

AD-A081 809

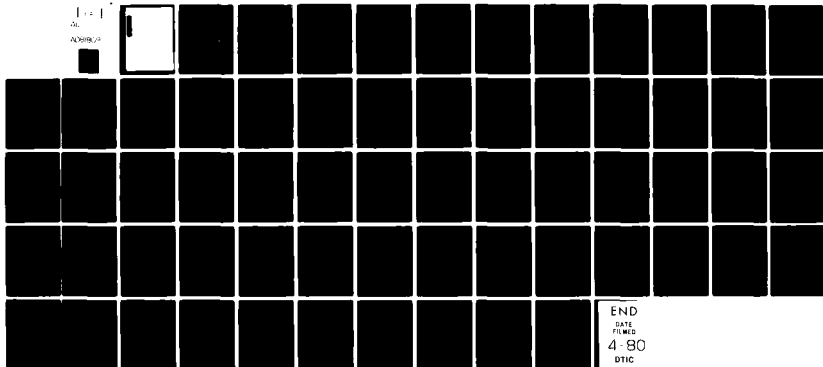
NAVAL SURFACE WEAPONS CENTER DAHLGREN LAB VA
POLYNOMIAL ARITHMETIC AND CONTOUR CONSTRUCTION, (U)
OCT 77 A V HERSHEY
NSWC/DL-TR-3688

F/G 12/1

UNCLASSIFIED

NL

1-1
AL
N/10/10/10



END
DATE
FILMED
4-80
DTIC

2
14

NSWC/DL-TR-3688

11

Oct 1977

12/63

6

POLYNOMIAL ARITHMETIC
AND
CONTOUR CONSTRUCTION,

By

10

A. V. HERSHEY

Science and Mathematics Research Group

DTIC
SELECTED
MAR 12 1969

63

Approved for public release; distribution unlimited.

291715

TABLE OF CONTENTS

	Page
TABLE OF CONTENTS	i
FOREWORD	ii
ABSTRACT	iii
INTRODUCTION	1
POLYNOMIAL ARITHMETIC	4
MULTIPLEX POLYNOMIAL EVALUATION	6
LAGRANGE POLYNOMIAL EXPANSION	7
LAGRANGE POLYNOMIAL EVALUATION	9
ORTHONORMAL POLYNOMIAL SYNTHESIS	10
ORTHONORMAL POLYNOMIAL EXPANSION	13
ORTHONORMAL POLYNOMIAL EVALUATION	14
ISOMETRIC REPRESENTATION	15
LAGRANGE INTEGRATION	17
ORTHONORMAL INTEGRATION	18
TRAPEZOIDAL INTEGRATION	18
CHEBYSHEV INTEGRATION	20
HIGH-ACCURACY QUADRATURE	21
Christoffel-Darboux Identity	21
Chebyshev Quadrature	23
Gauss Quadrature	25
COMPLEX POWER POLYNOMIAL EVALUATION	27
COMPLEX FOURIER SERIES EXPANSION	28
COMPLEX FOURIER SERIES EVALUATION	28
FAST FOURIER TRANSFORM	29
ROOTS OF POLYNOMIALS	32
COMPLEX ROOTS OF A QUADRATIC EQUATION	33
COMPLEX ROOTS OF A CUBIC EQUATION	34
COMPLEX ROOTS OF A QUARTIC EQUATION	35
COMPLEX ROOTS OF AN ANALYTIC FUNCTION	37
CONTOURS AMONG DATA	41
CONTOURS OF FUNCTIONS	43
DISCUSSION	51
CONCLUSION	53
BIBLIOGRAPHY	54
APPENDICES	
A. DISTRIBUTION	

Accession For	
Math. Sci.	☒
Eng. 143	☐
Un. sponsored	☐
Classification	
By	
Distributor/	
Availability of	
Dist	Available/or special
A	

FOREWORD

Investigations of polynomial operations have been made in this laboratory for a number of years. Subroutines have been prepared on the basis of several methods of computation. It is the purpose of this report to provide analysis and documentation of the best subroutines. The manuscript was completed by 27 October 1977.

Released by:



Ralph A. Niemann

Head, Strategic Systems Department

ABSTRACT

Power polynomials and trigonometric polynomials are used for the approximation of general functions. Analysis and documentation are given for a set of subroutines which perform polynomial arithmetic, find roots of a function, or construct contours of constant function.

INTRODUCTION

There was a library of subroutines on the Naval Ordnance Research Calculator. The subroutines were written in machine language. The library no longer was useful after NORC was dismantled. Some of the subroutines have been transcribed into FORTRAN, and are described herewith. In the meantime, the NORC library has been eclipsed by giant library projects. There are the System/360 Scientific Subroutine Package of the International Business Machines Corporation¹, and the Subroutine Library of the International Mathematical and Statistical Libraries, Inc.² Nevertheless, the subroutines from the NORC library present capabilities which are not exactly covered by the giant libraries.

