NASA TECHNICAL NOTE

NASA TN D-6401

C. 1

NASA TN D-6401

FITLOS: A FORTRAN PROGRAM FOR FITTING LOW-ORDER POLYNOMIAL SPLINES BY THE METHOD OF LEAST SQUARES

by Patricia J. Smith

Lewis Research Center

Cleveland, Ohio 44135

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • AUGUST 1971

1. Report No. NASA TN D-6401	2. Government Accession No.	3. Recipient's
4. Title and Subtitle FITLOS: A FORTRAN PROGRAM FOR FITTING LOW-ORDER POLYNOMIAL SPLINES BY THE METHOD OF LEAST SQUARES		5. Report Date August 1971
		6. Performing Organization Code
7. Author(s) Patricia J. Smith		8. Performing Organization Report No. E-5292
		10. Work Unit No. 129-04
9. Performing Organization Name and Address Lewis Research Center, National Aeronautics and Space Administration, Cleveland, Ohio 44135		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration, Washington, D. C. 20546		13. Type of Report and Period Covered Technical Note
		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

FITLOS is a FORTRAN IV program to fit polynomial splines of degrees two and three. It combines some of the advantages of the method of least squares with the segmented curve of the theory of splines. FITLOS divides a set of data points into subsets and fits a polynomial of degree two or three on each subset by the method of least squares. The total curve is made smooth by making the polynomials on adjacent subsets and their first derivatives equal at the break point between the segments. For third-degree polynomials, the second derivatives are also made equal. These constraints are imposed by the method of Langrangian multipliers. This report describes the mathematical analysis of the least squares polynomial spline fit, gives complete documentation of program FITLOS, and serves as a user's guide for FITLOS. The program listing, flow charts, and example problems are included.

17. Key Words (Suggested by Author(s)) Curve fitting; Spline function; Least squares polynomials	18. Distribution Statement Unclassified - unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	93	$3.00

*For sale by the National Technical Information Service, Springfield, Virginia 22151

CONTENTS

FITLOS: A FORTRAN PROGRAM FOR FITTING LOW-ORDER POLYNOMIAL SPLINES BY THE METHOD OF LEAST SQUARES

by Patricia J. Smith

Lewis Research Center

SUMMARY

FITLOS is a FORTRAN IV program to fit polynomial splines of degrees two and three. It combines some of the advantages of the method of least squares with the segmented curve of the theory of splines. FITLOS divides a set of data points into subsets and fits a polynomial of degree two or three on each subset by the method of least squares. The total curve is made smooth by making the polynomials on adjacent subsets and their first derivatives equal at the break point between the segments of the curve. For third-degree polynomials, the second derivatives are also made equal. These constraints are imposed by the method of Lagrangian multipliers.

FITLOS was written to complement other types of curve-fitting programs. This report describes the mathematical analysis of the least squares polynomial spline fit, gives complete documentation of the program FITLOS, and is intended to serve as a user's guide for FITLOS. To augment this last purpose, the report includes examples of problems for which this type of curve-fit is useful.

INTRODUCTION

FITLOS was written to complement other curve-fitting programs. A new method of curve-fitting was needed that would combine some of the advantages of a least squares polynomial with the segmented curve of the theory of splines. Segmenting the curve gives it more freedom than a single polynomial over the entire range of the data, while fitting by the method of least squares smooths any small fluctuations in the data.

The name "spline" is derived from the draftsman's spline which is used to fair curves. Like the draftsman's spline, the spline function is smooth. DeBoor's definition of a spline function is used for this report (ref. 1). It is as follows: A function f(x) is a spline function of degree M with joints $x_1 < x_2 < \cdots < x_n$ if it has these two properties:

(1) In each of the intervals $(-\infty, x_1)$, $[x_1, x_2)$, . . . , $[x_{n}, \infty)$, f(x) is a polynomial of degree M.

(2) The first M - 1 derivatives are continuous.

In FITLOS, the continuity of the curve and its derivatives is imposed by the method of Lagrangian multipliers (ref. 2).

The use of low-degree polynomials has two advantages. First, they have relatively few local maxima and minima. Second, they are easily differentiated and integrated. Second-degree polynomials have a third advantage; namely, their roots are easily found. Consequently, a FITLOS curve fit can be used readily for further applications.

This report is intended to serve three purposes. First, it describes the details of the mathematical analysis of the least squares polynomial spline fit. Second, it presents the program FITLOS, which makes this type of curve fit, and gives instructions for using the program. Third, it presents two problems for which the least squares polynomial spline fit is applicable and compares the results with fits made by other methods.

Notation in the section MATHEMATICAL DERIVATION follows conventions in standard mathematics textbooks. Involved proofs and mathematical details are given in the appendixes.

To clarify the vocabulary, the word "order" refers to the sequence of points or numbers, while the word "degree" refers to the highest power of x in the polynomials. For example, the values $x_1 < x_2 < x_{NX}$ are in order, while FITLOS fits polynomials of degree two or three. The difference between "subsets" and "segments" is a little more subtle. The set of data points is divided into subsets, while the fitted curve is divided into segments. However, the subsets of data correspond to the segments of the curve.

MATHEMATICAL DERIVATION

Curve Fit

Consider a set of NX data points $Z = \{(x_i, y_i) \mid i = 1, 2, \ldots, NX\}$ where $x_i < x_{i+1}$. For a weighted least squares polynomial fit of degree M, a matrix **X** can be defined:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 1 & x_{NX} & x_{NX}^2 & \cdots & x_{NX}^M \end{bmatrix}$$

Let W be the matrix of weights which has only diagonal elements,

$$W = \text{diag}\ (w_1,\ w_2,\ .\ .\ .\ .\ ,\ w_{NX})$$

Let Y be the column vector

$$Y = \text{col}\ (y_1,\ y_2,\ .\ .\ .\ .\ ,\ y_{NX})$$

where the y's have the same order as the x's in the matrix X. Let A be the column vector of undetermined coefficients. Then let Y^* be the column vector such that $Y^* = XA$. For a weighted least squares fit, the scalar

$$\epsilon = (Y^* - Y)^T W(Y^* - Y)$$

must be minimized with respect to each element of A.

The weighted least squares polynomial spline fit can be described in a similar manner. First, however, a set of spline joints XM must be defined. Let XM be the set of x-values of the break points between the NS segments of the curve $XM = \left\{(xm)_n \mid n = 1,\ 2,\ .\ .\ .\ .\ ,\ NS - 1\right\}$. Now set Z can be divided into NS subsets such that

$$Z_1 = \left\{(x_i,\ y_i) \mid x_1 \le x_i \le (xm)_1\right\}$$

$$Z_2 = \left\{(x_i,\ y_i) \mid (xm)_1 \le x_i \le (xm)_2\right\}$$

.

.

.

$$Z_{NS} = \left\{(x_i,\ y_i) \mid (xm)_{NS-1} \le x_i \le x_{NX}\right\}$$

For convenience, two sets of data point indices can be defined. Let F be the set of indices of the first data point in each subset $F = \{F_n\}$, where F_n is the smallest i such that (x_i, y_i) is an element of Z_n. Similarly, let L be the set of indices of the last data point in each subset $L = \{L_n\}$, where L_N is the largest i such that (x_i, y_i) is an element of Z_n. From these definitions it can be seen that if any of the $(xm)_n$ is an x-value of a data point, that data point is the last point in the n^{th} subset and the first point in the $(n + 1)^{th}$ subset. However, the (xm) do not have to correspond to data points.

When the data have been divided into subsets, a matrix X can be defined which is composed of submatrices X_{ij} such that

$$X_{ij} = \begin{bmatrix} 1 & x_{F_i} & \cdots & x_{F_i}^M \\ 1 & x_{F_i+1} & \cdots & x_{F_i+1}^M \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ 1 & x_{L_i} & \cdots & x_{L_i}^M \end{bmatrix}$$

for $j = i$, and X_{ij} is null for $j \neq i$. Matrix X has NS nonzero rectangular block submatrices on its diagonal and null submatrices elsewhere. The notation can be simplified a little at this point by dropping the second subscript on the submatrices of X since only diagonal elements are present.

$$X = \text{diag}\,(X_1, X_2, \ldots X_{NS})$$

Similarly, let Y be a column vector which is composed of NS subvectors Y_i of the form

$$Y_i = \text{col}\,(y_{F_i}, y_{F_i+1}, \ldots, y_{L_i})$$

Vector Y has the form

$$Y = \text{col}\,(Y_1, Y_2, \ldots, Y_{NS})$$

Let W be the matrix of weights which is composed of square submatrices W_{ij} of the form

$$W_{ij} = \text{diag}\,(w_{F_i}, w_{F_i+1}, \ldots, w_{L_i})$$

for $j = i$, and W_{ij} is null for $j \neq i$. Again dropping the second subscript, W has the form

$$W = \text{diag}\,(W_1, W_2, \ldots, W_{NS})$$

Let A be a column vector of undetermined coefficients composed of NS subvectors of the form

$$A_i = \text{col}\,(a_{i1}, a_{i2}, \ldots, a_{i,M+1})$$

Vector A has the form

$$A = \text{col}\,(A_1, A_2, \ldots, A_{NS})$$

Let Y^* be the column vector defined by the matrix product

$$Y^* = XA$$

The scaler

$$\epsilon = (Y^* - Y)^T W(Y^* - Y)$$

must be minimized with respect to each element of A, but subject to the constraints that the first M - 1 derivatives of Y^* must be continuous at the break points between the segments of the curve. These constraints can be expressed in matrix form by defining the matrix C which is composed of submatrices

$$C_{ij} = \begin{bmatrix} 1 & (xm)_i & (xm)_i^2 \\ 0 & 1 & 2(xm)_i \end{bmatrix}$$

for a quadratic fit, and

$$C_{ij} = \begin{bmatrix} 1 & (xm)_i & (xm)_i^2 & (xm)_i^3 \\ 0 & 1 & 2(xm)_i & 3(xm)_i^2 \\ 0 & 0 & 2 & 6(xm)_i \end{bmatrix}$$

for a cubic fit for $j = i$. For $j = i + 1$, $C_{ij} = -C_{i,j-1}$. For other combinations of i and j, C_{ij} is null. Again dropping the second subscript, C has the form

$$C = \begin{bmatrix} C_1 & -C_1 & 0 \rightarrow & & & \\ 0 & C_2 & -C_2 & & & \\ \downarrow & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & \cdot \\ & & & & C_{NS-1} & -C_{NS-1} \end{bmatrix}$$

The constraints take the form

$$CA = 0 \tag{1}$$

The set of Langrangian multipliers can be introduced as a row vector Λ composed of NS - 1 subvectors Λ_i of the form

$$\Lambda_i = (\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{iM})$$

Vector Λ has the form

$$\Lambda = (\Lambda_1, \Lambda_2, \ldots, \Lambda_{NS-1})$$

The scalar ϵ then becomes

$$\epsilon = (Y^* - Y)^T W(Y^* - Y) + \Lambda CA$$

and must be minimized with respect to each element of A

Substituting for Y^*

$$\epsilon = (XA - Y)^T W(XA - Y) + \Lambda CA$$

To minimize ϵ with respect to a_{ij}, the derivative of ϵ with respect to a_{ij} is set to zero; that is,

$$0 = \frac{\partial \epsilon}{\partial a_{ij}} = \frac{\partial (XA - Y)^T}{\partial a_{ij}} W(XA - Y) + (XA - Y)^T W \frac{\partial (XA - Y)}{\partial a_{ij}} + \Lambda C \frac{\partial A}{\partial a_{ij}}$$

Since

$$\frac{\partial (XA - Y)^T}{\partial a_{ij}} = \left[\frac{\partial (XA - Y)}{\partial a_{ij}}\right]^T$$

$W = W^T$, and a scalar is equal to its own transpose, we have

$$\frac{\partial (XA - Y)^T}{\partial a_{ij}} W(XA - Y) = (XA - Y)^T W^T \frac{\partial (XA - Y)}{\partial a_{ij}} = (XA - Y)^T WX \frac{\partial A}{\partial a_{ij}}$$

Therefore,

$$\left[2(XA - Y)^T WX + \Lambda C\right] \frac{\partial A}{\partial a_{ij}} = 0 \qquad (2)$$

Equations (1) and (2) must be solved for A and Λ. Since $\partial A/\partial a_{ij} \neq 0$ for any a_{ij}, it must be true that

$$2(XA - Y)^T WX + \Lambda C = 2A^T(X^T WX) - 2(Y^T WX) + \Lambda C = 0 \qquad (3)$$

Since the matrix $X^T WX$ has an inverse, right multiplying by $(X^T WX)^{-1}$, dividing by the scalar 2, and separating the unknowns A^T and Λ gives

$$A^T + \frac{1}{2} \Lambda C \, (X^T WX)^{-1} = (Y^T WX)(X^T WX)^{-1} \qquad (4)$$

The proof that X^TWX has an inverse is given in appendix A.

Since $CA = 0$, $(CA)^T = A^TC^T = 0$. Right multiplying equation (4) by C^T gives

$$\frac{1}{2} \Lambda C(X^TWX)^{-1}C^T = (Y^TWX)(X^TWX)^{-1}C^T$$

Since the matrix $C(X^TWX)^{-1}C^T$ has an inverse, right multiplying by $\left[C(X^TWX)^{-1}C^T\right]^{-1}$ gives

$$\frac{1}{2} \Lambda = (Y^TWX)(X^TWX)^{-1}C^T \left[C(X^TWX)^{-1}C^T\right]^{-1}$$

The proof that $C(X^TWX)^{-1}C^T$ has an inverse is also given in appendix A.

Substituting for $1/2\ \Lambda$ in equation (4) and solving for A^T gives

$$A^T = (Y^TWX)\left\{I - (X^TWX)^{-1}C^T \left[C(X^TWX)^{-1}C^T\right]^{-1} C\right\}(XWX)^{-1} \tag{5}$$

The details of this matrix manipulation and a method of finding $\left[C(X^TWX)^{-1}C^T\right]^{-1}$ is given in appendix B.

Statistical Analysis

FITLOS makes a rudimentary statistical analysis of the curve-fit. It calculates the deviation and relative error between the given data and the fitted curve, the variance and the standard deviation, and Pearson's correlation coefficient. The formulas were taken from reference 3, but they are standard in any statistics textbook. The formulas are as follows:

Deviation:

$$d_i = y_i^* - y_i$$

Relative error:

$$e_i = \frac{d_i}{y_i^*}$$

Variance:

$$\sigma_2 = \frac{\sum_{i=1}^{NX} (d_i - \bar{d})^2}{F}$$

where

$$\bar{d} = \frac{\sum_{i=1}^{NX} d_i}{NX}$$

and

$$\begin{aligned} F &= \text{Number of degrees of freedom} \\ &= \text{Number of points - Number of constraints} \\ &= NX - M(NS - 1) \end{aligned}$$

Standard deviation:

$$\sigma = \sqrt{\sigma^2}$$

Correlation coefficient:

$$r = \frac{1}{NX} \sum_{i=1}^{NX} \left(\frac{y_i - \bar{y}}{\sigma_y} \right) \left(\frac{y_i^* - \bar{y}^*}{\sigma_{y^*}} \right)$$

$$= \frac{F}{NX} \frac{NX \sum_{i=1}^{NX} y_i y_i^* - \left(\sum_{i=1}^{NX} y_i \right) \left(\sum_{i=1}^{NX} y_i^* \right)}{\sqrt{\left[NX \sum_{i=1}^{NX} y_i^2 - \left(\sum_{i=1}^{NX} y_i \right)^2 \right] \left[NX \sum_{i=1}^{NX} (y_i^*)^2 - \left(\sum_{i=1}^{NX} y_i^* \right)^2 \right]}}$$

Since the number of constraints is large, and hence, the number of degrees of freedom is small, the correlation coefficient can be deceptively small. For this reason, FITLOS also calculates the maximum possible correlation coefficient, which is r for $y_i^* = y_i$ for all i. The maximum r is equal to F/NX.

GENERAL DESCRIPTION OF PROGRAM

FITLOS was written in FORTRAN IV for the computer at the Lewis Research Center, which is an IBM 7094 II/7044 or 7040 Direct Couple computer under IBSYS version 13 using ALTIO.

Computer storage for the program as it is presented here, with 350 data points and 10 spline joints, is around 20 000 locations. Since the Lewis computer has 32 000 locations, the program could be expanded to fit more data points or to fit the curve in more segments.

The program is written as a series of subroutines so the actual curve-fitting routine could be used as part of another program. The actual fit requires only three subroutines: one to divide the data into subsets, one to define the matrices in equation (5), and one to solve equation (5).

In order to make the subroutines as flexible as possible, their arrays have variable dimensions. To conserve execution time, every subprogram with variably dimensioned arrays is called only once by its subroutine name. These calling vectors contain only the array names and the dimensions of the corresponding arrays in the main program FITLOS. Afterwards, the subroutines are called by entry names which do not disturb the size of the variably dimensioned arrays set up by the first call by the subroutine names.

The main program FITLOS reads input data, calls the subroutines to make the fit (see tree diagram for hierarchy of subroutines, fig. 1), makes a statistical analysis of the fitted curve, and writes the output data.

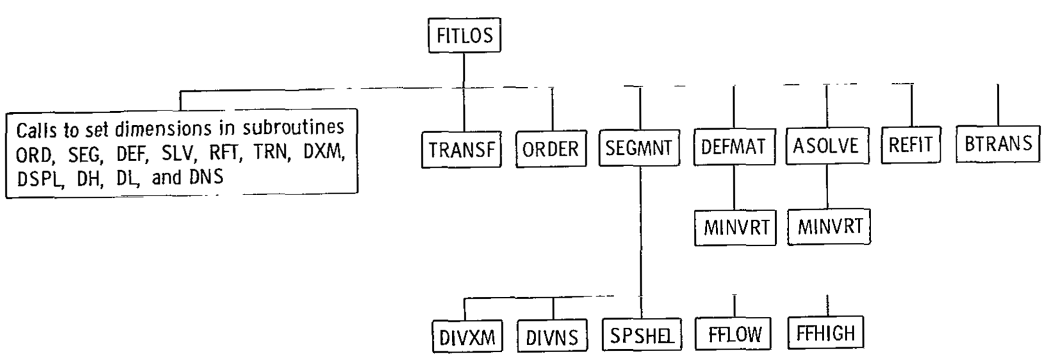


Figure 1. - Tree diagram of subroutine calls.

FITLOS uses the following procedure to fit a curve: It reads input data, which are described in the section INPUT DATA. After the data are read, FITLOS checks logical variables TRANX and TRANY. If either is .TRUE., subroutine TRANSF is called to make a log transformation on x or y.

The next subroutine called is ORDER, which arranges the data points in order of ascending x. The next subroutine called is SEGMNT, which divides the data into subsets. SEGMNT is a monitoring routine which controls calls to small subroutines (DIVXM, DIVNS, FFLOW, FFHIGH, and SPESHL) which actually allot the data to subsets. There are four methods of dividing the data into subsets. These are described in the section HOW DATA ARE DIVIDED INTO SUBSETS.

When the data have been divided into subsets, subroutine DEFMAT is called to define the matrices $(X^TWX)^{-1}$, Y^TWX, and C. Then subroutine ASOLVE is called to perform the matrix manipulation involved in solving equation (5).

FITLOS can check whether the curve was fit in more segments than were necessary. If the logical variable LREFIT is .TRUE., subroutine REFIT is called to do this checking. When the refit checking is complete, FITLOS again interrogates logical variables TRANX and TRANY. If either is .TRUE., subroutine BTRANS is called to transform the data back to its original form. Then the statistical analysis is made and the output data are written with descriptive labels and headings.

If REFIT has indicated there were too many segments in the first fit, new spline joints are determined, subroutines DEFMAT and ASOLVE are called again, a new statistical analysis is made, and the new output data are printed.

A listing of each subroutine, along with a flow chart and a description of its operation, is provided in appendix E. Variable names and their limitations or special features are found in the program listings. More details of how FITLOS works can be found in the section INPUT DATA.

HOW DATA ARE DIVIDED INTO SUBSETS

FITLOS provides four methods of dividing the data into subsets. The user determines which method is used by proper setting of the input variables.

The user has the option of selecting the spline joints, of selecting the number of segments, or of choosing one of the methods the program does automatically.

If the user selects the spline joints, he must supply these data as part of the input. Then subroutine DIVXM searches the x-array to determine the index of the first and last point in each subset.

If the user chooses the number of segments, subroutine DIVNS is called to divide the data as evenly as possible among the subsets. If the user does not specify the number of segments, the program will automatically choose the largest possible number of segments based on the number of data points and the degree of the polynomials. Again, subroutine DIVNS is used to divide the data into subsets.

If the user does not specify either the spline joints or the number of segments, subroutine SEGMNT checks the number of data points. If there are less than $3M + 1$ points, subroutine SPESHL is called to make a special division of the data into subsets. For $M = 2$, the division is as follows:

Index of first points	Index of last points	Number of subsets	Spline joints
1	3 or 4	1	x_3 or x_4
1, 3	3, 5	2	x_3, x_5
1, 3	4, 6	2	$\frac{x_3 + x_4}{2}, x_6$

For $M = 3$, SPESHL divides the data as follows:

Index of first points	Index of last points	Number of subsets	Spline joints
1	4, 5, or 6	1	$x_4, x_5,$ or x_6
1, 4	4, 7	2	x_4, x_7
1, 4	5, 8	2	$\frac{x_4 + x_5}{2}, x_8$
1, 5	5, 9	2	x_5, x_9

The final method of dividing the data into subsets is by force-fitting. The first $M + 1$ data points are used to determine a Lagrange interpolation polynomial. If the next point, the $(M + 2)^{th}$ point, falls on the polynomial, it is accepted in the first subset. Then the next point is examined, and so on to the end of the set of data points. If a point does not fall on the polynomial, a new subset is started with the next $M + 1$ points. There are two subroutines to do a force-fit division of the data. FFLOW starts at the low end of the data set and FFHIGH starts at the high end.

INPUT DATA

Input to FITLOS is by punched cards. The order of these cards, their formats, the variables they contain, and the use of these variables in controlling how the curve is fit are as follows:

Card	Format of card	Variable	Description
1	(12A6)	TITLE	Alphanumeric identification of the data. The title must be confined to columns 1 to 72 of one card.
2	(5I3,4L3,F12.6)	M	Degree of the polynomial. M must be 2 or 3.
		NX	Number of (x, y, w) data points.
		NS	Number of segments if the user selects the number of segments. NS $\neq 0$ means the curve will be fit in NS segments. If NS = 0, the program will select the largest possible number of segments.
		NB	Number of spline joints. NB $\neq 0$ indicates the user has selected the spline joints and these data will be read as part of the input data. NB = 0 means the program will set NB = NS - 1.
		NF	Numerical variable which indicates whether force-fitting is used to divide the data into subsets. If NF < 0, the data are divided into subsets by force-fitting starting at the low end of the data. If NF > 0, the data are divided by force-fitting starting at the high end of the data. If NF = 0, the program divides the data as evenly as possible among the maximum possible number of subsets.

Card	Format of card	Variable	Description
2	(5I3,4L3,F12.6)	LREFIT	Logical variable which indicates if FITLOS should check whether the curve was fit in more segments than were necessary. LREFIT = .TRUE. means a check should be made. LREFIT = .FALSE. means no check should be made. The write-up of subroutine REFIT gives details of how the check is made.
		TRANX	Logical variable which indicates if a log transformation should be made on x and (xm). TRANX = .TRUE. means the transformation should be made. TRANX = .FALSE. means the transformation should not be made.
		TRANY	Logical variable which indicates if a log transformation should be made on y and y*. TRANY = .TRUE. means the transformation should be made. TRANY = .FALSE. means the transformation should not be made.
		NPUNCH	Logical variable which indicates if cards containing the coefficients should be punched. NPUNCH = .TRUE. means no cards should be punched. NPUNCH = .FALSE. means cards should be punched with all the coefficients for one segment on a card.
		TOL	Tolerance acceptable for refit checking or for force-fitting. Details of how TOL is used are found in the descriptions of subroutine FFLOW.

Card	Format of card	Variable	Description
3	(12A6)	FMT	Variable format for reading (x, y, w) data points.
4 to 3 + n (n = number of data cards)	(FMT)	X Y W	Independent variable array. Dependent variable array. Array of weights. Since FITLOS makes a weighted least squares fit, each point must have a weight. However, if all the weights are zero, FITLOS will make all the weights 1.

The (x, y, w) data are read in the order (x_1, y_1, w_1), (x_2, y_2, w_2), . . . , (x_{NX}, y_{NX}, w_{NX}). If NB ≠ 0, the following data are needed:

Card	Format of card	Variable	Description
4 + n	(12A6)	FMTM	Variable format for reading the spline joints selected by the user.
5+n to 3+n+m (m = number of spline joint cards)	(FMTM)	XM	Array of spline joints. If NB = 0, FMTM and XM are not needed.
4 + n + m	(I3)	KASES	The number of additional fits to be made with the current (x, y, w) data. KASES is originally set to zero by FITLOS so a title card and (x, y, w) data are read in. If the KASES card contains zero or is blank, FITLOS will transfer to read a new title card and new (x, y, w) data. If KASES ≠ 0, FITLOS will transfer to read a new card 2. KASES is reduced by 1 each time a new card 2 is read in until KASES finally becomes 0.

Variables NS and NB are not independent. FITLOS interrogates NB to determine if more input cards should be read. Subroutine SEGMNT interrogates NB first. If NB ≠ 0, NS is set equal to NB + 1, and the division into subsets is based on NB and the chosen spline joints. If NB = 0, SEGMNT interrogates NS. If NS ≠ 0, NB is set equal to NS - 1, and the division into subsets is based on NS. If both NB and NS are zero, SEGMNT sets NS equal to the maximum possible number of subsets and then interrogates NF. More specific details of how these input variables are used can be found in the descriptions of the individual subroutines.

TYPICAL APPLICATIONS

One typical application for FITLOS is fitting experimental data. An example of this is the calibration of a multiplier phototube-capacitor, where the independent variable is time and the dependent variable is digitizer counts. Data from several different light sources are translated until the curves coincide as nearly as possible. Since the curves do not coincide exactly, there are small fluctuations in the data. For such a calibration to be useful, these fluctuations must be eliminated.

Obviously, any least squares fit would do that. However, fitting this curve with a single least squares polynomial of degree one, two, or three did not give satisfactory results. Figures 2 to 4 (appendix F) show the relatively large deviations between the data points and the fitted curve. The curve was then fit using FITLOS with three polynomials of degree two. The deviations between the data points and the fitted curve are sufficiently small, as figure 5 shows.

The curves in figures 2 to 5 (appendix F) are plotted on a log-log scale to emphasize these deviations. The plots of the deviations are made on a semi-log scale because they are both positive and negative. The computer listings from which these plots were made and the computer input sheet for the FITLOS fit can be found in appendix F.

Another application for FITLOS is approximating a curve to obtain further information about it, such as the derivative and the definite integral. The source of the data points is immaterial. They could be experimental data points or they could be generated from some complicated function. The points for this example were generated from the function

$$f(x) = x \sin x - 1$$

This function was chosen for the example because it is not a polynomial and yet it is simple enough to be differentiated and integrated analytically for comparison with the results from FITLOS. The derivative of $f(x)$ is

$$f'(x) = x \cos x + \sin x$$

and the definite integral is

$$\int_{x_o}^{x_f} f(x)dx = (\sin x - x \cos x - x) \Big|_{x_o}^{x_f}$$

For finding the derivative and the definite integral using a FITLOS curve, the third-degree polynomials yield smoother curves. The derivative is

$$y^{*\prime} = a_2 + 2a_3 x + 3a_4 x^2$$

The integral is a little more complicated because each segment of the curve must be integrated separately. Consequently, the definite integral takes the form

$$\int_{x_o}^{x_f} y^* \, dx = \int_{x_o}^{(xm)_i} y_i^* \, dx + \sum_{n=i+1}^{i+N} \int_{(xm)_{n-1}}^{(xm)_n} y_n^* \, dx + \int_{(xm)_{i+N}}^{x_f} y_{i+N}^* \, dx$$

where i is the number of the first spline joint such that $(xm)_{i-1} < x_o < (xm)_i$, and N is the number of additional segments such that $i + N \leq NS$ and $(xm)_{i+N} < x_f$. Tables I to III (appendix G) compare the FITLOS curve y^* with f(x), $y^{*\prime}$ with f'(x), and $\int_{x_o}^{x_f} y^* \, dx$ with $\int_{x_o}^{x_f} f(x) \, dx$. Figures 6 to 8 (appendix G) are plots of the data in these three tables.

To determine the roots of this curve, the curve should be fitted with second-degree polynomials. The roots of y^* can be found by the quadratic formula. The roots of f(x) can be found numerically (by the Newton-Raphson method) for comparison. The following table compares the Newton-Raphson roots with the FITLOS roots:

Newton-Raphson root	FITLOS root	Deviation
1.1141571	1.1143261	-0.0001690
2.7726047	2.7714741	0.0011306

The computer listings and the input sheet for FITLOS for this example are presented in appendix G.

Another application for FITLOS, one that is shared by all curve-fitting schemes, is generating points for mechanical plotting. Automatic plotting devices such as the Calcomp Plotter or the DD80 Microfilm Plotter require a method of generating points close together. Figures 2 to 5 illustrate this application, since these plots were done on the Calcomp Plotter at the Lewis Research Center. Figures 6 to 8 were done on the DD80 Microfilm Plotter.

CONCLUDING REMARKS

This report has described the mathematical analysis of the least squares polynomial spline method of curve fitting; has presented the FORTRAN program FITLOS, which makes this type of curve fit; and is intended to serve as a user's guide for FITLOS. The sample problems included show problems for which this type of curve fit is useful. They also show how the curve fit may be used for further applications such as integration, differentiation, root finding, and plotting.

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, March 23, 1971,
129-04.

APPENDIX A

PROOF THAT MATRICES X^TWX AND $C(X^TWX)^{-1}C^T$ HAVE INVERSES

Matrix X^TWX

Let X and W be the matrices in the main-text section Curve Fit. Since X is block diagonal and W is diagonal, the product matrix X^TWX is block diagonal with diagonal blocks of the form

$$(X^TWX)_i = X_i^T W_i X_i$$

$$= \begin{bmatrix} \sum_{k=F_i}^{L_i} w_k & \sum_{k=F_i}^{L_i} w_k x_k & \cdots & \sum_{k=F_i}^{L_i} w_k x_k^M \\ \vdots & & & \vdots \\ \sum_{k=F_i}^{L_i} w_k x_k^M & \cdots & & \sum_{k=F_i}^{L_i} w_k x_k^{2M} \end{bmatrix}$$

Therefore, $(X^TWX)^{-1}$, if it exists, is block diagonal with diagonal blocks $(X_i^TW_iX_i)^{-1}$.

Let U be the diagonal matrix

$$U = \sqrt{W_i} = \text{diag}\left(\sqrt{w_{F_i}}, \ . \ . \ . \ , \sqrt{w_{L_i}}\right)$$

Then $(X^TWX)_i$ becomes $(X^TWX)_i = X_i^TU^TUX_i$. If P is defined as $P = UX_i$, then $(X^TWX)_i = P^TP$.

Since the leading principal minor of X_i is Vandermonde of order $M + 1$ and since none of the X_i are equal, by the definition of the spline function, X_i has rank $M + 1$. Since premultiplying X_i by the nonsingular matrix U does not change the rank, the product P has rank $M + 1$, by theorem 5.6.3 of reference 4. The matrix $(X^TWX)_i = P^TP$ then has rank $M + 1$, by theorem 5.5.4 of reference 4. Therefore, since $(X^TWX)_i$ has dimension $(M + 1) \times (M + 1)$ and has rank $M + 1$, it is nonsingular. Consequently, the inverse $(XWX)_i^{-1}$ exists.

