 , a pivot elements if found with a smaller absolute value than eps, procedure Abort is called.

(b) Output of procedure CSCOUPLEDPOLY

b the vector of solution values.

(ii) Algol-W program

```
procedure cscoupledpoly (real array a(*,*); real array b(*);
            integer value n,r,sw; real value eps);
comment if sw=1, then procedure prelim is invoked to convert
          the equation system to strict  band form.;
begin if sw=1 then prelim (a,b,n,r,eps);
      belim (a,b,n,r,eps);
      dcssolve (a,b,n,r,eps)
end;
```

```
      preserved in the centre portion and  is  used  in
      procedure dcsolve.;
  begin integer i,j,k nelim; real quot;
     nelim:=(n-2*r) div 2;
     for i:=1 step 1 until nelim do
     begin
     for j:=i+1 step 1 until i+r do
     begin
        if abs (a(i,1) < = eps then abort (2,eps);
        quot:=a(j,i-j+1)/a(i,1);
        for k:=i+1 step 1 until r+i do
        a(j,k-j+1):=a(j,k-j+1)-quot * a(i,k-i+1);
        b(j):=b(j)-quot*b(i);
        b(n+1-j):=b(n+1-j)-quot*b(n+1-i)
     end
   end
end;
```

```
procedure dcssolve (real array a(*,*); real array b(*);
         integer value n,r; real value eps);
comment dcsolve solves the set of equations that are in the centre
       of the bitriangular form.  Only the elements in the upper
       half of the coefficient matrix are used since the centro-
       symmetric property gives the rest.  dcssolve then performs
       the back-substitution process for the entire equation system.;
begin real alpha, beta, den, t1, t2, t3, t4, t5, z,s;
```

```
procedure prelim (real array a(*,*); real array b(*);
        integer value n,r; real value eps);
comment prelim converts the coefficient matrix to band form.;
begin integer j,rp2,i,ℓ,f; real quot,t;
    begin j:=2*r;rp2:=r+2;f:=0;
        for i:=r step -1 until 2 do
        begin t:=a(i,j-i+1);f:=f+1;
            if abs(t) < = eps then abort (1,eps);
            for ℓ:=1 step 1 until i-1 do
              begin quot:=a(ℓ,rp2-j)/t;
                  for k:=1 step 1 until j-f do
                  begin if (k-ℓ) > r then
                        a(1,rp2-k):=a(ℓ,rp2-k)-quot*a(i,k-i+1)
                  else a(ℓ,k-ℓ+1):=a(ℓ,k-ℓ+1)-quot*a(i,k-i+1)
                  end;
                  b(ℓ):=b(ℓ)-quot*b(i);
                  b(n+1-ℓ);b(n+1-ℓ)-quot*b(n-i+1)
              end;
              j:=j-1
        end
    end
end;
```

```
procedure belim(real array a(*,*); real array b(*);
        integer value n,r; real value eps);
comment belim reduces the         system so that matrix is in
        bitriangular form.  The centrosymmetric property of a is
```

```
integer q, m, nq, dim2;
q:=n div 2 - r; m:=2*r+n-2*(n div 2);
nq:=m+2*(q+1);dim2:= r+1;
for i:=1+q step 1 until (m-1) div 2 + q do
begin if (nq-2*i) > dim2 then t1:=0 else t1:=a(i,nq-2*i);
    if abs (a(i,1) < = eps then abort (3,eps);
    den:=a(i,1)*a(i,1)-t1*t1;
    for j:=i+1 step 1 until m-m div 2+q do
    begin if (nq-i-j) > dim2 then t2:=0. else t2:=a(j,nq-i-j);
          alpha:=(t1*t2-a(i,1)*a(j,i-j+1))/den;
          beta:=(a(j,i-j+1)*t1-a(i,1)*t2)/den;
          if m+1-j+2*q ¬ = j then b(j):=b(j)+alpha*b(i)+
                                          beta*b(m-i+1+2*q);
          b(m+1-j+2*q):=b(m+1-j+2*q)+beta*b(i)
                          + alpha * b(m+1-i+2*q);
          for k:=i+1 step 1 until m-i+2*q do
          begin if (k-j+1) < = dim2 then
                begin if(k-i+1) > dim2 then t4:=0 else t4:
                                     =a(i,k-i+1);
                      if (nq-k-i) > dim2 then t5:=0 else t5:
                                     =a(i,nq-k-i);
                      a(j,k-j+1):=a(j,k-j+1)+alpha*t4+beta*t5
                end
          end
    end
end;
```

```
comment commence the back-substitution.;
if m div 2*2 < m then
b(m div 2+1+q):=b(m div 2+1+q)/a(m div 2 + 1 + q, 1);
for i:=m div 2+q step -1 until 1+q do
begin for j:=i+1 step 1 until m-i+2*q do
      begin t3:=0; t4:=0.;
            if (j-i+1) < = dim2 then t4:= a(i,j-i+1);
            b(i):=b(i) - t4*b(j);
            if (m-i-j+2*(q+1)) < = dim2 then t3:=a(i,m-i-j+2*(q+1));
            b(m+1-i+2*q):=b(m+1-i+2*q)-t3*b(j)
      end;
      nq:= m+2*(q+1-i);
      if nq > dim2 then t1:=0. else t1:=a(i,nq);
      t2:=a(i,1);z:=t2*t2-t1*t1;t3:=b(m+1-i+2*q);
      b(m+1-i+2*q):=(t2*t3-t1*b(i))/z;
      b(i):=(t2*b(i)-t1*t3)/z
end;
comment complete the back-substitution for the coupled stage.;
for i:=n div 2-r step -1 until 1 do
begin for j:=i+1 step 1 until i+r do
      begin b(i):=b(i)-a(i,j-i+1)*b(j);
            b(n+1-i):=b(n+1-i)-a(i,j-i+1)*b(n+1-j)
      end
      b(i):= b(i)/a(i,1);
      b(n+1-i):=b(n+1-i)/a(i,1);
   end
end;
```

(iii) Organizational and Notational Details

The procedure cscoupledpoly requires that the coefficient matrix A of size [(n+1)/2] by 2r+1 be dimensioned as (1 to (n+1)/2,-r+1 to r+1) . This permits the band portion of the coefficient matrix to be stored economically using the transformation $a(i, j-i+1) \leftarrow a(i, j)$ (Forsythe and Moler [1967]). Since the first r and last r equations may have up to 2r entries, some of these elements are outside the band structure; however they may be stored in the given array using the transformation $a(\ell, r+2-k) \leftarrow a(\ell,k)$. The procedure Abort is a user-specified routine.

4.4.6 A Note on CACM Algorithm 472, Procedures for Natural Spline Interpolation

In the calculation of the polynomial spline on a set of equidistant knots and the subsequent procedure NATSPLINEQ, described by Herriot & Reinsch [1973], solution of a symmetric system of equations is accomplished using Gaussian elimination with no pivoting. Although effective use is made of the symmetry of the coefficient matrix, further substantial savings can be made if the *coupledsolve* technique for poly-diagonal equations described in Section 4.4 is also used.

In this case, the Algol statements in procedure NATSPLINEEQ,

"for i:=N1 step 1 until n do A[i,j]:=f;"

may be replaced by

"for i:=N1 step 1 until N1 + ((n-N1+1) DIV 2) do A[i,j]: = f"

and the

statements commencing with "comment Gaussian elimination..." and finishing with the second occurrence of "end i;" can be replaced by the following statements:

```
begin integer nmnl, ne, nms, nlml; q, nq, jl;
      real zl, z2, z3, s, f, tl, t2, t3, t4, t5, z;
      nmnl:=n-m-nl;nlml =nl-l;ne:=n-nlml;ml:=m-l;
      q:=(ne div 2) - ml+nlml;mm:=2*m-l;
      for i:=nl step l until (ne div 2) - m + nl do
      begin ℓ:=i+ml; ℓ:= if ℓ< (n+l) then
           ℓ else n;
           for k:=i+l step l until ℓ do
           if k < (n+l) then
           begin f:=a(i,k-i+l)/a(i,l);
                d(k):=d(k)-f*d(i);jl:=n+l+nlml;
                d(jl-k):=d(jl-k)-f*d(jl-i);
                for j:=k until ℓ do
                    a(k,j-k+l):=a(k,j-k+l)-f*a(i,j-i+l)
              end
           end;
           nms:=mm-l+ne-2*(ne div 2); nq:=nms+2*(q+l);
           for i:=l+q step l until (nms div 2)+q do
           begin if (nq - 2* i) > m then tl:=0. else tl:=a(i,nq-2*i);
                zl:=a(i,l)*a(i,l)-tl*tl;
                for j:=i+l step l until nms - (nms div 2) + q do
                begin if (nq-i-j) > m then t2:=0. else t2:=a(i,nq-i-j);
```

```
z2:=a(i,1)*a(i,j-i+1)-t1*t2;z2:=z2/z1;
z3:=-t1*a(i,j-i+1)+a(i,1)*t2; z3:=z3/z1;
if (nms+1-j+2*q)¬ = j then
d(j):= d(j)-z2*d(i)-z3*d(nms-i+1+2*q);
d(nms+1-j+2*q):=d(nms+1-j+2*q)-z3*d(i)
                    -z2*d(nms+1-i+2*q);
for k:=j until nms-i+2*q do
begin if (k-j+1) < = m then
   begin if (k-j+1) <= m then
      begin t3:=a(j,k-j+1);
         if (k-i+1) > m then t4:=0 else
                 t4:=a(i,k-i+1);
         if (nq-k-i) > m then t5:=0 else
                 t5:=a(i,nq-k-i);
         a(j,k-j+1):=t3-z2*t4-z3*t5
      end
   end
end
end;
comment begin the back-substitution phase.;
if nms div 2*2 < nms
then d(nms div 2+1+q):= d(nms div 2+1+q)/a(nms div 2+1+q,1);
for i:=nms div 2+q step -1 until 1+q do
begin for j:=i+1 step 1 until nms - i + 2*q do
```

```
            begin t3:=0.; t4:=0.;
                  if (j-i+1) < = m then t4:=a(i,j-i+1);
                  d(i):=d(i)-t4*d(j);
                  if (nms-j-i+2* (q+1))<= m then t3:=a(i,nms-i-j+2*(q+1));
                  d(nms+1-i+2*q):=d(nms+1-i+2*q)-t3*d(j)
            end;
            nq:=nms + 2*(q+1-i); t1:=0.; t2:=a(i,1);
            if nq < = m then t1:=a(i,nq);
            z:=t2*t2-t1*t1; s:=t2*d(i)-t1*d(nms+1-i+2*q);
            d(nms+1-i+2*q):=(t2*d(nms+1-i+2*q)-t1 * d(i))/z;
            d(i):=s/z
       end;
       for i:=q step -1 until n1 do
       begin for j:=i+1 step 1 until i+m1 do
            begin d(i):=d(i)-a(i,j-i+1)*d(j);
                  d(n+1-i+n1m1):=d(n+1-i+n1m1)-a(i,j-i+1)*d(n+1-j+n1m1)
            end;
            d(i):=d(i)/a(i,1);
            d(n+1-i+n1m1):=d(n+1-i+n1m1)/a(i,1)
    end
end;
```

(iii) Organizational and Notational Details

The subscripting technique used in storing the coefficient matrix is described in detail in Herriot and Reinsch [1973]. Through the use of the coupled technique presented here, only the upper half of the coefficient matrix need be specified, and the appropriate modification to the Herriot and Reinsch algorithm can be easily made in the specification step.

Chapter 5

Properties of Some Classes of Band Matrices Arising in Spline Computation

5.1 *Introduction*

In the application of splines to curve fitting, polynomial splines of degree three(or perhaps five)are most often used (Cox [1975]). In this chapter, we obtain useful properties of the coefficient matrices of the resulting systems of equations for the spline paramaters.