Power polynomials and trigonometric polynomials are used extensively in numerical analysis because they can be evaluated with efficiency on digital computers. They are used often to approximate other functions which are more expensive to compute. They give continuous interpolations of a function which is specified only at discrete values of its argument. The function is illustrated graphically as a set of ordinates at a set of abscissae. The function can be represented as a set of coefficients of the powers of its argument or of the trigonometric functions of its argument. Power polynomial expansions are available through Lagrange polynomial approximation or orthonormal polynomial approximation for arbitrary arrangements of the abscissae. The power polynomial expansions can be differentiated, evaluated, or integrated.

Lagrange polynomials are convenient because the polynomial approximation of an arbitrary function is given directly by the products of the polynomials and discrete values of the function. In the construction of the polynomials, binomials are multiplied in succession to form a polynomial which has all the abscissae for roots, then this polynomial is divided in succession by binomials to obtain the Lagrange polynomials. The binomials may be symbolic insofar as they are represented by linear functions of their argument. There is a loss of accuracy to rounding error in the derivation of polynomials of high order. The binomials may be numeric insofar as they are represented by their values for a specific value of their argument. There is no significant loss of accuracy to rounding error in the evaluation of polynomials of any order.

The orthonormal polynomials are convenient because they provide a least squares approximation for a function when the number of discrete values of the function is greater than the order of the approximation. The polynomials may be orthonormal with respect to integration between two limits of integration, or they may be orthonormal with respect to summation over a discrete set of arguments. In the construction of the polynomials which are orthonormal with respect to summation, a matrix of values of the polynomials is synthesized by orthogonalization and the coefficients of a three-term recurrence are derived by summation. The coefficients of recurrence may be used in the expansion of the polynomials into representations as coefficients of the powers of the argument, or the coefficients of recurrence may be used in the evaluation of the polynomials for specific values of the argument.

Many high-order polynomials have such large coefficients that they can be computed accurately only at small values of their arguments. Otherwise the rounding error in the largest term of their expansion exceeds the values of the polynomials. Expansion of polynomials into representations by coefficients is feasible only for polynomials of low degree.

A high-order polynomial may be represented as a set of discrete values with an algorithm for interpolation, or as a recurrence formulation with a set of recurrence coefficients. The interpolation and the recurrence can be differentiated or evaluated but not integrated.

High-accuracy algorithms for integration to any order are available for special spacing between abscissae. Of especial importance are equal spacing, Chebyshev spacing, and Gauss spacing.

Equal spacing has the disadvantage that Lagrange polynomials of high degree have large amplitudes of oscillation near the ends of a set of abscissae. Small errors in the values of a function near the center of the set of abscissae cause violent oscillations near the ends.

In Chebyshev spacing the abscissae are proportional to the cosines of equally spaced angles. The orthonormal polynomials for Chebyshev spacing are nearly equal to the Chebyshev polynomials. The amplitude of oscillation of the polynomials is nearly uniform.

In Gauss spacing the abscissae are proportional to the roots of a Legendre polynomial. The orthonormal polynomials for Gauss spacing are nearly equal to the Legendre polynomials. The order of accuracy of the integration of a function is nearly twice the number of values of the function.

In the discrete Fourier transform the orthogonality of trigonometric functions with respect to summation is used to obtain the coefficients of a Fourier series. If the order of the Fourier series is a power of 2, then there is available the fast Fourier transform⁶. Two versions of the FFT are known as the Cooley-Tukey⁷ algorithm and the Sande-Tukey⁸ algorithm.

Various tactics for finding roots have been proposed in the literature. It is beyond the scope of the present investigation to provide a literature survey. Libraries of subroutines have been assembled by various organizations. The presence of an algorithm in such libraries is evidence that the algorithm is worthy of consideration.

If a function is irregular it may have roots anywhere. A set of values of the function must be computed at a closely spaced set of abscissae. The values are sorted in order to determine where the function changes sign.

If a function is represented by a smooth curve of ordinates, then the presence of roots can be predicted from the behavior of the function between roots. Only a few computations of ordinates, slopes, and curvatures are required in order to locate and determine the roots.

In the method of false position or regula falsi^{3,4}, two abscissae which straddle a root must first be found. The values of the function are opposite in sign at the two abscissae. Linear interpolation is used to obtain a new abscissa. The interpolation is repeated between the nearest abscissae for which the function has opposite signs. The curvature of the curve of ordinates tends to make the sign of the function at each new abscissa uniformly the same and opposite to the sign of the function at a fixed abscissa. Although the abscissae improve gradually in accuracy, the rate of convergence is geometric, because each new abscissa is obtained with a slope which does not continue to improve in accuracy.

In the Pegasus¹⁸ method, the location of the new abscissa overshoots the regula falsi by an amount which gives improved convergence.