Since $(X^TWX)_i$ is defined for all i, all the submatrices $(X^TWX)_i^{-1}$ exist and, hence, the entire inverse $(X^TWX)^{-1}$ exists.

Matrix $C(X^TWX)^{-1}C^T$

Let $(X^TWX)^{-1}$ be the inverse matrix found in the preceding section. Let C be the matrix of constraints defined in the main-text section Curve Fit. Since C_i has rank M and since there are NS - 1 rows of blocks in C, the rank of C is (M)(NS - 1).

It was shown that X^TWX is positive definite since it can be decomposed into the form

$$(X^TWX) = P^TP$$

Consequently, its inverse $(X^TWX)^{-1}$ is also positive definite. Therefore, $(X^TWX)^{-1}$ possesses a positive definite square root Q (see pp. 92 to 93 of ref. 5), and $(X^TWX)^{-1}$ can be written as

$$(X^TWX)^{-1} = QQ^T$$

where Q has the same rank as $(X^TWX)^{-1}$, which is (M + 1)(NS). Therefore, the matrix $C(X^TWX)^{-1}C^T$ can be written as

$$C(X^TWX)^{-1}C^T = CQQ^TC^T$$

or as

$$C(X^TWX)^{-1}C^T = PP^T$$

where P = CQ. Since postmultiplying C by the nonsingular matrix Q leaves the rank of the product unchanged, P has the same rank as C, which is (M)(NS - 1), by theorem 5.6.3 of reference 4. The matrix $C(X^TWX)^{-1}C^T = PP^T$ then has the rank (M)(NS - 1) by theorem 5.5.4 of reference 4.

Since $C(X^TWX)^{-1}C^T$ also has dimension $(M)(NS - 1) \times (M)(NS - 1)$, it is nonsingular. Therefore, its inverse $\left[C(X^TWX)^{-1}C^T\right]^{-1}$ exists.

APPENDIX B

DETAILS OF SOLUTION OF EQUATION (5)

The solution of equation (5) requires some rather involved matrix manipulation. The calculation of the matrix $C(X^TWX)^{-1}C^T$ and its inverse is particularly complicated.

Let us define the matrix B to be

$$B = C(X^TWX)^{-1}C^T \tag{6}$$

and its inverse to be $B^{-1} = D$. From the definition of the partitioned matrices C and $(X^TWX)^{-1}$, it can be seen that B is composed of submatrices of the form

$$B_{ij} = C_i\left[(X^TWX)_j^{-1} + (X^TWX)_{j+1}^{-1}\right]C_j^T$$

for $j = i$ and $i = 1, \ldots, NS - 1$;

$$B_{ij} = -C_i(X^TWX)_j^{-1}C_j^T$$

for $j = i - 1$ with $i = 2, \ldots, NS - 1$ and for $j = i + 1$ with $i = 1, \ldots, NS - 2$;

$$B_{ij} = 0$$

for other combinations of i and j. Since there are at most only three nonzero submatrices in each row of B, these can be redefined as follows:

$$B_{i1} = -C_i(X^TWX)_i^{-1}C_{i-1}^T \qquad \text{for } i = 2, \ldots, NS - 1$$

$$B_{i2} = C_i\left[(X^TWX)_i^{-1} + (X^TWX)_{i+1}^{-1}\right]C_i^T \qquad \text{for } i = 1, \ldots, NS - 1$$

$$B_{i3} = -C_i(X^TWX)_{i+1}^{-1}C_{i+1}^T \qquad \text{for } i = 1, \ldots, NS - 2$$

Consequently, matrix B takes the form

$$
B = \begin{bmatrix}
B_{12} & B_{13} & 0 \longrightarrow & & & & & \\
B_{21} & B_{22} & B_{23} & 0 \longrightarrow & & & & \\
0 & B_{31} & B_{32} & B_{33} & 0 \longrightarrow & & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & & & \\
0 \longrightarrow & & & & B_{NS-3,1} & B_{NS-3,1} & B_{NS-3,3} & 0 \\
0 \longrightarrow & & & & 0 & B_{NS-2,1} & B_{NS-2,2} & B_{NS-2,3} \\
0 \longrightarrow & & & & & 0 & B_{NS-1,1} & B_{NS-1,2}
\end{bmatrix}
$$

Matrix D has the form

$$
D = \begin{bmatrix}
D_{11} \cdots & D_{1,NS-1} \\
\cdot & \cdot \\
\cdot & \cdot \\
\cdot & \cdot \\
D_{NS-1,1} \cdots & D_{NS-1,NS-1}
\end{bmatrix}
$$

From the definition of the inverse of a matrix, $BD = I$, it can be seen that the product matrix BD has the form

$$
BD = \begin{bmatrix}
(B_{12}D_{11} + B_{13}D_{21}) & (B_{12}D_{12} + B_{13}D_{22}) \cdots \\
(B_{21}D_{11} + B_{22}D_{21} + B_{23}D_{31}) & \\
(B_{31}D_{21} + B_{32}D_{31} + B_{33}D_{41}) & \\
\cdot & \\
\cdot & \\
\cdot & \\
(B_{NS-2,1}D_{NS-3,1} + B_{NS-2,2}D_{NS-2,1} + B_{NS-2,3}D_{NS-1,1}) & \\
(B_{NS-1,1}D_{NS-2,1} + B_{NS-1,2}D_{NS-1,1}) & \cdots
\end{bmatrix}
$$

If I is partitioned into submatrices of the form δ_{ij}, where δ_{ij} is the identity matrix for $j = i$ and δ_{ij} is null for $j \neq i$, then each column k of the product matrix BD becomes a series of simultaneous linear matrix equations of the form

$$B_{12}D_{1k} + B_{13}D_{2k} = \delta_{1k} \tag{7}$$

$$B_{21}D_{1k} + B_{22}D_{2k} + B_{23}D_{3k} = \delta_{2k} \tag{8}$$

$$B_{31}D_{2k} + B_{32}D_{3k} + B_{33}D_{4k} = \delta_{3k} \tag{9}$$

$$\vdots$$

$$B_{NS-2,1}D_{NS-3,k} + B_{NS-2,2}D_{NS-2,k} + B_{NS-2,3}D_{NS-1,k} = \delta_{NS-2,k}$$

$$B_{NS-1,1}D_{NS-2,k} + B_{NS-1,2}D_{NS-1,k} = \delta_{NS-1,k} \tag{10}$$

The solution of this system is begun by left multiplying equation (7) by B_{12}^{-1}. (The proof that B_{12} has an inverse is the same as the proof for the existence of $\left[C(X^TWX)^{-1}C^T\right]^{-1}$ found in appendix A.) Solving equation (7) for D_{1k} gives

$$D_{1k} = B_{12}^{-1}\delta_{1k} - B_{12}^{-1}B_{13}D_{2k} = B_{12}^{-1}\left(\delta_{1k} - B_{13}D_{2k}\right)$$

Substituting for D_{1k} in equation (8) and solving for D_{2k} gives

$$D_{2k} = \left(B_{22} - B_{21}B_{12}^{-1}B_{13}\right)^{-1}\left(\delta_{2k} - B_{21}B_{12}^{-1}\delta_{1k} - B_{23}D_{3k}\right)$$

Similarly, substitution of D_{2k} into equation (9) will give a similar solution for D_{3k}. This process can be repeated for the entire set of equations.

However, the matrix algebra can be simplified by defining two auxiliary matrices E and Δ. Let $E_1 = B_{12}$ and let $E_l = B_{l2} - B_{l1}E_{l-1}^{-1}B_{l-1,3}$ for $l \neq 1$. Let $\Delta_1 = \delta_{1k}$ and let $\Delta_l = \delta_{lk} - B_{l1}E_{l-1}^{-1}\Delta_{l-1}$ for $l \neq 1$. Then the solution of the first NS - 2 equations can be written as

$$D_{lk} = E_l^{-1}\left(\Delta_l - B_{l,3}D_{l+1,k}\right) \tag{11}$$

Substituting for $D_{NS-2,k}$ into equation (10) and solving for $D_{NS-1,k}$ gives

$$D_{NS-1,k} = E_{NS-1}^{-1}\,\Delta_{NS-1}$$

Since all the E_l and Δ_l can be found in terms of known quantities, $D_{NS-1,k}$ can be determined uniquely. Similarly, all the D_{lk} can be found by substitution into equation (11) for $l = NS - 2, \ldots, 1$.

This scheme is easily programmed. Since the B_{ij} have dimensions 3×3 for a quadratic and 4×4 for a cubic, the matrix E is either a 3×3 or a 4×4. Consequently, the largest matrix that must be inverted by numerical methods is a 4×4. This inversion can be done with good accuracy by any standard numerical matrix inversion technique. Even though this scheme involves many arithmetic operations, the round-off error in the final answers becomes apparent only in the 12th or 13th significant figure.

Once the matrix D is determined, equation (5) becomes

$$A^T = (Y^TWX)\left[I - (X^TWX)^{-1}C^TDC\right](X^TWX)^{-1}$$

The next problem is to form the matrix product $T = C^TDC$. This is somewhat complicated because the submatrices of T take special forms depending on their location in the matrix. These forms are readily seen from the definition of the partitioned matrices C and D. The forms are summarized as follows:

$$T_{jn} = s\left(C_j^TD_{jn}C_n\right)$$

for the corner elements. The scalar s equals +1 if $n = j$ and -1 if $n \neq j$.

$$T_{jn} = sC_j^T(D_{jn}C_n - D_{jn-1}C_{n-1})$$

for the noncorner elements of the top and bottom rows. The scalar s equals +1 for $j = 1$ and -1 for $j = NS$.

$$T_{jn} = s\left(C_j^TD_{jn} - C_{jn}^TD_{j-1,n}\right)C_n$$

for the noncorner elements of the first and last columns. The scalar s equals +1 for $n = 1$ and -1 for $n = NS$.

$$T_{j,n} = \left(C_{j-1}^T D_{j-1,n-1} - C_j^T D_{j-1,n-1}\right) C_{n-1} + \left(C_j^T D_{jn} - C_{j-1}^T D_{j,n-1}\right) C_n$$

for the elements in the "middle" of the matrix.

Once T is defined, the solution of equation (5) is quite straightforward. Since the matrices are partitioned, the matrix multiplication can be done in several steps which can be programmed easily. These steps are as follows: Since equation (5) becomes

$$A^T = (Y^T WX)\left[I - (X^T WX)^{-1} T\right](X^T WX)^{-1}$$

the first step is to carry out the multiplication by $Y^T WX$. Equation (5) becomes

$$A^T = \left[(Y^T WX) - (Y^T WX)(X^T WX)^{-1} T\right](X^T WX)^{-1}$$

Letting the product $(Y^T WX)(X^T WX)^{-1}$ define the vector VV, equation (5) becomes

$$A^T = \left[(Y^T WX) - (VV)T\right](X^T WX)^{-1}$$

Letting the vector V be defined as $(Y^T WX) - (VV)T$, equation (5) becomes

$$A^T = V(X^T WX)^{-1}$$

Writing out each of these steps in terms of the partitioned matrices and vectors gives

$$A^T = \left(A_1^T, \ . \ . \ . \ , A_{NS}^T\right) = \left(V_1, \ . \ . \ . \ , V_{NS}\right)\begin{bmatrix} (X^T WX)_1^{-1} & & & 0 \\ & \cdot & & \\ & & \cdot & \\ 0 & & \cdot & (X^T WX)_{NS}^{-1} \end{bmatrix}$$

$$= \left[V_1 (X^T WX)_1^{-1}, \ . \ . \ . \ , V_{NS}(X^T WX)_{NS}^{-1}\right]$$

Consequently, each subvector of A^T can be determined independently of the others. The n^{th} subvector of A^T can be written as

$$A_n^T = V_n (X^T W X)_n^{-1}$$

Writing out V in terms of its definition gives

$$V = (Y^T W X) - (VV)T$$

$$\left(V_1, \ldots, V_{NS}\right) = \left[(Y^T W X)_1, \ldots, (Y^T W X)_{NS}\right] - \left(VV_1, \ldots, VV_{NS}\right) \begin{bmatrix} T_{11} \cdots & T_{1,NS} \\ \vdots & \vdots \\ T_{NS,1} \cdots & T_{NS,NS} \end{bmatrix}$$

$$= \left[(Y^T W X)_1 - \sum_{j=1}^{NS} VV_j T_{j1}, \ldots, (Y^T W X)_{NS} - \sum_{j=1}^{NS} VV_j T_{jNS} \right]$$

Therefore, V_n becomes

$$V_n = (Y^T W X)_n - \sum_{j=1}^{NS} VV_j T_{jn}$$

Writing out VV in terms of its definition gives

$$VV = (Y^T W X)(X^T W X)^{-1}$$

$$\left(VV_1, \ldots, VV_{NS}\right) = \left[(Y^T W X)_1, \ldots, (Y^T W X)_{NS}\right] \begin{bmatrix} (X^T W X)_1^{-1} & & 0 \\ & \ddots & \\ 0 & & (X^T W X)_{NS}^{-1} \end{bmatrix}$$

$$= \left[(Y^T W X)_1 (X^T W X)_1^{-1}, \ldots, (Y^T W X)_{NS} (X^T W X)_{NS}^{-1} \right]$$

Consequently,

$$VV_j = (Y^T W X)_j (X^T W X)_j^{-1}$$

Writing $(Y^T W X)_j$, $(X^T W X)_j^{-1}$, and T_{jn} in terms of their row and column elements, VV_j becomes

$$VV_j = \left[(Y^TWX)_{j,1}, \ldots, (Y^TWX)_{j,M+1}\right] \begin{bmatrix} (X^TWX)^{-1}_{j,1,1} & \cdots & (X^TWX)^{-1}_{j,1,M+1} \\ \vdots & & \vdots \\ (X^TWX)^{-1}_{j,M+1,1} & \cdots & (X^TWX)^{-1}_{j,M+1,M+1} \end{bmatrix}$$

$$\left(VV_{j,1}, \ldots, VV_{j,M+1}\right)$$

$$= \left[\sum_{k=1}^{M+1} (Y^TWX)_{j,k}(X^TWX)^{-1}_{j,k,1}, \ldots, \sum_{k=1}^{M+1} (Y^TWX)_{j,k}(X^TWX)^{-1}_{j,k,M+1}\right]$$

Similarly, V_n becomes

$$V_n = (Y^TWX)_n - \sum_{j=1}^{NS} \left(VV_{j,1}, \ldots, VV_{j,M+1}\right) \begin{bmatrix} T_{j,n,1,1} & \cdots & T_{j,n,1,M+1} \\ \vdots & & \vdots \\ T_{j,n,M+1,1} & \cdots & T_{j,n,M+1,M+1} \end{bmatrix}$$

$$\left(V_{n,1}, \ldots, V_{n,M+1}\right) = \left[(Y^TWX)_{n,1}, \ldots, (Y^TWX)_{n,M+1}\right]$$

$$- \sum_{j=1}^{NS} \left(\sum_{k=1}^{M+1} VV_{j,k}T_{j,n,k,1}, \ldots, \sum_{k=1}^{NS} V_{j,k}T_{j,n,k,M+1} \right)$$

$$= \left[(Y^TWX)_{n,1} - \sum_{j=1}^{NS} \sum_{k=1}^{M+1} VV_{j,k}T_{j,n,k,1}, \ldots, (Y^TWX)_{n,M+1} - \sum_{j=1}^{NS} \sum_{k=1}^{M+1} VV_{j,k}T_{j,n,k,M+1}\right]$$

Once the vector V_n has been formed, it is a simple matter to combine it with the sub-matrix $(X^TWX)^{-1}_n$ to get A^T_n.

APPENDIX C

PROGRAM LISTING AND FLOW CHART FOR FITLOS

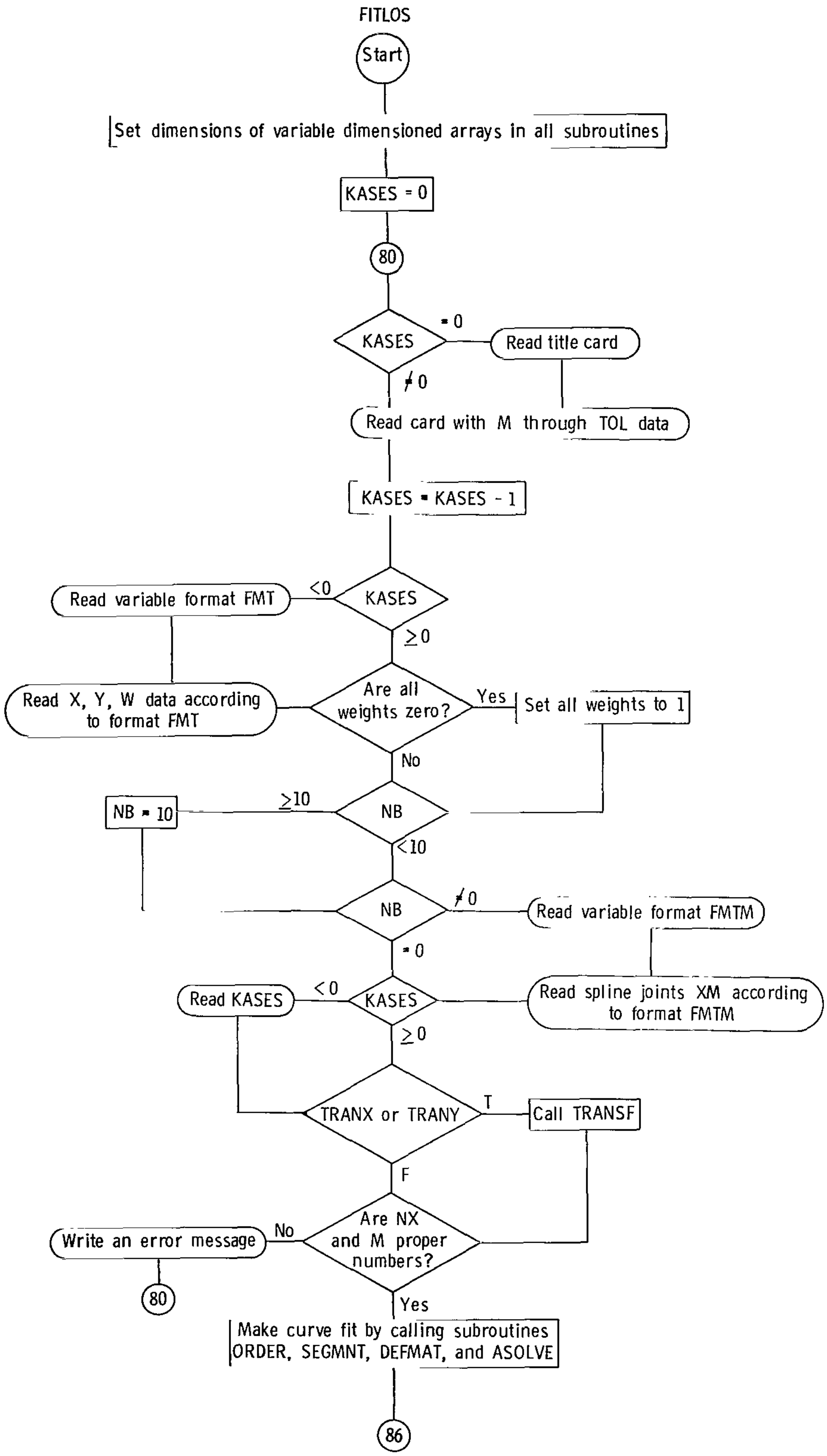
```
$IBFTC FITLOS
C
C     INPUT VARIABLES
C     ***************
C      TITLE - HOLLERITH IDENTIFICATION OF PROBLEM
C          M - DEGREE OF POLYNOMIAL
C         NX - NUMBER OF DATA POINTS
C         NS - NUMBER OF SEGMENTS CHOSEN BY USER
C         NB - NUMBER OF SPLINE JOINTS (INTERIOR) IF USER CHOOSES THEM
C         NF - NUMERICAL SIGNAL  TO DETERMINE WHICH AUTOMATIC METHOD OF
C                DIVIDING THE DATA INTO SUBSETS - USED ONLY IF BOTH NS
C                AND NB ARE ZERO
C     LREFIT - LOGICAL VARIABLE
C                IF LREFIT IS TRUE, PROGRAM WILL CHECK FOR DUPLICATION
C                 OF COEFFICIENTS
C                IF LREFIT IS FALSE, NO CHECK WILL BE MADE
C      TRANX - LOGICAL VARIABLE
C                IF TRANX IS TRUE, A LOG(10) TRANSFORMATION WILL BE
C                 MADE ON X AND XM
C                IF TRANX IS FALSE, NO TRANSFORMATION WILL BE MADE
C      TRANY - LOGICAL VARIABLE
C                IF TRANY IS TRUE, A LOG(10) TRANSFORMATION WILL BE
C                  MADE ON Y AND YC
C                IF TRANY IS FALSE, NO TRANSFORMATION WILL BE MADE
C     NPUNCH - LOGICAL VARIABLE
C                IF NPUNCH IS TRUE, NO COEFFICIENT CARDS WILL BE PUNCHED
C                IF NPUNCH IS FALSE, SEGMENT COEFFICIENTS WILL BE
C                  PUNCHED ON CARDS
C          X - ARRAY OF INDEPENDENT VARIABLES
C          Y - ARRAY OF DEPENDENT VARIABLES
C          W - ARRAY OF WEIGHTS - MAY BE READ AS ALL ZEROS
C         XM - ARRAY OF SPLINE JOINTS
C        TOL - TOLERANCE FOR FORCE FITTING AND REFIT CHECKING
C      KASES - NUMBER OF ADDITIONAL CASES USING SAME DATA
C
C     VARIABLES USED IN SUBROUTINE CALLS
C     **********************************
C
C         XX - ARRAY OF ORDERED INDEPENDENT VARIABLES
C         YY - CORRESPONDING ARRAY OF DEPENDENT VARIABLES
C         WW - CORRESPONDING ARRAY OF WEIGHTS
C        NXX - NUMBER OF POINTS IN XX, YY, AND WW ARRAYS
C         XM - ARRAY OF SPLINE JOINTS
C       LLOW - ARRAY OF INDICES OF FIRST POINTS IN EACH SUBSET
C      LHIGH - ARRAY OF INDICES OF LAST POINT IN EACH SUBSET
C        XWX - MATRIX (X-TRANSPOSE*W*X)-INVERSE
C        YWX - VECTOR (Y-TRANSPOSE*W*X)
C          C - MATRIX OF CONSTRAINTS
C          A - VECTOR OF UNDETERMINED COEFFICIENTS
C         YC - ARRAY OF DEPENDENT VARIABLES CALCULATED FROM EQUATION
C                YC = XA
C
C     PROGRAM VARIABLES
C     *****************
C
C         NW - NUMBER OF POINTS WITH ZERO WEIGHT
C        NSS - NUMBER OF SEGMENTS FOR NEW FIT (RETURNED FROM
C                SUBROUTINE REFIT)
C        DEV - DEVIATION OF FITTED CURVE FROM ORIGINAL DATA POINTS
C        ERR - RELATIVE ERROR
```