In Section 5.2, the region where the tridiagonal coefficient matrix resulting from a cubic spline fit with specified boundary conditions is positive definite is determined. Hence, it may be calculated *a priori* whether given boundary conditions will result in the parameters of the spline being determined in a manner stable with respect to rounding errors (Wilkinson [1965]). In Section 5.3, the symmetric Toeplitz quindiagonal matrix arising from a quintic polynomial spline fit with natural boundary conditions is examined. Such matrices occur in a variety of contexts (Stanton and Sprott [1962], Varga [1962], Fox [1962]), and have most recently been examined by Hoskins and Ponzo [1972], and Hoskins and McMaster [1975]. In particular, the region where the elements of the inverse alternate in sign is determined. Application of a simple similarity transformation to the inverse results in a positive matrix form. As well, a result permitting the economical evaluation of the infinity norm of the inverse (useful in obtaining error estimates for the solution of systems with this coefficient matrix (Wilkinson [1965])) is obtained. In Section 5.4, a matrix form arising from a third order finite difference approximation is examined, and the region where the elements of the inverse alternate in sign is obtained.

5.2 Analysis of the Coefficient Matrix of the Tridiagonal System of Equations Arising from a Cubic Spline Fit on Equally-spaced Knots with Specified Boundary Conditions

5.2.1 Introduction

The extended Malcolm-Palmer technique, the coupled technique, the coupled procedures Generaltrcple and Tridiagdualsym (slightly modified), and the procedure Madison of Chapter 4 can be used to solve the system having, as coefficient matrix, the form (4.3-9). These methods are variations on the standard Gaussian elimination algorithm, and are known to be stable for coefficient matrices that are positive definite (Wilkinson [1965]). Hence, the following question may be posed: for what values of z and w is the matrix form (4.3-9) positive definite (if it is assumed that the same matrix with the value 'a' replacing both z and w is positive definite)? The Madison algorithm, by initiating the elimination stage from the centre of the system, minimizes the effect of the boundary conditions on the accuracy of the solution. However, in order to guarantee a stable solution, those regions where the matrix (4.3-9) is positive definite must be determined.

5.2.2 Derivation of the Result

Assume that the system to be solved is $A\underline{x} = \underline{d}$, where A is described in (4.3-9) with $b = 1$. We determine the values for z and w that ensure that A is positive definite (if A were negative definite, then $(-A)\underline{x} = -d$ would be a positive definite case).

For A to be positive definite, the leading principal minors must be positive, and we immediately have that

$$a,\ z,\ w > 0 \quad \text{and}$$

$$az > 1$$

$$aw > 1\ .$$

However, these are not the most stringent conditions on z and w.

We first define $A_n(a, z, w)$ to be the determinant of the matrix given in (4.3-9), $A_n(a, w)$ as the determinant of the following matrix

$$\begin{pmatrix} a & 1 & & & & & \\ 1 & a & 1 & & & \cdot\cdot & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & 1 & a & 1 \\ & & & & & 1 & w \end{pmatrix}_{n\times n} ,$$

and $A_n = A_n(a, a)$.

Evaluating $A_n(a, z, w)$, by cofactors gives

$$A_n(a, z, w) = z\, A_{n-1}(a, w) - A_{n-2}(a, w) \quad .$$

Evaluating $A_n(a, w)$ in a similar manner leads to

$$A_n(a, w) = w\, A_{n-1} - A_{n-2} \quad .$$

Hence

$$\text{(5.2-1)} \qquad \begin{aligned} A_n(a, z, w) &= z(w\, A_{n-2} - A_{n-3}) - (w\, A_{n-3} - A_{n-4}) \\ &= z\, w\, A_{n-2} - (z+w)\, A_{n-3} + A_{n-4} \ . \end{aligned}$$

For A to be positive definite,

(5.2-2) $$A_r(a, w) > 0 \qquad r = 1\ (1)\ n-1$$

(5.2-3) $$A_r(a, z, w) > 0 \qquad r = 1\ (1)\ n\ .$$

Since

$$A_r(a, w) = w\ A_{r-1} - A_{r-2}\ ,$$

then (5.2-2) gives

$$w\ A_{r-1} - A_{r-2} > 0 \quad \text{for}\ r = 1\ (1)\ n-1\ .$$

Thus,

(5.2-4) $$w > \frac{A_{r-2}}{A_{r-1}}\ , \quad r = 2\ (1)\ n-1\ .$$

In a similar manner

(5.2-5) $$z > \frac{A_{r-2}}{A_{r-1}}\ , \quad r = 2\ (1)\ n-1\ .$$

In order to evaluate the bounds (5.2-4) and (5.2-5) economically, observe that

$$A_n = a\ A_{n-1} - A_{n-2} \qquad (n = 2, 3, \ldots)\ .$$

Hence,

(5.2-6) $$A_n = \alpha\lambda_1^n + \beta\lambda_2^n$$

where

(5.2-7) $$\lambda_1 = \frac{a + \sqrt{a^2 - 4}}{2}\ , \qquad \lambda_2 = \frac{a - \sqrt{a^2 - 4}}{2}\ .$$

Using the conditions that $A_1 = a$ and $A_2 = a^2 - 1$, we obtain

$$\alpha = \frac{a\lambda_2 - a^2 + 1}{\lambda_1 (\lambda_2 - \lambda_1)} ,$$

(5.2-8)

$$\beta = \frac{a\lambda_1 - a^2 + 1}{\lambda_2 (\lambda_1 - \lambda_2)} .$$

Using (5.2-6) in (5.2-5), obtain

(5.2-9) $$z > \frac{\alpha\lambda_1^{r-2} + \beta\lambda_2^{r-2}}{\alpha\lambda_1^{r-1} + \beta\lambda_2^{r-1}}$$

It can be shown that the ratio A_{r-2}/A_{r-1} is monotone increasing for increasing r and has limit $1/\lambda_1$. It follows that

(5.2-10) $$z > \frac{1}{\lambda_1} .$$

A similar inequality holds for w.

We also require that condition (5.2-3) be valid. Using (5.2-1) and (5.2-6) in (5.2-3), we obtain

$$\alpha\lambda_1^{n-4} (z.w\lambda_1^2 - (z+w) \lambda_1 + 1) + \beta\lambda_2^{n-4} (z.w\lambda_2^2 - (z+w) \lambda_2 + 1) > 0 .$$

Since $\beta < 0$; $\lambda_1 \geq 1$; $\lambda_1 = \frac{1}{\lambda_2}$; and $z, w > 0$; this expression takes the form

(5.2-11) $$(\lambda_1 - \frac{1}{z}) (\lambda_1 - \frac{1}{w}) > \left| \frac{\beta}{\alpha} \right| \frac{1}{\lambda_1^{2n-8}} (\frac{1}{\lambda_1} - \frac{1}{z}) (\frac{1}{\lambda_1} - \frac{1}{w}) .$$

The inequality (5.2-11) leads to the restriction $w, z > \frac{1}{\lambda_1}$ which agrees with that obtained in (5.2-10).

Hence for $w, z > (a - \sqrt{a^2 - 4})/2$, the tridiagonal matrix in the special form (4.3-9) with $b = 1$ is positive definite.

5.3 Some Results on Quindiagonal Spline Matrices

5.3.1 Introduction

Consider the following positive definite band matrix M. The case $a = 66$, $b = 26$, arises from the use of quintic polynomial splines (Ahlberg et al [1967]) defined on a uniform partition with natural boundary conditions.

$$(5.3\text{-}1) \qquad \begin{pmatrix} a & b & 1 & & & & & & \\ b & a & b & 1 & & & & & \\ 1 & b & a & b & 1 & & & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & & 1 & b & a & b & 1 \\ & & & & & 1 & b & a & b \\ & & & & & & 1 & b & a \end{pmatrix}$$

Such matrices occur quite frequently in a variety of contexts (Varga [1962], Fox [1962]) and have been studied by a number of different workers (Rutherford [1952], Gregory and Karney [1971], Hoskins and Ponzo [1972], Hoskins and Thurgur [1973], Hoskins and McMaster [1975]).

In Section 5.3.2 it is proved that the elements m_{ij} of M^{-1}

are such that

$$m_{ij} = (-1)^{i+j}|m_{ij}| \tag{5.3-2}$$

provided that $b^2 \geq 4\,(a - 2)$ and $2b \leq a + 2$.

Using the matrix S with diagonal elements $1, -1, 1, -1, \ldots$ and off-diagonal elements zero, then a similarity transformation applied to M^{-1} produces $SM^{-1}S^{-1}$. If SMS^{-1} is a positive matrix, then the result given by equation (5.3-2) follows immediately. The following theorem (Varga [1962]) can be made to apply.

Theorem 5.3.1

If $A = (a_{ij})$ is a real non-singular $n \times n$ irreducible matrix where $a_{ij} \leq 0$ for all $i \neq j$ and $a_{ii} > 0$ for all i, then A^{-1} is a non-negative matrix.

If we consider the matrix SMS^{-1} , then it satisfies all the conditions of Theorem 5.3.1 except that the elements in the second off-diagonal are positive. The result then follows by writing SMS^{-1} as the product of matrices satisfying the conditions of Theorem 5.3.1.

In Section 5.3.3, a condition dependent on M is established where the infinity norm of M^{-1} can be obtained by merely summing the centre row or column of M^{-1} . A similar result is obtained for the infinity norm of a variant of M .

5.3.2 *Determination of the Region Where the Elements of* M^{-1} *Alternate in Sign*

The matrix M can be expressed in a factored form using the following result .

Theorem 5.3.2

If P is the lower bidiagonal matrix

$$\begin{bmatrix} 1 & & & & \\ p_1 & 1 & & & \\ & p_2 & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & & p_{n-1} & 1 \end{bmatrix}$$

and T is the tridiagonal matrix

$$\begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & b_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix}.$$

then M can be written as $M = PTP^*$.

Proof:

If the product PTP^* is formed, then comparison of elements in PTP^* with M leads to the following recurrence relations for the quantities

p_i and a_i :

$$p_1 = \frac{b - b_1}{a} ; \quad p_{k+1} = \frac{1}{b_k} , \quad k \geq 1 ;$$

$$a_1 = a ; \quad a_{k+1} = (b - b_{k+1} - p_k)b_k , \quad k \geq 1 .$$

For the b_i , we obtain

$$b_2 = b - \frac{b}{a} + \frac{b^2}{ab_1} - \frac{a}{b_1} ,$$

$$b_3 = b + \frac{b}{b_1 b_2} - \frac{b}{ab_1 b_2} + \frac{1}{b_2} (\frac{1}{a} - a) ,$$

and, in general,

$$b_{k+1} = b - \frac{a}{b_k} + \frac{b}{b_k . b_{k-1}} - \frac{1}{b_k . b_{k-1} . b_{k-2}} , \quad k \geq 3 .$$

Once b_1 is specified, the remaining quantities b_i are defined as well as the values of p_i and a_i . The choice for b_1 is restricted to a non-zero value by the expression for p_2 . Also, the remaining b_k $(k > 2)$ which are dependent on b_1 must be non-zero and in fact will be shown to be positive in a later theorem. A convenient choice for b_1 is b since this make $p_1 = 0$ and satisfies the above requirements.

Theorem 5.3.3

The determinant of the matrix A obtained from M by deleting the first row and last column is greater than zero provided that $b^2 \geq 4(a-2)$ and $2b \leq a + 2$.