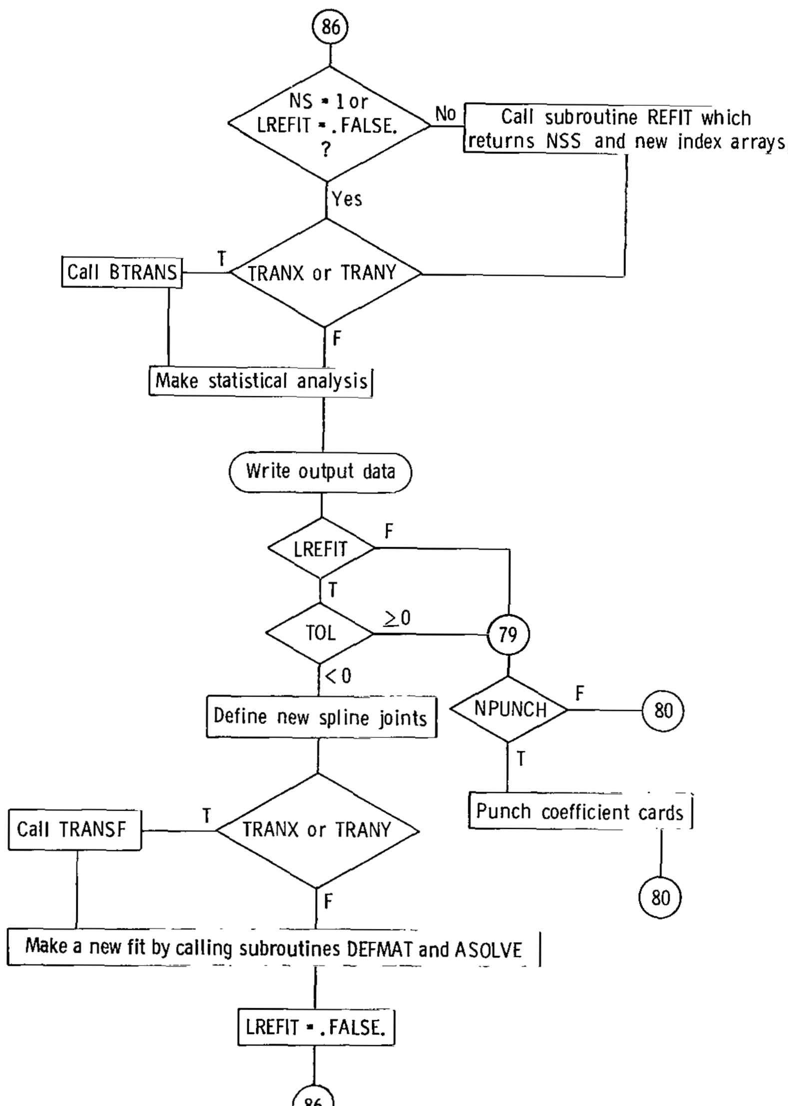
```
C                               ************
C        SUMX - SUM OF Y                   *
C       SUMX2 - SUM OF Y SQUARED           *
C        SUMY - SUM OF YC                  ** - USED IN CALCULATING
C       SUMY2 - SUM OF YC SQUARED          *      CORRELATION COEFFICIENT
C       SUMXY - SUM OF Y TIMES YC          *
C                               ************
C                               ************
C        SUMD - SUM OF DEVIATIONS          ** - USED IN CALCULATING
C       SUMD2 - SUM OF DEVIATIONS SQUARED *      VARIANCE AND STANDARD
C                               ************      DEVIATION
C          FN - NUMBER OF DEGREES OF FREEDOM
C         VAR - VARIANCE
C       STDEV - STANDARD DEVIATION
C        CORR - CORRELATION COEFFICIENT
C      CORMAX - MAXIMUM POSSIBLE CORRELATION COEFFICIENT
C
C          IM - MAXIMUM ORDER OF POLYNOMIALS (IM=3)
C
C      ****************************************************************
C
C      TO CHANGE THE MAXIMUM NUMBER OF POINTS OR THE MAXIMUM NUMBER OF
C      SEGMENTS THE PROGRAM WILL FIT, THE FOLLOWING TWO VARIABLES MUST BE
C      CHANGED -
C
C         INX - MAXIMUM NUMBER OF DATA POINTS (INX IS NOW SET AT 350)
C         IXM - MAXIMUM NUMBER OF SEGMENTS (IXM IS NOW SET AT 10)
C
C      THE FOLLOWING DIMENSIONED VARIABLES MUST BE CHANGED ALSO -
C
C      X, Y, W, XX, YY, WW, YC, AND NBLANK MUST HAVE DIMENSION INX
C      XM MUST HAVE DIMENSION (IXM+1)
C      LLOW AND LHIGH MUST HAVE DIMENSION IXM
C      THE REMAINING ARRAYS MUST HAVE DIMENSIONS THAT CORRESPOND TO THE
C      NUMBER OF SEGMENTS AND THE HIGHEST ORDER POLYNOMIAL -
C            A(IXM,IM+1)              XWX(IXM,IM+1,IM+1)
C            YWX(IXM,IM+1)            C(IXM-1,IM,IM+1)
C            B(IXM-1,IM,IM,IM)        BB(IXM-1,IXM-1,IM,IM)
C
C      ****************************************************************
C
C      THE SUBROUTINE DUBIO IS NECESSARY FOR DOUBLE PRECISION OUTPUT ON
C      THE LEWIS COMPUTER
C
C      ****************************************************************
C
      DIMENSION TITLE(12),FMT(12),FMTM(12)
      DIMENSION X(350),Y(350),W(350),XX(350),YY(350),WW(350),YC(350),
     1      NBLANK(350)
      DIMENSION XM(11),LLOW(10),LHIGH(10)
      DIMENSION A(10,4),XWX(10,4,4),YWX(10,4),C(9,3,4),B(9,3,3,3),
     1      BB(9,9,3,3)
      DOUBLE PRECISION B,BB,A,YC,XWX,YWX,C
      DOUBLE PRECISION DEV,ERR,VAR,STDEV,CORR,CORMAX,SUMX,SUMY,SUMXY,
     1      SUMX2,SUMY2,SUMD,SUMD2
      EXTERNAL DUBIO
      LOGICAL LREFIT,TRANX,TRANY,NPUNCH
      INX = 350
      IXM = 10
      IM = 3
C
C      SET DIMENSIONS OF ARRAYS IN SUBROUTINES
C
      CALL ORD(X,Y,W,XX,YY,WW,NBLANK,350)
      CALL SEG(XX,YY,XM,LLOW,LHIGH,350,11,10)
      CALL DEF(XX,YY,WW,XM,LLOW,LHIGH,XWX,YWX,C,350,11,10,4,3,9)
      CALL SLV(C,XWX,YWX,A,B,BB,9,3,4,10)
      CALL RFT(XX,A,XM,LLOW,LHIGH,350,10,11,4)
      CALL TRN(X,Y,XM,YC,350,11)
      CALL DXM(XX,XM,LLOW,LHIGH,350,11,10)
      CALL DSPL(XX,XM,LLOW,LHIGH,350,11,10)
      CALL DNS(XX,XM,LLOW,LHIGH,350,11,10)
      CALL DH(XX,YY,XM,LLOW,LHIGH,350,11,10)
      CALL DL(XX,YY,XM,LLOW,LHIGH,350,11,10)
```

```
C
      KASES = 0
   80 WRITE (6,24)
C
C     IF KASES = 0, READ A NEW TITLE AND A NEW SET OF X,Y,W DATA
C           IN ANY CASE, READ NEW VALUES FOR M,NX,NS,NP,NB,NF,REFIT,
C           TRANX,TRANY,TOL
C
      IF (KASES.EQ.0)      READ (5,4) TITLE
   81 READ (5,1) M,NX,NS,NB,NF,LREFIT,TRANX,TRANY,NPUNCH,TOL
      KASES = KASES-1
      IF (KASES.GE.0)      GO TO 84
      READ (5,4) FMT
      READ (5,FMT) (X(I),Y(I),W(I),I=1,NX)
C
C     IF ALL WEIGHTS ARE READ AS 0, SET ALL WEIGHTS TO 1
C
      NW = 0
      DO 82 I=1,NX
      IF (W(I).LE.0.)      NW = NW+1
   82 CONTINUE
      IF (NW.NE.NX)      GO TO 84
      DO 83 I=1,NX
   83 W(I) = 1.
C
C     TEST INPUT TRIGGERS TO SEE IF ADDITIONAL DATA IS NEEDED TO DIVIDE
C     THE CURVE INTO SEGMENTS
C
   84 IF (NB.NE.0)      READ (5,4) FMTM
      IF (NB.NE.0) READ (5,FMTM) (XM(I),I=1,NB)
      IF (NB.GT.IXM)  NB=IXM
      IF (KASES.LT.0)      READ (5,1) KASES
      IF(TRANX.OR.TRANY)      CALL TRANSF(X,Y,NX,XM,NB,TRANX,TRANY)
      WRITE (6,10) TITLE
C
C     CHECK FOR SUFFICIENT NUMBER OF DATA POINTS AND CORRECT ORDER OF
C     POLYNOMIALS
C
      IF (NX.GT.M.AND.(M.EQ.2.OR.M.EQ.3))      GO TO 85
      WRITE (6,23) M,NX
      GO TO 80
C
   85 CALL ORDER(X,Y,W,NX,XX,YY,WW,NXX)
      CALL SEGMNT(XX,YY,XM,LLOW,LHIGH,NXX,NS,NB,NF,M,TOL,IXM)
      CALL DEFMAT(XX,YY,WW,XM,LLOW,LHIGH,NXX,NS,M,XWX,YWX,C)
      XM(IXM+1) = XX(1)
      CALL ASOLVE(C,XWX,YWX,A,NS,M)
      IF (NS.EQ.1)   LREFIT=.FALSE.
      IF (LREFIT)   CALL REFIT(XX,A,XM,LLOW,LHIGH,NXX,NS,NSS,M,TOL)
C
C     WRITE OUTPUT DATA
C           WRITE ORDER OF POLYNOMIALS AND NUMBER OF SEGMENTS
C
   86 WRITE (6,11) M,NS
      IF (NS.EQ.0)      GO TO 80
      IF (M.EQ.3)       GO TO 90
C
C           WRITE EQUATION FOR FITTED CURVE FOR M=2
C           WRITE COEFFICIENTS IN STYLE FOR M=2
C
      IF (TRANX.AND. TRANY)  WRITE (6,39)
      IF (TRANX.AND..NOT.TRANY)  WRITE (6,40)
      IF (.NOT.TRANX.AND.TRANY)  WRITE (6,41)
      IF (.NOT.TRANX.AND..NOT.TRANY)  WRITE (6,42)
      WRITE (6,12)
      WRITE (6,44)
      WRITE (6,13) ((A(N,J),J=1,3),N=1,NS)
      GO TO 91
C
C           WRITE EQUATION FOR FITTED CURVE FOR M=3
C           WRITE COEFFICIENTS IN STYLE FOR M=3
```

```
C
   90 IF (TRANX.AND.TRANY)    WRITE (6,46)
      IF (TRANX.AND..NOT.TRANY)  WRITE (6,47)
      IF (.NOT.TRANX.AND.TRANY)  WRITE (6,48)
      IF (.NOT.TRANX.AND..NOT.TRANY)   WRITE (6,49)
      WRITE (6,12)
      WRITE (6,45)
      WRITE (6,14) ((A(N,J),J=1,4),N=1,NS)
C
C     CALCULATE FITTED VALUES OF Y AND BACK TRANSFORM DATA
C
   91 DO 103 I=1,NX
      DO 100 N=1,NS
      NN = N
      IF (X(I).LE.XM(N)) GO TO 101
  100 CONTINUE
  101 YC(I) = A(NN,M+1)
      DO 102 J=1,M
      JJ = M+1-J
      YC(I) = YC(I)*X(I)+A(NN,JJ)
  102 CONTINUE
  103 CONTINUE
      IF (TRANX.OR.TRANY)     CALL BTRANS(X,Y,XM,YC,NX,NS,TRANX,TRANY)
C
C               WRITE SPLINE JOINTS
C
      WRITE (6,15)
      WRITE (6,16) XM(IXM+1),(XM(I),I=1,NS)
C
C
C     CALCULATE DEVIATION AND RELATIVE ERROR
C     CALCULATE SUMS FOR VARIANCE AND CORRELATION COEFFICIENT
C           WRITE X,Y,Y*,DEVIATION, AND RELATIVE ERROR
C
      SUMX = 0.0D0
      SUMY = 0.0D0
      SUMXY = 0.0D0
      SUMX2 = 0.0D0
      SUMY2 = 0.0D0
      SUMD = 0.0D0
      SUMD2 = 0.0D0
      WRITE (6,21)
      DO 110 I=1,NX
      DEV = YC(I)-Y(I)
      IF (Y(I).NE.0.0)      GO TO 111
      ERR = DEV/YC(I)
      GO TO 112
  111 ERR = DEV/Y(I)
  112 SUMX = SUMX+Y(I)
      SUMY = SUMY+YC(I)
      SUMXY = SUMXY+Y(I)*YC(I)
      SUMX2 = SUMX2+Y(I)*Y(I)
      SUMY2 = SUMY2+YC(I)*YC(I)
      SUMD = SUMD+DEV
      SUMD2= SUMD2+DEV*DEV
  110 WRITE (6,20)  X(I),Y(I),YC(I),DEV,ERR
C
C     CALCULATE AND WRITE VARIANCE, STANDARD DEVIATION, AND CORRELATION
C     COEFFICIENT
C
      FN = FLOAT(NX-M*(NS-1))
      FX = FLOAT(NX)
      VAR = (SUMD2-SUMD*SUMD/FX)/FN
      STDEV = SQRT(VAR)
      CORR = FN*(FX*SUMXY-SUMX*SUMY)/FX/SQRT((FX*SUMX2-SUMX*SUMX)*
     1       (FX*SUMY2-SUMY*SUMY))
      CORMAX = FN/FX
      WRITE (6,22) VAR,CORR,STDEV,CORMAX
C
      IF (LREFIT)    GO TO 88
      WRITE (6,43)
      GO TO 79
   88 IF (TOL.LT.0.) GO TO 78
      WRITE (6,37)
```

```
      GO TO 79
   78 NS = NSS
      NB = NS-1
      DO 77 I=1,NS
      NPTS = LHIGH(I)
      XM(I) = XX(NPTS)
   77 CONTINUE
      IF (TRANX.OR.TRANY)   CALL TRANSF(X,Y,NX,XM,NB,TRANX,TRANY)
      CALL DEFMAT(XX,YY,WW,XM,LLOW,LHIGH,NXX,NS,M,XWX,YWX,C)
      CALL ASOLVE(C,XWX,YWX,A,NS,M)
      WRITE (6,38)
      WRITE (6,36)
      WRITE (6,24)
      WRITE (6,10) TITLE
      WRITE (6,36)
      LREFIT = .FALSE.
      GO TO 86
   79 IF (NPUNCH) GO TO 80
      WRITE (6,32) TITLE
      WRITE (6,33) NS,(XM(N),N=1,NS)
      IF (M.EQ.3) GO TO 89
      WRITE (6,30) ((A(N,I),I=1,3),N=1,NS)
      GO TO 80
   89 WRITE (6,31) ((A(N,I),I=1,4),N=1,NS)
      GO TO 80
C
    1 FORMAT (5I3,4L3,F12.6)
    3 FORMAT (24I3)
    4 FORMAT (12A6)
   10 FORMAT (1H ,5X,12A6)
   11 FORMAT (1H0,22HDEGREE OF POLYNOMIAL =,I5,10X,20HNUMBER OF SEGMENTS
     1 =,I5)
   12 FORMAT (1H0,41HSEGMENT COEFFICIENTS IN ASCENDING ORDER -    )
   13 FORMAT (1H ,1P3D25.15)
   14 FORMAT (1H ,1P4D25.15)
   15 FORMAT (1H0,19HSPLINE JOINTS ARE -)
   16 FORMAT (1H ,21X,7G15.7)
   20 FORMAT (1H ,1P2E17.7,1P2D25.15,0PD25.15)
   21 FORMAT (1H0,9X,1HX,14X,1HY,23X,2HY*,23X,3HDEV,20X,5HR-ERR)
   22 FORMAT (1H0,/,1H0,50HCORRELATION OF FITTED DATA TO ORIGINAL DATA
     1          ,/,1H0,16X,10HVARIANCE =,1PD25.15,12X,19HCORRELATION IND
     2EX =,0PD25.15,/,1H ,6X,20HSTANDARD DEVIATION =,1PD25.15,10X,
     321HMAXIMUM CORRELATION =,0PD25.15)
   23 FORMAT (1H0,30HCANNOT MAKE VALID FIT WITH M =,I3,9H AND NX =,I3)
   24 FORMAT (1H1)
   30 FORMAT (1H$,3D20.13)
   31 FORMAT (1H$,4D20.13)
   32 FORMAT (1H$,12A6)
   33 FORMAT (1H$,I3,(/,1H$,5E14.7))
   36 FORMAT (1H ,82HDUPLICATION OCCURED IN FIRST SET OF COEFFICIENTS -
     1CURVE WAS REFIT IN NEW SEGMENTS)
   37 FORMAT (1H ,43HNO DUPLICATION IN FIRST SET OF COEFFICIENTS)
   38 FORMAT (1H0,20HREFIT CHECK WAS MADE)
   39 FORMAT (1H0,69HEQUATION FITTED IS        LOG Y  =  A0 + A1 (LOG X
     1) + A2 (LOG X)**2 )
   40 FORMAT (1H0,64HEQUATION FITTED IS        Y  =  A0 + A1 (LOG X) +
     1A2 (LOG X)**2)
   41 FORMAT (1H0,56HEQUATION FITTED IS        LOG Y  =  A0 + A1 X + A2
     1 X**2)
   42 FORMAT (1H0,52HEQUATION FITTED IS        Y  =  A0 + A1 X + A2 X**
     12)
   43 FORMAT (1H0,19HNO REFIT CHECK MADE)
   44 FORMAT (1H0,10X,2HA0,23X,2HA1,23X,2HA2)
   45 FORMAT (1H0,10X,2HA0,23X,2HA1,23X,2HA2,23X,2HA3)
   46 FORMAT (1H0,84HEQUATION FITTED IS        LOG Y  =  A0 + A1 (LOG X
     1) + A2 (LOG X)**2 + A3 (LOG X)**3)
   47 FORMAT (1H0,81HEQUATION FITTED IS        Y  =  A0 + A1 (LOG X) +
     1A2 (LOG X)**2 + A3 (LOG X)**3 )
   48 FORMAT (1H0,66HEQUATION FITTED IS        LOG Y  =  A0 + A1 X + A2
     1 X**2 + A3 X**3)
   49 FORMAT (1H0,63HEQUATION FITTED IS        Y  =  A0 + A1 X + A2 X**
     12 + A3 X**3 )
C
      END
```


FITLOS
Start
Set dimensions of variable dimensioned arrays in all subroutines
KASES = 0
80
KASES
= 0
Read title card
≠ 0
Read card with M through TOL data
KASES = KASES - 1
KASES
<0
Read variable format FMT
≥0
Are all weights zero?
Yes
Set all weights to 1
Read X, Y, W data according to format FMT
No
NB
≥10
NB = 10
<10
NB
≠ 0
Read variable format FMTM
= 0
Read spline joints XM according to format FMTM
KASES
< 0
Read KASES
≥ 0
TRANX or TRANY
T
Call TRANSF
F
Are NX and M proper numbers?
No
Write an error message
80
Yes
Make curve fit by calling subroutines ORDER, SEGMNT, DEFMAT, and ASOLVE
86


86
NS = 1 or LREFIT = .FALSE. ?
No
Call subroutine REFIT which returns NSS and new index arrays
Yes
Call BTRANS
T
TRANX or TRANY
F
Make statistical analysis
Write output data
LREFIT
F
T
TOL
≥0
79
<0
Define new spline joints
NPUNCH
F
80
T
Punch coefficient cards
80
Call TRANSF
T
TRANX or TRANY
F
Make a new fit by calling subroutines DEFMAT and ASOLVE
LREFIT = .FALSE.
86

APPENDIX D

VARIABLES USED BY SEVERAL SUBROUTINES

The variables used by several subroutines of the program FITLOS are defined as follows:

XM	Array of spline joints.
LLOW	Array of indices of the first point in each subset. LLOW(1) = 1 and LLOW(N) = lowest I such that XM(N - 1) $\leq$ X(I) $\leq$ XM(N) for N = 2, . . . , NS.
LHIGH	Array of indices of the last point in each subset. LHIGH(N) = highest I such that XM(N) $\leq$ X(I) $\leq$ XM(N + 1) for N = 1, . . . , NS - 1, and LHIGH(NS) = NX.
XWX	Multidimensioned array $(X^T WX)^{-1}$. The subscripts on XWX have the same order as the subscripts on matrix $(X^T WX)^{-1}$ of appendix B.
YWX	Multidimensioned array $Y^T WX$. The subscripts correspond to the subscripts on vector $Y^T WX$ of appendix B.
C	Multidimensioned array of constraints. The subscripts correspond to the subscripts on matrix C of appendix B.
A	Multidimensioned array of undetermined coefficients. The subscripts correspond to the subscripts of vector A of appendix B.
X	Array of the ordered independent variable.
Y	Array of dependent variable that corresponds to X.

In the main program FITLOS, X and Y are the names of the input arrays, while XX and YY are the names of the ordered data.

APPENDIX E

DESCRIPTION OF SUBROUTINES

The subroutines of the program FITLOS are described in this appendix. After the descriptions, all the subroutines are listed followed by all the flow charts.

TRANSF

Subroutine TRANSF makes a base 10 log transformation on the input data. If any of these data are not greater than zero, TRANSF changes that number to 10^{-30}.

BTRANS

Subroutine BTRANS converts the transformed data back to its original form. Since YC, the calculated values of y^*, are actually $\log_{10}(YC)$, these data are also back transformed so they have the same form as the input data.

ORDER

Subroutine ORDER arranges the input data in order of ascending x. Since the definition of a spline function requires that $x_i < x_{i+1}$, ORDER averages the y's for which duplicate x's occur. This average y is a weighted average,

$$\bar{y} = \frac{\sum_j y_j w_j}{\sum_j w_j}$$

The total weight, $\sum_j w_j$, becomes the weight of the average point. To preserve the input data, the ordered data are put into new arrays.

SEGMNT

Subroutine SEGMNT determines the spline joints and the low and high indices of the points in each subset. SEGMNT first tests the variable NB. If $NB \neq 0$, the spline joints have been supplied by the user. In that case, SEGMNT calls subroutine DIVXM to determine the index arrays. If $NB = 0$, SEGMNT then tests the variable NS.

If $NS \neq 0$, the number of segments has been chosen by the user. In that case, SEGMNT calls subroutine DIVNS to divide the data as evenly as possible among the NS subsets. Subroutine DIVNS calculates the index arrays and determines the spline joints. If NS = 0, SEGMNT tests the number of data points NX.

If NX ≤ 3M, SEGMNT calls subroutine SPESHL to make a special division of the data into one or two subsets with specially determined spline joints. These special spline joints and the index arrays are listed in the main-text section HOW DATA ARE DIVIDED INTO SUBSETS.

If NX > 3M, SEGMNT sets the number of subsets to be the maximum possible number based on the number of data points and the degree of the polynomial. This number is (NX - 1)/M.

SEGMNT then tests the variable NF. If NF < 0, subroutine FFLOW is called to do a force-fit division starting at the low end of the data. If NF > 0, subroutine FFHIGH is called to do a force-fit division starting at the high end of the data. If NF = 0, subroutine DIVNS is called with NS = (NX - 1)/M.

DIVXM

Subroutine DIVXM divides the data into subsets according to spline joints (xm) chosen by the user. DIVXM first puts the (xm) in ascending order. Then it eliminates any of the (xm) that are outside the range of x and adjusts the number of spline joints NB accordingly.

DIVXM then determines the indices of the first and last points in each subset. Then it checks whether each subset has a sufficient number of points. If LHIGH(I) - LLOW(I) + 1 ≤ M, there are not enough points in subset I and that subset must be combined with its neighbors. DIVXM also changes the spline joints to correspond to the new index arrays.

DIVNS

Subroutine DIVNS divides the data into NS subsets as evenly as possible. DIVNS first makes NS a "proper" number. It chooses the smallest of three possible values which are as follows: the chosen NS, the maximum number of subsets based on the number of data points and the degree of the polynomial, and the dimension of the arrays LLOW and LHIGH, which is called LIM in the program.

In dividing the data as evenly as possible, DIVNS uses fixed-point arithmetic to eliminate the possibility of a fractional number of points in a subset. The spline joints and the index arrays are determined as the division takes place.

SPESHL

Subroutine SPESHL makes an arbitrary division of the data into one or two subsets. If the number of points NX is not greater than 2M, only one segment is possible. If

NX is between 2M and 3M, SPESHL divides the data into two subsets where the spline joints and index arrays are defined in the main-text section HOW DATA ARE DIVIDED INTO SUBSETS.

FFLOW

Subroutine FFLOW divides the data into subsets by force-fitting starting at the low end of the data. The maximum possible number of subsets NS appears in the calling vector. FFLOW first sets the index arrays LLOW and LHIGH to zero. It then starts force-fitting as described in the main body of the report. For a point to be accepted in a subset it must fall on the Lagrange polynomial within a given amount of precision TOL; that is, the ratio $|y(calc)/y(given)| = 1 \pm TOL$. For $y(given) = 0$, the acceptance criterion is $|y(calc) - y(given)| \leq TOL$. Spline joints are determined as the last point in each subset.

FFHIGH

Subroutine FFHIGH does a force-fit division of the data into subsets starting at the high end of the data. It first sets the index arrays to zero. Then it starts force-fitting, but begins with the NS^{th} segment. Consequently, some low-order elements of LLOW and LHIGH could remain zero. If they do, the nonzero elements are moved down so that LLOW(1) = 1. The elements of LHIGH and XM are moved down simultaneously and the value of NS is reduced accordingly. In all other respects, however, FFHIGH and FFLOW are essentially the same.

REFIT

Subroutine REFIT checks whether the curve was fit in more segments than were necessary. To do this, it checks whether the coefficients for a low-order segment would give the same value of y^* for points in a higher order subset as the coefficients for the higher order subset. Subroutine REFIT works in essentially the same way as the force-fitting subroutines except the test polynomial is defined by the coefficients from the lower order segment instead of a Lagrange polynomial. The use of TOL is the same as in subroutine FFLOW.

MINVRT

Subroutine MINVRT inverts a double-precision matrix by Gaussian elimination (ref. 6). It also calculates the determinant of the matrix. If the determinant is zero, that is, if the matrix is singular, an error message is printed and the null matrix is returned to the calling program. If the matrix is nonsingular, MINVRT finishes the Gaussian elimination. Pivoting is not necessary since the matrices are small and well conditioned. Then the inverse is multiplied by the input matrix and the maximum deviation of the elements of the product matrix from the elements of the identity matrix is returned to the calling program. This measures the accuracy of the inverse. Finally, the inverse is transferred to the input matrix for return to the calling program.

DEFMAT

Subroutine DEFMAT defines the matrices $(X^TWX)^{-1}$, Y^TWX, and C from the arrays of ordered data and the array of spline joints. The multiple subscripts on the arrays XWX, YWX, and C correspond to the subscripts on matrices $(X^TWX)^{-1}$, Y^TWX, and C of appendix B.

ASOLVE

Subroutine ASOLVE solves equation (5). If there is only one segment, the simple matrix multiplication $A = (X^TWX)_1^{-1}(Y^TWX)_1$ is performed. For more than one segment, matrix B is defined by equation (6) of appendix B. Since each row of B has only three nonzero submatrices, only these three submatrices are calculated. Then B is inverted by the process described in appendix B. The E and DELTA matrices are the same as the E and Δ matrices defined in appendix B.

Beginning with statement 500, the matrix multiplication of equation (5) is performed. Since there are four types of elements in the matrix product $C^TB^{-1}C$, there are four separate techniques used for calculating these elements. These four types of elements are defined in appendix B. When the matrix product $C^TB^{-1}C$ has been formed, the remaining multiplication is finished. The vectors V and VV are the same as defined in appendix B.

```
$IBFTC TR1
C
C     PROGRAM VARIABLES
C     *****************
C
C         YC - CALCULATED VALUES OF Y
C
      SUBROUTINE TRN(X,Y,XM,YC,IX,IXM)
      DIMENSION X(IX),Y(IX),YC(IX),XM(IXM)
      DOUBLE PRECISION YC
      GO TO 3
C
C     TRANSFORMATION SUBROUTINE
C
      ENTRY TRANSF(X,Y,NX,XM,NB,TRANX,TRANY)
      LOGICAL TRANX,TRANY
      IF (.NOT.TRANX)   GO TO 1
      DO 10 I=1,NX
      IF (X(I).LE.0.)   X(I)=1.E-30
   10 X(I) = ALOG10(X(I))
      IF (NB.EQ.0)       GO TO 1
      DO 11 I=1,NB
      IF (XM(I).LE.0.)   XM(I)=1.E-30
   11 XM(I) = ALOG10(XM(I))
    1 IF (.NOT.TRANY)   GO TO 3
      DO 12 I=1,NX
      IF (Y(I).LE.0.)   Y(I)=1.E-30
   12 Y(I) = ALOG10(Y(I))
      GO TO 3
C
      ENTRY BTRANS(X,Y,XM,YC,NX,NS,TRANX,TRANY)
      IF (.NOT.TRANX)   GO TO 2
      DO 13 I=1,NX
   13 X(I) = 10.**X(I)
      DO 14 I=1,NS
   14 XM(I) = 10.**XM(I)
      XM(11) = 10.**XM(11)
    2 IF (.NOT.TRANY)   GO TO 3
      DO 15 I=1,NX
      Y(I) = 10.**Y(I)
   15 YC(I) = 10.**YC(I)
C
    3 RETURN
```

```
$IBFTC ORDR
C
C     PROGRAM VARIABLES
C     *****************
C
C         XT - ORIGINAL VALUES OF THE INDEPENDENT VARIABLE
C         YT - ORIGINAL VALUES OF THE DEPENDENT VARIABLE
C         WT - ORIGINAL WEIGHTS
C        NXT - NUMBER OF ORIGINAL POINTS
C
C          X - ORDERED ARRAY OF INDEPENDENT VARIABLES
C          Y - ORDERED ARRAY OF DEPENDENT VARIABLES
C          W - ORDERED ARRAY OF WEIGHTS
C         NX - NUMBER OF ORDERED DATA POINTS
C
C     NBLANK - BOOKKEEPING ARRAY, NBLANK(I)=0 MEANS POINT I HAS
C                BEEN TRANSFERED TO THE NEW ARRAYS
C         KK - INDEX OF THE AVERAGED POINT IN THE NEW ARRAYS
C          N - NUMBER OF POINTS WITH SAME XT VALUE
C       SUMY - SUM OF YT VALUES FOR POINTS WITH SAME XT VALUE
C       SUMW - SUM OF WEIGHTS FOR POINTS WITH SAME XT VALUE
C
      SUBROUTINE ORD(XT,YT,WT,X,Y,W,NBLANK,IX)
      DIMENSION XT(IX),YT(IX),WT(IX),X(IX),Y(IX),W(IX),NBLANK(IX)
      GO TO 300
```

```
C
C     ARRANGE DATA IN ORDER OF ASCENDING X AND AVERAGE Y FOR WHICH
C     DUPLICATE VALUES OF X OCCUR
C
      ENTRY ORDER(XT,YT,WT,NXT,X,Y,W,NX)
C
      DEBUG (XT(I),YT(I),WT(I),I=1,NXT)
      DO 100 I=1,NXT
  100 NBLANK(I) = 1
      NX = 0
      KK = 1
      DO 230 I=1,NXT
      IF (NBLANK(I).EQ.0)     GO TO 230
      N = 1
      SUMY = YT(I)*WT(I)
      SUMW = WT(I)
      NBLANK(I) = 0
      II = I+1
      DO 200 J=II,NXT
      IF (NBLANK(J).EQ.0)     GO TO 200
      IF (XT(J).NE.XT(I))     GO TO 200
      SUMY = SUMY+YT(J)*WT(J)
      SUMW = SUMW+WT(J)
      N = N+1
      NBLANK(J) = 0
  200 CONTINUE
      IF (KK.EQ.1)      GO TO 221
      DO 220 J=1,NX
      IF (X(J).LE.XT(I))      GO TO 220
      KN = NX-J+1
      DO 210 K=1,KN
      KK = NX+2-K
      X(KK) = X(KK-1)
      Y(KK) = Y(KK-1)
  210 W(KK) = W(KK-1)
      GO TO 222
  220 CONTINUE
  222 KK = KK-1
  221 X(KK) = XT(I)
      Y(KK) = SUMY/SUMW
      W(KK) = SUMW
      NX = NX+1
      KK = NX+2
  230 CONTINUE
      DEBUG (X(I),Y(I),W(I),I=1,NX)
C
  300 RETURN
      END
```

```
$IBFTC SGMNT
      SUBROUTINE SEG(X,Y,XM,LLOW,LHIGH,IX,IXM,IL)
      DIMENSION X(IX),Y(IX),XM(IXM),LLOW(IL),LHIGH(IL)
      RETURN
C
C     DIVIDE DATA INTO SUBSETS BY DETERMINING SPLINE JOINTS AND
C     THE NUMBER OF POINTS IN EACH SUBSET
C
      ENTRY SEGMNT(X,Y,XM,LLOW,LHIGH,NX,NS,NB,NF,M,TOL,LIM)
C
C     DIVIDE ACCORDING TO PREDETERMINED BREAK POINTS
C
  200 IF (NB.EQ.0)    GO TO 400
      CALL DIVXM(X,XM,LLOW,LHIGH,NX,NS,NB,M)
      WRITE (6,21)
      RETURN
C
C     DIVIDE ACCORDING TO PREDETERMINED NUMBER OF SEGMENTS
C
  400 IF (NS.EQ.0) GO TO 500
      CALL DIVNS(X,XM,LLOW,LHIGH,NX,NS,NB,M,LIM).
```

```
      WRITE (6,23)
      RETURN
C
C     DO THE NUMBER OF POINTS REQUIRE A SPECIAL DIVISION
C
  500 IF (NX.GT.3*M)  GO TO 600
      CALL SPESHL(X,XM,LLOW,LHIGH,NX,NS,NB,M)
      WRITE (6,27) NX
      RETURN
C
C     DIVIDE ACCORDING TO FORCE FIT SCHEME OR AS EVENLY AS POSSIBLE
C     AMONG SEGMENTS
C
  600 NS = (NX-1)/M
      IF (NF)  610,620,630
  610 CALL FFLOW(X,Y,XM,LLOW,LHIGH,NX,NS,NB,M,TOL)
      WRITE (6,24)
      RETURN
C
  620 CALL DIVNS(X,XM,LLOW,LHIGH,NX,NS,NB,M,LIM)
      WRITE (6,25)
      RETURN
C
  630 CALL FFHIGH(X,Y,XM,LLOW,LHIGH,NX,NS,NB,M,TOL)
      WRITE (6,26)
      RETURN
C
   20 FORMAT (1H0,47HNUMBER OF DATA POINTS REQUIRES SPECIAL DIVISION)
   21 FORMAT (1H0,34HSPLINE JOINTS CHOSEN BY PROGRAMMER)
   22 FORMAT (1H0,52HNUMBER OF POINTS IN EACH SUBSET CHOSEN BY PROGRAMME
     1R)
   23 FORMAT (1H0, 83HDATA DIVIDED AS EVENLY AS POSSIBLE AMONG THE NUMBE
     1R OF SUBSETS CHOSEN BY PROGRAMMER)
   24 FORMAT (1H0,74HDATA DIVIDED INTO SUBSETS BY FORCE FITTING STARTING
     1 AT THE LOW END OF DATA)
   25 FORMAT (1H0,70HDATA DIVIDED AS EVENLY AS POSSIBLE AMONG THE MAXIMU
     1M NUMBER OF SUBSETS)
   26 FORMAT (1H0,75HDATA DIVIDED INTO SUBSETS BY FORCE FITTING STARTING
     1 AT THE HIGH END OF DATA)
   27 FORMAT (1H0,I5,2X,32HPOINTS REQUIRES SPECIAL DIVISION)
C
      END
```

```
$IBFTC DVXM
C
C     PROGRAM VARIABLES
C     *****************
C
C          T - TEMPORARY STORAGE USED IN ORDERING SPLINE JOINTS
C        KST - INDEX OF FIRST POINT IN NEW SUBSET
C        NSS - SUBSET COUNTER WHEN A SUBSET DOES NOT HAVE SUFFICIENT
C               POINTS
C      NPLUS - NUMBER OF POINTS IN (I+1) SUBSET
C        NP2 - ONE HALF THE NUMBER OF POINTS IN SUBSET I
C
      SUBROUTINE DXM(X,XM,LLOW,LHIGH,IX,IXM,IL)
      DIMENSION X(IX),XM(IXM),LLOW(IL),LHIGH(IL)
      RETURN
C
C     DIVIDE ACCORDING TO PRECHOSEN SPLINE JOINTS
C
      ENTRY DIVXM(X,XM,LLOW,LHIGH,NX,NS,NB,M)
C
C
C     CHECK THAT SPLINE JOINTS MATCH THE RANGE OF X
C
      DEBUG (XM(I),I=1,NB)
  300 DO 310 I=1,NB
      IF (I.EQ.NB)   GO TO 310
```

```
      II = I+1
      DO 305 J=II,NB
      IF (XM(I).LE.XM(J))   GO TO 305
      T = XM(I)
      XM(I) = XM(J)
      XM(J) = T
  305 CONTINUE
  310 CONTINUE
      DEBUG (XM(I),I=1,NB)
C
  311 DO 320 I=1,NB
      IF (I.EQ.NB)   GO TO 320
      IF (XM(I).GE.X(M+1))   GO TO 320
      II = I+1
      DO 315 J=II,NB
  315 XM(J-1) = XM(J)
      NB = NB-1
      GO TO 311
  320 CONTINUE
      DEBUG NB,(XM(I),I=1,NB)
C
      NS = NB+1
      DO 330 I=1,NB
      IF (XM(I).LT.X(NX))   GO TO 330
      NS = NS-1
  330 CONTINUE
      NB = NS-1
      XM(NS) = X(NX)
      DEBUG NS,(XM(I),I=1,NS)
C
C     DETERMINE LOW AND HIGH INDICES
C
      LLOW(1)= 1
      KST=1
      DO 350  I=1,NS
      DO 340 K=KST,NX
      IF (XM(I).GT.X(K))   GO TO 340
      LHIGH(I) = K-1
      IF(K.EQ.NX)  GO TO 340
      IF (XM(I).EQ.X(K))   LHIGH(I)=K
      LLOW(I+1) = LHIGH(I)
      IF (XM(I).NE.X(K))   LLOW(I+1)=LHIGH(I)+1
      KST= K+1
      GO TO 350
  340 CONTINUE
  350 CONTINUE
      LHIGH(NS) = NX
      DEBUG (LLOW(I), I=1,NS)
      DEBUG (LHIGH(I),I=1,NS)
C
C     CHECK FOR SUFFICIENT POINTS IN EACH SUBSET
C
      II=0
      DO 360  I=1,NS
      IF(LHIGH(I)-LLOW(I)+1.GT.M)  GO TO 360
      II= I
      WRITE(6,10) I
   10 FORMAT (1H0,29HINSUFFICIENT POINTS IN SUBSET,I5)
  360 CONTINUE
C
C     IF ANY SUBSETS  ARE DEFICIENT, COMBINE THEM WITH OTHER SUBSETS
C
      IF(II.EQ.0) RETURN
      NSS= NS
  400 DO 470  I=1,NSS
      NPTS= LHIGH(I)-LLOW(I)+1
      IF(NPTS.GT.M) GO TO 470
      IF(I.NE.1)  GO TO 410
      LHIGH(1)= LHIGH(2)
      LHIGH(2)=0
      LLOW(2)=0
      XM(1)= XM(2)
      DEBUG I
      GO TO 480
```

```
  410 IF(I.NE.NSS)  GO TO 420
      LHIGH(NSS-1) = NX
      LHIGH(NSS)=0
      LLOW(NSS)=0
      XM(NSS-1)= X(NX)
      DEBUG I,I
      GO TO 480
  420 NPLUS= LHIGH(I+1)-LLOW(I+1)+1
      IF(NPLUS.GT.M)  GO TO 430
      LHIGH(I)= LHIGH(I+1)
      LHIGH(I+1)= 0
      LLOW(I+1)=0
      XM(I)= XM(I+1)
      DEBUG  I,I,I
      GO TO 480
  430 NP2 = (LHIGH(I)-LLOW(I)+1)/2
      IF (NP2*2.NE.LHIGH(I)-LLOW(I)+1)   GO TO 460
      LHIGH(I-1)= LHIGH(I-1)+NP2
      LHIGH(I)=0
      LLOW(I+1)= LLOW(I+1)-NP2
      IF (LLOW(I+1).LT.LHIGH(I-1))   LLOW(I+1)=LHIGH(I-1)
      LLOW(I)=0
      DEBUG I,I,I,I
  440 IF(LLOW(I+1).NE.LHIGH(I-1)) GO TO 450
      NP2= LLOW(I+1)
      XM(I-1)= X(NP2)
      DEBUG I,I,I,I,I
      GO TO 480
  450 NP2= LLOW(I+1)
      XM(I-1)= X(NP2)
      NP2= LHIGH(I-1)
      XM(I-1)= .5*(XM(I-1)+X(NP2))
      DEBUG I,I,I,I,I,I
      GO TO 480
  460 LHIGH(I-1)= LHIGH(I-1)+NP2
      LHIGH(I)=0
      LLOW(I+1) = LLOW(I+1)-NP2
      LLOW(I)=0
      GO TO 440
  470 CONTINUE
      NS= NSS
      RETURN
C
C     COMPACT INDEX AND SPLINE JOINT ARRAYS AND CHECK AGAIN
C
  480 DO 500  I=1,NSS
      DEBUG I,LLOW(I),LHIGH(I)
      IF(LLOW(I).GT.0.AND.LHIGH(I).GT.0)  GO TO500
      II= I
      NST = NSS-1
      DO 490 J=II,NST
      LLOW(J)= LLOW(J+1)
      LHIGH(J)= LHIGH(J+1)
      XM(J)= XM(J+1)
  490 CONTINUE
      NSS= NSS-1
      DEBUG NSS
      DEBUG (LLOW(J),J=1,NSS)
      DEBUG (LHIGH(J),J=1,NSS)
      DEBUG (XM(J),J=1,NSS)
      GO TO 400
  500 CONTINUE
      NS = NSS
      DEBUG (XM(I),I=1,NS)
      DEBUG (LHIGH(I),I=1,NS)
      DEBUG  (LLOW(I),I=1,NS)
C
      RETURN
      END
```

```
$IBFTC DVNS
C
C     PROGRAM VARIABLES
C     *****************
C
C     NSCRIT - MAXIMUM NUMBER OF SEGMENTS BASED ON DEGREE OF
C                POLYNOMIAL AND NUMBER OF DATA POINTS
C         NS - SMALLEST OF THE THREE POSSIBLE NUMBER OF SEGMENTS
C      NPLFT - NUMBER OF POINTS THAT HAVE NOT BEEN ASSIGNED TO A
C                SUBSET
C      NSLFT - NUMBER OF AVAILABLE SUBSETS
C          I - SUBSET INDEX
C       NPTS - NUMBER OF POINTS THAT WILL BE IN THE I(TH) SUBSET
C        LIM - DIMENSION OF ARRAYS LLOW AND LHIGH IN MAIN PROGRAM
C
      SUBROUTINE DNS(X,XM,LLOW,LHIGH,IX,IXM,IL)
      DIMENSION X(IX),XM(IXM),LLOW(IL),LHIGH(IL)
      GO TO 610
C
C     DIVIDE ACCORDING TO PREDETERMINED NUMBER OF SEGMENTS
C
      ENTRY DIVNS(X,XM,LLOW,LHIGH,NX,NS,NB,M,ILL)
      IF (NS.EQ.0)  GO TO 600
C
C     ONE SEGMENT REQUIRES SPECIAL HANDLING
C
      IF (NS.NE.1)    GO TO 500
      LLOW(1) = 1
      LHIGH(1) = NX
      NB = 0
      RETURN
C
C     MORE THAN ONE SEGMENT MEANS DIVIDING THE DATA AS EVENLY AS
C     POSSIBLE AMONG THE SEGMENTS
C
  500 NSCRIT = (NX-1)/M
  510 NS = MIN0(NS,NSCRIT,LIM)
      DEBUG  NS,NSCRIT,LIM
  511 NPLFT = NX
      NSLFT = NS
      NN = NS-1
      LLOW(1) = 1
      DO 520 I=1,NN
      NPTS = NPLFT/NSLFT+1
      NSLFT = NSLFT-1
      NPLFT = NPLFT-NPTS+1
      LHIGH(I) = LLOW(I)+NPTS-1
      IF (I.LT.NS) LLOW(I+1) =LHIGH(I)
      NPTS =LHIGH(I)
      XM(I) = X(NPTS)
  520 CONTINUE
      LHIGH(NS) = NX
      XM(NS) = X(NX)
  522 NB =NS-1
C
      DEBUG NS,NB
      DEBUG(LLOW(I),I=1,NS)
      DEBUG (LHIGH(I),I=1,NS)
      DEBUG  (XM(I),I=1,NS)
  610 RETURN
C
  600 WRITE (6,10)
   10 FORMAT (1H0,15HNS = 0 IN DIVNS)
      RETURN
      END
```

```
$IBFTC SPSHL
      SUBROUTINE DSPL(X,XM,LLOW,LHIGH,IX,IXM,IL)
      DIMENSION X(IX),XM(IXM),LLOW(IL),LHIGH(IL)
      RETURN
C
C     SPECIAL DIVISION INTO SEGMENTS WHEN NUMBER OF POINTS IS BETWEEN
C     M+1 AND 3M
C
      ENTRY SPESHL(X,XM,LLOW,LHIGH,NX,NS,NB,M)
C
C     NUMBER OF POINTS LESS THAN 2M+1 REQUIRES ONE SEGMENT
C
  100 IF (NX.GT.2*M)   GO TO 200
      NS = 1
      NB = 0
      XM(1) = X(NX)
      LLOW(1) = 1
      LHIGH(1) = NX
      RETURN
C
  200 NS = 2
      NB = 1
      GO TO (1,2,3),M
C
    1 WRITE (6,10)
   10 FORMAT (1H0,13HM CANNOT BE 1)
      CALL EXIT
C
C     M = 2
C
    2 LLOW(1) = 1
      LLOW(2) = 3
      LHIGH(2) = NX
      XM(2) = X(NX)
      IF (NX.GT.5)  GO TO 210
      XM(1) = X(3)
      LHIGH(1) = 3
      RETURN
C
  210 XM(1) = .5*(X(3)+X(4))
      LHIGH(1) = 4
      RETURN
C
C     M = 3
C
    3 LLOW(1) = 1
      XM(2) = X(NX)
      LHIGH(2) = NX
      IF (NX.GT.8) GO TO 310
      IF (NX.GT.7)  GO TO 300
      XM(1) = X(4)
      LLOW(2) = 4
      LHIGH(1) = 4
      RETURN
C
  300 XM(1) = .5*(X(4)+X(5))
      LLOW(2) = 5
      LHIGH(1) = 4
      RETURN
C
  310 LLOW(2) = 5
      LHIGH(1) = 5
      XM(1) = X(5)
      RETURN
      END
```

```
$IBFTC FFLW
C
C     PROGRAM VARIABLES
C     *****************
C
C         NN - TRIAL NUMBER OF SEGMENTS
C         NS - NEW SEGMENT COUNTER
C         IO - INDEX OF FIRST POINT USED TO DETERMINE LAGRANGE
C                POLYNOMIAL
C         I1 - INDEX OF SECOND POINT
C         I2 - INDEX OF THIRD POINT - LAST POINT FOR A QUADRATIC
C         I3 - INDEX OF LAST POINT FOR CUBIC
C        NST - INDEX OF FIRST POINT TO BE TESTED
C     *******
C         A *
C         B *- INTERMEDIATE VALUES TO SIMPLIFY CODING OF THE
C         C *    LAGRANGE POLYNOMIAL
C         D *
C     *******
C         YJ - Y AT X(J) EVALUATED BY THE LAGRANGE POLYNOMIAL
C
      SUBROUTINE DL(X,Y,XM,LLOW,LHIGH,IX,IXM,IL)
      DIMENSION X(IX),Y(IX),XM(IXM),LLOW(IL),LHIGH(IL)
      RETURN
C
C     DETERMINE SUBSETS BY FORCE FITTING STARTING AT LOW END OF DATA
C
      ENTRY FFLOW(X,Y,XM,LLOW,LHIGH,NX,NS,NB,M,TOL)
C
  700 JJ = 1
      NN = NS
      NS = 0
      DO 710 I=1,NN
      LLOW(I) = 0
      LHIGH(I) = 0
  710 CONTINUE
      LLOW(1)= 1
      DO 750 N=1,NN
      IO = JJ
      NS = NS+1
      I1 = IO+1
      I2 = IO+2
      I3 = IO+3
      NST = IO+M+1
      DEBUG NS,NST,IO,X(IO)
      IF (NST.GT.NX-M)   GO TO 760
      LHIGH(N) = NST-1
      DO 740 J=NST,NX
      JJ = J-1
      A = (X(J)-X(IO))/(X(I2)-X(I1))
      B = (X(J)-X(I1))/(X(I2)-X(IO))
      C = (X(J)-X(I2))/(X(I1)-X(IO))
      IF (M.EQ.3)            GO TO 720
      YJ = Y(IO)*B*C-Y(I1)*C*A+Y(I2)*A*B
      DEBUG A,B,C,YJ,Y(J),X(J)
      GO TO 730
  720 D = X(J)-X(I3)
      YJ = D*(-Y(IO)*B*C/(X(I3)-X(IO))+Y(I1)*C*A/(X(I3)-X(I1))-Y(I2)*
     1 A*B/(X(I3)-X(I2)))+Y(I3)*(X(J)-X(IO))*(X(J)-X(I1))*(X(J)-X(I2))/
     2 (X(I3)-X(IO))/(X(I3)-X(I1))/(X(I3)-X(I2))
      DEBUG A,B,C,D,X(J),Y(J),YJ
  730 IF (Y(J).EQ.0.)       GO TO 731
      IF (ABS(1.-YJ/Y(J)).GT.TOL)   GO TO 735
      LHIGH(N) = J
      GO TO 740
  731 IF (ABS(YJ-Y(J)).GT.TOL)   GO TO 735
      LHIGH(N)= J
  740 CONTINUE
  735 IF (N.NE.NN)   LLOW(N+1)=LHIGH(N)
  750 CONTINUE
  760 LHIGH(NS) = NX
C
C     SELECT SPLINE JOINTS
```

```
C
      NB = NS-1
      DO 910 I=1,NS
      LL= LHIGH(I)
  910 XM(I)= X(LL)
      DEBUG NB,NS,(XM(I),I=1,NS)
      DEBUG (LLOW(I),I=1,NS)
      DEBUG (LHIGH(I),I=1,NS)
C
      RETURN
      END
```

```
$IBFTC FFHGH
C
C     PROGRAM VARIABLES
C     *****************
C
C      NSTRL - TRIAL NUMBER OF SEGMENTS
C         NS - NEW SEGMENT COUNTER
C      NPLFT - NUMBER OF POINTS LEFT THAT HAVE NOT BEEN ASSIGNED
C                TO A SUBSET
C          N - INDEX OF HIGHEST POINT USED FOR LAGRANGE POLYNOMIAL
C         N1 - INDEX OF SECOND HIGHEST POINT
C         N2 - INDEX OF THIRD HIGHEST POINT - LOWEST POINT FOR QUADRATIC
C         N3 - INDEX OF LOWEST POINT FOR CUBIC
C         NM - INDEX OF FIRST POINT TO BE TESTED
C     *******
C         A *
C         B *- INTERMEDIATE VALUES TO SIMPLIFY CODING OF THE
C         C *   LAGRANGE POLYNOMIAL
C         D *
C     *******
C         YJ - Y AT X(J) EVALUATED BY LAGRANGE POLYNOMIAL
C
      SUBROUTINE DH(X,Y,XM,LLOW,LHIGH,IX,IXM,IL)
      DIMENSION X(IX), Y(IX),XM(IXM),LLOW(IL), LHIGH(IL)
      RETURN
C
C     DETERMINE SUBSETS BY FORCE FITTING STARTING AT HIGH END OF DATA
C
      ENTRY FFHIGH(X,Y,XM,LLOW,LHIGH,NX,NS,NB,M,TOL)
C
      NSTRL = NS
      NN = NS
      NS = 0
  800 NPLFT = NX
      DEBUG NPLFT,NSTRL
      DO 810 I=1,NSTRL
      LHIGH(I)= 0
  810 LLOW(I)= 0
      LHIGH(NN)= NX
      DO 860 I=1,NN
      IF (NPLFT.LE.M)       GO TO 870
  820 II = NSTRL-I+1
      NS = NS+1
      DEBUG I,II,NS,NPLFT
      N = NPLFT
      N1 = N-1
      N2 = N-2
      N3 = N-3
      NM = N3-M+2
      NPLFT = NPLFT-M
      DEBUG I,N,N1,N2,N3,NM
      DO 850 J=1,NM
      JJ = NM-J+1
      IF (J.EQ.1)    LLOW(II)=JJ
      DEBUG J,JJ,X(JJ),Y(JJ)
      IF (M.EQ.3)       GO TO 830
```

```
      A = (X(N1)-X(JJ))/(X(N)-X(N2))
      B = (X(N)-X(JJ))/(X(N1)-X(N2))
      C = (X(N2)-X(JJ))/(X(N)-X(N1))
      YJ = Y(N2)*A*B-Y(N1)*B*C+Y(N)*C*A
      DEBUG A,B,C,YJ
      GO TO 840
  830 A = (X(N2)-X(JJ))/(X(N)-X(N3))
      B = (X(N1)-X(JJ))/(X(N2)-X(N3))
      C = (X(N)-X(JJ))/(X(N1)-X(N3))
      D =(X(N3)-X(JJ))/(X(N1)-X(N2))
      YJ = Y(N3)*A*B*C-Y(N2)*D*B*(X(N)-X(JJ))/(X(N)-X(N2))+Y(N1)*D*C*
     1 (X(N2)-X(JJ))/(X(N)-X(N1))-Y(N)*A*(X(N3)-X(JJ))*(X(N1)-X(JJ))/
     2 (X(N)-X(N2))/(X(N)-X(N1))
      DEBUG A,B,C,D,YJ
  840 IF (Y(JJ).EQ.0.)       GO TO 841
      IF (ABS(1.-YJ/Y(JJ)).GT.TOL)    GO TO 859
      LLOW(II) = JJ
      GO TO 850
  841 IF (ABS(YJ-Y(JJ)).GT.TOL)    GO TO 859
      LLOW(II)= JJ
  850 NPLFT = NPLFT-1
  859 IF (II.NE.1)    LHIGH(II-1)=LLOW(II)
  860 CONTINUE
C
  870 LLOW(II) = 1
      DEBUG (LLOW(I),I=1,NSTRL)
      DEBUG (LHIGH(I),I=1,NSTRL)
      DO 880 I=II,NSTRL
      IJ = I-II+1
      LLOW(IJ)= LLOW(I)
      LHIGH(IJ)= LHIGH(I)
      JJ= LHIGH(IJ)
      XM(IJ)= X(JJ)
  880 CONTINUE
      NB= NS-1
      DEBUG (XM(I),I=1,NS)
      DEBUG(LLOW(I),I=1,NS)
      DEBUG (LHIGH(I),I=1,NS)
C
      RETURN
      END
```

```
$IBFTC REFT
C
C     PROGRAM VARIABLES
C     *****************
C
C        NST - NUMBER OF THE SUBSET FROM WHICH POINTS ARE BEING
C               CHECKED
C        NSS - NEW SUBSET COUNTER
C     *******
C        AA *
C        BB *- COEFFICIENTS FOR TESTING POLYNOMIAL,
C        CC *    Y = AA + BB*X + CC*X**2 + DD*X**3
C        DD *    (IF M=2, DD=0.)
C     *******
C          I - INDEX OF POINT BEING TESTED
C         YI - Y EVALUATED AT X(I) BY TESTING POLYNOMIAL
C         YJ - Y EVALUATED BY COEFFIENTS FOR SEGMENT NST
C         NS - NUMBER OF SEGMENTS IN FIRST FIT
C
      SUBROUTINE RFT(X,A,XM,LLOW,LHIGH,IX,IA,IXM,IM1)
      DIMENSION X(IX), A(IA,IM1),XM(IXM),LLOW(IA),LHIGH(IA)
      DOUBLE PRECISION A,AA, BB,CC,DD
      GO TO 140
C
C     CHECK IF DATA SHOULD BE REFITTED AND DETERMINE NEW SUBSETS
C
      ENTRY REFIT(X,A,XM,LLOW,LHIGH,NX,NS,NSS,M,TOL)
      DEBUG (LLOW(I),I=1,NS)
```

```
      DEBUG (LHIGH(I),I=1,NS)
      NST=2
      NSS=1
      IST= LHIGH(1)+1
      AA= A(1,1)
      BB= A(1,2)
      CC= A(1,3)
      DD= 0.D0
      IF(M.EQ.3)  DD=A(1,4)
      DEBUG AA,BB,CC,DD,NSS,NST
C
  100 DO 130  I= IST,NX
      IF(X(I).GT.XM(NST))  NST= NST+1
      YI= AA+X(I)*(BB+X(I)*(CC+DD*X(I)))
      YJ= A(NST,M+1)
      DO 110  J=1,M
      IJ= M-J+1
      YJ= YJ*X(I)+A(NST,IJ)
  110 CONTINUE
      DEBUG NSS, NST,I,YI,YJ
      IF (YI.NE.0.)   GO TO 120
      IF (ABS(YI-YJ).LE.ABS(TOL))   GO TO 130
      GO TO 125
  120 IF (ABS(1.-YJ/YI).LE.ABS(TOL))   GO TO 130
C
C     END OF NSS SUBSET
C
  125 LHIGH(NSS)= I-1
      IF (NSS.NE.NS)   LLOW(NSS+1)=I-1
      IST= I+M
      IF (IST.GT.NX)   GO TO 135
      NSS= NSS+1
      AA= A(NST,1)
      BB= A(NST,2)
      CC= A(NST,3)
      IF(M.FQ.3)  DD=A(NST,4)
      DEBUG I,NSS,NST,AA,BB,CC,DD
      GO TO 100
  130 TOL = -ABS(TOL)
  135 LHIGH(NSS) = NX
      DEBUG (LLOW(I),I=1,NSS)
      DEBUG (LHIGH(I),I=1,NSS)
C
  140 RETURN
      END
```

```
$IBFTC MINV
C
C     PROGRAM VARIABLES
C     *****************
C
C        AIN - MATRIX TO BE INVERTED
C         NN - ORDER OF AIN
C        DET - VALUES OF THE DETERMINANT
C        ERR - MAXIMUM DEVIATION OF ELEMENTS OF AIN*AIN(INVERSE)
C
C          A - WORKING MATRIX
C          N - NUMBER OF ROWS IN WORKING MATRIX
C        JND - NUMBER OF COLUMNS IN WORKING MATRIX
C         AK - VALUE OF THE FIRST ELEMENT IN PIVOTAL ROW
C              FROM UNIT MATRIX
C       ERR1 - SCALAR PRODUCT OF I(TH) ROW OF AIN AND J(TH) COLUMN
C              OF AIN(INVERSE). ALSO, THE DEVIATION OF THE I,J(TH)
C              ELEMENT FROM UNIT MATRIX
C
C     MATRIX INVERSION BY GAUSSIAN ELIMINATION
C
      SUBROUTINE MINVRT(AIN,NN,DET,ERR)
      DOUBLE PRECISION AIN,A,AK,DET,ERR,ERR1
      DIMENSION AIN(4,4),A(4,8)
```

```
C
C     TRANSFER INPUT MATRIX (AIN) TO WORKING ARRAY (A) AND FILL
C     REMAINDER OF WORKING ARRAY WITH UNIT MATRIX
C
      N= NN
      JND= 2*N
      DO 110  I=1,N
      DO 100  J=1,N
      A(I,J)= AIN(I,J)
      JN= J+N
      A(I,JN)= 0.D0
      IF(I.EQ.J)  A(I,JN)= 1.D0
  100 CONTINUE
      DEBUG (A(I,J), J=1,JND)
  110 CONTINUE
C
C     CALCULATE DETERMINANT AND ELIMINATE THE .BELOW THE DIAGONAL.
C     ELEMENTS OF LEFT SIDE OF A
C
      DET = 1.D0
      DO 230  I=1,N
      DET = DET*A(I,I)
      IF (A(I,I).EQ.0.D0)  GO TO 600
      JST = I
      AK = A(I,I)
      DO 200  J= JST,JND
  200 A(I,J)= A(I,J)/AK
      DEBUG I,(A(I,J),J=1,JND)
      IF(I.EQ.N)  GO TO 300
      KST= I+1
      DO 220  K=KST,N
      AK= A(K,I)
      DO 210  J= JST,JND
  210 A(K,J)= A(K,J) -A(I,J)*AK
      DEBUG K, (A(K,J),J=1,JND)
  220 CONTINUE
  230 CONTINUE
      DEBUG DET
C
C     ELIMINATE THE ABOVE THE DIAGONAL ELEMENTS OF LEFT SIDE OF A
C
  300 DO 330  I=2,N
      KND= I-1
      DO 320  K=1,KND
      AK= A(K,I)
      JST=I
      DO 310  J= JST,JND
      A(K,J)= A(K,J)-A(I,J)*AK
  310 CONTINUE
      DEBUG K,(A(K,J),J=1,JND)
  320 CONTINUE
  330 CONTINUE
      DEBUG N,(A(N,J),J=1,JND)
C
C     INVERSE IS IN RIGHT SIDE OF A. MULTIPLY INPUT MATRIX BY
C     THE INVERSE AND FIND THE MAXIMUM DEVIATION OF ELEMENTS
C     OF THE PRODUCT MATRIX FROM THE UNIT MATRIX
C
      ERR= 0.D0
      DO 420  I=1,N
      DO 410  J=1,N
      JN= J+N
      ERR1= 0.D0
      DO 400  K=1,N
  400 ERR1= ERR1+AIN(I,K)*A(K,JN)
      ERR1= DABS(ERR1)
      IF(J.EQ.I)  ERR1= ERR1-1.D0
      IF(ERR1.GT.ERR)  ERR= ERR1
      DEBUG I,J,ERR,ERR1
  410 CONTINUE
  420 CONTINUE
C
C     TRANSFER INVERSE TO INPUT ARRAY
```

```
C
      DO 510  I=1,N
      DO 500  J=1,N
      JN= J+N
      AIN(I,J)= A(I,JN)
  500 CONTINUE
  510 CONTINUE
      RETURN
C
C     FOR SINGULAR MATRIX, RETURN NULL MATRIX
C
  600 DO 620  I=1,N
      DO 610  J=1,N
  610 A(I,J)= 0.D0
  620 CONTINUE
      RETURN
C
      END
```

```
$IBFTC MATDEF
C
C     PROGRAM VARIABLES
C     *****************
C
C         T - ONE BLOCK OF MATRIX PRODUCT X-TRANSPOSE *W*X
C        TT - POWER OF X(I) IN FORMATION OF T
C       DET - DETERMINANT OF T
C       ERR - VALUE THAT MEASURES ACCURACY OF T-INVERSE
C
      SUBROUTINE DEF(XX,YY,W,XM,LLOW,LHIGH,XWX,YWX,C,IX,IXM,IL,IM1,IM,
     1      IN1)
      DIMENSION XX(IX),YY(IX),W(IX),XM(IXM),LLOW(IL),LHIGH(IL),
     1      YWX(IL,IM1),XWX(IL,IM1,IM1),C(IN1,IM,IM1),T(4,4)
      DOUBLE PRECISION YWX,XWX,C,T,TT,DET,ERR
      GO TO 400
C
C     DEFINE THREE MULTIDIMENSIONAL MATRICES REQUIRED FOR SOLUTION
C     OF VECTOR A-TRANSPOSE
C
      ENTRY DEFMAT(XX,YY,W,XM,LLOW,LHIGH,NX,NS,M,XWX,YWX,C)
C
C     DEFINE MATRIX OF CONSTRAINTS IF THERE IS MORE THAN ONE SEGMENT
C
  100 NN = NS-1
      M1 = M+1
      MM = M
      IF (NS.EQ.1)   GO TO 200
      DO 120 N=1,NN
C
C     FIRST ROW
C
      C(N,1,1) = 1.D0
      DO 110 K=2,M1
  110 C(N,1,K) = C(N,1,K-1)*XM(N)
C
C     SECCND ROW
C
      C(N,2,1) = 0.D0
      C(N,2,2) = 1.D0
      C(N,2,3) = 2.D0*XM(N)
      IF (MM.EQ.2) GO TO 120
      C(N,2,4) = 3.D0*XM(N)**2
C
C     THIRD ROW
C
      C(N,3,1) = 0.D0
      C(N,3,2) = 0.D0
      C(N,3,3) = 2.D0
      C(N,3,4) = 6.D0*XM(N)
```

```
  120 CONTINUE
      DO 130 N=1,NN
      DEBUG N
      DO 130 J=1,MM
      DEBUG J,(C(N,J,K),K=1,M1)
  130 CONTINUE
C
C     DEFINE MATRIX XWX AND VECTOR YWX
C
  200 NN = NS
      DO 300 N=1,NN
      KST= LLOW(N)
      KND= LHIGH(N)
      DEBUG KST,KND
      DO 210 J=1,M1
      T(1,J) = 0.D0
      T(J,M1) = 0.D0
  210 YWX(N,J) = 0.D0
      DO 240 K=KST,KND
      TT = W(K)
      DO 220 J=1,M1
      T(1,J) = T(1,J)+TT
      YWX(N,J) = YWX(N,J)+TT*YY(K)
  220 TT = TT*XX(K)
      DO 230 I=2,M1
      T(I,M1) = T(I,M1)+TT
      TT = TT*XX(K)
  230 CONTINUE
  240 CONTINUE
C
      DO 270 I=2,M1
      DO 260 J=1,MM
  260 T(I,J) = T(I-1,J+1)
  270 CONTINUE
      DEBUG N
      DO 271 I=1,M1
      DEBUG I,(T(I,J),J=1,M1)
  271 CONTINUE
      DEBUG (YWX(N,I),I=1,M1)
C
      CALL MINVRT(T,M1,DET,ERR)
      DEBUG DET,ERR
      DO 290 I=1,M1
      DEBUG I,(T(I,J),J=1,M1)
      DO 290 J=1,M1
      XWX(N,I,J) = T(I,J)
  290 CONTINUE
  300 CONTINUE
C
  400 RETURN
      END
```

```
$IBFTC ASLV
C
C     PROGRAM VARIABLES
C     *****************
C          B - MATRIX PRODUCT C*(X-TRANSPOSE*W*X)*C-TRANSPOSE
C          N - ROW INDEX OF SUBMATRICES OF B
C          L - COLUMN INDEX OF SUBMATRICES OF B (L=1,2,3)
C          I - ROW INDEX OF ELEMENTS OF THE SUBMATRICES B(N,L)
C          J - COLUMN INDEX OF ELEMENTS OF THE SUBMATRICES B(N,L)
C       SIGN - PLUS OR MINUS 1 - CHANGES SIGN OF MATRIX PRODUCT
C         TT - INTERMEDIATE MATRIX IN THE DOUBLE MULTIPLICATION
C          T - INTERMEDIATE MATRIX IN THE DOUBLE MULTIPLICATION
C       BINV - INVERSE OF B
C          E - MATRIX E FROM SOLUTION FOR B-INVERSE IN APPENDIX B
C      DELTA - MATRIX DELTA FROM SOLUTION FOR B-INVERSE
C         TS - INTERMEDIATE VALUE RELATED TO THE IDENTITY SUBMATRICES
C                OF APPENDIX B
```

```
C          V - VECTOR V OF APPENDIX B
C         VV - VECTOR VV OF APPENDIX B
C
      SUBROUTINE SLV(C,XWX,YWX,A,B,BINV,IC,IM,IM1,IL)
      DIMENSION C(IC,IM,IM1),XWX(IL,IM1,IM1),YWX(IL,IM1),A(IL,IM1),
     1      B(IC,IM,IM,IM),BINV(IC,IC,IM,IM)
      DIMENSION T(4,4),TT(4,4),E(4,4),DELTA(3,3),V(4),VV(4)
      DOUBLE PRECISION C,XWX,YWX,A,B,BINV,T,TT,E,DELTA,V,VV,TS,DET,DIV,
     1      SIGN
      GO TO 940
C
C     SOLVE MATRIX EQUATION FOR A-TRANSPOSE
C
      ENTRY ASOLVE(C,XWX,YWX,A,NS,M)
C
C     FOR ONE SEGMENT, DO A SIMPLE LEAST SQUARES FIT
C
      IF (NS.GT.1)   GO TO 100
      M1 = M+1
      DO 95 I=1,M1
      A(1,I) = 0.D0
      DO 90 K=1,M1
      A(1,I) = A(1,I)+XWX(1,I,K)*YWX(1,K)
   90 CONTINUE
   95 CONTINUE
      RETURN
C
C     DEFINE B MATRIX
C
  100 NN = NS-1
      MM = M
      M1 = M+1
      DO 200 N=1,NN
      DO 190 L=1,3
      NIND = (L-1)/2+N
      GO TO (1,2,3),L
    1 IF (N.EQ.1) GO TO 190
      JIND = N-1
      GO TO 120
C
    2 SIGN = 1.D0
      JIND = N
      DO 110 I=1,M1
      DO 110 J=1,M1
      T(I,J) = XWX(N,I,J)+XWX(N+1,I,J)
  110 CONTINUE
      GO TO 140
C
    3 IF (N.EQ.NN)   GO TO 200
      JIND = N+1
  120 SIGN = -1.D0
      DO 130 I=1,M1
      DO 130 J=1,M1
      T(I,J) = XWX(NIND,I,J)
  130 CONTINUE
C
  140 DO 160 I=1,M1
      DO 150 J=1,MM
      TT(I,J) = 0.D0
      DO 150 K=1,M1
  150 TT(I,J) = TT(I,J)+T(I,K)*C(JIND,J,K)
      DEBUG I,(TT(I,J),J=1,MM)
  160 CONTINUE
C
      DO 180 I=1,MM
      DO 175 J=1,MM
      B(N,L,I,J) = 0.D0
      DO 170 K=1,M1
  170 B(N,L,I,J) = B(N,L,I,J)+C(N,I,K)*TT(K,J)
  175 B(N,L,I,J) = B(N,L,I,J)*SIGN
  180 CONTINUE
  190 CONTINUE
  200 CONTINUE
C
```