Proof:

The LU decomposition

(5.3-3) $$LA = U$$

may be performed on A to give

$$
(5.3\text{-}4) \qquad
\begin{pmatrix}
1 & & & & & & \\
1 & \ell_1 & & 0 & & & \\
1 & \ell_1 & \ell_2 & & & & \\
\cdot & \cdot & \cdot & \cdot & & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
1 & \ell_1 & \cdot & \cdot & \cdot & \cdot & \ell_{n-1}
\end{pmatrix}
\begin{pmatrix}
b & a & b & 1 & & & \\
1 & b & a & b & 1 & & \\
 & \cdot & \cdot & \cdot & \cdot & \cdot & \\
 & & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & & & 1 & b & a & b \\
 & & & & 1 & b & a \\
 & & & & & 1 & b
\end{pmatrix}
=
\begin{pmatrix}
d_1 & u_{12} & \cdot & \cdot & \cdot & \cdot & u_{1n} \\
 & d_2 & u_{23} & & & & \\
 & & d_3 & \cdot & & & \\
 & & & \cdot & \cdot & & \\
 & & & & \cdot & \cdot & \\
 & & & & & \cdot & u_{n-1,n} \\
 & & & & & & d_n
\end{pmatrix}
$$

The ℓ_i are determined by comparing the zero elements in U element by element with the corresponding elements in the product LA , and the following recurrence relations are obtained.

$$
(5.3\text{-}5) \qquad
\begin{aligned}
& b + \ell_1 = 0 \\
& a + b\,\ell_1 + \ell_2 = 0 \\
& b + a\,\ell_1 + b\,\ell_2 + \ell_3 = 0 \\
& 1 + b\,\ell_1 + a\,\ell_2 + b\,\ell_3 + \ell_4 = 0 \\
& \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot
\end{aligned}
$$

These conditions may be expressed in a more compact form as

$$
(5.3\text{-}6) \qquad (1 + \ell_1 t + \ell_2 t^2 + \ldots).(1 + bt + at^2 + bt^3 + t^4) = 1
$$

where the relations (5.3-5) may be obtained from (5.3-6) by equating coefficients of like powers of t on both sides of the equation (5.3-6). Equation (5.3-6) may be abbreviated as

$$
(5.3\text{-}7) \qquad \sum_{k=0}^{\infty} \ell_k t^k = \frac{1}{1 + bt + at^2 + bt^3 + t^4} \quad , \quad \ell_0 = 1 .
$$

Applying the same method used to give (5.3-5), the following recurrence relations are obtained for the d_i

$$
\begin{aligned}
& b = d_1 \\
& a + b\,\ell_1 = d_2 \\
(5.3\text{-}8) \qquad & b + a\,\ell_1 + b\,\ell_2 = d_3 \\
& 1 + b\,\ell_1 + a\,\ell_2 + b\,\ell_3 = d_4 \\
& \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot
\end{aligned}
$$

The recurrence relations (5.3-8) may be expressed more concisely as

$$(5.3\text{-}9) \qquad (1 + \ell_1 t + \ell_2 t^2 + \ldots).(b + at + bt^2 + t^3) = (d_1 + d_2 t + \ldots)$$

or, in a more abbreviated form, as

$$(5.3\text{-}10) \qquad (b + at + bt^2 + t^3) \sum_{k=0}^{\infty} \ell_k t^k = \sum_{k=1}^{\infty} d_k\, t^{k-1}$$

Substituting equation (5.3-7) in equation (5.3-10) gives the power series in the quantities d_i as

$$(5.3\text{-}11) \qquad \sum_{k=1}^{\infty} d_k t^{k-1} = \frac{b + at + bt^2 + t^3}{1 + bt + at^2 + bt^3 + t^4} \; .$$

Now we define

$$f(t) = 1 + bt + at^2 + bt^3 + t^4 \; .$$

Then

$$b + at + bt^2 + t^3 = \frac{f(t) - 1}{t} \; .$$

The right-hand side of (5.3-11) may be expressed as

$$(5.3\text{-}12) \qquad \frac{1}{t} - \frac{1}{t\,f(t)} \; .$$

Substitution of expression (5.3-12) into equation (5.3-11) and application of the result (5.3-7) gives

$$\sum_{k=1}^{\infty} d_k t^{k-1} = \frac{1}{t} - \frac{1}{t} \sum_{k=0}^{\infty} \ell_k t^k .$$

Cross-multiplication by t and use of the fact that $\ell_0 = 1$ leads to

(5.3-13) $$\sum_{k=1}^{\infty} d_k t^k = - \sum_{k=1}^{\infty} \ell_k t^k .$$

Hence, for all k,

(5.3-14) $$d_k = - \ell_k .$$

The determinant of A may be found from (5.3-3) provided that the ℓ_i are non-zero.

$$\det A = \frac{\det U}{\det L}$$

$$= \frac{\prod_{j=1}^{n} d_j}{\prod_{j=1}^{n-1} \ell_j} .$$

On application of (5.3-14), det A becomes

(5.3-15) $$\det A = (-1)^n \ell_n .$$

In order to prove that $\det A > 0$, we show that

(5.3-16) $$\ell_k = (-1)^k |\ell_k| \quad \text{for all } k .$$

Substituting $-t$ for t in (5.3-7) gives

(5.3-17) $$\sum_{k=0}^{\infty} \ell_k (-t)^k = \frac{1}{f(-t)} .$$

In order to find the solutions of the equation

(5.3-18) $$f(-t) = 0 ,$$

rewrite (5.3-18) as

(5.3-19) $$(t^2 + \frac{1}{t^2}) - b(t + \frac{1}{t}) + a = 0 \quad .$$

Let $z = t + \frac{1}{t}$ in (5.3-19) ; then the roots of the resulting equation are

$$\alpha = \frac{b + \sqrt{b^2 - 4(a-2)}}{2}, \qquad \beta = \frac{b - \sqrt{b^2 - 4(a-2)}}{2} \quad .$$

Solving for t in terms of α, we obtain

$$t = \frac{\alpha \pm \sqrt{\alpha^2 - 4}}{2} \quad .$$

For the zeros of $f(-t)$ to be positive

$$\alpha^2, \beta^2 \geq 4 \;, \quad b^2 \geq 4(a-2) \quad .$$

This occurs in the region enclosed by the curves $b^2 = 4(a - 2)$ and $2b - a = 2$ in the upper half of the a, b plane of Figure 5.3.1. The two curves are tangent at the point (6,4).

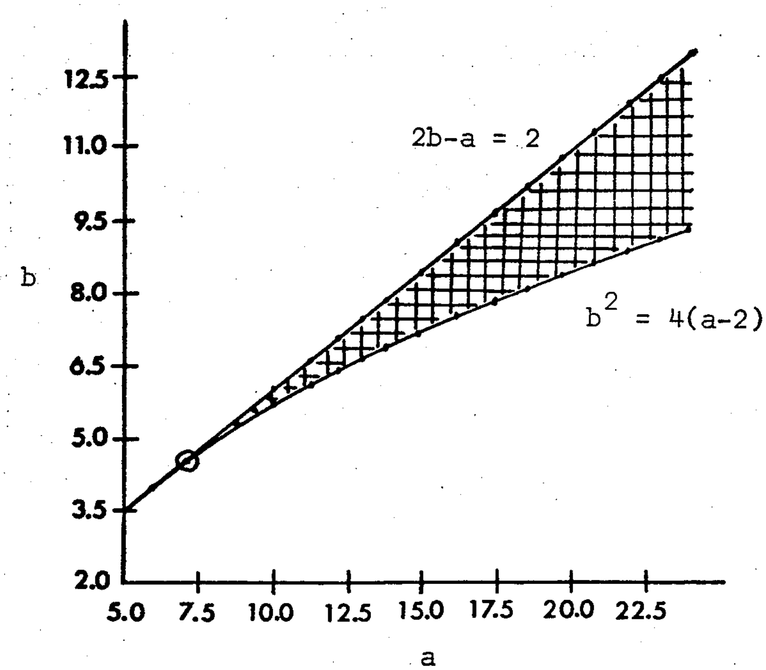


Figure 5.3.1

Also, the roots of $f(-t)$ occur in reciprocal pairs; for, if we define

$$g(t) = t^2 - bt + a - \frac{b}{t} + \frac{1}{t^2} ,$$

then

$$g(t) = g\left(\frac{1}{t}\right) , \tag{5.3-20}$$

and the zeros of $g(t)$ are the same as the zeros of $f(-t)$. From equation (5.3-20), it is evident that the zeros of $f(-t)$ occur in reciprocal pairs

$$e, \quad 1/e, \quad r, \quad 1/r .$$

Hence,

$$f(-t) = (t-e)\left(t - \frac{1}{e}\right)(t-r)\left(t - \frac{1}{r}\right) .$$

Substituting in (5.3-17) gives

$$\sum_{k=0}^{\infty} \ell_k (-t)^k = \frac{1}{(1-\frac{t}{e})(1-et)(1-t/r)(1-rt)}$$

$$= \sum_{k=0}^{\infty} c_k t^k \quad \text{where} \quad c_k > 0 \quad \text{and thus}$$

the $\ell_i \neq 0$. Therefore

$$\ell_k = (-1)^k |\ell_k|$$

and using this result in (5.3-15), we obtain

$$\det A = (-1)^n \ell_n$$

$$= |\ell_n|$$

in the region $b^2 \geq 4(a-2)$ and $2b \leq a + 2$.

Theorem 5.3.4

The b_i in the matrix T defined in Theorem 5.3.2 are all positive provided that $b^2 \geq 4(a-2)$ and $2b \leq a + 2$.

Proof:

Delete the first column and last row of M , and take the determinant of the resulting matrix given by

$$\text{(5.3-21)} \qquad \begin{vmatrix} b & 1 & & & & & \\ a & b & 1 & & & & \\ b & a & b & 1 & & & \\ 1 & b & a & b & 1 & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & 1 & b & a & b & 1 \\ & & & 1 & b & a & b \end{vmatrix}$$

The determinant (5.3-21) is merely $\det A^T$, and $\det A$ was proved positive in Theorem 5.3.3.

If the corresponding row and column of PTP^* is deleted, we have

$$\text{(5.3-22)} \qquad \det \begin{pmatrix} b_1 & & & & & \\ a_2 & b_2 & & & & \\ b_2 & a_3 & b_3 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & b_{n-2} & a_{n-1} & b_{n-1} \end{pmatrix} = \prod_{i=1}^{n-1} b_i \; .$$

Thus, on comparing expression (5.3-21) with expression (5.3-22), we obtain

$$\prod_{i=1}^{n-1} b_i = \det A^T > 0 \quad \text{for arbitrary } n \, .$$

Hence, the b_i are positive for all i . Q.E.D.

Corollary

The matrix P is non-negative (Theorems (5.3.2) and (5.3.4)). provided that $b^2 \geq 4(a - 2)$ and $2b \leq a + 2$.

Theorem 5.3.5

If the matrix M is positive definite, and $M = PTP^*$, then T is positive definite.

Proof:

Let $\underline{x}^* = (x_1, x_2, \ldots, x_n)$; then

$$\underline{x}^* M \underline{x} > 0 \quad \text{for all } \underline{x} \neq \underline{0} \, .$$

Since $M = PTP^*$, then

$$\underline{x}^* PTP^* \underline{x} > 0 \quad \text{for all } \underline{x} \neq \underline{0} \, .$$

Let $P^* \underline{x} = \underline{y}$, then

$$\underline{y}^* T \underline{y} > 0 \quad \text{for all } \underline{y} \neq \underline{0} \, .$$

Q.E.D.

Corollary

The matrix T is non-negative provided that the restrictions of Theorem 5.3.5 apply.