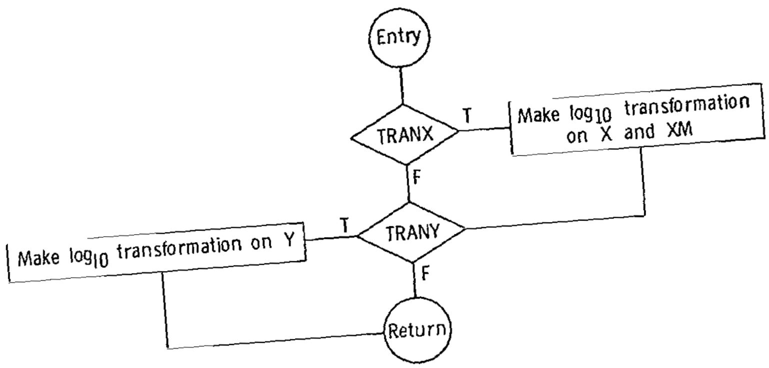
```
      DO 370  K=1,MM
      DELTA(I,J)= DELTA(I,J)+B(LL,1,I,K)*T(K,J)
  370 E(I,J)= E(I,J)+B(LL,1,I,K)*TT(K,J)
      E(I,J)= B(LL,2,I,J)-E(I,J)
      TS= 0.D0
      IF(LL.EQ.N.AND.I.EQ.J)  TS=1.D0
      DELTA(I,J)= TS-DELTA(I,J)
  380 CONTINUE
  390 CALL MINVRT(E,MM,DET,DEV)
  396 CONTINUE
C
  400 DO 420  I=1,MM
      DO 410  J=1,MM
      DO 410  K=1,MM
      DELTA(I,J)= DELTA(I,J)-B(L,3,I,K) *BINV(L+1,N,K,J)
  410 CONTINUE
  420 CONTINUE
C
      DO 440  I=1,MM
      DO 430  J=1,MM
      BINV(L,N,I,J)= 0.D0
      DO 430  K=1,MM
      BINV(L,N,I,J)= BINV(L,N,I,J)+E(I,K) *DELTA(K,J)
  430 CONTINUE
      DEBUG L,N, (BINV(L,N,I,J),J=1,MM)
  440 CONTINUE
C
      IF (L.EQ.1)  GO TO 450
      L= L-1
      GO TO 310
  450 CONTINUE
C
C     CARRY OUT MATRIX MULTIPLICATIONS FOR A-TRANSPOSE
C
  500 DO 930  N=1,NS
      DO 510 L=1,M1
      V(L) = YWX(N,L)
  510 CONTINUE
      DEBUG N,(V(L),L=1,M1)
      DO 850 JJ=1,NS
C
C     FORM MATRIX D(JJ,N) BY 4 SEPARATE TECHNIQUES
C
      IF(N.NE.1)  GO TO 550
      NDUM= 1
      IF(JJ.NE.1)  GO TO 530
      JDUM= 1
      SIGN = 1.D0
      GO TO 570
  530 IF(JJ.NE.NS)  GO TO 540
      JDUM= NS-1
      SIGN = -1.D0
      GO TO 570
  540 SIGN=1.D0
      GO TO 630
  550 IF(N.NE.NS)  GO TO 561
      NDUM= NS-1
      IF(JJ.NE.1)  GO TO 560
      JDUM=1
      SIGN= -1.D0
      GO TO 570
  560 IF(JJ.NE.NS)  GO TO 563
      JDUM= NS-1
      SIGN = 1.D0
      GO TO 570
  561 IF (JJ.NE.1)   GO TO 562
      JDUM = 1
      SIGN = 1.D0
      GO TO 615
  562 IF (JJ.NE.NS)   GO TO 680
      JDUM = NS-1
      SIGN = -1.D0
      GO TO 615
```

```
C      FIND INVERSE OF B
C
       DO 450 N=1,NN
       DO 280 L=1,NN
       IF (L.NE.1) GO TO 220
       DO 210 I=1,MM
       DO 210 J=1,MM
       E(I,J) = B(1,2,I,J)
       DELTA(I,J) = 0.D0
       IF (L.EQ.N.AND.I.EQ.J)  DELTA(I,J)=1.D0
   210 CONTINUE
       GO TO 270
C
   220 DO 240 I=1,MM
       DO 230 J=1,MM
       T(I,J) = 0.D0
       TT(I,J) = 0.D0
       DO 230 K=1,MM
       T(I,J) = T(I,J)+E(I,K)*DELTA(K,J)
       TT(I,J) = TT(I,J)+E(I,K)*B(L-1,3,K,J)
   230 CONTINUE
   240 CONTINUE
C
       DO 265 I=1,MM
       DO 260 J=1,MM
       E(I,J) = 0.D0
       DELTA(I,J) = 0.D0
       DO 250 K=1,MM
       E(I,J) = E(I,J)+B(L,1,I,K)*TT(K,J)
   250 DELTA(I,J) = DELTA(I,J)+B(L,1,I,K)*T(K,J)
       TS = 0.D0
       IF (L.EQ.N.AND.I.EQ.J) TS=1.D0
       DELTA(I,J) = TS-DELTA(I,J)
       E(I,J) = B(L,2,I,J)-E(I,J)
   260 CONTINUE
   265 CONTINUE
C
   270 CALL MINVRT(E,MM,DET,DEV)
   280 CONTINUE
C
       DO 300  I=1,MM
       DO 290  J=1,MM
       BINV(NN,N,I,J) = 0.D0
       DO 290  K=1,MM
   290 BINV(NN,N,I,J)= BINV(NN,N,I,J) + E(I,K)*DELTA(K,J)
       DEBUG NN,N,(BINV(NN,N,I,J),J=1,MM)
   300 CONTINUE
C
       IF (NN.EQ.1)   GO TO 500
       L= NN-1
   310 LAST = L
       DO 396 LL=1,LAST
   320 IF(LL.NE.1)  GO TO 340
       DO 330  I=1,MM
       DO 330  J=1,MM
       E(I,J) = B(1,2,I,J)
       DELTA(I,J)= 0.D0
       IF(LL.EQ.N.AND.I.EQ.J)  DELTA(I,J)= 1.D0
   330 CONTINUE
       GO TO 390
C
   340 DO 360  I=1,MM
       DO 350  J=1,MM
       T(I,J)= 0.D0
       TT(I,J)= 0.D0
       DO 350  K=1,MM
       T(I,J)= T(I,J)+E(I,K)*DELTA(K,J)
   350 TT(I,J) = TT(I,J)+E(I,K)*B(LL-1,3,K,J)
   360 CONTINUE
       DO 380  I=1,MM
       DO 380  J=1,MM
       DELTA(I,J)= 0.D0
       E(I,J)= 0.D0
```

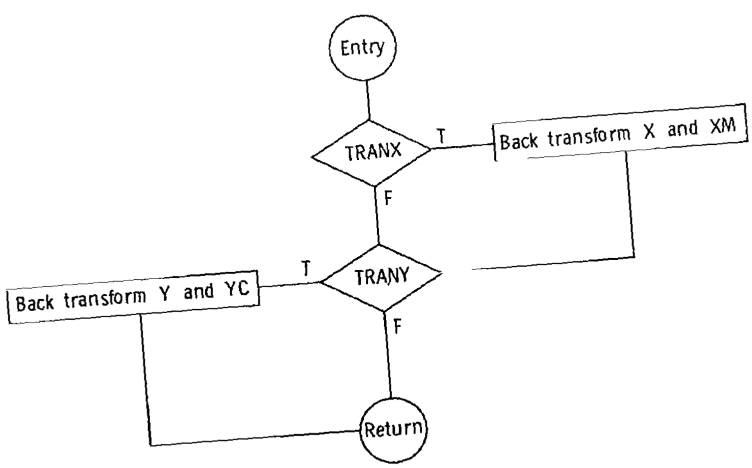
```
  563 SIGN = -1.D0
      GO TO 630
C
  570 DO 590  I=1,M1
      DO 580  J=1,MM
      T(I,J)= 0.D0
      DO 580  K=1,MM
      T(I,J)= T(I,J)+C(JDUM,K,I)*BINV(JDUM,NDUM,K,J)
  580 CONTINUE
      DEBUG (T(I,J), J=1,MM)
  590 CONTINUE
C
      DO 610  I=1,M1
      DO 605  J=1,M1
      TT(I,J)= 0.D0
      DO 600  K=1,MM
      TT(I,J)= TT(I,J) +T(I,K)*C(NDUM,K,J)
  600 CONTINUE
      TT(I,J)= TT(I,J)*SIGN
  605 CONTINUE
      DEBUG (TT(I,J),J=1,M1)
  610 CONTINUE
      GO TO 800
C
  615 DO 617 I=1,MM
      DO 616 J=1,M1
      T(I,J) = 0.D0
      DO 616 K=1,MM
      T(I,J) = T(I,J)+BINV(JDUM,N,I,K)*C(N,K,J)-BINV(JDUM,N-1,I,K)*
     1      C(N-1,K,J)
  616 CONTINUE
      DEBUG (T(I,J),J=1,M1)
  617 CONTINUE
C
      DO 620 I=1,M1
      DO 619 J=1,M1
      TT(I,J) = 0.D0
      DO 618 K=1,MM
      TT(I,J) = TT(I,J)+C(JDUM,K,I)*T(K,J)
  618 CONTINUE
      TT(I,J) = TT(I,J)*SIGN
  619 CONTINUE
      DEBUG (TT(I,J),J=1,M1)
  620 CONTINUE
      GO TO 800
C
  630 DO 650  I=1,M1
      DO 640  J=1,MM
      T(I,J)= 0.D0
      DO 640  K=1,MM
      T(I,J) = T(I,J)+C(JJ,K,I)*BINV(JJ,NDUM,K,J)-C(JJ-1,K,I)*
     1      BINV(JJ-1,NDUM,K,J)
  640 CONTINUE
      DEBUG (T(I,J), J=1,MM)
  650 CONTINUE
C
      DO 670  I=1,M1
      DO 665  J=1,M1
      TT(I,J)= 0.D0
      DO 660  K=1,MM
      TT(I,J)= TT(I,J)+T(I,K)*C(NDUM,K,J)
  660 CONTINUE
      TT(I,J) = TT(I,J)*SIGN
  665 CONTINUE
      DEBUG (TT(I,J), J=1,M1)
  670 CONTINUE
      GO TO 800
C
  680 DO 700  I=1,M1
      DO 690  J=1,MM
      T(I,J)= 0.D0
      DO 690  K=1,MM
      T(I,J) = T(I,J)+C(JJ-1,K,I)*BINV(JJ-1,N-1,K,J)-C(JJ,K,I)*
     1      BINV(JJ,N-1,K,J)
```

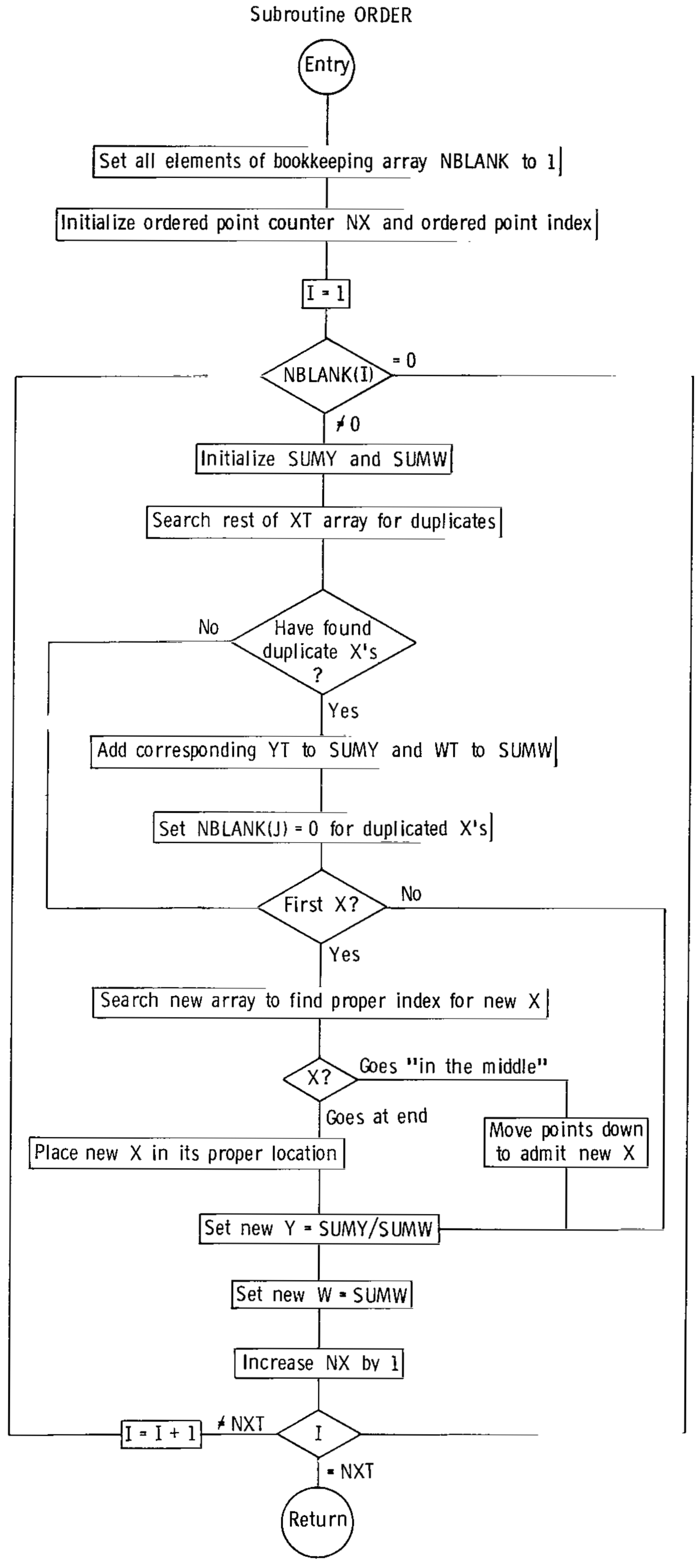
```
  690 CONTINUE
      DEBUG (T(I,J),J=1,MM)
  700 CONTINUE
C
      DO 720  I=1,M1
      DO 710  J=1,M1
      TT(I,J)= 0.D0
      DO 710  K=1,MM
      TT(I,J)= TT(I,J)+T(I,K)*C(N-1,K,J)
  710 CONTINUE
      DEBUG (TT(I,J),J=1,M1)
  720 CONTINUE
C
      DO 740  I=1,M1
      DO 730  J=1,MM
      T(I,J)= 0.D0
      DO 730  K=1,MM
      T(I,J) = T(I,J)+C(JJ,K,I)*BINV(JJ,N,K,J)-C(JJ-1,K,I)*
     1       BINV(JJ-1,N,K,J)
  730 CONTINUE
      DEBUG (T(I,J),J=1,MM)
  740 CONTINUE
C
      DO 760  I=1,M1
      DO 750  J=1,M1
      DO 750  K=1,MM
      TT(I,J)= TT(I,J) + T(I,K)*C(N,K,J)
  750 CONTINUE
      DEBUG (TT(I,J),J=1,M1)
  760 CONTINUE
C
C     MATRIX D(JJ,N) IS STORED IN TT
C
  800 DO 820 L=1,M1
      VV(L) = 0.D0
      DO 810 K=1,M1
      VV(L) = VV(L)+YWX(JJ,K)*XWX(JJ,K,L)
  810 CONTINUE
  820 CONTINUE
      DEBUG JJ,(VV(L),L=1,M1)
      DO 840 L=1,M1
      DO 830 K=1,M1
      V(L) = V(L)-VV(K)*TT(K,L)
  830 CONTINUE
  840 CONTINUE
      DEBUG JJ,(V(L),L=1,M1)
  850 CONTINUE
C
C     FINAL MULTIPLICATION FOR A-TRANSPOSE
C
  900 DO 920 I=1,M1
      A(N,I) = 0.D0
      DO 910 K=1,M1
      A(N,I) = A(N,I)+V(K)*XWX(N,K,I)
  910 CONTINUE
  920 CONTINUE
      DEBUG N
      DEBUG (A(N,I),I=1,M1)
  930 CONTINUE
C
  940 RETURN
      END
```

Subroutine TRANSF

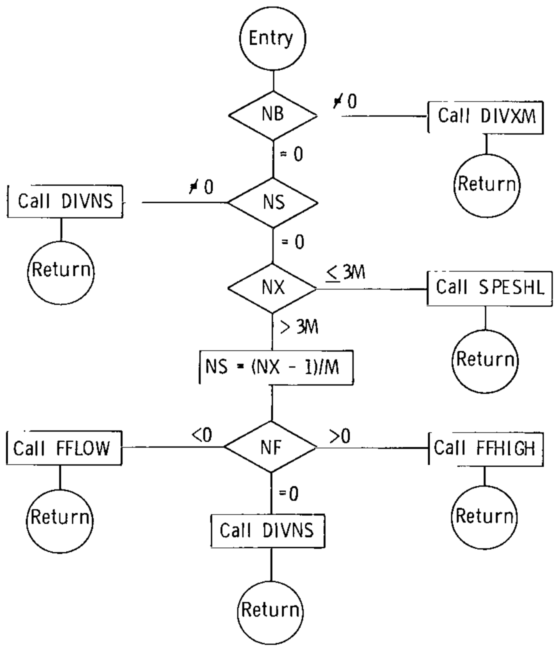


Subroutine BTRANS

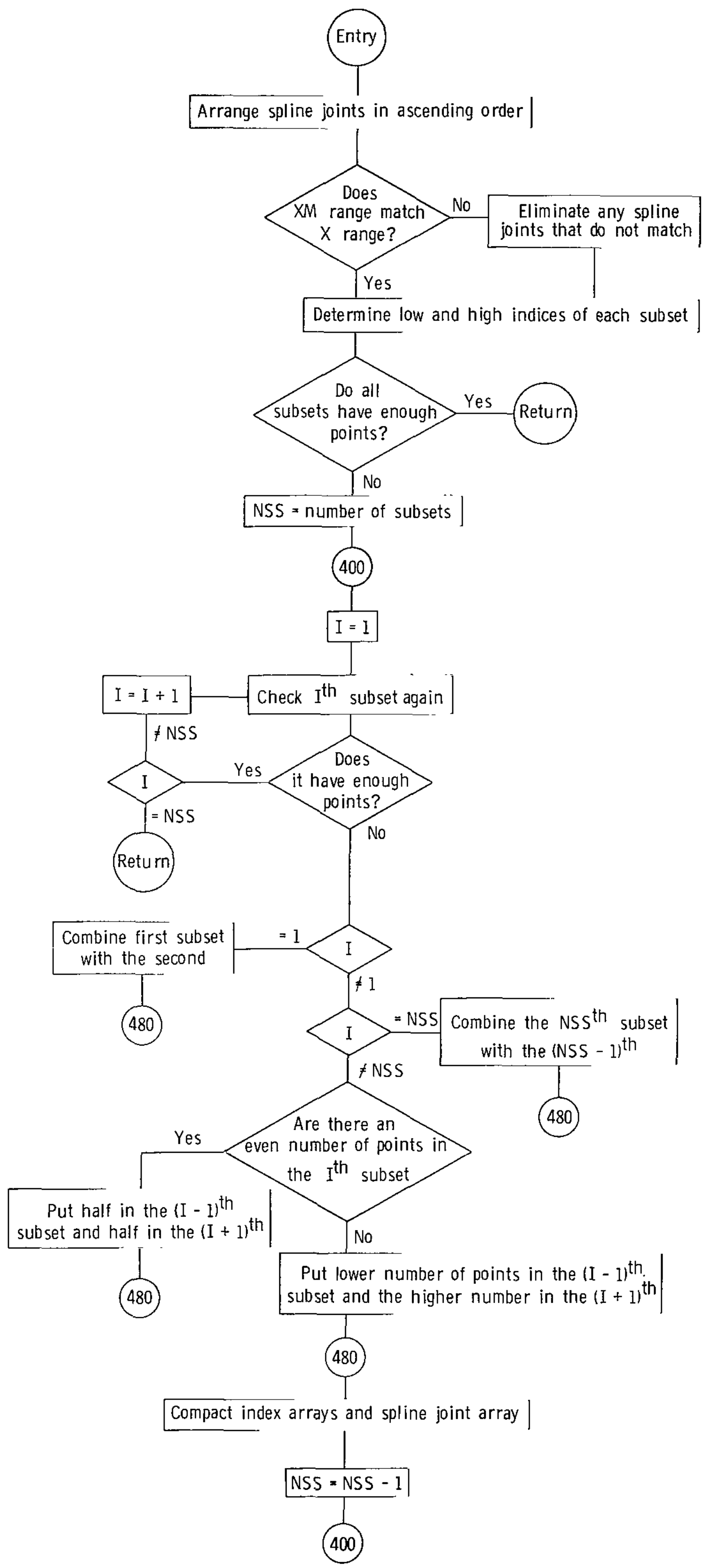



Subroutine ORDER
Entry
Set all elements of bookkeeping array NBLANK to 1
Initialize ordered point counter NX and ordered point index
I = 1
NBLANK(I)
= 0
≠ 0
Initialize SUMY and SUMW
Search rest of XT array for duplicates
Have found duplicate X's ?
No
Yes
Add corresponding YT to SUMY and WT to SUMW
Set NBLANK(J) = 0 for duplicated X's
First X?
No
Yes
Search new array to find proper index for new X
X?
Goes "in the middle"
Goes at end
Place new X in its proper location
Move points down to admit new X
Set new Y = SUMY/SUMW
Set new W = SUMW
Increase NX by 1
I = I + 1
≠ NXT
I
= NXT
Return

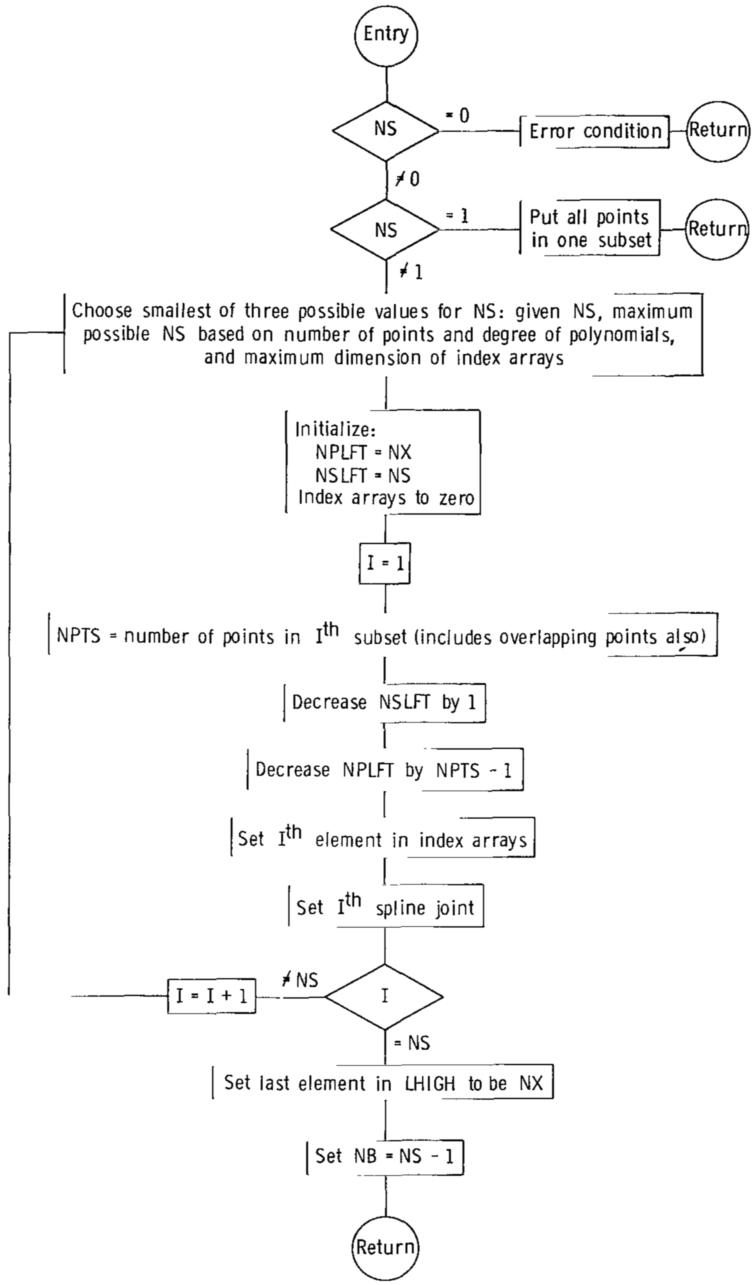
Subroutine SEGMNT



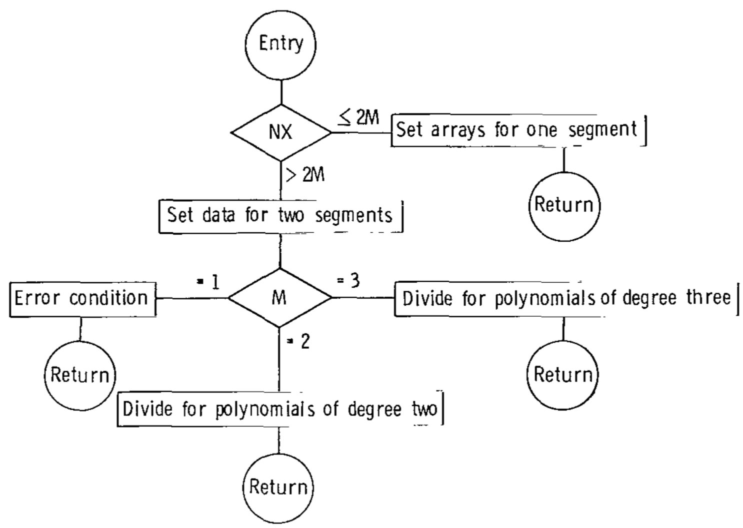
Subroutine DIVXM



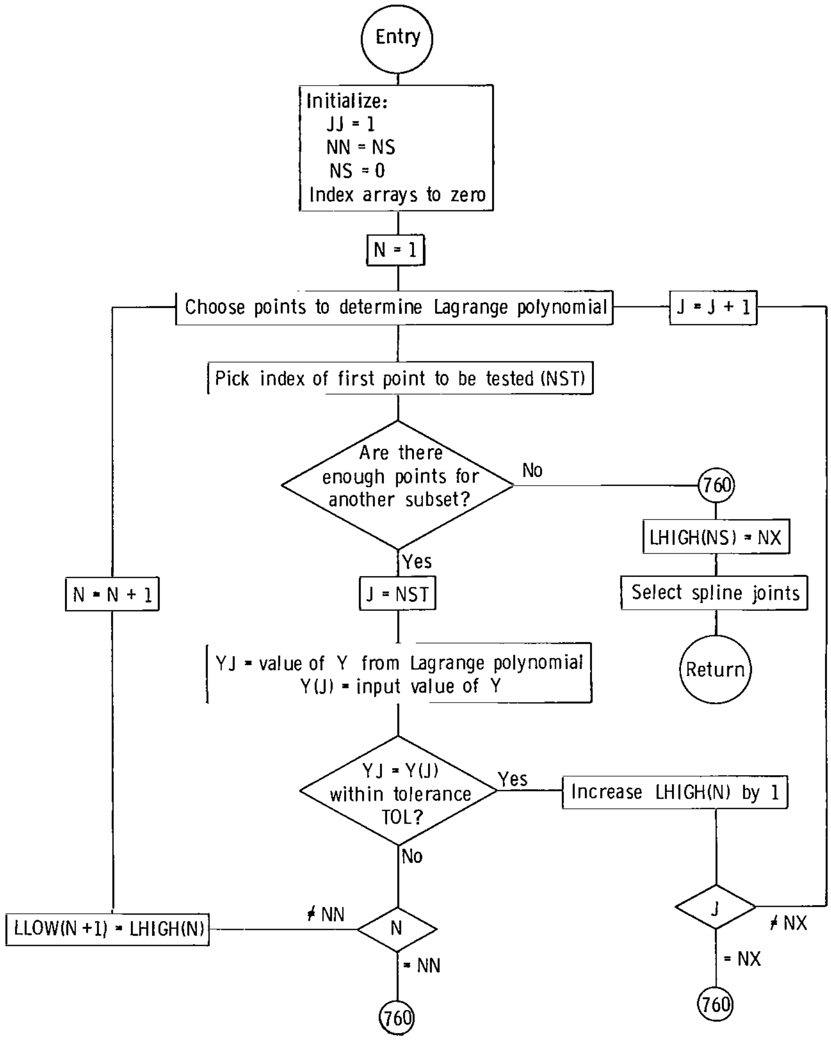
Subroutine DIVNS



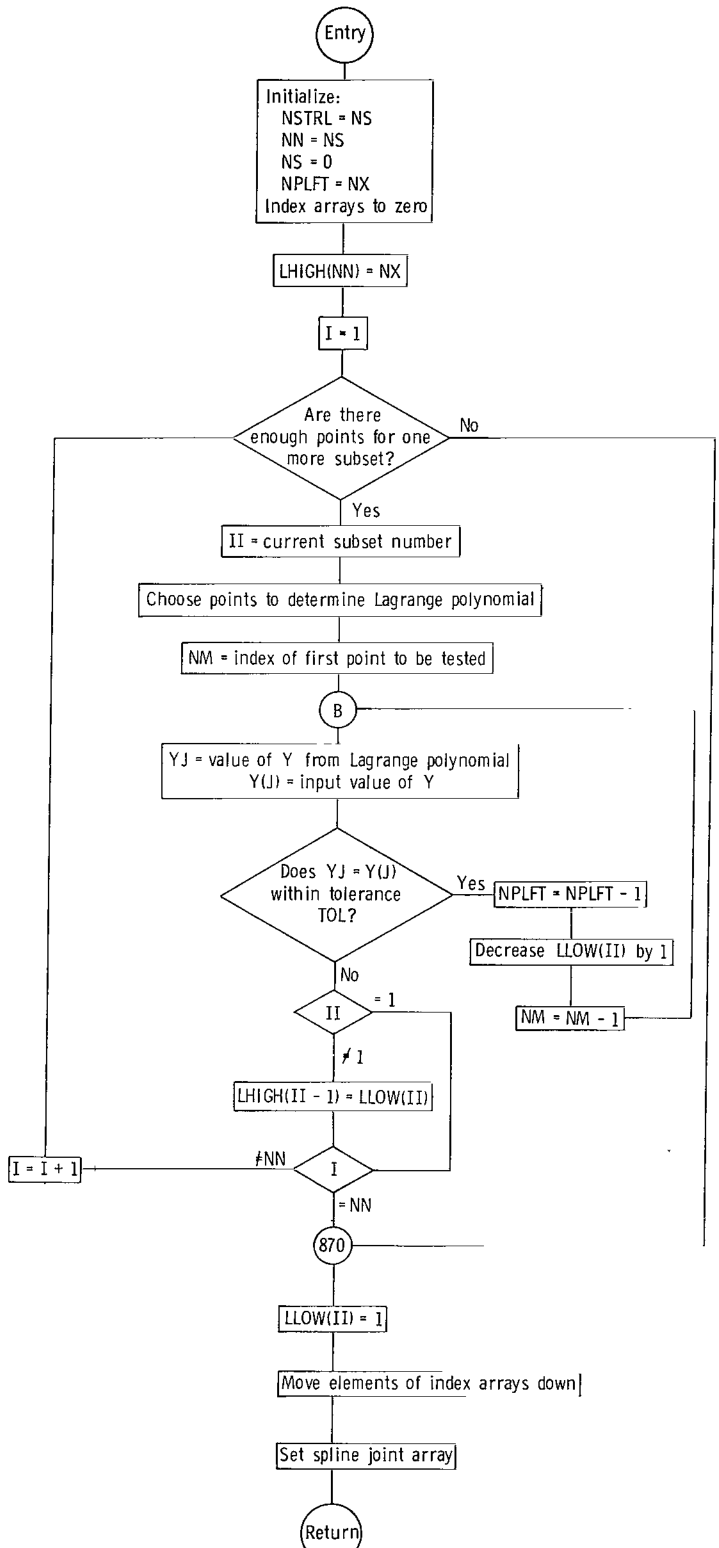
Subroutine SPESHL



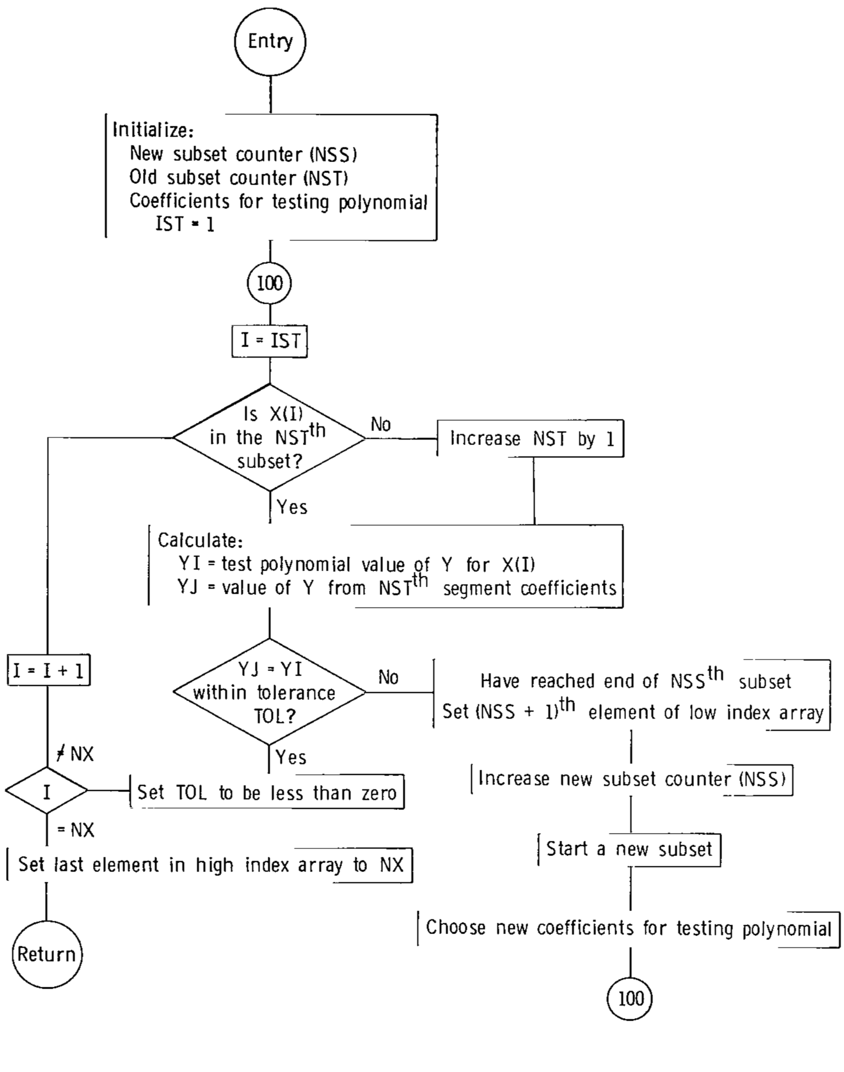
Subroutine FFLOW



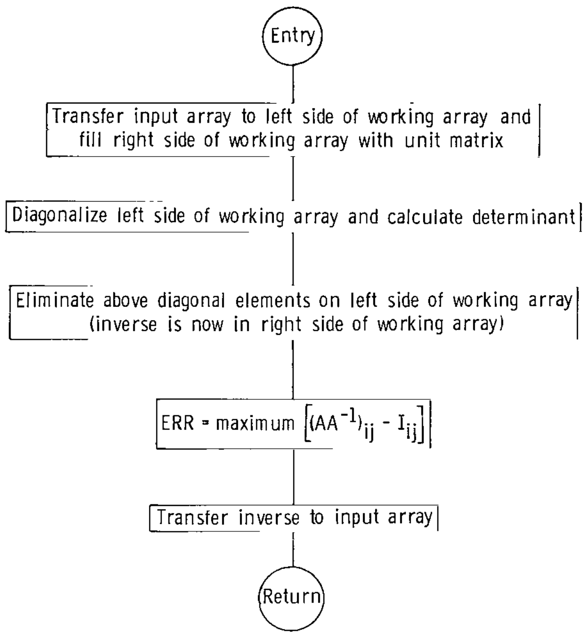
Subroutine FFHIGH



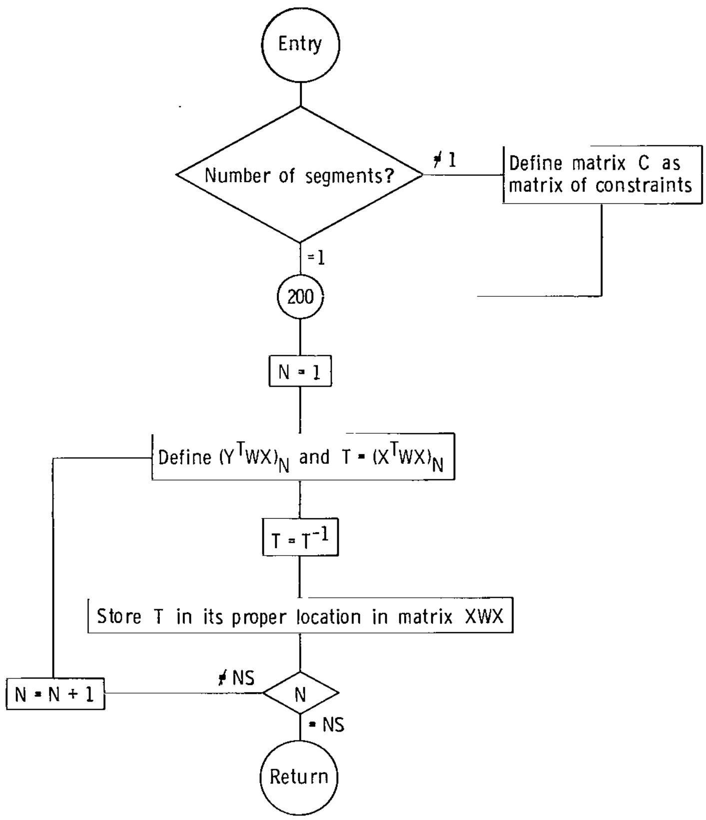
Subroutine REFIT



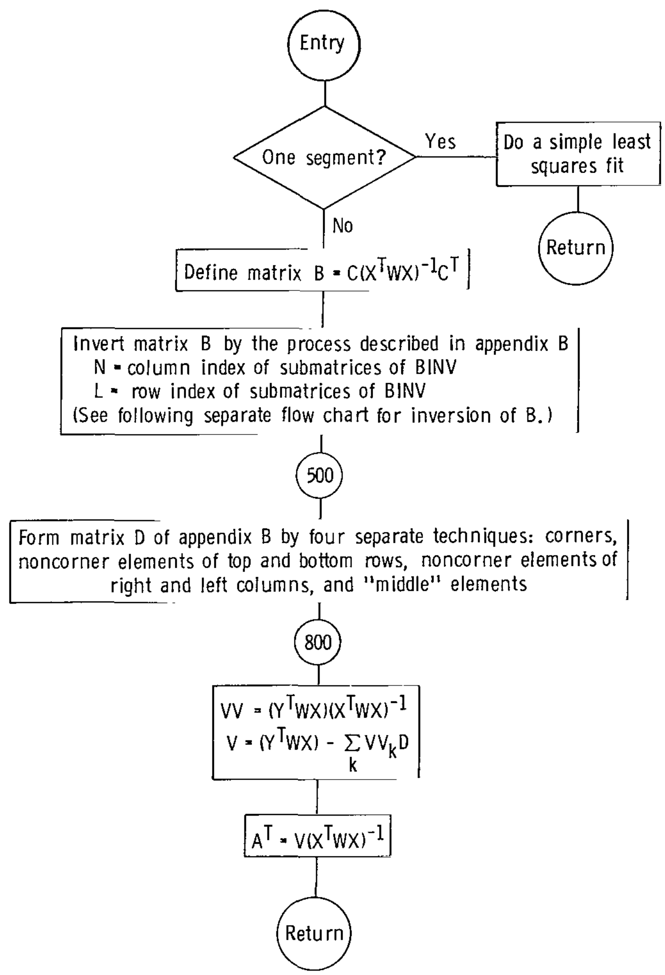
Subroutine MINVRT



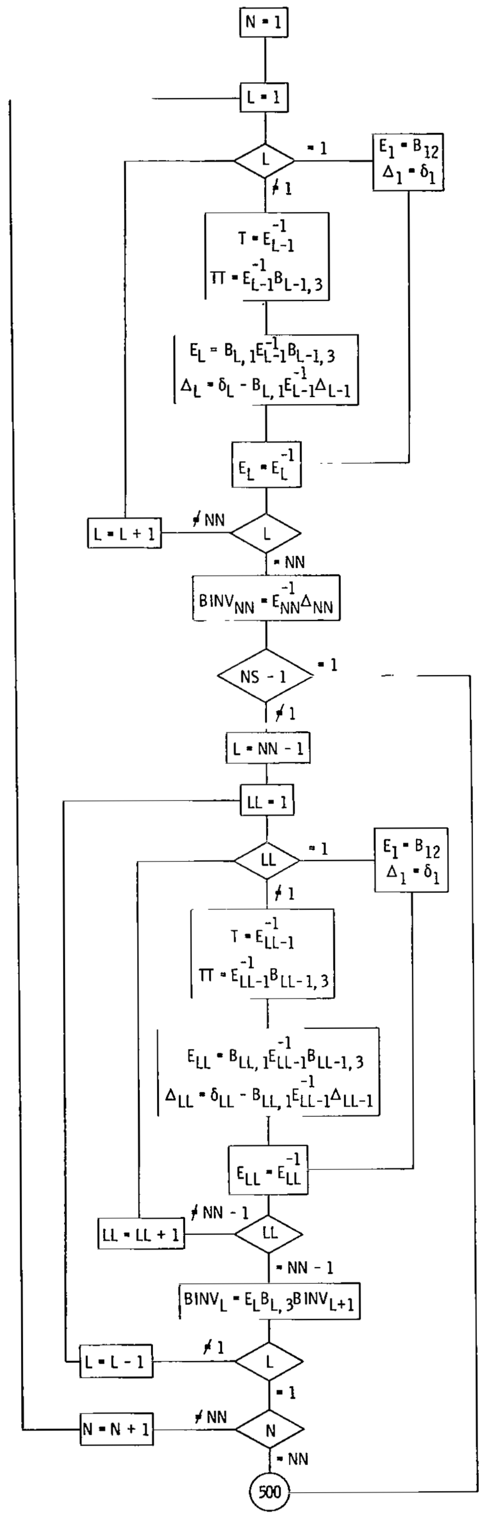
Subroutine DEFMAT



Subroutine ASOLVE



Inversion of Matrix B



APPENDIX F

COMPUTER INPUT AND OUTPUT SHEETS FOR SAMPLE PROBLEM 1

INPUT FOR SAMPLE PROBLEM 1

Card column									
1-6	7-12	13-18	19-24	25-30	31-36	37-42	43-48	49-60	61-66
	SAMPLE	PROBL	EM 1						
2 45	0 2	0 F	F F	F .0	01				
(6F12	.5)								
8	.86	5	.556			10	.98	7	.6798
12	.90	13	.116			16	.81	11	.967
60	.805	43	.043			120	.86	86	.038
180	.88	116	.49			240	.82	161	.03
248	.09	158	.00			255	.89	159	.89
265	.94	168	.00			275	.78	174	.92
279	.30	181	.84			300	.85	210	.51
360	.89	244	.47			374	.74	238	.03
406	.99	260	.51			415	.79	275	.91
445	.53	287	.06			480	.48	320	.67
506	.57	324	.27			513	.07	327	.02
573	.47	372	.89			600	.79	400	.49
626	.50	407	.06			644	.26	414	.99
699	.31	455	.99			722	.16	489	.77
900	.80	598	.51			968	.34	624	.53
2865	.4	1837	.9			3598	.7	2386	.4
3957	.1	2554	.0			4281	.0	2821	.0
7200	.0	4349	.0			8503	.6	5426	.0
11750	.0	7009	.0			12730	.0	7587	.0
15640	.0	8788	.0			16940	.0	9427	.5
19550	.0	10360	.0			21160	.0	11060	.0
23440	.0	11720	.0			31230	.0	14020	.0
47300	.0	14970	.0						
(12F6	.0)								
200.	7000.								
0									

OUTPUT FOR SAMPLE PROBLEM 1

Linear Fit

IND. VAR.	DEP.VAR.	CALC. FUNC.	DEVIATION	RELATIVE ERROR
8.85999990	5.55599999	473.876369	-468.320366	-0.98827541
10.9799999	7.67979997	474.759743	-467.079941	-0.98382381
12.8999997	13.1160001	475.559776	-462.443775	-0.97241987
16.8099997	11.9670000	477.189011	-465.222008	-0.97492187
60.8049994	43.0430002	495.521046	-452.478043	-0.91313587
120.859998	86.0380001	520.545029	-434.507027	-0.83471554
180.879997	116.490000	545.554428	-429.064426	-0.78647409
240.819998	161.030001	570.530502	-409.500500	-0.71775391
248.089996	158.000000	573.559792	-415.559792	-0.72452741
255.889997	159.889999	576.809937	-416.919937	-0.72280297
265.939995	168.000000	580.997612	-412.997612	-0.71084218
275.779995	174.920000	585.097786	-410.177784	-0.70104142
279.299995	181.840000	586.564514	-404.724514	-0.68999147
300.849995	210.510000	595.544067	-385.034065	-0.64652489
360.889996	244.469999	620.561806	-376.091805	-0.60605051
374.739998	238.030001	626.332886	-388.302883	-0.61996247
406.989994	260.509998	639.770958	-379.260960	-0.59280740
415.789997	275.910000	643.437782	-367.527782	-0.57119397
445.529991	287.060001	655.829979	-368.769978	-0.56229509
480.479996	320.669998	670.393097	-349.723099	-0.52166870
506.569996	324.270000	681.264397	-356.994396	-0.52401739
513.069992	327.020000	683.972847	-356.952847	-0.52188160
573.469994	372.889999	709.140587	-336.250587	-0.47416633
600.789993	400.490002	720.524406	-320.034405	-0.44416872
626.499992	407.060001	731.237366	-324.177364	-0.44332713
644.259995	414.990002	738.637680	-323.647678	-0.43816838
699.309990	455.990002	761.576164	-305.586163	-0.40125489
722.159996	489.770000	771.097404	-281.327404	-0.36484029
900.799995	598.510002	845.533920	-247.023918	-0.29215140
968.339981	624.529999	873.676781	-249.146782	-0.28517043
2865.39996	1837.89999	1664.15222	173.747772	0.10440617
3598.69998	2386.39999	1969.70695	416.693039	0.21155077
3957.09998	2554.00000	2119.04666	434.953339	0.20525897
4280.99994	2821.00000	2254.01074	566.989258	0.25154683
7199.99994	4349.00000	3470.31274	878.687256	0.25320117
8503.59985	5426.00000	4013.50259	1412.49741	0.35193633
11749.9999	7009.00000	5366.22705	1642.77295	0.30613183
12729.9999	7587.00000	5774.57782	1812.42218	0.31386228
15639.9998	8788.00000	6987.12964	1800.87036	0.25774108
16939.9998	9427.50000	7528.81946	1898.68054	0.25218835
19549.9998	10360.0000	8616.36584	1743.63416	0.20236306
21159.9998	11060.0000	9287.22791	1772.77209	0.19088280
23439.9998	11720.0000	10237.2684	1482.73157	0.14483663
31229.9998	14020.0000	13483.2406	536.759399	0.39809376E-01
47300.0000	14970.0000	20179.3601	-5209.36011	-0.25815289

THE REGRESSION EQUATION FOR THE ABOVE IS

Y = A0 + A1 X + ...

THE PARAMETERS (A0-A1) ARE
470.184547   0.41668448

THE VARIANCE= 1346098.9   STANDARD DEVIATION= 1160.2150
DETERMINANT= 86.72037

Parabolic Fit

IND. VAR.	DEP.VAR.	CALC. FUNC.	DEVIATION	RELATIVE ERROR
8.85999990	5.55599999	-7.82075906	13.3767591	-1.71041697
10.9799999	7.67979997	-6.36010355	14.0399035	-2.20749605
12.8999997	13.1160001	-5.03730673	18.1533067	-3.60377234
16.8099997	11.9670000	-2.34366494	14.3106649	-6.10610527
60.8049994	43.0430002	27.9484172	15.0945830	0.54008722
120.859998	86.0380001	69.2492838	16.7887163	0.24243884
180.879997	116.490000	110.469475	6.02052498	0.54499444E-01
240.819998	161.030001	151.578245	9.45175552	0.62355620E-01
248.089996	158.000000	156.560404	1.43959618	0.91951485E-02
255.889997	159.889999	161.904854	-2.01485443	-0.12444682E-01
265.939995	168.000000	168.789560	-0.78956032	-0.46777793E-02
275.779995	174.920000	175.528872	-0.60887146	-0.34687824E-02
279.299995	181.840000	177.939314	3.90068626	0.21921441E-01
300.849995	210.510000	192.692173	17.8178272	0.92467831E-01
360.889996	244.469999	233.756330	10.7136688	0.45832636E-01
374.739998	238.030001	243.220953	-5.19095230	-0.21342537E-01
406.989994	260.509998	265.247829	-4.73783112	-0.17861903E-01
415.789997	275.910000	271.255432	4.65456772	0.17159353E-01
445.529991	287.060001	291.549381	-4.48937988	-0.15398352E-01
480.479996	320.669998	315.380772	5.28922653	0.16770923E-01
506.569996	324.270000	333.158276	-8.88827515	-0.26678836E-01
513.069992	327.020000	337.585655	-10.5656548	-0.31297700E-01
573.469994	372.889999	378.694500	-5.80450058	-0.15327660E-01
600.789993	400.490002	397.269943	3.22005844	0.81054670E-02
626.499992	407.060001	414.740002	-7.68000031	-0.18517626E-01
644.259995	414.990002	426.801937	-11.8119354	-0.27675449E-01
699.309990	455.990002	464.158379	-8.16837692	-0.17598254E-01
722.159996	489.770000	479.650211	10.1197891	0.21098269E-01
900.799995	598.510002	600.481705	-1.97170258	-0.32835348E-02
968.339981	624.529999	646.034935	-21.5049362	-0.33287575E-01
2865.39996	1837.89999	1896.25816	-58.3581696	-0.30775435E-01
3598.69998	2386.39999	2364.37613	22.0238647	0.93148735E-02
3957.09998	2554.00000	2590.09534	-36.0953369	-0.13935911E-01
4280.99994	2821.00000	2792.35071	28.6492920	0.10259919E-01
7199.99994	4349.00000	4540.73303	-191.733032	-0.42225128E-01
8503.59985	5426.00000	5278.30939	147.690613	0.27980666E-01
11749.9999	7009.00000	6999.10071	9.89929199	0.14143663E-02
12729.9999	7587.00000	7486.02826	100.971741	0.13488026E-01
15639.9998	8788.00000	8842.99243	-54.9924316	-0.62187582E-02
16939.9998	9427.50000	9406.20850	21.2915039	0.22635586E-02
19549.9998	10360.0000	10456.8165	-96.8165283	-0.92587001E-02
21159.9998	11060.0000	11051.5273	8.47265625	0.76665025E-03
23439.9998	11720.0000	11824.0630	-104.062988	-0.88009500E-02
31229.9998	14020.0000	13847.4071	172.592896	0.12463914E-01
47300.0000	14970.0000	15009.6951	-39.6950684	-0.26446285E-02

THE REGRESSION EQUATION FOR THE ABOVE IS

Y = A0 + A1 X + ...

THE PARAMETERS (A0-A2) ARE

-13.9259609 0.68914430 -0.78545492E-05

THE VARIANCE= 3248.4323 STANDARD DEVIATION= 56.995019

DETERMINANT= 15.83930

Cubic Fit

IND. VAR.	DEP.VAR.	CALC. FUNC.	DEVIATION	RELATIVE ERROR
8.85999990	5.55599999	-1.94968595	7.50568593	-3.84968969
10.9799999	7.67979997	-0.50512648	8.18492639	-16.2037168
12.8999997	13.1160001	0.80309701	12.3129030	15.3317754
16.8099997	11.9670000	3.46707362	8.49992633	2.45161402
60.8049994	43.0430002	33.4265404	9.61645985	0.28768935
120.859998	86.0380001	74.2768517	11.7611485	0.15834204
180.879997	116.490000	115.050751	1.43924904	0.12509688E-01
240.819998	161.030001	155.717804	5.31219673	0.34114254E-01
248.089996	158.000000	160.646660	-2.64665985	-0.16475038E-01
255.889997	159.889999	165.933983	-6.04398346	-0.36424024E-01
265.939995	168.000000	172.745182	-4.74518204	-0.27469258E-01
275.779995	174.920000	179.412632	-4.49263191	-0.25040778E-01
279.299995	181.840000	181.797392	0.42608261E-01	0.23437223E-03
300.849995	210.510000	196.393326	14.1166744	0.71879603E-01
360.889996	244.469999	237.022972	7.44702721	0.31419010E-01
374.739998	238.030001	246.387926	-8.35792542	-0.33921814E-01
406.989994	260.509998	268.183537	-7.67353821	-0.28613010E-01
415.789997	275.910000	274.128231	1.78176880	0.64997639E-02
445.529991	287.060001	294.210201	-7.15019989	-0.24303032E-01
480.479996	320.669998	317.793720	2.87627792	0.90507701E-02
506.569996	324.270000	335.387054	-11.1170540	-0.33146938E-01
513.069992	327.020000	339.768669	-12.7486687	-0.37521613E-01
573.469994	372.889999	380.454430	-7.56443024	-0.19882618E-01
600.789993	400.490002	398.839809	1.65019226	0.41374813E-02
626.499992	407.060001	416.131741	-9.07173920	-0.21800162E-01
644.259995	414.990002	428.071049	-13.0810471	-0.30558121E-01
699.309990	455.990002	465.049557	-9.05955505	-0.19480838E-01
722.159996	489.770000	480.385475	9.38452530	0.19535406E-01
900.799995	598.510002	600.017441	-1.50743866	-0.25123247E-02
968.339981	624.529999	645.126060	-20.5960617	-0.31925639E-01
2865.39996	1837.89999	1884.78838	-46.8883820	-0.24877266E-01
3598.69998	2386.39999	2349.77625	36.6237488	0.15586058E-01
3957.09998	2554.00000	2574.14865	-20.1486511	-0.78273067E-02
4280.99994	2821.00000	2775.28751	45.7124939	0.16471264E-01
7199.99994	4349.00000	4517.62238	-168.622375	-0.37325469E-01
8503.59985	5426.00000	5254.61517	171.384827	0.32616057E-01
11749.9999	7009.00000	6978.67584	30.3241577	0.43452595E-02
12729.9999	7587.00000	7467.71558	119.284424	0.15973348E-01
15639.9998	8788.00000	8833.32520	-45.3251953	-0.51311589E-02
16939.9998	9427.50000	9401.30420	26.1958008	0.27864007E-02
19549.9998	10360.0000	10462.5498	-102.549805	-0.98016072E-02
21159.9998	11060.0000	11064.2125	-4.21252441	-0.38073423E-03
23439.9998	11720.0000	11846.6007	-126.600708	-0.10686670E-01
31229.9998	14020.0000	13895.7180	124.281982	0.89439050E-02
47300.0000	14970.0000	14985.5291	-15.5290527	-0.10362699E-02

THE REGRESSION EQUATION FOR THE ABOVE IS

Y = A0 + A1 X + ...

THE PARAMETERS (A0-A3) ARE
-7.98756391   0.68154073   -0.72963022E-05   -0.86881731E-11

THE VARIANCE= 3157.9090   STANDARD DEVIATION= 56.195276
DETERMINANT= 0.128715

SAMPLE PROBLEM 1

SPLINE JOINTS CHOSEN BY PROGRAMMER

DEGREE OF POLYNOMIAL = 2 NUMBER OF SEGMENTS = 3

EQUATION FITTED IS Y = A0 + A1 X + A2 X**2

SEGMENT COEFFICIENTS IN ASCENDING ORDER -

A0	A1	A2
4.056095511918016D 00	5.379181567783746D-01	3.485214235920547D-04
-1.016499811302856D 01	6.801290930278334D-01	-7.005917031613871D-06
-5.500887565268203D 01	6.929416294677520D-01	-7.921098205894840D-06

SPLINE JOINTS ARE -
8.8600000 200.00000 7000.0000 47300.000

X	Y	Y*	DEV	R-ERR
8.8600000E+00	5.5560000E+00	8.849409181101148D 00	3.293409186823194D 00	0.592766232940070D 00
1.0980000E+01	7.6798000E+00	1.000445476598735D 01	2.324654792022660D 00	0.302697309813208D 00
1.2900000E+01	1.3116000E+01	1.105323717141965D 01	-2.062762884847134D 00	-0.157270728575634D 00
1.6810000E+01	1.1967000E+01	1.319698352055719D 01	1.229983512927797D 00	0.102781274516891D 00
6.0804999E+01	4.3043000E+01	3.805277908025758D 01	-4.990221140994862D 00	-0.115935718127078D 00
1.2086000E+02	8.6037999E+01	7.415978481454962D 01	-1.187821529226191D 01	-0.138057780021801D 00
1.8088000E+02	1.1649000E+02	1.127575067591070D 02	-3.732493012011147D 00	-0.320413170172961D-01
2.4082000E+02	1.6103000E+02	1.532173868026619D 02	-7.812613883983602D 00	-0.485165115237531D-01
2.4809000E+02	1.5800000E+02	1.581370239574991D 02	1.370239574990819D-01	0.867240237335961D-03
2.5589000E+02	1.5989000E+02	1.634144897988261D 02	3.524490409177631D 00	0.220432198550988D-01
2.6594000E+02	1.6800000E+02	1.702130448960571D 02	2.213044896057056D 00	0.131728862860539D-01
2.7578000E+02	1.7492000E+02	1.768681700603464D 02	1.948169984052413D 00	0.111374913286227D-01
2.7930000E+02	1.8184000E+02	1.792485360450323D 02	-2.591464107555623D 00	-0.142513424185055D-01
3.0085000E+02	2.1051000E+02	1.938177278807263D 02	-1.669227234815548D 01	-0.792944388865442D-01
3.6089000E+02	2.4447000E+02	2.343743280694369D 02	-1.009567124391762D 01	-0.412961560611668D-01
3.7474000E+02	2.3803000E+02	2.437227379386330D 02	5.692737251987523D 00	0.239160493869078D-01
4.0699000E+02	2.6051000E+02	2.654802764674659D 02	4.970278145932753D 00	0.190790302789001D-01
4.1579000E+02	2.7591000E+02	2.714146858813971D 02	-4.495313966015033D 00	-0.162926822822699D-01
4.4553000E+02	2.8706000E+02	2.914622625018356D 02	4.402261128544581D 00	0.153356828101589D-01
4.8048000E+02	3.2067000E+02	3.150060349719130D 02	-5.663963197032302D 00	-0.176629033878256D-01
5.0657000E+02	3.2427000E+02	3.325701857941790D 02	8.300185336415296D 00	0.255965255025076D-01
5.1307000E+02	3.2702000E+02	3.369445920628139D 02	9.924591605050239D 00	0.303485768184140D-01
5.7347000E+02	3.7289000E+02	3.775646128783585D 02	4.674613488710122D 00	0.125361728562353D-01
6.0079000E+02	4.0049000E+02	3.959209841890134D 02	-4.569017489453415D 00	-0.114085681797411D-01
6.2650000E+02	4.0706000E+02	4.131860404706873D 02	6.126039097396301D 00	0.150494744674716D-01
6.4426000E+02	4.1499000E+02	4.251070201735926D 02	1.011701849512582D 01	0.243789451654415D-01
6.5930999E+02	4.5599000E+02	4.620299413305613D 02	6.039939652094461D 00	0.132457721218927D-01
7.2216000E+02	4.8977000E+02	4.773433388812723D 02	-1.242666157649131D 01	-0.253724433200823D-01
9.0080000E+02	5.9851000E+02	5.968104051235945D 02	-1.699597012635934D 00	-0.283971363313838D-02
9.6833999E+02	6.2452999E+02	6.418618806031829D 02	1.733188182388608D 01	0.277518803864712D-01
2.8654000E+03	1.8379000E+03	1.881154699133198D 03	4.325470523671356D 01	0.235348524840083D-01
3.5987000E+03	2.3864000E+03	2.346684455466761D 03	-3.971553842972344D 01	-0.166424482615240D-01
3.9571000E+03	2.5540000E+03	2.571470704179836D 03	1.747070417983650D 01	0.684052630377310D-02
4.2810000E+03	2.8210000E+03	2.773070480931503D 03	-4.792951906849703D 01	-0.169902584432815D-01
7.2000000E+03	4.3490000E+03	4.523541125521543D 03	1.745411255215436D 02	0.401336227918012D-01
8.5035999E+03	5.4260000E+03	5.264705331817194D 03	-1.612946681828052D 02	-0.297262565762634D-01
1.1750000E+04	7.0089999E+03	6.993448649542047D 03	-1.555135045795305D 01	-0.221876879126167D-02
1.2730000E+04	7.5870000F+03	7.482501132021744D 03	-1.044988679782555D 02	-0.137734108314558D-01
1.5640000E+04	8.7880000E+03	8.845021545518303D 03	5.702154551830335D 01	0.648856913043962D-02
1.6940000E+04	9.4274999E+03	9.410355470413911D 03	-1.714452958608854D 01	-0.181856585373519D-02
1.9550000E+04	1.0360000E+04	1.046453644340334D 04	1.045364434033454D 02	0.100903902898982D-01
2.1160000E+04	1.1060000E+04	1.106099913502764D 04	9.991350276395679D-01	0.903377059348615D-04
2.3440000E+04	1.1720000E+04	1.183542541585308D 04	1.154254158530803D 02	0.984858497039934D-02
3.1230000E+04	1.4020000E+04	1.386000895024912D 04	-1.599910497508818D 02	-0.114116297967819D-01
4.7300000E+04	1.4970000E+04	1.499933639310552D 04	2.933639310551916D 01	0.195967889816427D-01

CORRELATION OF FITTED DATA TO ORIGINAL DATA

VARIANCE = 3.160794765434042D 03 CORRELATION INDEX = 0.911037794155017D 00
STANDARD DEVIATION = 5.622094583511352D 01 MAXIMUM CORRELATION = 0.911111111111111D 00

NO REFIT CHECK MADE

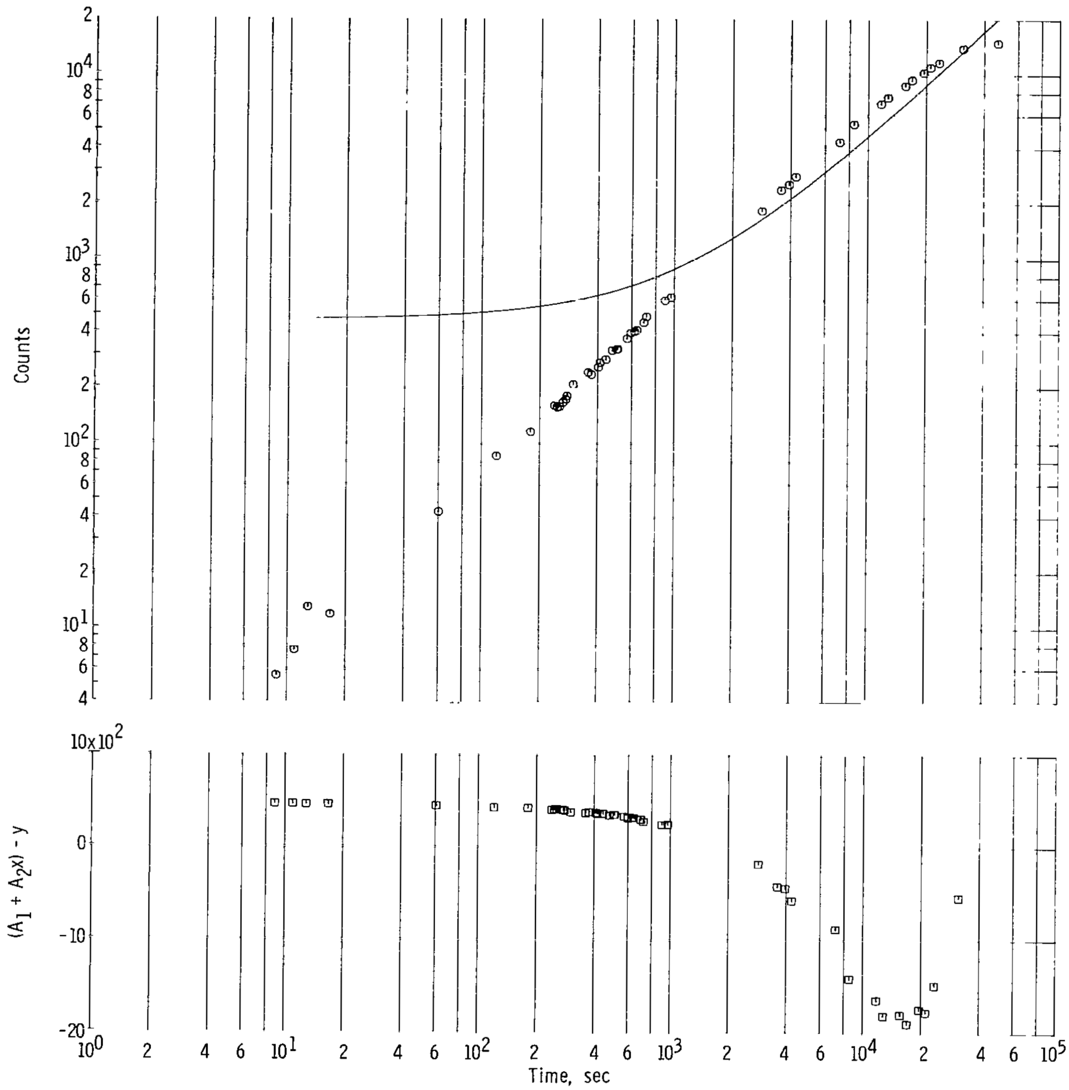


Figure 2. - Linear fit.