Finally, we have

Theorem 5.3.6

The matrix M^{-1} has an element m_{ij} in position (i, j) such that $m_{ij} = (-1)^{i+j} |m_{ij}|$ provided that $b^2 \geq 4(a - 2)$ and $2b \leq a + 2$.

Proof:

Determine matrices P and T as in Theorem 5.3.2 such that

$$M = PTP^* \quad .$$

Let S be the matrix with diagonal entries

$$1, -1, +1, \ldots \quad .$$

Then the matrices SPS^{-1}, STS^{-1}, and SP^*S^{-1} all satisfy the requirements of Theorem 5.3.1 (since P and T are non-negative and tridiagonal, all the off-diagonal elements in the products are negative or zero), and the elements in all the inverses are non-negative.

Since $SMS^{-1} = (SPS^{-1})(STS^{-1})(SP^*S^{-1})$, then $(SMS^{-1})^{-1}$ is non-negative.

This is only possible if

$$m_{ij} = (-1)^{i+j} |m_{ij}| \quad .$$

A similar result may be obtained concerning the signs of the elements in the inverse of matrix (5.3-1) for the special case $b = 0$.

Theorem 5.3.7

The elements m_{ij} of the inverse of the matrix M (which is a special case of (5.3.1) with $b = 0$)

$$\begin{bmatrix} a & 0 & 1 & & & & \\ 0 & a & 0 & 1 & & & \\ 1 & 0 & a & 0 & 1 & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & 1 & 0 & a & 0 & 1 \\ & & & 1 & 0 & a & 0 \\ & & & & 1 & 0 & a \end{bmatrix}$$

are such that

$$m_{ij} > 0 \quad \text{if} \quad |i-j| \equiv 0 \pmod 4$$
$$m_{ij} < 0 \quad \text{if} \quad |i-j| \equiv 2 \pmod 4$$
$$m_{ij} = 0 \quad \text{if} \quad |i-j| \equiv 1 \text{ or } 3 \pmod 4$$

provided that $a > 2$.

Proof:

Let D be the matrix with diagonal elements

$$1,\ i,\ i^2,\ i^3,\ \ldots$$

and zeros elsewhere, where $i = \sqrt{-1}$.

Then $DM\bar{D} = aI - Q$ where

$$Q = \begin{bmatrix} 0 & 0 & 1 & & & & \\ 0 & 0 & 0 & 1 & & & \\ 1 & 0 & 0 & 0 & 1 & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & 1 & 0 & 0 & 0 & 1 & \\ & & 1 & 0 & 0 & 0 & \\ & & & 1 & 0 & 0 & \end{bmatrix}$$

Hence, $M^{-1} = \bar{D}(aI - Q)^{-1} D$.

Since the elements in M are real, then the elements in M^{-1} are also real. For this to be possible, the following must be true:

$$m_{ij} = 0 \quad \text{if} \quad |i-j| \equiv 1 \text{ or } 3 \pmod 4 .$$

Now $(aI - Q)^{-1}$, where $a > 2$, is a non-negative matrix (Theorem 5.3-1).

Hence $\bar{D}(aI - Q)^{-1} D$ is such that

$$m_{ij} > 0 \quad \text{if} \quad |i-j| \equiv 0 \pmod 4$$

$$m_{ij} < 0 \quad \text{if} \quad |i-j| \equiv 2 \pmod 4 .$$

Q.E.D.

5.3.3 Results on the Infinity Norm of the Inverse of Symmetric Quindiagonal Toeplitz Matrices

In obtaining the parameters of the natural quintic spline, it is necessary to solve systems of the form $A\underline{x} = \underline{d}$ where A is of the form (5.3-1). The infinity norm of the inverse of A plays a decisive role in determining the magnitude of the errors in the solution; however, in general, it is impossible to say anything about the norm of the inverse of A without computing the inverse, a procedure which is closely connected with the solution of the system (Wilkinson [1965]). The computation of the general inverse is an expensive undertaking and it would be preferable to be able to calculate the norm of the inverse without computing the entire inverse. We consider this problem in this section.

The infinity norm of the $n \times n$ matrix $A = (a_{ij})$ is defined as

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}| ,$$

and it is easily seen that, if all the elements of A are of the same sign, then the largest row sum of A, if the sign is ignored, is the infinity norm.

In Theorem 5.3.6, it was established that the inverse of the matrix A, when pre-multiplied and post-multiplied by the matrix with diagonal entries $1, -1, 1, \ldots$ is always such that the elements are all of the same sign. If it may further be shown that the largest row sum occurs in the centre row, then the norm of A^{-1} may be readily obtained. We first establish a preliminary result.

Theorem 5.3.8

If A is a symmetric Toeplitz matrix with first row $(a, b, 1, 0, .., 0)$ with $a > b > 1$, $a > 2$, and $\underline{v}$ is a vector

$$v_1 < v_2 < \ldots < v_{[(n+1)/2]}$$

with $P\underline{v} = \underline{v}$ (P the skew identity matrix), then

$$\underline{u} = A\underline{v}$$

is such that $P\underline{u} = \underline{u}$, and

$$u_1 < u_2 < \ldots < u_{[(n+1)/2]}.$$

Proof:

If the product $\underline{u} = A\underline{v}$ is formed, then

$$u_k = v_{k-2} + b\, v_{k-1} + a\, v_k + b\, v_{k+1} + v_{k+2}$$

and

$$u_{k+1} = v_{k-1} + b\, v_k + a\, v_{k+1} + b\, v_{k+2} + v_{k+3} \quad .$$

It is evident that

$$u_{k+1} - u_k > 0 \quad \text{if} \quad k + 3 \leq [n/2] \quad .$$

Let n be odd. The remaining cases must be examined in detail.

Case (i)

Assume that $k + 2 = [n/2] + 1$, then

$$v_{k+1} = v_{k+3}$$

and we have that

$$u_k = v_{k+2} + b\ v_{k+1} + a\ v_k + b\ v_{k-1} + v_{k-2}$$

and

$$u_{k+1} = b\ v_{k+2} + (a+1)\ v_{k+1} + b\ v_k + v_{k-1}\ .$$

Then

$$u_{k+1} - u_k = (b-1)\ v_{k+2} + (a+1-b)\ v_{k+1} + (b-a)\ v_k + (1-b)\ v_{k-1} - v_{k-2}\ .$$

Since $b > 1$ and $v_{k+2} > v_{k+1}$, $v_{k+1} > v_k$, $v_k > v_{k-1}$, we obtain the relation

$$u_{k+1} - u_k > v_{k-1} - v_{k-2} > 0\ .$$

Case (ii)

Assume that $k + 1 = [n/2] + 1$, then

$$v_k = v_{k+2}$$

and

$$v_{k-1} = v_{k+3}\ .$$

Hence

$$u_k = b\ v_{k+1} + (a+1)\ v_k + b\ v_{k-1} + v_{k-2}$$

and

$$u_{k+1} = av_{k+1} + 2bv_k + 2v_{k-1}\ .$$

Then

$$u_{k+1} - u_k > (a-b)\ v_{k+1} + (2b-a-1)\ v_k + (2-b)\ v_{k-1} - v_{k-2}\ .$$

Since $a > b$, $v_{k+1} > v_k$, $v_k > v_{k-1}$, and $v_{k-1} > v_{k-2}$, then $u_{k+1} - u_k > 0$. We can conclude that the vector $\underline{u}$ has elements that are monotone increasing to the midpoint from u_1 to $u_{[(n+1)/2]}$.

Since $\underline{u} = A\underline{v}$, then $P\underline{u} = PAPP\underline{v} = PAP\underline{v} = A\underline{v}$. Hence $\underline{u} = P\underline{u}$.

For n even, a similar analysis may be used and leads to the result provided that $a > 2$.

Lemma 1

Given the conditions of the theorem, then $\underline{u} = A^k\underline{v}$ $(k = 1,2,\ldots)$ is such that $P\underline{u} = \underline{u}$ and $u_1 < u_2 < \ldots < u_{[n/2]}$.

Lemma 2

Given A as in the theorem, and $\underline{v}$ a vector such that $v_1 \leq v_2 \leq \cdots \leq v_{[m/2]}$ with $P\underline{v} = \underline{v}$, then $\underline{u} = A^k\underline{v}$ $(k = 1, 2, \ldots)$ is such that $P\underline{u} = \underline{u}$, and $u_1 \leq u_2 \leq \cdots \leq u_{[m/2]}$ where $m = n$, n even, and $m = n + 1$, n odd .

Lemma 3

If $A = (a, 1, 0, \ldots)_{n\times n}$ is an $n\times n$ symmetric Toeplitz matrix with $a > 2$, and $\underline{v}$ is a vector such that the elements of $D\underline{v}$ are positive (D is the matrix with diagonal entries $1, -1, 1, \ldots$), $P(D\underline{v}) = D\underline{v}$ and $|v_1| \leq |v_2| \leq \cdots < |v_{[(n+1)/2]}|$, then $\underline{u}$ where $A\underline{u} = \underline{v}$ is such that $P(D\underline{u}) = D\underline{u}$ and

$$|u_1| \leq |u_2| \leq \cdots \leq |u_{[(n+1)/2]}| .$$

Proof:

First let $\underline{s} = D\underline{u}$ and $D\underline{v} = \underline{t}$ where t is a positive vector and $\underline{t}$ is such that $t_1 \leq t_2 \leq \cdots \leq t_{[(n+1)/2]}$ and $P\underline{t} = \underline{t}$. The equation

$\underline{Au} = \underline{v}$ may be written as

$$DAD^{-1}\underline{s} = \underline{t}$$

Hence,

$$\underline{s} = [D^{-1}AD]^{-1}\,\underline{t}\ .$$

By theorem 5.3.1, $[D^{-1}AD]^{-1}$ is positive and since $\underline{t}$ is positive then the elements of $\underline{s}$ are all positive. The equation for $\underline{s}$ may be rewritten as

$$\underline{s} = \frac{1}{2a}\left[I - \frac{B}{2a}\right]^{-1}\underline{t}$$

where B is the symmetric toeplitz matrix with first row $(a, 1, 0, \ldots, 0)$. As was done in the main theorem, it may be shown if $a > 1$ that for $\underline{t}$ monotone increasing to the midpoint, then $\underline{r} = B^k\underline{t}$, $k = 1, 2, \ldots$ is also monotone increasing to the midpoint and $P\underline{r} = \underline{r}$. Now if $||B||_\infty \leq 2a$, that is $a > 2$ then

$$\underline{s} = \frac{1}{2a}\sum_{k=0}^{\infty}\left(\frac{B}{2a}\right)^k\underline{t}\ .$$

is a convergent sequence and $\underline{s}$ is monotone increasing to its midpoint and $P\underline{s} = \underline{s}$. Recalling that $\underline{u} = D\underline{s}$ immediately leads to the result.

Q.E.D.

Theorem 5.3.9

The infinity norm of the inverse of the matrix A defined in (5.3-1) is found by summing the absolute values of the elements in the centre row or column of A^{-1} provided that $\|A^{-1}\|_\infty \leq 1$, $b^2 \geq 4(a-2)$ and $a \geq 2b - 2$.