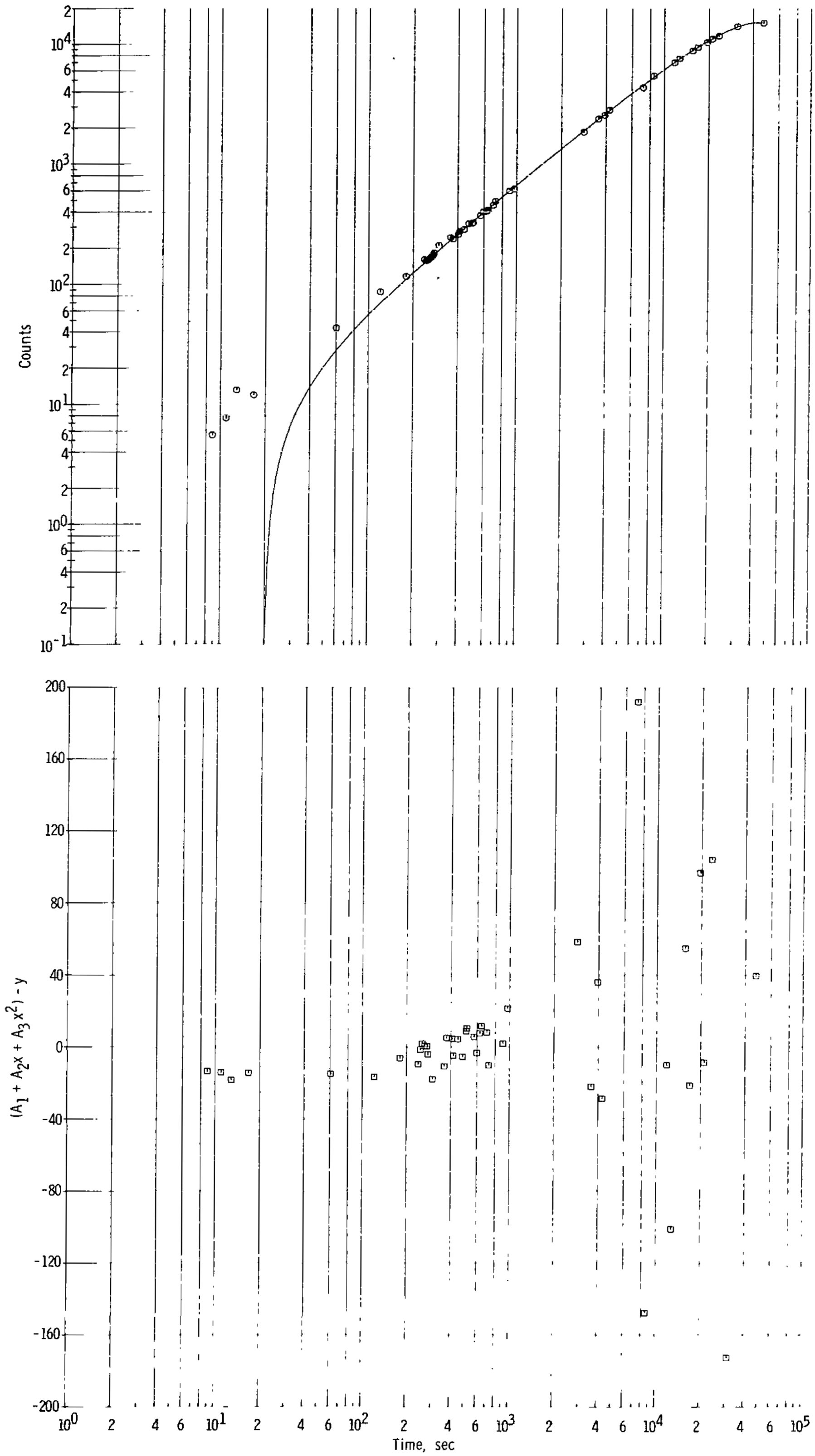


Figure 3. - Parabolic fit.

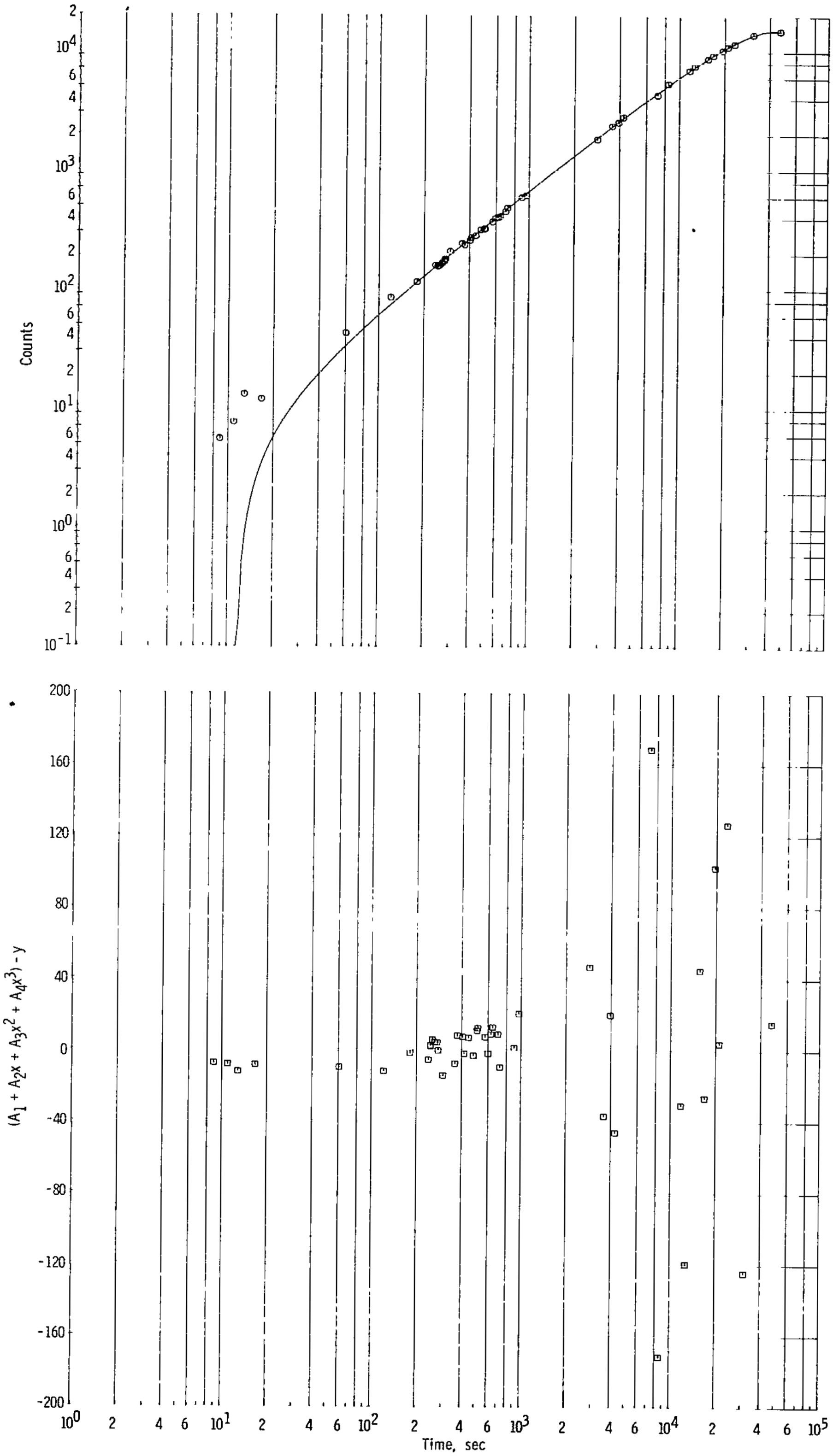


Figure 4. - Cubic fit.

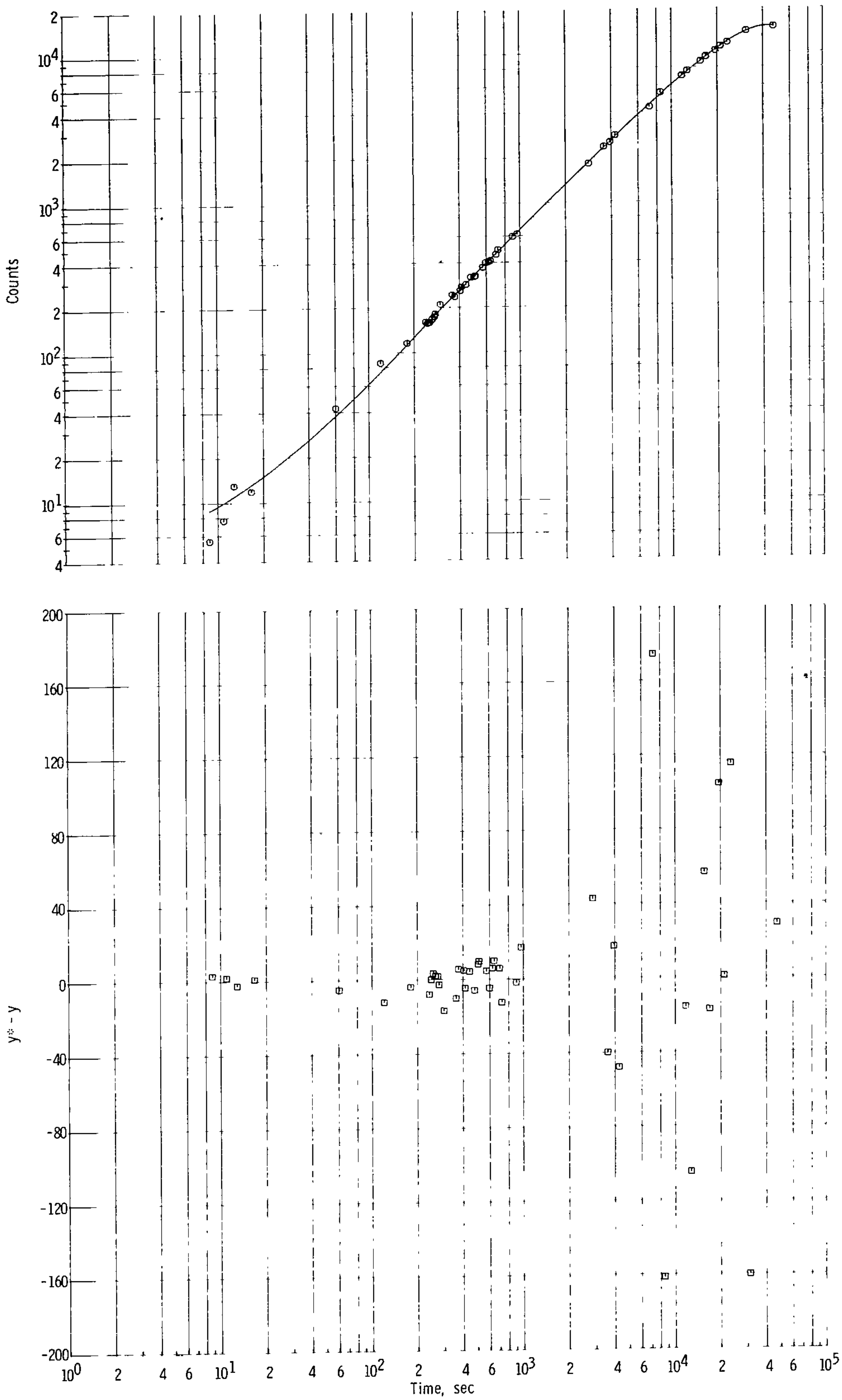


Figure 5. - FITLOS fit.

APPENDIX G

COMPUTER INPUT AND OUTPUT SHEETS FOR SAMPLE PROBLEM 2

INPUT FOR SAMPLE PROBLEM 2

Card column					
1-6	7-12	13-18	19-24	25-30	31-36
	SAMPLE	PROBL	EM 2		
3 51	0 0	0 T	F F	F .0	1
(2E18	.7,F6.	0)			
0	.0		-1	.0	
6	.28318	E-02	-9	.96054	8 E-01
1	.25663	6 E-01	-9	.84250	2 E-01
1	.88495	4 E-01	-9	.64679	5 E-01
2	.51327	2 E-01	-9	.37497	5 E-01
3	.14159	E-01	-9	.02919	6 E-01
3	.76990	8 E-01	-8	.61220	5 E-01
4	.39822	6 E-01	-8	.12732	8 E-01
5	.02654	4 E-01	-7	.57844	6 E-01
5	.65486	2 E-01	-6	.96997	6 E-01
6	.28318	E-01	-6	.30684	2 E-01
6	.91149	8 E-01	-5	.59444	9 E-01
7	.53981	6 E-01	-4	.83864	4 E-01
8	.16813	4 E-01	-4	.04569	1 E-01
8	.79645	1 E-01	-3	.22222	2 E-01
9	.42477	E-01	-2	.37520	5 E-01
1	.00530	9	-1	.51190	2 E-01
1	.06814	1	-6	.39817	5 E-02
1	.13097	2	2	.33339	4 E-02
1	.19380	4	1	.09970	6 E-01
1	.25663	6	1	.95131	4 E-01
1	.31946	8	2	.78013	9 E-01
1	.3823		3	.57815	E-01
1	.44513	1	4	.33736	E-01
1	.50796	3	5	.04987	4 E-01
1	.57079	5	5	.70795	E-01
1	.63362	7	6	.30403	3 E-01
1	.69645	9	6	.83081	8 E-01
1	.75929		7	.2813	E-01
1	.82212	2	7	.64877	6 E-01
1	.88495	4	7	.92698	7 E-01
1	.94778	6	8	.11006	6 E-01
2	.01061	8	8	.19262	6 E-01
2	.07344	9	8	.16979	3 E-01
2	.13628	1	8	.03724	E-01
2	.19911	3	7	.79122	2 E-01
2	.26194	5	7	.42861	2 E-01
2	.32477	7	6	.94692	3 E-01
2	.38760	8	6	.34434	E-01
2	.45044		5	.61973	3 E-01
2	.51327	2	4	.77268	5 E-01
2	.57610	4	3	.80350	2 E-01
2	.63893	6	2	.71322	1 E-01
2	.70176	7	1	.50362	2 E-01
2	.76459	9	1	.77228	9 E-02
2	.82743	1	-1	.26269	3 E-01
2	.89026	3	-2	.81214	E-01
2	.95309	5	-4	.46638	E-01
3	.01592	6	-6	.21996	5 E-01
3	.07875	8	-8	.06675	E-01
3	.14159		-1	.0	
1					
2 51	0 0	0 F	F F	T .0	1

FITLOS OUTPUT FOR SAMPLE PROBLEM 2

SAMPLE PROBLEM 2

DATA DIVIDED AS EVENLY AS POSSIBLE AMONG THE MAXIMUM NUMBER OF SUBSETS

DEGREE OF POLYNOMIAL = 3 NUMBER OF SEGMENTS = 10

EQUATION FITTED IS Y = A0 + A1 X + A2 X**2 + A3 X**3

SEGMENT COEFFICIENTS IN ASCENDING ORDER -

A0	A1	A2	A3
-9.9999926188269760-01	-1.2196927085028620-03	1.0223985051097770 00	-1.1231313538701220-01
-9.9427620283313440-01	-5.5870927477371880-02	1.1963589360821060 00	-2.9689104217686690-01
-9.5637609886762220-01	-2.3683074797736480-01	1.4843656807206570 00	-4.4968349530245180-01
-8.8763751721489830-01	-4.5563263211806770-01	1.7165218839945740 00	-5.3179202490355240-01
-8.8269209746658800-01	-4.6765383146703240-01	1.7260880507528780 00	-5.3432953194715080-01
-1.2012250572443010 00	1.4087177813053130-01	1.3386883027851580 00	-4.5212068222463130-01
-2.2666095048189160 00	1.8364852219820020 00	4.3913665413856510-01	-2.9304489120841030-01
-4.5532967150211330 00	4.9559525400400160 00	-9.7937496006488800-01	-7.8032205812633040-02
-8.5195615738630290 00	9.6903365254402160 00	-2.8631280437111850 00	1.7180851614102720-01
-1.3815877854824070 01	1.5309907317161560 01	-4.8506462275981900 00	4.0612229146063330-01

SPLINE JOINTS ARE -

0	0.3141590	0.6283180	0.9424770	1.2566360	1.5707950	1.8849540
2.1991130	2.5132720	2.8274310	3.1415900			

X	Y	Y*	DEV	R-ERR
0.	-1.0000000E+00	-9.9999926188269760-01	7.3811730239015110-07	-0.7381173023901510-06
6.2831800E-02	-9.9605479E-01	-9.9606749584764920-01	-1.2697483009971310-05	0.1274777555493790-04
1.2566360E-01	-9.8425020E-01	-9.8423036367354150-01	1.9839101459973030-05	-0.2015656324381600-04
1.8849540E-01	-9.6467949E-01	-9.6465502141683190-01	2.4480533908910470-05	-0.2537685714209410-04
2.5132720E-01	-9.3749750E-01	-9.3750862236054730-01	-1.1125755627949820-05	0.1186750435946860-04
3.1415900E-01	-9.0291959E-01	-9.0295832621252830-01	-3.8728288772715840-05	0.4289228948155600-04
3.7699080E-01	-8.6122049E-01	-8.6121706802426390-01	3.4333390135365960-06	-0.3986596937836200-05
4.3982260E-01	-8.1273279E-01	-8.1268092733527220-01	5.1873506059174620-05	-0.6382602745388860-04
5.0265440E-01	-7.5784460E-01	-7.5779176293126580-01	5.2834169945725760-05	-0.6971636420952100-04
5.6548620E-01	-6.9699759E-01	-6.9699144732187040-01	6.1504917358012890-06	-0.8824265327591670-05
6.2831799E-01	-6.3068420E-01	-6.3072183649538820-01	-3.7639546383083640-05	0.5968049709374130-04
6.9114980E-01	-5.5944489E-01	-5.5946268372075970-01	-1.7786843947764640-05	0.3179373705446680-04
7.5398160E-01	-4.8386440E-01	-4.8384536901716050-01	1.9031518661449810-05	-0.3933233906105650-04
8.1681340E-01	-4.0456910E-01	-4.0453913714703600-01	2.9963441524116790-05	-0.7406260507914840-04
8.7964510E-01	-3.2222220E-01	-3.2221339859744750-01	8.8006935433093680-06	-0.2731249914709220-04
9.4247700E-01	-2.3752050E-01	-2.3753701006096250-01	-1.6510906037159320-05	0.6951360449267960-04
1.0053090E+00	-1.5119020E-01	-1.5119973027934770-01	-9.5294587825200950-06	0.6302960595858850-04
1.0681410E+00	-6.3981750E-02	-6.3972792275453490-02	8.9577732435008880-06	-0.1400051301610700-03
1.1309720E+00	2.3333940E-02	2.3350947147213090-02	1.7007190711959550-05	0.7288606529229170-03
1.1938040E+00	1.0997060E-01	1.0998280137472020-01	1.2201030479497280-05	0.1109481119617830-03
1.2566360E+00	1.9513140E-01	1.9512990930271300-01	-1.4912973627101420-06	-0.7642528870925160-05
1.3194680E+00	2.7801390E-01	2.7800174725965900-01	-1.3725196404967830-05	-0.4936874166651480-04
1.3823000E+00	3.5781500E-01	3.5779896578810440-01	-1.6035371801947920-05	-0.4481469963519430-04
1.4451310E+00	4.3373600E-01	4.3372986887726760-01	-6.1313327791578940-06	-0.1413609379020570-04
1.5079630E+00	5.0498739E-01	5.0500005876655920-01	1.2662466720159140-05	0.2507481733789790-04
1.5707950E+00	5.7079500E-01	5.7081304727103830-01	1.8047671581533640-05	0.3161848228207710-04
1.6336270E+00	6.3040330E-01	6.3039403994778470-01	-9.2626621357450030-06	-0.1469323224893780-04
1.6964590E+00	6.8308179E-01	6.8304962532391510-01	-3.2172931528462460-05	-0.4709967621832480-04
1.7592900E+00	7.2812999E-01	7.2810631560778090-01	-2.3682241683475880-05	-0.3252474386911880-04
1.8221220E+00	7.6487760E-01	7.6489265515878380-01	1.5052700783613200-05	0.1967988176832480-04
1.8849540E+00	7.9269870E-01	7.9273497492909770-01	3.6271222833228480-05	0.4575663195062930-04
1.9477860E+00	8.1100659E-01	8.1099977515802130-01	-6.8230165506388830-06	-0.8413022244203990-05
2.0106180E+00	8.1926260E-01	8.1921155793454400-01	-5.1043500640446380-05	-0.6230419959488980-04
2.0734490E+00	8.1697929E-01	8.1693425828156660-01	-4.5038223884574790-05	-0.5512774200915660-04
2.1362810E+00	8.0372399E-01	8.0373165758366840-01	7.6592158819721590-06	0.9529659307830300-05
2.1991130E+00	7.7912220E-01	7.7916757706596410-01	4.5373477478349860-05	0.5823666334930310-04
2.2619450E+00	7.4286119E-01	7.4285894970995780-01	-2.2466887772409900-06	-0.3024372235530080-05
2.3247770E+00	6.9469230E-01	6.9463680949411180-01	-5.5489275839093470-05	-0.7987604862950820-04
2.3876080E+00	6.3443400E-01	6.3438584029427060-01	-4.8159363360333880-05	-0.7590917792287750-04
2.4504400E+00	5.6197330E-01	5.6198796409461330-01	1.4660538171007430-05	0.2608760608062410-04
2.5132720E+00	4.7726849E-01	4.7732791901726210-01	5.9420626349115930-05	0.1245014631165680-03
2.5761040E+00	3.8035020E-01	3.8035174464841680-01	1.5460517008847320-06	0.4064811078287370-05
2.6389360E+00	2.7132210E-01	2.7125271751287630-01	-6.9383841722725490-05	-0.2557249902470930-03
2.7017670E+00	1.5036220E-01	1.5028888842788040-01	-7.3310744497234740-05	-0.4875610020387510-03
2.7645990E+00	1.7722890E-02	1.7712008085193350-02	-1.0881928635519960-05	-0.6140041848157360-03
2.8274310E+00	-1.2626930E-01	-1.2622036312224960-01	4.8936414693834250-05	-0.3875559211407250-03
2.8902630E+00	-2.8121400E-01	-2.8119419091196110-01	1.9807882594591320-05	-0.7043704324642180-04
2.9530950E+00	-4.4663799E-01	-4.4666357614331890-01	-2.5576876932831280-05	0.5726534010729480-04
3.0159260E+00	-6.2199650E-01	-6.2202105627815920-01	-2.4556680132903350-05	0.3948041532190850-04
3.0787580E+00	-8.0667499E-01	-8.0666769234033000-01	7.3096385442283920-06	-0.9061441753242550-05
3.1415900E+00	-1.0000000E+00	-9.9999637883384200-01	3.6211661580409780-06	-0.3621166158040980-05

CORRELATION OF FITTED DATA TO ORIGINAL DATA

VARIANCE = 1.9405527708483540-09 CORRELATION INDEX = 0.4705882322388560 00

STANDARD DEVIATION = 4.4051705572201170-05 MAXIMUM CORRELATION = 0.4705882352941180 00

REFIT CHECK WAS MADE
DUPLICATION OCCURED IN FIRST SET OF COEFFICIENTS - CURVE WAS REFIT IN NEW SEGMENTS

SAMPLE PROBLEM 2

DUPLICATION OCCURED IN FIRST SET OF COEFFICIENTS - CURVE WAS REFIT IN NEW SEGMENTS

DEGREE OF POLYNOMIAL = 3 NUMBER OF SEGMENTS = 6

EQUATION FITTED IS Y = A0 + A1 X + A2 X**2 + A3 X**3

SEGMENT COEFFICIENTS IN ASCENDING ORDER -

A0	A1	A2	A3
-1.000050371376718D 00	-6.664472752618167D-03	1.076414097009774D 00	-2.109179147087161D-01
-9.222209269792074D-01	-3.782730124657974D-01	1.667847959426581D 00	-5.246835903199099D-01
-9.598718869565346D-01	-2.725258497230243D-01	1.568846832342388D 00	-4.937884409919207D-01
-4.228361103683710D 00	4.604316819459200D 00	-8.566972818225622D-01	-9.166595758870244D-02
-4.694865703582764D 00	5.175445079803467D 00	-1.089768171310425D 00	-5.996146798133850D-02
-1.200020630657673D 01	1.348031432926655D 01	-4.236820912919939D 00	3.375540029956028D-01