Proof:

First, the matrix A may be decomposed in the following manner

(5.3-23) $$A = P(x)\, P^*(y)$$

where the rectangular matrix $P(x)$ is defined as

(5.3-24) $$\begin{bmatrix} 1 & x & 1 & & & & & & \\ & 1 & x & 1 & & & & & \\ & & \cdot & \cdot & \cdot & & & & \\ & & & \cdot & \cdot & \cdot & & & \\ & & & & \cdot & \cdot & \cdot & & \\ & & & & & 1 & x & 1 & \\ & & & & & & 1 & x & 1 \end{bmatrix}_{n\times(n+2)}$$

Let $\underline{u}$ be a vector such that

$$\underline{u}^* = (1, -1, 1, \ldots) ,$$

and consider the solution of the linear system

$$P(x)\, P^*(y)\, \underline{v} = \underline{u} \quad . \tag{5.3-25}$$

This system (5.3-25) may be solved in two stages by setting

$$P^*(y)\, \underline{v} = \underline{w} , \tag{5.3-26}$$

where $\underline{w}^* = (w_1, w_2, \ldots, w_{n+1}, w_{n+2})$; we first solve the linear system

$$P(x)\, \underline{w} = \underline{u} , \tag{5.3-27}$$

and then solve the system (5.3-26). The system (5.3-27) may be rearranged as

$$T(x).\begin{pmatrix} w_2 \\ \cdot \\ \vdots \\ \cdot \\ w_{n+1} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} - \begin{pmatrix} w_1 \\ 0 \\ \cdot \\ \cdot \\ w_{n+2} \end{pmatrix} \tag{5.3-28}$$

where $T(x)$ is defined as

$$\begin{bmatrix} x & 1 & & & & & \\ 1 & x & 1 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & 1 & x & 1 \\ & & & & & 1 & x \end{bmatrix}_{n\times n} \tag{5.3-29}$$

From (5.3-26), it is evident that

$$w_1 = v_1 \quad \text{and} \quad w_{n+2} = v_n \ .$$

Thus

$$(5.3\text{-}30) \qquad T(y).\begin{pmatrix} v_1 \\ \cdot \\ \cdot \\ \cdot \\ v_n \end{pmatrix} = \begin{pmatrix} w_2 \\ \cdot \\ \cdot \\ \cdot \\ w_{n+1} \end{pmatrix} .$$

By defining $\underline{v}_s = (v_1, v_2, \ldots, v_n)^*$ and $\underline{w}_s = (w_2, w_3, \ldots, w_{n+1})^*$, (5.3-30) may be written as

$$T(y).\underline{v}_s = \underline{w}_s$$

and (5.3-28) may be re-expressed as

$$(5.3\text{-}31) \qquad T(x)\begin{pmatrix} w_2 \\ \cdot \\ \cdot \\ \cdot \\ w_{n+1} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} - \begin{pmatrix} v_1 \\ 0 \\ \cdot \\ 0 \\ v_n \end{pmatrix} .$$

From the symmetry of A, we have

$$v_1 = \alpha v_n$$

where $\alpha = 1$ (n odd); $\alpha = -1$ (n even), and $v_1 > 0$.

Thus the equations to be solved are

$$(5.3\text{-}32) \qquad T(y).\underline{v}_s = \underline{w}_s$$

and

$$T(x)\,\underline{w}_s = \underline{u} - v_1 \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ \alpha \end{bmatrix} = \underline{q}_s \,. \tag{5.3-33}$$

If $v_1 \leq 1$, then $\underline{q}_s$ is a vector that is monotone increasing in absolute size to its centre, and symmetric (we call it type M), and alternating in sign. By equating corresponding elements on both sides of (5.3-23), it is evident that $x > 2$ and $y > 2$ for the region defined by Theorem 5.3.6. Then, applying Lemma 3 of Theorem 5.3.8 to (5.3-32) and (5.3-33) consecutively, we find that $\underline{w}_s$ is alternating in sign and ~~of type~~ M and $\underline{v}_s$ is alternating in sign and of type M .

Now, if $||A^{-1}||_\infty \leq 1$, then $v_1 \leq 1$ and the infinity norm of A^{-1} is obtained from the centre row and column. This gives one lower bound $a \geq 2b+3$. The result obtained in Theorem 5.3.9 is dependent on the fact that $v_1 \leq 1$, which is assured provided that $||A^{-1}||_\infty \leq 1$. This bound on v_1 may be sharpened.

Lemma 1

The infinity norm of the inverse of the matrix A is found by summing the absolute values of the elements in the centre row or column provided that

(i) for $x > 2$, $v_1 \leq \dfrac{2\sinh(\Theta/2)e^{\Theta/2}}{(x-2)}$ where $2\cosh\Theta = x$

(ii) for $x = 2$, $v_1 \leq n/2$.

Proof:

From Equation (5.3-33), we have

(5.3-34) $$T(x)\underline{w}_s = \underline{u} - v_1 (e_1 + \alpha\, e_n) ,$$

where $\alpha = 1$ (n odd), $\alpha = -1$ (n even) .

From Equation (5.3-34), $\underline{w}_s$ can be written as

(5.3-35) $$\underline{w}_s = T^{-1}(x)\underline{u} - v_1\, T^{-1}(x)\,(e_1 + \alpha\, e_n).$$

Denote the i'th element in the vector $\underline{w}_s$ by $\underline{w}_{si}$. Then

(5.3-36) $$(-1)^{i+1}\, \underline{w}_{si} = \sum_{k=1}^{n} |t_{ik}| - v_1 \left(|t_{i1}| + |t_{in}|\right)$$

where t_{ij} represented the i j'th element in $T^{-1}(x)$.

Define S_i by

(5.3-37) $$S_i = \sum_{k=1}^{n} |t_{ik}|$$

and R_i by

(5.3-38) $$R_i = |t_{i1}| + |t_{in}|.$$

We first consider case (i) where $x > 2$.

(i) The elements t_{ij} may be determined explicitly (Fischer and Usmani [1969]). We define

(5.3-39) $$D_j = \frac{\sinh\,(j+1)\theta}{\sinh\,\Theta}$$

where $2 \cosh \Theta = x$.

Then

$$t_{ij} = \frac{D_{j-1}\, d_{n-i}}{D_n}\ ,\ i \geq j\ ,$$

(5.3-40)

$$t_{ij} = \frac{D_{i-1}\, D_{n-j}}{D_n}\ ,\ i \leq j\ .$$

Apply (5.3-40) to (5.3-38) to give

$$R_i = \frac{D_0\, D_{n-i}}{D_n} + \frac{D_{i-1}\, D_0}{D_n} = \frac{D_{n-i} + D_{i-1}}{D_n} \tag{5.3-41}$$

If (5.3-39) is used, then

$$R_i = \frac{\cosh\ (n - 2i + 1)\ \Theta/2}{\cosh\ (n + 1)\ \Theta/2}\ . \tag{5.3-42}$$

The R_i $(i = 1, 2, \ldots, n)$ in (5.3-42) are concave, and may be shown to be symmetric about $i = [(n+1)/2]$ by differentiation with respect to i. In a similar manner, the S_i $(i = 1, 2, \ldots, n)$ defined in (5.3-37) may be shown to be convex and symmetric about $i=[(n+1)/2]$.

For the vector $\underline{w}_s$ defined in (5.3-34) to be convex in modulus (the elements of $\underline{w}_s$ alternate in sign), it is required that

$$v_1 \leq \min_i \frac{S_i}{R_i}\ .$$

Hence

$$v_1 \leq \frac{S_1}{R_1}\ . \tag{5.3-43}$$

In order to determine S_1, apply Equations (5.3-40) to (5.3-37) to give

$$S_1 = \frac{D_0 \quad D_{n-1}}{D_n} + \frac{D_0 \quad D_{n-2}}{D_n} + \cdots + \frac{D_0 \quad D_0}{D_n} \tag{5.3-44}$$

On applying the result (Fisher and Usmani [1969])

$$\sum_{k=1}^{n-1} D_k = (D_{n-1} - D_{n-2} - D_0)/(x-2)$$

to Equation (5.4-44), we obtain

$$S_1 = \frac{1}{(x-2)} \left[\frac{D_n - (D_{n-1} - D_0)}{D_n} \right] . \tag{5.3-45}$$

If (5.3-41) and (5.3-45) are used, then S_1/R_1 becomes

$$\frac{S_1}{R_1} = \frac{1}{(x-2)} \left[\frac{D_n}{D_{n-1} + 1} - 1 \right] . \tag{5.3-46}$$

Substituting the value for D_j from (5.3-39) gives

$$\begin{aligned} \frac{S_1}{R_1} &= \frac{1}{(x-2)} \left[\frac{\sinh (n+1)\theta}{\sinh n\theta + \sinh \theta} - 1 \right] \\ &= \frac{2\sinh \theta/2}{(x-2)} \left[\frac{\sinh n\theta/2}{\cosh (n-1)\theta/2} \right] . \end{aligned} \tag{5.3-47}$$

The quotient S_1/R_1 is monotone increasing for increasing n; also, we have

$$\lim_{n\to\infty} \frac{S_1}{R_1} = \frac{2\sinh (\theta/2)e^{\theta/2}}{(x-2)} .$$

Then

$$v_1 \sim \frac{e^{\theta} - 1}{(x-2)} ;$$

this result permits quick evaluation of an approximate bound for v_1. The following table gives an upper bound on v_1 for varying x and N from Equation (5.3-47).

x	N = 10	N = 100
2.05	3.93	5.00
2.1	3.34	3.70
2.2	2.71	2.79
2.3	2.36	2.39
2.4	2.15	2.16
2.5	1.99	2.00
3.0	1.62	1.62
4.0	1.36	1.36
5.0	1.26	1.26
6.0	1.21	1.21
7.0	1.17	1.17

Case (ii)

When $x = 2$, then the D_j in (5.3-41) are given (Fisher and Usmani [1969]) by

(5.3-48) $$D_j = j + 1 \quad .$$

Using (5.3-48) in (5.3-44) gives

$$S_1 = n/2 \quad .$$

Applying (5.3-48) to (5.3-41) gives

$$R_1 = 1 \quad .$$

Hence

$$\frac{S_1}{R_1} = \frac{n}{2}$$

and $v_1 \le n/2$ for $x = 2$.

A similar result may be obtained for the quindiagonal matrix (5.3-1) when the off-diagonal elements are non-positive.

Theorem 5.3.10

If A is a symmetric Toeplitz matrix defined by specifying the first row,

$$A = (a, -b, -1, 0, \ldots, 0)_n$$

with $a > b > 1$ and $a > 2b + 2$, then $\|A^{-1}\|_\infty$ is given by the centre row and column of A^{-1} .

Proof:

$$A = (a, -b, -1, 0, \ldots, 0)$$

$$= 2aI - (a, b, 1, 0, \ldots, 0)_n \; .$$

If we let $B = (a, b, 1, 0, \ldots, 0)$, then

$$A^{-1} = (2aI - B)^{-1}$$

$$= \frac{1}{2a} (I - \frac{B}{2a})^{-1} \; .$$

Since $\|B\|_\infty < 2a$, then A^{-1} can be written as

$$A^{-1} = \frac{1}{2a} \sum_{k=0}^{\infty} (\frac{B}{2a})^k \; .$$

If we let $e^T = (1, 1, \ldots, 1)$, then

$$A^{-1} e = \frac{1}{2a} \sum_{k=0}^{\infty} (\frac{B}{2a})^k e \; .$$

Applying Theorem 5.3.7, we find that

$$\|A^{-1}e\|_\infty = \|A^{-1}\|_\infty$$

and the infinity norm is then given by the centre row and column.

5.4 On the Inverse of a Matrix Arising from a Third-Order Finite Difference Approximation

5.4.1 Introduction

The following matrix A arises in a third-order finite difference approximation.

$$(5.4\text{-}1) \qquad A = \begin{bmatrix} a & b & 1 & & & & & \\ 1 & a & b & 1 & & & & \\ & 1 & a & b & 1 & & & \\ & & \cdot & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \cdot & \cdot \\ & & & & 1 & a & b & 1 \\ & & & & & 1 & a & b \\ & & & & & & 1 & a \end{bmatrix}_{n\times n}$$

It is demonstrated that the elements α_{ij} of A^{-1} (A^{-1} is assumed to exist) are such that

$$\alpha_{ij} = (-1)^{i+j}\,|\alpha_{ij}|$$

in the region given by $4b^3 - a^2b^2 - 18ab + 27 + 4a^3 \le 0$. If the similarity transformation $DA^{-1}D^{-1}$ is applied to A^{-1}, where D is the matrix with diagonal elements $1, -1, 1, \ldots$ and off-diagonal elements zero, then (Theorem 5.3.1) a proof that $DA^{-1}D^{-1}$ is positive would lead to the result.