SPLINE JOINTS ARE -

0	0.6283180	1.0681410	2.0106180	2.4504400	2.6389360	3.1415900

X	Y	Y*	DEV	R-ERR
0.	-1.0000000E+00	-1.000050371376718D 00	-5.037137671770608D-05	0.503713767177061D-04
6.2831800E-02	-9.9605479E-01	-9.962719248977712D-01	-2.171265331319949D-04	0.217986533962269D-03
1.2566360E-01	-9.8425020E-01	-9.843083762260516D-01	-5.817345105008087D-05	0.591043322989076D-04
1.8849540E-01	-9.6467949E-01	-9.644736346767179D-01	2.058672740229417D-04	-0.213404839230692D-03
2.5132720E-01	-9.3749750E-01	-9.370816067586558D-01	4.158898462636174D-04	-0.443617020599770D-03
3.1415900E-01	-9.0291959E-01	-9.024462054522163D-01	4.733924715393245D-04	-0.524290836778690D-03
3.7699080E-01	-8.6122049E-01	-8.608813345468538D-01	3.391668164236528D-04	-0.393821113044528D-03
4.3982260E-01	-8.1273279E-01	-8.127009044980607D-01	3.189634327072088D-05	-0.392457930056498D-04
5.0265440E-01	-7.5784460E-01	-7.582188205085834D-01	-3.742234073719208D-04	0.493799663946067D-03
5.6548620E-01	-6.9699759E-01	-6.977490014245465D-01	-7.514036109402822D-04	0.107805767666537D-02
6.2831799E-01	-6.3068420E-01	-6.316053497106535D-01	-9.211527616483917D-04	0.146056103213709D-02
6.9114980E-01	-5.5944489E-01	-5.601795946239015D-01	-7.346977470896188D-04	0.131326204098238D-02
7.5398160E-01	-4.8386440E-01	-4.841748088571494D-01	-3.104083213274489D-04	0.641519237587450D-03
8.1681340E-01	-4.0456910E-01	-4.043718597686763D-01	1.972408198837594D-04	-0.487533080496797D-03
8.7964510E-01	-3.2222220E-01	-3.215517811142874D-01	6.704181767034356D-04	-0.208060828266521D-02
9.4247700E-01	-2.3752050E-01	-2.364950498264767D-01	1.025449328448669D-03	-0.431730874639080D-02
1.0053090E+00	-1.5119020E-01	-1.499826816181615D-01	1.207519202403695D-03	-0.798675572788475D-02
1.0681410E+00	-6.3981750E-02	-6.279569874234348D-02	1.186051306353519D-03	-0.185373376853682D-01
1.1309720E+00	2.3333940E-02	2.429129064808322D-02	9.573506915820928D-04	0.410282486955386D-01
1.1938040E+00	1.0997060E-01	1.105384892889192D-01	5.678889446785185D-04	0.516400695186584D-02
1.2566360E+00	1.9513140E-01	1.952095995154609D-01	7.819891538518586D-05	0.400750033796229D-03
1.3194680E+00	2.7801390E-01	2.775697294779885D-01	-4.441704443823857D-04	-0.159765552911711D-02
1.3823000E+00	3.5781500E-01	3.568839289163991D-01	-9.310722435073027D-04	-0.260210511154956D-02
1.4451310E+00	4.3373600E-01	4.324161403479121D-01	-1.319859862134676D-03	-0.304300279777446D-02
1.5079630E+00	5.0498739E-01	5.034338628510856D-01	-1.553533448753419D-03	-0.307638063867836D-02
1.5707950E+00	5.7079500E-01	5.692009531930913D-01	-1.594046406365401D-03	-0.279267759437975D-02
1.6336270E+00	6.3040330E-01	6.289825003239178D-01	-1.420802286002609D-03	-0.225379892541865D-02
1.6964590E+00	6.8308179E-01	6.820435931935545D-01	-1.038205061889119D-03	-0.151988395026868D-02
1.7592900E+00	7.2812999E-01	7.276486592164074D-01	-4.813386330569935D-04	-0.661061396287242D-03
1.8221220E+00	7.6487760E-01	7.650642463928135D-01	1.866439348133797D-04	0.244018041858702D-03
1.8849540E+00	7.9269870E-01	7.935546634757814D-01	8.559597695169119D-04	0.107980467927457D-02
1.9477860E+00	8.1100659E-01	8.123849857565544D-01	1.378387581982388D-03	0.169960094663211D-02
2.0106180E+00	8.1926260E-01	8.208203100292639D-01	1.557708594079443D-03	0.190135445136962D-02
2.0734490E+00	8.1697929E-01	8.182256187695032D-01	1.246322264051969D-03	0.152552490544496D-02
2.1362810E+00	8.0372399E-01	8.043646184879609D-01	6.406201201745176D-04	0.797064815129942D-03
2.1991130E+00	7.7912220E-01	7.791008601438314D-01	-2.134344465432036D-05	-0.273942194895956D-04
2.2619450E+00	7.4286119E-01	7.422979123400122D-01	-5.632840587228571D-04	-0.758262864521074D-03
2.3247770E+00	6.9469230E-01	6.938193475712238D-01	-8.729511987271010D-04	-0.125660123233377D-02
2.3876080E+00	6.3443400E-01	6.335298066037369D-01	-9.041930538939980D-04	-0.142519640243420D-02
2.4504400E+00	5.6197330E-01	5.612909191798110D-01	-6.823843766312621D-04	-0.121426475655124D-02
2.5132720E+00	4.7726849E-01	4.769767727884773D-01	-2.917256024357329D-04	-0.611240011480479D-03
2.5761040E+00	3.8035020E-01	3.804866687487092D-01	1.364701519932510D-04	0.358801316515020D-03
2.6389360E+00	2.7132210E-01	2.717331220020811D-01	4.110206474821076D-04	0.151488082036093D-02
2.7017670E+00	1.5036220E-01	1.507260357413340D-01	3.638365689564438D-04	0.241973428799971D-02
2.7645990E+00	1.7722890E-02	1.786852324620725D-02	1.456332323783727D-04	0.821723952835782D-02
2.8274310E+00	-1.2626930E-01	-1.263368485042928D-01	-6.754896734939564D-05	0.534959547547283D-03
2.8902630E+00	-2.8121400E-01	-2.813876939819000D-01	-1.736951873443360D-04	0.617661951712550D-03
2.9530950E+00	-4.4663799E-01	-4.467816276583525D-01	-1.436283919664660D-04	0.321576740452848D-03
3.0159260E+00	-6.2199650E-01	-6.220134467552478D-01	-1.694715722155848D-05	0.272463868084641D-04
3.0787580E+00	-8.0667499E-01	-8.065862580085081D-01	8.874397036606751D-05	-0.110012049646224D-03
3.1415900E+00	-1.0000000E+00	-9.999950087394129D-01	4.991260587061674D-06	-0.499126058706167D-05

CORRELATION OF FITTED DATA TO ORIGINAL DATA

VARIANCE = 8.166388743421532D-07 CORRELATION INDEX = 0.705881863095631D 00

STANDARD DEVIATION = 9.036807387019508D-04 MAXIMUM CORRELATION = 0.705882352941176D 00

NO REFIT CHECK MADE

SAMPLE PROBLEM 2

DATA DIVIDED AS EVENLY AS POSSIBLE AMONG THE MAXIMUM NUMBER OF SUBSETS

DEGREE OF POLYNOMIAL = 2 NUMBER OF SEGMENTS = 10

EQUATION FITTED IS Y = A0 + A1 X + A2 X**2

SEGMENT COEFFICIENTS IN ASCENDING ORDER -

A0	A1	A2
-9.9997303068974430-01	1.5051882719490100-03	9.8019617612206390-01
-1.0185126112251060 00	1.1953192395549190-01	7.9235063250621350-01
-1.1601840887528850 00	5.7048662598936060-01	4.3349197790189460-01
-1.5752865214463010 00	1.4513620881729570 00	-3.3827362316472430-02
-2.3800351341269560 00	2.7321603729542400 00	-5.4344124063300110-01
-3.5423488166634340 00	4.2120653673700870 00	-1.0145100289810220 00
-4.8097794992208950 00	5.5568522204412150 00	-1.3712261210203000 00
-5.6720548603625500 00	6.3410550845583200 00	-1.5495259310300750 00
-5.4018257222269310 00	6.1260133863543160 00	-1.5067447081228240 00
-3.2217167986673300 00	4.5839004725567070 00	-1.2340390404242500 00

SPLINE JOINTS ARE -

0	0.3141590	0.6283180	0.9424770	1.2566360	1.5707950	1.8849540
2.1991130	2.5132720	2.8274310	3.1415900			

X	Y	Y*	DEV	R-ERR
0.	-1.0000000E+00	-9.9997303068974430-01	2.6969310255697820-05	-0.2696931025569780-04
6.2831800E-02	-9.9605479E-01	-9.9600880408714610-01	4.5994277493133230-05	-0.4617645291067160-04
1.2566360E-01	-9.8425020E-01	-9.8430527165758000-01	-5.5068882578501730-05	0.5595008507312490-04
1.8849540E-01	-9.6467949E-01	-9.6486243409214330-01	-1.8293214140258090-04	0.1896299662558000-03
2.5132720E-01	-9.3749750E-01	-9.3768028931754380-01	-1.8279271262444660-04	0.1949794141172830-03
3.1415900E-01	-9.0291959E-01	-9.0275884170699390-01	1.6075621676170380-04	-0.1780404557962410-03
3.7699080E-01	-8.6122049E-01	-8.6083966966767030-01	3.8083169560709780-04	-0.4421999882774000-03
4.3982260E-01	-8.1273279E-01	-8.1266435827911880-01	6.8442562212611690-05	-0.8421287062828120-04
5.0265440E-01	-7.5784460E-01	-7.5823290412888980-01	-3.8830702767833620-04	0.5123834479570450-03
5.6548620E-01	-6.9699759E-01	-6.9754531745507570-01	-5.4771964146949070-04	0.7858271580671410-03
6.2831799E-01	-6.3068420E-01	-6.3060159217413290-01	8.2604774872208080-05	-0.1309764463289500-03
6.9114980E-01	-5.5944489E-01	-5.5881843421142880-01	6.2646266538313180-04	-0.1119793332427300-02
7.5398160E-01	-4.8386440E-01	-4.8361257523299060-01	2.5182530283129890-04	-0.5204460227957100-03
8.1681340E-01	-4.0456910E-01	-4.0498399740250380-01	-4.1489681394368820-04	0.1025527686963990-02
8.7964510E-01	-3.2222220E-01	-3.2293284849068940-01	-7.1064919969854310-04	0.2205463190500950-02
9.4247700E-01	-2.3752050E-01	-2.3745872166868900-01	6.1777486236369580-05	-0.2600932822900250-03
1.0053090E+00	-1.5119020E-01	-1.5040664579161330-01	7.8355502895188780-04	-0.5182578134688920-02
1.0681410E+00	-6.3981750E-02	-6.3621661292708960-02	3.6008875598803060-04	-0.5627991665028920-02
1.1309720E+00	2.3333940E-02	2.2894859211768810-02	-4.3908074473231200-04	-0.1881725698921150-01
1.1938040E+00	1.0997060E-01	1.0914566519816440-01	-8.2493514607628880-04	-0.7501415319130720-02
1.2566360E+00	1.9513140E-01	1.9512937980631270-01	-2.0207937629912460-06	-0.1035606651096040-04
1.3194680E+00	2.7801390E-01	2.7883413814854350-01	8.2023822617260930-04	0.2950349699787100-02
1.3823000E+00	3.5781500E-01	3.5824801631203030-01	4.3301515212390560-04	0.1210164891690480-02
1.4451310E+00	4.3373600E-01	4.3336987506217420-01	-3.6612514787259800-04	-0.8441198048935140-03
1.5079630E+00	5.0498739E-01	5.0420210175542880-01	-7.8529454441023110-04	-0.1555077513150360-02
1.5707950E+00	5.7079500E-01	5.7074346862346000-01	-5.1530975996749720-05	-0.9027930523727520-04
1.6336270E+00	6.3040330E-01	6.3113426208474490-01	7.3095947482437750-04	0.1159510858839330-02
1.6964590E+00	6.8308179E-01	6.8351476848392110-01	4.3297022847754720-04	0.6338482881308380-03
1.7592900E+00	7.2812999E-01	7.2788434645779220-01	-2.4565139167220810-04	-0.3373729861394810-03
1.8221220E+00	7.6487760E-01	7.6424440606196730-01	-6.3319639603287570-04	-0.8278401589980470-03
1.8849540E+00	7.9269870E-01	7.9259418440189890-01	-1.0451930436561340-04	-0.1318524981521140-03
1.9477860E+00	8.1100659E-01	8.1152540295856170-01	5.1880478398969030-04	0.6397047633859270-03
2.0106180E+00	8.1926260E-01	8.1962980402624730-01	3.6720259106282070-04	0.4482110991268920-03
2.0734490E+00	8.1697929E-01	8.1690751890833550-01	-7.1777597115740830-05	-0.8785730240994170-04
2.1362810E+00	8.0372399E-01	8.0335845080569520-01	-3.6554756209117300-04	-0.4548172791076690-03
2.1991130E+00	7.7912220E-01	7.7898256917293910-01	-1.3963441554665220-04	-0.1792201722701820-03
2.2619450E+00	7.4286119E-01	7.4307596720976580-01	2.1477081103071780-04	0.2891129757105230-03
2.3247770E+00	6.9469230E-01	6.9493474275128090-01	2.4244398132999830-04	0.3489947733108300-03
2.3876080E+00	6.3443400E-01	6.3455996786358780-01	1.2596820595689450-04	0.1985521047498600-03
2.4504400E+00	5.6197330E-01	5.6194969521163700-01	-2.3608344805303720-05	-0.4200972654020850-04
2.5132720E+00	4.7726849E-01	4.7710479981030750-01	-1.6369858060549800-04	-0.3429905412936320-03
2.5761040E+00	3.8035020E-01	3.8019417600890650-01	-1.5602258780944570-04	-0.4102077201092140-03
2.6389360E+00	2.7132210E-01	2.7138666351090900-01	6.4562156310010720-05	0.2379539152456750-03
2.7017670E+00	1.5036220E-01	1.5068440810625190-01	3.2220893387435720-04	0.2142885217480570-02
2.7645990E+00	1.7722890E-02	1.8083467862419770-02	3.6057784859089280-04	0.2034531886783360-01
2.8274310E+00	-1.2626930E-01	-1.2641430657717790-01	-1.4500704023445720-04	0.1148395063299070-02
2.8902630E+00	-2.8121400E-01	-2.8173231168454380-01	-5.1831288998815950-04	0.1843126203567200-02
2.9530950E+00	-4.4663799E-01	-4.4679394315868910-01	-1.5594389230311200-04	0.3491505258380470-03
3.0159260E+00	-6.2199650E-01	-6.2159638886704680-01	4.0011073097945270-04	-0.6432684608965320-03
3.0787580E+00	-8.0667499E-01	-8.0614512107911460-01	5.2988089975958270-04	-0.6568703610000540-03
3.1415900E+00	-1.0000000E+00	-1.0004374799161070 00	-4.3747991610665340-04	0.4374799161066530-03

CORRELATION OF FITTED DATA TO ORIGINAL DATA

VARIANCE = 2.4801717257076730-07 CORRELATION INDEX = 0.6470587006306660 00

STANDARD DEVIATION = 4.9801322165876630-04 MAXIMUM CORRELATION = 0.6470588235294120 00

NO REFIT CHECK MADE

TABLE I. - COMPARISON OF FITLOS CURVE WITH f(x)

x	y	y*	y*-y
0	-1.0000000	-1.0000504	-0.5036592E-04
0.6283180E-01	-0.9960548	-0.9962719	-0.2171248E-03
0.1256636	-0.9842502	-0.9843084	-0.5816668E-04
0.1884954	-0.9646795	-0.9644736	0.2058744E-03
0.2513272	-0.9374975	-0.9370816	0.4158914E-03
0.3141590	-0.9029196	-0.9024462	0.4733950E-03
0.3769908	-0.8612205	-0.8608813	0.3391728E-03
0.4398226	-0.8127328	-0.8127009	0.3190339E-04
0.5026544	-0.7578446	-0.7582188	-0.3742203E-03
0.5654862	-0.6969976	-0.6977490	-0.7513985E-03
0.6283180	-0.6306842	-0.6316053	-0.9211525E-03
0.6911498	-0.5594449	-0.5601796	-0.7346943E-03
0.7539816	-0.4838644	-0.4841748	-0.3104061E-03
0.8168134	-0.4045691	-0.4043719	0.1972429E-03
0.8796451	-0.3222222	-0.3215518	0.6704219E-03
0.9424770	-0.2375205	-0.2364950	0.1025449E-02
1.0053090	-0.1511902	-0.1499827	0.1207519E-02
1.0681410	-0.6398175E-01	-0.6279570E-01	0.1186051E-02
1.1309720	0.2333394E-01	0.2429129E-01	0.9573505E-03
1.1938040	0.1099706	0.1105385	0.5678888E-03
1.2566360	0.1951314	0.1952096	0.7819757E-04
1.3194680	0.2780139	0.2775697	-0.4441738E-03
1.3823000	0.3578150	0.3568839	-0.9310730E-03
1.4451310	0.4337360	0.4324161	-0.1319863E-02
1.5079630	0.5049874	0.5034339	-0.1553535E-02
1.5707950	0.5707950	0.5692009	-0.1594052E-02
1.6336270	0.6304033	0.6289825	-0.1420803E-02
1.6964590	0.6830818	0.6820436	-0.1038209E-02
1.7592900	0.7281300	0.7276487	-0.4813448E-03
1.8221220	0.7648776	0.7650642	0.1866370E-03
1.8849540	0.7926987	0.7935547	0.8559525E-03
1.9477860	0.8110066	0.8123850	0.1378387E-02
2.0106180	0.8192626	0.8208203	0.1557715E-02
2.0734490	0.8169793	0.8182256	0.1246318E-02
2.1362810	0.8037240	0.8043646	0.6406158E-03
2.1991130	0.7791222	0.7791009	-0.2134591E-04
2.2619450	0.7428612	0.7422979	-0.5632862E-03
2.3247770	0.6946923	0.6938193	-0.8729547E-03
2.3876080	0.6344340	0.6335298	-0.9041950E-03
2.4504400	0.5619733	0.5612927	-0.6805807E-03
2.5132720	0.4772685	0.4769768	-0.2917275E-03
2.5761040	0.3803502	0.3804867	0.1364686E-03
2.6389360	0.2713221	0.2717314	0.4093461E-03
2.7017670	0.1503622	0.1507260	0.3638361E-03
2.7645990	0.1772289E-01	0.1786852E-01	0.1456330E-03
2.8274310	-0.1262693	-0.1263368	-0.6754883E-04
2.8902630	-0.2812140	-0.2813877	-0.1736917E-03
2.9530950	-0.4466380	-0.4467816	-0.1436248E-03
3.0159260	-0.6219965	-0.6220134	-0.1694262E-04
3.0787580	-0.8066750	-0.8065863	0.8875132E-04
3.1415900	-1.0000000	-0	1.0000000

TABLE II. - COMPARISON OF $y^{*'}$ WITH $f'(x)$

x	dy/dx	dy*/dx	dy*/dx - dy/dx
0	0	-0.6664473E-02	-0.6664473E-02
0.6283180E-01	0.1254983	0.1261036	0.6053094E-03
0.1256636	0.2500058	0.2538756	0.3869805E-02
0.1884954	0.3725378	0.3766517	0.4113883E-02
0.2513272	0.4921210	0.4944317	0.2310704E-02
0.3141590	0.6077997	0.6072157	-0.5840361E-03
0.3769908	0.7186415	0.7150037	-0.3637798E-02
0.4398226	0.8237424	0.8177956	-0.5946755E-02
0.5026544	0.9222328	0.9155916	-0.6641194E-02
0.5654862	1.0132823	1.0083916	-0.4890755E-02
0.6283180	1.0961049	1.0961955	0.9052455E-04
0.6911498	1.1699639	1.1752873	0.5323440E-02
0.7539816	1.2341759	1.2419509	0.7775053E-02
0.8168134	1.2881158	1.2961864	0.8070603E-02
0.8796451	1.3312202	1.3379936	0.6773457E-02
0.9424770	1.3629912	1.3673728	0.4381567E-02
1.0053090	1.3829996	1.3843237	0.1324087E-02
1.0681410	1.3908878	1.3888464	-0.2041385E-02
1.1309720	1.3863723	1.3813069	-0.5065411E-02
1.1938040	1.3692459	1.3620709	-0.7174999E-02
1.2566360	1.3393793	1.3311386	-0.8240774E-02
1.3194680	1.2967224	1.2885097	-0.8212730E-02
1.3823000	1.2413053	1.2341844	-0.7120892E-02
1.4451310	1.1732398	1.1681638	-0.5075917E-02
1.5079630	1.0927146	1.0904458	-0.2268776E-02
1.5707950	1.0000021	1.0010314	0.1029342E-02
1.6336270	0.8954524	0.8999205	0.4468091E-02
1.6964590	0.7794939	0.7871132	0.7619314E-02
1.7592900	0.6526328	0.6626115	0.9978689E-02
1.8221220	0.5154434	0.5264115	0.1096810E-01
1.8849540	0.3685770	0.3785150	0.9937912E-02
1.9477860	0.2127518	0.2189220	0.6170245E-02
2.0106180	0.4875046E-01	0.4763266E-01	-0.1117799E-02
2.0734490	-0.1225800	-0.1305877	-0.8007713E-02
2.1362810	-0.3003440	-0.3109823	-0.1063830E-01
2.1991130	-0.4835848	-0.4935481	-0.9963315E-02
2.2619450	-0.6713007	-0.6782853	-0.6984554E-02
2.3247770	-0.8624470	-0.8651937	-0.2746679E-02
2.3876080	-1.0559385	-1.0542704	0.1668110E-02
2.4504400	-1.2506672	-1.2455219	0.5145237E-02
2.5132720	-1.4454896	-1.4385689	0.6920710E-02
2.5761040	-1.6392453	-1.6330362	0.6209150E-02
2.6389360	-1.8307597	-1.8289229	0.1836717E-02
2.7017670	-2.0188470	-2.0215197	-0.2672642E-02
2.7645990	-2.2023296	-2.2061239	-0.3794283E-02
2.8274310	-2.3800253	-2.3827324	-0.2707154E-02
2.8902630	-2.5507663	-2.5513453	-0.5789101E-03
2.9530950	-2.7134030	-2.7119624	0.1440585E-02
3.0159260	-2.8668072	-2.8645815	0.2225727E-02
3.0787580	-3.0098889	-3.0092074	0.6815195E-03
3.1415900	-3.1415873	-0	3.1415873

TABLE III. - COMPARISON OF $\int_{x_o}^{x_f} y^* \, dx$ WITH $\int_{x_o}^{x_f} f(x) \, dx$

x	$\int y \, dx$	$\int y^* \, dx$	$\int y \, dx - \int y^* \, dx$
0	-0	0.4712424E-09	0.4712424E-09
0.6283180E-01	-0.6274915E-01	-0.6275994E-01	-0.1079123E-04
0.1256636	-0.1250032	-0.1250237	-0.2050772E-04
0.1884954	-0.1862709	-0.1862868	-0.1594424E-04
0.2513272	-0.2460688	-0.2460646	0.4187226E-05
0.3141590	-0.3039252	-0.3038922	0.3308058E-04
0.3769908	-0.3593838	-0.3593241	0.5961955E-04
0.4398226	-0.4120071	-0.4119351	0.7204339E-04
0.5026544	-0.4613806	-0.4613191	0.6152689E-04
0.5654862	-0.5071157	-0.5070902	0.2558529E-04
0.6283180	-0.5488533	-0.5488819	-0.2858788E-04
0.6911498	-0.5862666	-0.5863489	-0.8233637E-04
0.7539816	-0.6190642	-0.6191802	-0.1159683E-03
0.8168134	-0.6469929	-0.6471125	-0.1196191E-03
0.8796451	-0.6698399	-0.6699318	-0.9193271E-04
0.9424770	-0.6874352	-0.6874730	-0.3786385E-04
1.0053090	-0.6996535	-0.6996202	0.3328919E-04
1.0681410	-0.7064159	-0.7063063	0.1095608E-03
1.1309720	-0.7076914	-0.7075135	0.1778826E-03
1.1938040	-0.7034979	-0.7032713	0.2265275E-03
1.2566360	-0.6939030	-0.6936558	0.2471805E-03
1.3194680	-0.6790246	-0.6787889	0.2356768E-03
1.3823000	-0.6590311	-0.6588390	0.1920834E-03
1.4451310	-0.6341418	-0.6340211	0.1206994E-03
1.5079630	-0.6046244	-0.6045948	0.2952665E-04
1.5707950	-0.5707971	-0.5708675	-0.7043779E-04
1.6336270	-0.5330258	-0.5331921	-0.1662821E-03
1.6964590	-0.4917232	-0.4919678	-0.2445988E-03
1.7592900	-0.4473476	-0.4476407	-0.2930835E-03
1.8221220	-0.4003982	-0.4007008	-0.3026240E-03
1.8849540	-0.3514170	-0.3516866	-0.2695322E-03
1.9477860	-0.3009838	-0.3011819	-0.1981072E-03
2.0106180	-0.2497133	-0.2498167	-0.1034811E-03
2.0734490	-0.1982536	-0.1982667	-0.1311488E-04
2.1362810	-0.1472790	-0.1472320	0.4703179E-04
2.1991130	-0.9749204E-01	-0.9742580E-01	0.6624311E-04
2.2619450	-0.4961565E-01	-0.4956876E-01	0.4689349E-04
2.3247770	-0.4390597E-02	-0.4390209E-02	0.3881287E-06
2.3876080	0.3742823E-01	0.3737132E-01	-0.5691033E-04
2.4504400	0.7507864E-01	0.7497072E-01	-0.1079114E-03
2.5132720	0.1077915	0.1076525	-0.1390344E-03
2.5761040	0.1347983	0.1346546	-0.1437012E-03
2.6389360	0.1553342	0.1552091	-0.1250263E-03
2.7017670	0.1686435	0.1685442	-0.9925663E-04
2.7645990	0.1739844	0.1739015	-0.8292124E-04
2.8274310	0.1706328	0.1705520	-0.8086115E-04
2.8902630	0.1578875	0.1577984	-0.8911267E-04
2.9530950	0.1350748	0.1349750	-0.9971485E-04
3.0159260	0.1015535	0.1014485	-0.1050094E-03
3.0787580	0.5671749E-01	0.5661521E-01	-0.1022718E-03
3.1415900	0.2652407E-05	-0.9539165E-04	-0.9804405E-04

01 EXIT IN SAM2

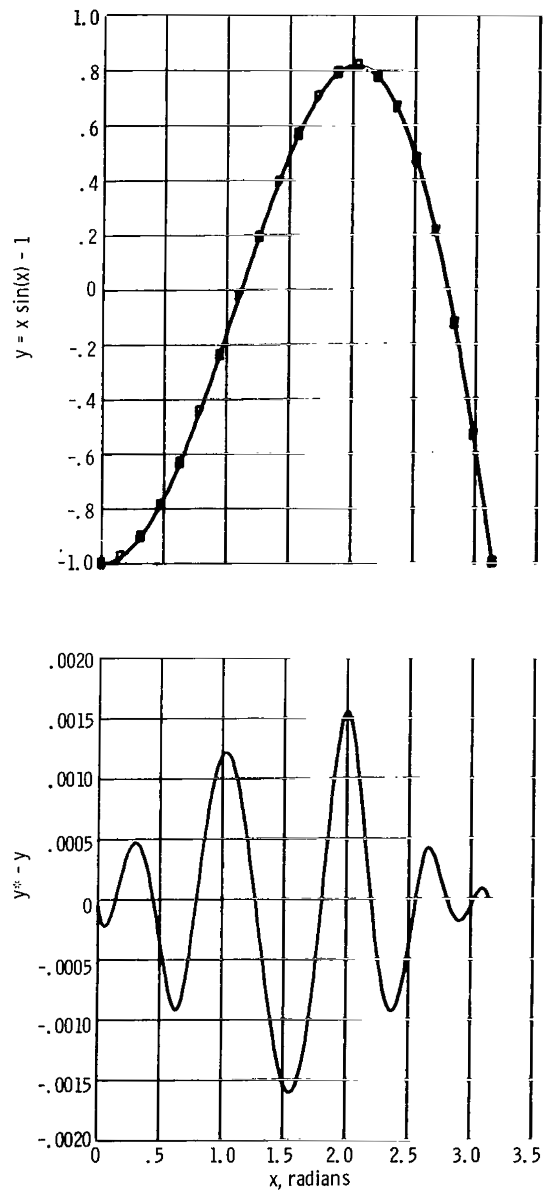


Figure 6. - Comparison of FITLOS curve with f(x).

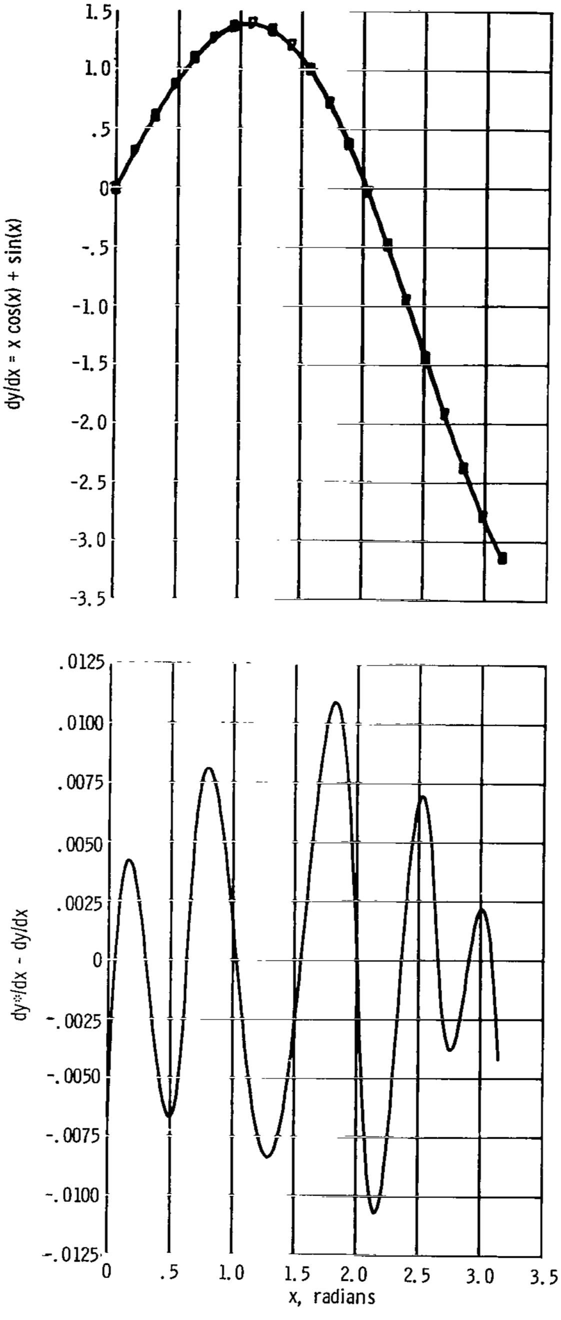


Figure 7. - Comparison of y*' with f'(x).

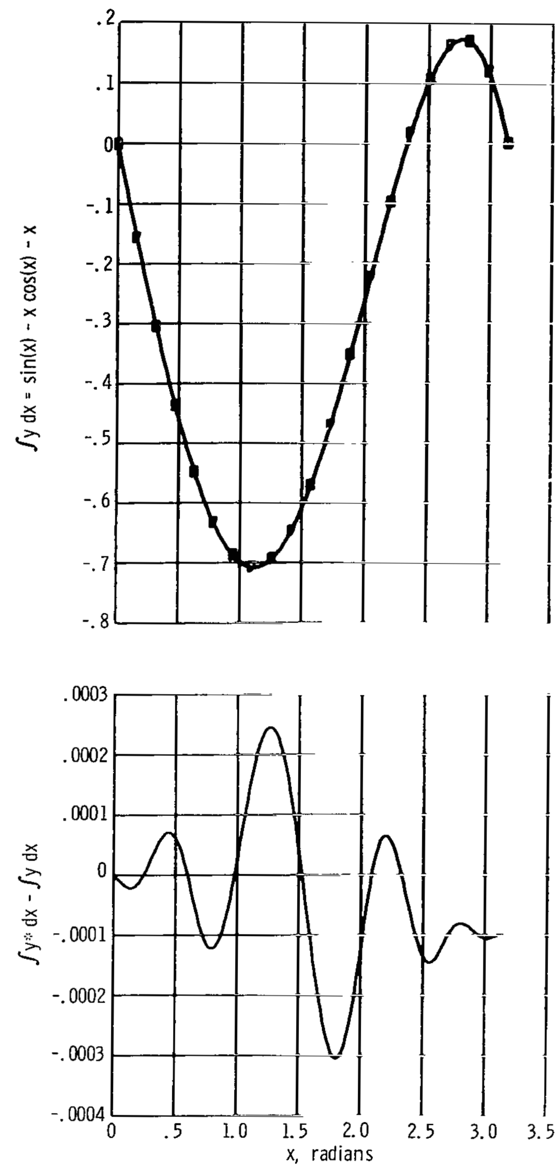


Figure 8. - Comparison of $\int_{x_0}^{x_f} y^* \, dx$ with $\int_{x_0}^{x_f} f(x)dx$.

REFERENCES

1. Greville, T. N. E.: Numerical Procedures for Interpolation by Spline Functions. J. SIAM Numerical Anal., Ser. B, vol. 1, 1964, pp. 53-68.
2. Nielsen, Kaj L.: Methods in Numerical Analysis. Macmillan Co., 1956.
3. Kenney, John F.; and Keeping, E. S.: Mathematics of Statistics. Part I. Third ed., D. Van Nostrand Co., Inc., 1954.
4. Mirsky, Leonid: An Introduction to Linear Algebra. Clarendon Press, Oxford, 1955.
5. Bellman, Richard E.: Introduction to Matrix Analysis. McGraw-Hill Book Co., Inc., 1960.
6. Scarborough, James B.: Numerical Mathematical Analysis. Sixth ed., Johns Hopkins Press, 1966.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D. C. 20546

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

FIRST CLASS MAIL

POSTAGE AND FEES PAID
NATIONAL AERONAUTICS AN
SPACE ADMINISTRATION

008 001 C1 U 08 710730 S00903DS
DEPT OF THE AIR FORCE
WEAPONS LABORATORY /WLOL/
ATTN: E LOU BOWMAN, CHIEF TECH LIBRARY
KIRTLAND AFB NM 87117

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Retur

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546