5.4.2 *Derivation of the Result*

It is possible to express A in a factored form using the following theorem.

Theorem 5.4.1

If L is the lower bidiagonal matrix

$$(5.4\text{-}2) \qquad \begin{bmatrix} 1 & & & & & \\ c_1 & 1 & & & & \\ & c_2 & \cdot & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & c_{n-1} & 1 \end{bmatrix}$$

and T is the tridiagonal matrix

$$(5.4\text{-}3) \qquad \begin{bmatrix} a_1 & b_1 & & & \\ 1 & a_2 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & b_{n-1} \\ & & & 1 & a_n \end{bmatrix}$$

then the matrix A in (5.4-1) can be factored as

$$A = TL^* \quad .$$

Proof:

If the product TL^* is formed, then comparison of elements in this product with those in A gives the following recurrence relations in a_i, c_i, and b_i :

$$a_1 = a, \quad c_1 = 0, \quad b_1 = b\ ,$$

$$c_{k+1} = \frac{1}{b_k}\ ,$$

$$a_{k+1} = a - c_k\ ,$$

$$b_k = b - a_k \cdot c_k \quad .$$

Lemma 1

The matrix Q obtained from A by deleting the first column and last row has determinant

$$\det\ Q = \prod_{i=1}^{n-1} b_i \quad .$$

Proof:

Q may be written in the factored form

$$\begin{pmatrix} b_1 & & & & & \\ a_2 & b_2 & & & & \\ 1 & a_3 & \cdot & & & \\ & 1 & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & 1 & a_{n-1} & b_{n-1} \end{pmatrix} \begin{pmatrix} 1 & c_2 & & & & \\ & 1 & c_3 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & 1 & c_{n-1} \\ & & & & & 1 \end{pmatrix}$$

Theorem 4.5.2

The determinants of the matrix A and of the matrix Q obtained from A by deleting the first column and the last row are greater than zero provided that

$$4b^3 - a^2 b^2 - 18ab + 27 + 4a^3 \leq 0 \quad .$$

Proof:

(ii) The LU decomposition $LQ^T = U$ (Forsythe and Moler [1967]) may be applied to Q^T to give

$$\text{(5.4-4)} \quad \begin{bmatrix} 1 & & & & & & \\ 1 & r_1 & & & & & \\ 1 & r_1 & r_2 & & & & \\ \cdot & \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\ 1 & r_1 & r_2 & r_2 & \cdot & \cdot & r_{n-2} \end{bmatrix} \begin{bmatrix} b & a & 1 & & & & \\ 1 & b & a & 1 & & & \\ & 1 & b & a & 1 & & \\ & & \cdot & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot & \cdot \\ & & & & 1 & b & a \\ & & & & & 1 & b \end{bmatrix} = \begin{bmatrix} s_0 & u_{12} & & & & & u_{1,n-1} \\ & s_1 & \cdot & & & & \\ & & s_2 & \cdot & & & \\ & & & \cdot & \cdot & & \\ & & & & \cdot & \cdot & \\ & & & & & \cdot & u_{n-1,n} \\ & & & & & & s_{n-2} \end{bmatrix} .$$

The r_i may be determined by comparing the zero elements in U element by element with the corresponding elements in the product LQ^T, and the following recurrence relations are obtained.

$$\text{(5.4-5)} \quad \begin{aligned} & b + r_1 = 0 \\ & a + br_1 + r_2 = 0 \\ & 1 + ar_1 + br_2 + r_3 = 0 \\ & \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ & \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \end{aligned}$$

These conditions may be expressed in more compact form as

(5.4-6) $$(1 + r_1 t + r_2 t^2 + \ldots)(1 + bt + at^2 + t^3) = 1 ,$$

where powers of t may be compared on both sides of (5.4-6) to give (5.4-5). If we let $f(t) = 1 + bt + at^2 + t^3$, then (5.4-6) may be expressed as

(5.4-7) $$\sum_{k=0}^{\infty} r_k t^k = \frac{1}{f(t)} ,$$

where r_0 is defined to be 1 .

In a like manner, the s_i may be evaluated and expressed as

$$\sum_{k=0}^{\infty} s_k t^k = \frac{b + at + t^2}{1 + bt + at^2 + t^3}$$

$$= \frac{1}{t}\left(1 - \frac{1}{f(t)}\right)$$

or

$$\sum_{k=0}^{\infty} s_k t^{k+1} = 1 - \frac{1}{f(t)} .$$

Hence

(5.4-8) $$\frac{1}{f(t)} = 1 - \sum_{k=0}^{\infty} s_k t^{k+1} .$$

Comparison of (5.4-7) and (5.4-8) gives

(5.4-9) $$r_k = -s_{k-1} .$$

Then (5.4-4) and (5.4-9) give (provided that the $r_i \neq 0$).

$$\det Q^T = \frac{\prod_{i=0}^{n-2} s_i}{\prod_{i=0}^{n-2} r_i}$$

or, on simplification

$$\text{(5.4-10)} \qquad \det\ Q^T = (-1)^{n-2}\, s_{n-2} \ .$$

For restricted values of a and b, $f(t)$ factors as

$$f(t) = (t + \alpha)\,(t + \beta)\,(t + \delta)$$

where $\alpha,\ \beta,\ \delta > 0$ (Lemma 1). Hence $1/f(t)$ may be expressed as a series

$$\text{(5.4-11)} \qquad \frac{1}{f(t)} = \sum_{k=0}^{\infty} \phi_k\, t^k$$

where ϕ_k are the coefficients in the power series and $\operatorname{sgn} \phi_k = (-1)^k$, and hence the $r_i \neq 0$. On comparison of (5.4-7) and (5.4-11), we have

$$\text{(5.4-12)} \qquad \operatorname{sgn}\ r_k = (-1)^k \ .$$

Substituting this result into (5.4-9) gives

$$\text{(5.4-13)} \qquad \operatorname{sgn}\ s_k = (-1)^k \ .$$

This result, when applied to (5.4-10), gives

$$\det\ Q^T > 0 \ .$$

(ii) In order to demonstrate that $\det\ A > 0$ and that the a_k in (5.4-3) are positive, we write $LA = U$ as

$$\text{(5.4-14)} \quad \begin{bmatrix} 1 & & & & & & \\ 1 & d_1 & & & & & \\ 1 & d_1 & d_2 & & & & \\ \cdot & \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\ 1 & d_1 & d_2 & \cdot & \cdot & \cdot & d_{n-1} \end{bmatrix} \begin{bmatrix} a & b & 1 & & & & \\ 1 & a & b & 1 & & & \\ & 1 & a & b & 1 & & \\ & & \cdot & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot & \cdot \\ & & & & 1 & a & b \\ & & & & & 1 & a \end{bmatrix} = \begin{bmatrix} e_0 & u_{12} & & & & u_{1,n-1} \\ & e_1 & \cdot & & & \\ & & e_2 & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & \cdot \\ & & & & \cdot & u_{n-1,n} \\ & & & & & e_{n-1} \end{bmatrix}$$

The d_i may be determined by comparing the zero elements in U with the corresponding elements in the product LA . The following recurrence relations in the d_i are then obtained.

$$
\begin{aligned}
&a + d_1 = 0 \\
&b + ad_1 + d_2 = 0 \\
&1 + bd_1 + ad_2 + d_3 = 0 \\
&\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot
\end{aligned}
\tag{5.4-15}
$$

The recurrence relations (5.4-15) may be expressed as

$$
\left(\sum_{k=0}^{\infty} d_k \, t^k \right) (1 + at + bt^2 + t^3) = 1 \tag{5.4-16}
$$

where powers of t on both sides of (5.4-16) may be compared to give the relations (5.4-15) and d_0 is defined to be 1 . Then

$$
\sum_{k=0}^{\infty} d_k \, t^k = \frac{1}{1 + at + bt^2 + t^3} = \frac{1}{g(t)} \tag{5.4-17}
$$

The e_i may be obtained as functions of the d_i by comparing the e_i with the corresponding elements in the product LA in (5.4-14). The recurrence relations for the e_i are

$$
\begin{aligned}
&a = e_0 \\
&b + d_1 a = e_1 \\
&1 + bd_1 + d_2 a = e_2 \\
&\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\
&\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot
\end{aligned}
\tag{5.4-18}
$$

The relations (5.4-18) may be expressed as

$$(a + bt + t^2)\left(\sum_{k=0}^{\infty} d_k t^k\right) = \sum_{k=0}^{\infty} e_k t^k ,$$

$$\sum_{k=0}^{\infty} e_k t^k = \frac{a + bt + t^2}{g(t)} = \frac{(g(t) - 1)/t}{g(t)} .$$

Hence,

$$\sum_{k=0}^{\infty} e_k t^{k+1} = 1 - \frac{1}{g(t)} \tag{5.4-19}$$

or

$$\frac{1}{g(t)} = 1 - \sum_{k=0}^{\infty} e_k t^{k+1} .$$

Comparing corresponding elements in (5.4-20) and (5.4-17) gives

$$d_k = -e_{k-1} . \tag{5.4-21}$$

Using (5.4-14), det A may be evaluated as

$$\det A = \prod_{r=0}^{n-1} \frac{e_r}{d_r} = (-1)^{n-1} e_{n-1} . \tag{5.4-22}$$

To obtain the sign of e_i from (5.4-20), we have

$$1 - \sum_{k=0}^{\infty} e_k t^{k+1} = \frac{1}{g(t)} = \frac{1}{t^3 f(1/t)} . \tag{5.4-23}$$

Now

$$\frac{1}{f(t)} = \frac{1}{(t + \alpha)\ (t + \beta)\ (t + \delta)}$$

where α, β, $\delta > 0$ for restricted a,b .

Hence

$$\frac{1}{f(\frac{1}{t})} = \frac{1}{(1/t + \alpha)\ (1/t + \beta)\ (1/t + \delta)} \quad .$$

Then

$$\text{(5.4-24)} \qquad \frac{1}{t^3\ f(1/t)} = \frac{1}{(1 + \alpha t)\ (1 + \beta t)\ (1 + \delta t)} = \sum_{k=0}^{\infty} \psi_k t^k \quad ,$$

where ψ_k is the coefficient of t^k when $1/(t^3 f(1/t))$ is expressed in power series form. It is evident that $\operatorname{sgn}(\psi_k) = (-1)^k$.

Comparing signs in (5.4-24) and (5.4-23), we have

$$\operatorname{sgn}(e_k) = (-1)^k \ .$$

This, with (5.4-22), gives $\det A > 0$. Every principal minor of A has the same form as A and hence is non-negative as well.

Q.E.D.

Lemma 1

The region where the cubic

$$f(t) = t^3 + at^2 + bt + 1 \qquad (a > b > 0)$$

has three negative real zeros is given by

$$\text{(5.4-25)} \qquad 4b^3 - a^2b^2 - 18ab + 27 + 4a^3 \le 0 \quad .$$

Proof:

If a and b are positive, then $f(t)$ has three negative zeros or one negative and two complex conjugate zeros.

$f(t)$ has real zeros provided that

$$q^3 + r^2 \le 0 \,,$$

where

$$q = \frac{b}{3} - \frac{a^2}{9}$$

and

$$r = \frac{1}{b}\,(ab - 3) - \frac{a^3}{27} \quad .$$

This gives the condition

$$\left(\frac{b}{3} - \frac{a^2}{9}\right)^3 + \left(\frac{1}{6}\,(ab - 3) - \frac{a^3}{27}\right) \le 0 \,,$$

which simplifies to give the result

$$4b^3 - a^2b^2 - 18ab + 27 + 4a^3 \le 0 \quad .$$

This region in the $a\ b$ plane is displayed in Figure 5.4.1; it is only the shaded region defined for $a > 0$, $b > 0$, that is of interest.

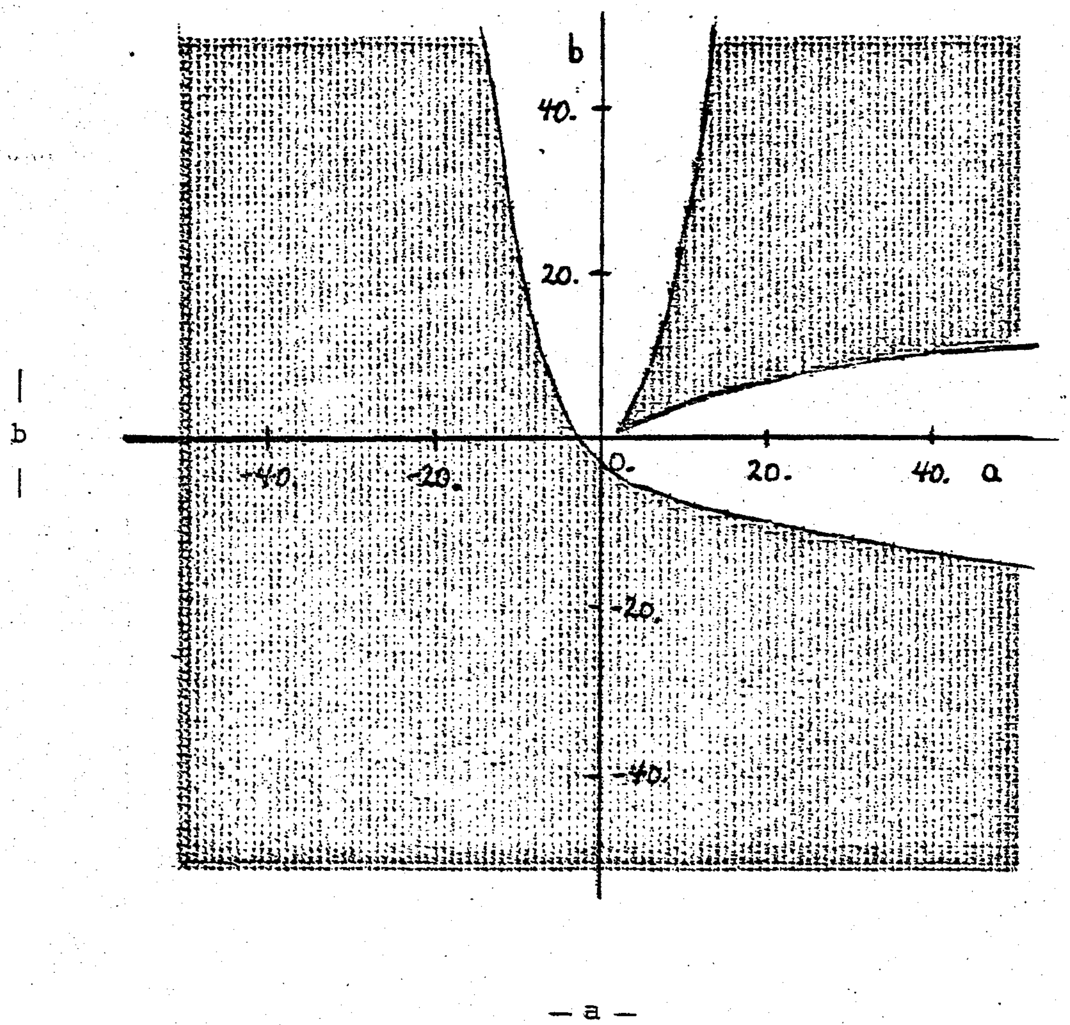


Figure 5.4.1

Theorem 5.4.3

The a_i, b_i, and c_i in the matrices T and L (Theorem 5.4.1) are all positive are all positive provided that $4b^3-a^2b^2-18ab+27+4a^3 \leq 0$.

Proof:

The determinant of Q^T is

$$\det \begin{pmatrix} b_1 & a_2 & 1 & & & \\ & b_2 & a_3 & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & 1 \\ & & & & \cdot & a_{n-1} \\ & & & & & b_{n-1} \end{pmatrix} = \prod_{i=1}^{n-1} b_i \; .$$

By Theorem 5.4.2, it is positive for Q of any order.

Hence, $b_i > 0$ for all i.

From Theorem 5.4.1, we have that $c_{i+1} = 1/b_i$, and so the c_i, excluding c_1 which is zero, are all positive.

In order to show that the a_i are positive, the matrix A is expressed in the form $A = TL^*$, where T is defined in (5.4-3) and L in (5.4-2). Let A_k denote the k'th order leading principal minor of A; then

$$A_k = a_k A_{k-1} - b_{k-1} A_{k-2} \quad .$$

By Theorem 5.4.2, A_k, A_{k-1}, $A_{k-2} > 0$. Since $b_{k-1} > 0$, it follows that

$$a_k > 0 \quad . \qquad \text{Q.E.D.}$$

Our final theorem is

Theorem 5.4.4

The matrix A^{-1} has an element a_{ij} in position (i,j) such that $a_{ij} = (-1)^{i+j} |a_{ij}|$ provided that a and b satisfy (5.4-25).

Proof:

Matrices T and L may be determined such that

$$A = TL^*$$

where $T \geq 0$ and $L^* \geq 0$, by Theorem (5.4.3), provided that a and b in A satisfy (5.4-25).

Let D be the matrix with diagonal entries

$$1, -1, 1, \ldots$$

and zeros elsewhere; then

$$DAD^{-1} = (DTD^{-1})\ (DL^*D^{-1}) \ .$$

The matrices DTD^{-1} and DL^*D^{-1} all satisfy the requirements of Theorem 5.2.1 since T and L are non-negative and tridiagonal(or of smaller band width). Hence, the elements in their inverses are non-negative and

$$DA^{-1}D^{-1} > 0$$

as well. This is only possible if

$$a_{ij} = (-1)^{i+j} \quad m_{ij} \quad . \qquad \text{Q.E.D.}$$

REFERENCES

Abramowitz, M. and Stegun, I.A. [1965], *Handbook of Mathematical Functions*, Dover Publications, Inc., New York.

Ahlberg, J.H., Nilson, E.N., and Walsh, J.L. [1967], *The Theory of Splines and Their Applications*. Academic Press, New York.

Albasiny, E.L. and Hoskins, W.D. [1969], *Cubic Spline Solution to Two-Point Boundary Value Problems*. Comp. J. 12, pp. 151-153.

Albasiny, E.L. and Hoskins, W.D. [1971], *Odd-degree Polynomial Splines With Equi-spaced Knots*. J. Inst. Math. Applics. 7, pp. 384-397.

Albasiny, E.L. and Hoskins, W.D. [1972], *Explicit Error Bounds for Periodic Splines of Odd Order on a Uniform Mesh*. Department of Computer Science, University of Manitoba Technical Report, No. 55.

Albasiny, E.L. and Hoskins, W.D., [1972b], *Increased Accuracy Cubic Spline Solutions to Two-point Boundary Value Problems*. J. Inst. Math. Applics. 9, pp. 47-55.

Andres, T.H., Hoskins, W.D., and McMaster, G.E. [1974], *A Coupled Algorithm for the Solution of Certain Tridiagonal Systems of Linear Equations*. Comp. J. 17, pp. 378-379.

Andrew, A.L. [1973], *Solution of Equations Involving Centrosymmetric Matrices*. Technometrics 15, No. 2, pp. 405-407.

Anselone, P.M. and Laurent, P.J. [1968], *A General Method for the Construction of Interpolating or Smoothing Spline Functions*. Num. Math. 12, pp. 66-82.

Atkinson, L.V. and Evans, D.J. [1970], *An Algorithm for the Solution of General Three Term Linear Systems*. Comp. J. 13, pp. 323-326.

Brandon, D.M. [1973], *The Implementation and Use of Sparse Matrix Techniques in General Simulation Programs*. Comp. J. 17, No. 2, pp. 165-171.

Businger , P. and Golub, G.H. [1965], *Linear Least Squares Solutions by Householder Transformations.* Num. Math. 7, pp. 269-276.

Buzbee, B.L., Golub, G.H., and Nielson, C.W. [1970], *On Direct Methods for Solving Posisson's Equations.* SIAM J. Numer. Anal. 7, No. 4, pp. 627-656.

Carasso, C. and Laurent, P.J. [1968], *On the Numerical Construction and the Practical Use of Interpolating Spline-Functions.* Proc. IFIP Congress, Edinburgh, A6-9.

Cody, W.J. [1973], *The Evaluation of Mathematical Software, in Program Test Methods.* Edited by William C. Hetzel, Prentice-Hall.

Cox, M.G. [1971], *Curve Fitting with Piecewise Polynomials.* J. Inst. Math. Applics. 8, No. 1, pp. 36-52.

Cox, M.G. [1972], *The Numerical Evaluation of B-Splines.* J. Inst. Maths. Applics. 10, pp. 134-149.

Cox, M.G. [1973a], *An Algorithm for Spline Interpolation.* National Physical Laboratory Report NAC 27, December.

Cox, M.G. [1973b], *A Data Fitting Package for the Non-Specialist User.* Software for Numerical Mathematics, edited by D.J. Evans, Academic Press.

Cox, M.G. [1975], *Private correspondence.*

Curry, H.B. and Schoenberg, I.J. [1966], *On Polya Frequency Functions IV. The Fundamental Spline Functions and Their Limits.* J. d'Analyse Math. 17, pp. 71-107.

Curtis, A.R. and Powell, M.J.D. [1967], *Using Cubic Splines to Approximate Functions of One Variable to Prescribed Accuracy.* AERE Report R5602, Harwell.

deBoor, C. [1968], *On the Convergence of Odd-Degree Spline Interpolation.* J. Approx. Theory 1, pp. 452-463.

de Boor, C. [1973], *On Calculating With B-Splines.* J. Approx. Theory 6, pp. 50-62.

Evans, D.J. [1972], *An Algorithm for the Solution of Certain Tridiagonal Systems of Linear Equations.* Comp. J. 15, No. 4, pp. 356-359.

Evans, D.J. and Forrington, C.V.D. [1962], *A Note on the Solution of Certain Tridiagonal Systems of Linear Equations.* Comp. J. 5, pp. 343-348.

Fischer, C.F. and Usmani, R.A. [1969], *Properties of Some Tridiagonal Matrices and Their Application to Boundary Value Problems.* SIAM J. Numer. Anal, 6. No. 1., pp. 127-142.

Flynn, M.J. [1966], *Very High Speed Computing Systems.* Proc. IEEE, Vol. 54, pp. 1901-1909.

Flynn, M.J. [1972], *Some Computer Organizations and Their Effectiveness.* IEEE. Trans. on Computers, Vol. C-21, No. 9, pp. 948-960.

Ford, W.S. [1975], *Periodic Cubic Spline Interpolation with Equidistant Nodes.* Comp. J. 18, No. 2., pp. 133-134.

Forsythe, G. and Moler, C.B. [1967], *Computer Solution of Linear Algebraic Systems.* Prentice-Hall.

Fox, L. (Editor), [1962], *Numerical Solution of Ordinary and Partial Differential Equations.* Pergamon Press, Oxford: Addison-Wesley, Reading, Mass.

Fyfe, D.J. [1969], *The Use of Cubic Splines in the Solution of Two-point Boundary Value Problems.* Comp. J. 12, pp. 188-192.

Fyfe, D.J. [1971], *Linear Dependence Relations Connecting Equal Interval* N^{th} *Degree Splines and Their Derivatives.* J. Inst. Math. Applics. 7, pp. 398-406.

Gabel, J.C. [1973], *A Description of Two Program Packages Which Manipulate Multi-Precise Integers.* University of Manitoba Scientific Report No. 64.

Good, I.J. [1970], *The Inverse of a Centrosymmetric Matrix.* Technometrics 12, pp. 925-928.

Golomb, M. [1968], *Approximation by Periodic Spline Interpolants on Uniform Meshes.* J. Approx. Theory 1, pp. 26-65.

Gregory, R.T. and Karney, D.L. [1969], *A Collection of Matrices for Testing Computational Algorithms.* Wiley-Interscience, New York.

Greville, T.N.E. [1967], *Spline Functions, Interpolation, and Numerical Quadrature in:* Ralston/Wilf. Mathematical Methods for Digital Computers, Vol. II., J. Wiley and Sons, pp. 156-168.

Greville, T.N.E. [1968], *Data Fitting by Spline Functions.* M.R.C. Tech. Summ. Report 893, Mathematics Research Centre, University of Wisconsin.

Greville, T.N.E. [1969], *Theory and Applications of Spline Functions.* Proc. Seminar Math. Res. Center, Univ. of Wisconsin, October, 1968. Ed. by T.N.E. Greville, Academic Press, New York.

Greville, T.N.E. [1972], *Spline Functions in Numerical Analysis.* Proc. Second Manitoba Conference on Numerical Mathematics, pp. 1-17.

Gryte, D.G. and Hoskins, W.D. [1971], *Some Multipoint Expansions for Odd-Degree Polynomial Splines with Equidistant Knots.* Proc. 25th Summer Meeting of the Canadian Mathematical Congress, p. 405.

Hall, C.A. [1967], *Error Bounds for Periodic Quintic Splines.* Communications of the ACM 12, pp. 450-452.

Hayes, J.G. [1973], *Available Algorithms for Curve and Surface Fitting.* Report NAC 39, National Physical Laboratory, Teddington (July).

Hayes, J.G. [1974a], *Numerical Methods for Curve and Surface Fitting.* NPL Report NAC 50, National Physical Laboratory, Teddington (April).

Hayes, J.G. [1974b], *New Shapes from Bicubic Splines.* NPL Report NAC 58, National Physical Laboratory, Teddington (September).

Hayes, J.G. and Halliday, J. [1974], *The Least-Squares Fitting of Cubic Spline Surfaces to General Data Sets*, J. Inst. Math, Applics. 14, No. 1.

Herriot, J.G. and Reinsch, C.H. [1971], *Algol 60 Procedures for the Calculation of Interpolating Natural Spline Functions.* Stanford Technical Report, STAN-CS-71-200.

Herriot, J.G. and Reinsch, C.H. [1973], *Procedures for Natural Spline Interpolation.* Communications of ACM 16, 12 , pp. 763-768.

Hoare, C.A.R. and Wirth, N. [1966], *A Contribution to the Development of ALGOL.* Communications of ACM 9, 6, pp. 413-431.

Holladay, J.A. [1957], *Smoothest Curve Approximation.* Math. Tables Aids Comp . 11, pp. 233-243.

Hoskins, W.D. [1970], *Studies in Spline Approximation and Variational Methods.* Ph.D. Thesis, Brunel University.

Hoskins, W.D. [1971], *Interpolating Quintic Splines on Equidistant Knots.* Comp. J. 13, pp. 437-438.

Hoskins, W.D. and King, P.R. [1972], *Periodic Cubic Spline Interpolation Using Parametric Splines (Algorithm 73).* Comp. J. 15, No. 3, pp. 282-283.

Hoskins, W.D., King, P.R., and Andres, T.H. [1972], *Interpolation Using Periodic Splines of ODD Order with Equidistant Knots (Algorithm 74).* Comp. J. 15, No. 3, pp. 283-285.

Hoskins, W.D. and McMaster, G.E. [1973], *A Smoothing Algorithm Using Piecewise Polynomials.* Presented at the Fourth Southeastern Conference on Combinatorics, Graph Theory and Computing, Boca Raton.

Hoskins, W.D. and McMaster, G.E. [1974], *An Economical Method of Solving Sets of Simultaneous Linear Equations with Coefficient Matrices of Band Structure and Symmetric about the Main and Skew Diagonals.* Presented at the Fourth Manitoba Conference on Numerical Mathematics, Oct. 2.

Hoskins, W.D. and McMaster, G.E. [1974], *A Note on Algorithm 74.* The Comp. J. 17, No. 1, pp. 91.

Hoskins, W.D. and McMaster, G.E. [1974b], *An Algorithm for the Rapid Calculation of Odd-Order Polynomial Splines with Equidistant Knots and Arbitrary Linear Boundary Equations.* Submitted to the Comp. J.

Hoskins, W.D. and McMaster, G.E. [1975], *On the Inverses of a Class of Toeplitz Matrices of Band Width Five.* Journal of Linear and Multi-Linear Algebra (to appear).

Hoskins, W.D. and Meek, D.S. [1971], *Iterative Solution of Equations Defining Odd-Degree Polynomial Splines with Equi-spaced Knots.* Proc. Twenty-fifth Canadian Mathematical Congress, Lakehead University, pp. 493-508.

Hoskins, W.D., McMaster, G.E. and Andres, T. [1974], *A Coupled Algorithm for the Solution of Certain Tridiagonal Systems of Linear Equations.* Comp. J. 17, No. 4., pp. 378-379.

Hoskins, W.D. and Ponzo, P.J. [1972], *Some Properties of a Class of Band Matrices.* Mathematics of Computation 26, No. 118, pp. 393-400.

Hoskins, W.D. and Thurgur, M.C. [1973], *Determinants and Norms for the Inverses of a Set of Band Matrices.* Utilitas Mathematica Vol. 3, pp. 33-47.

Kogge, P.M. and Stone, H.S. [1972], *An Algorithm for the Parallel Evaluation of a Class of Recurrence Relations.* Tech. Rept. 19, Digital Systems Laboratory, Stanford University, May 1972.

Lafata, P. and Rosen, J.B. [1970], *An Interactive Display for Approximation by Linear Programming.* Communications of ACM, pp. 651-659.

Loscalzo, F.R. and Talbot, T.D. [1967], *Spline Function Approximation for Solutions of Ordinary Differential Equations.* SIAM J. Num. Anal. 4, pp. 433-445.

Lyche, T. and Schumaker, L.L. [1971a], *Computation of Smoothing and Interpolating Natural Splines via Local Bases.* The University of Texas at Austin Technical Report, CNA-17.

Lyche, T. and Schumaker, L.L. [1971b], *Algol Procedures for Computing, Smoothing and Interpolating Natural Splines.* The University of Texas at Austin Technical Report, CNA-17.

Malcolm, M.A. and Palmer, J. [1974], *A Fast Method for Solving a Class of Tridiagonal Linear Systems.* Communications of ACM 17, No. 1., pp. 14-17.

Marsden, M.J. [1970], *An Identity for Spline Functions with Applications to Variation-Diminishing Spline Approximation.* J. Approx. Theory 3, pp. 7-49.

Meek, D. [1974], *On the Numerical Construction and Approximation of Some Piecewise Polynomial Functions.* Ph.D. Thesis, University of Manitoba.

Polya, G. [1931], *Bemerkung Fur Interpolation Und Fur Naherungstheorie der Balkenkigung.* Z. Angew. Math. Mech. 11, pp. 445-449.

Pooch, U.W. and Nieder, A. [1973], *A Survey of Indexing Techniques for Sparse Matrices.* Computing Surveys 15, No. 2, pp.109-133.

Powell, M.J.D. [1967], *The Local Dependence of Least Squares Cubic Splines.* TP308 AERE Report, Harwell.

Powell, M.J.D. [1969], *The Local Dependence of Least Squares Cubic Splines.* SIAM J. Numer. Anal. 6, pp. 398-413.

Ray, W.D. [1970], *The Inverse of a Finite Toeplitz Matrix.* Technometrics 12, pp. 153-156.

Reid, J.K. [1971], *Large Sparse Sets of Linear Equations.* Academic Press.

Rose, D.J. [1969], *An Algorithm for Solving a Special Class of Tridiagonal Systems of Linear Equations.* Communications of ACM 12, No. 4., pp. 234-236.

Rutherford, D.E. [1952], *Some Continuant Determinants Arising in Physics and Chemistry.* Proc. Roy. Soc. Edinburgh Sect. A., Vol. 63, 1952, pp. 232-241, MR 15, 495.

Schoenberg, I.J. [1946], *Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions.* Part A, Quart. Appl. Math. 4, pp. 45-99.

Schoenberg, I.J. and Whitney, A. [1953], *On Polya Frequency Functions III.* Trans. Amer. Math. Soc. 74, pp. 246-250.

Schoenberg, I.J. and Curry, H.B. [1966], *On Polya Frequency Functions IV, The Fundamental Spline Functions and Their Limits.* J. d'Analyse Math. 17, pp. 71-107.

Schoenberg, I.J. [1973], *Cardinal Spline Interpolation.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania.

Schumaker, L.L. [1969], *Some Algorithms for the Computation of Interpolating and Approximating Spline Functions.* Theory and Applications of Spline Functions. Ed. by T.N.E. Greville, Academic Press, New York.

Späth, H. [1969], *Die Numerische Berchnung Von Interpolierenden Splinefunktionen Mit Blockunturelaxation.* Doctoral Dissertation, Univ. of Karlsruhe.

Späth, H. [1971], *The Numerical Calculation of High Degree Lidstone Splines with Equidistant Knots by Blockunderrelaxation.* Computing 7, pp. 65-74.

Späth, H. [1974], *Spline Algorithms for Curves and Surfaces.* Translation by W. D. Hoskins and H. W. Sager, Utilitas Mathematica Publishing Inc., Winnipeg.

Stanton, R.G. and Sprott, D.A. [1962], *Some Finite Inversion Formulae.* Math. Gazette, V. 46, pp. 197-202.

Stone, H.S. [1973a], *An Efficient Parallel Algorithm for the Solution of Tridiagonal Linear Systems.* Journal of ACM 20, No. 1, pp. 27-38.

Stone, H.S. [1973b], *Problems of Parallel Computation.* Complexity of Sequential and Parallel Numerical Algorithms. Ed. by J.F. Traub, Academic Press.

Swartz, B. [1968], $O(h^{2n+2-\ell})$ *Bounds on Some Spline Interpolation Errors.* Los Alamos Scientific Laboratory Report No. LA-3886.

Traub, J.F. [1973], *Complexity of Sequential and Parallel Numerical Algorithms.* Proc. Symposium on Sequential and Parallel Numerical Algorithms, Carnegie-Mellon Univ., May 16. Ed. by J.F. Traub, Academic Press, New York.

Varga, R.S. [1962], *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey.

Whiten, W.J. [1972], *The Use of Periodic Spline Functions for Regression and Smoothing.* Australian Computer Journal 4, No. 1, pp. 31-34.

Wilkinson, J.H. [1963], *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, New Jersey.

Wilkinson, J.H. [1965], *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford.

Wilkinson, J.H. and Reinsch. C. [1971], *Linear Algebra.* Springer-Verlag, New York.