In the Lehmer-Schur¹⁹ method, the field is partitioned into areas by an arrangement of circles with various centers and radii. The coefficients of polynomials with complex argument are combined with the coefficients of polynomials with inverse argument so as to eliminate the terms of highest degree. When the polynomials have been reduced

to a constant, then the sign of the constant indicates whether a root is located within the circle of inversion. Repetition with the various circles identifies the areas within which the roots are located. The disadvantage of the method is that the time for reduction of polynomials for each circle is proportional to the square of the order, and the method is too expensive.

In the Newton-Raphson^{3,4} method, a value of the function and its derivative at an iterant is used. At the iterant a tangent to the curve of ordinates is constructed. The point of intersection of the tangent with the axis of abscissae is the new iterant.

In the secant method, the values of the function at two successive iterants are used. A chord is constructed between the ordinates at the two iterants. The point of intersection of the chord with the axis of abscissae is the new iterant.

The advantage of the tangent is that it does not require as many iterations. The advantage of the chord is that it does not require the differentiation of the function. The rates of convergence are quadratic, because the accuracy of the abscissae and the accuracy of the slope both improve together.

In the Muller method²⁰, the values of the function at three successive iterants are used. The three values of the function are fitted with a quadratic polynomial. The nearer root of the quadratic polynomial is the next iterant. Convergence is complete if the increment from the iterant to the Newton-Raphson intercept is less than a tolerance. The three consecutive iterants which define the quadratic polynomial must be distinct. Otherwise the roots of the quadratic polynomial may be out of bounds. In that case, the iterant is brought in to a circle with radius equal to the geometric mean of the distances to adjacent roots.

In the method of Jarrett and Nudds²¹, the quadratic polynomial is replaced by a linear rational function.

In the method of Jenkins and Traub²², the function is a polynomial. A set of auxiliary polynomials is constructed by a recurrence relation. A fraction of the function is subtracted from each auxiliary polynomial to give a difference which is zero at an iterant. The difference is divided by a binomial to give auxiliary polynomials of one lower degree than the function. At all roots of the function except the nearest root, the values of the auxiliary polynomials approach zero by a powering process. Newton-Raphson iteration is applied to the quotients between the function and the auxiliary polynomials. Insofar as all roots of the auxiliary polynomials except one approach the roots of the function, the Newton-Raphson iteration is applied to a progressively more nearly linear function.

In the present system, it is assumed that the best tactic is to go directly in that direction in which a root is most likely to lie. There are two stages in the search for roots. In the hunting stage, the function is approximated locally at an iterant by a quadratic polynomial. Steps of predetermined length are taken away from the iterant in the direction of the nearer root of the quadratic polynomial. Convergence of the Newton-Raphson iteration is tested with the aid of a criterion. The hunting stage terminates when a region of convergence has been reached. In the homing stage, the intercept of the tangent with the complex plane is computed at each iterant. The intercept becomes the new iterant as long as the displacement of the intercept is less than the displacement of the iterant. Termination of the homing stage occurs if the displacement of the intercept is equal to or greater than the displacement of the iterant. Termination may occur because rounding error has upset the convergence or because the iterant has moved out of the region of convergence. Whether termination is caused by rounding error or by nonconvergence is tested by comparison between the displacements and a tolerance. If the displacements are less than the tolerance,

the iterant is accepted as a root. If the displacements are more than the tolerance, the hunting stage is resumed with double the tolerance.

After each root of a polynomial has been determined, the root of the polynomial may be eliminated by a synthetic division to obtain a reduced polynomial of one lower degree. Experience has shown that with each reduction of the degree of the polynomial the remaining roots are disturbed and the accuracy deteriorates. Reduction of degree is not feasible for transcendental functions. In the present system the function is replaced by the quotient between the function and that polynomial whose roots are equal to all roots already determined.

Contours are used for the illustration of information which may range from irregular data to smooth functions. The contours for irregular data are based upon local interpolations within one element of a grid, whereas contours for smooth functions can be based upon global formulations for the entire field.

An equation between two coordinates defines implicitly a relationship which is represented in a graph of the coordinates by a contour line. The equation is solved by an iteration which starts with trial values of the coordinates and follows a hunting procedure and a homing procedure. During hunting the trial point is displaced by steps of predetermined length. Before the contour is reached the steps are perpendicular to the contour while after the contour is reached the steps are parallel to the contour. After a contour is reached it is followed until it goes out of bounds or closes on itself. If it does not close on itself, then the trial point returns to the initial point and the other half of the contour is followed until it also goes out of bounds.

Two programs on NORC were designed to trace section lines on a ship or contour lines in a LORAN map. These programs were prepared when the available cathode-ray printers were dot plotters. The rasters were coarse and the simulation of curves required care in the arrangement of dots. Later the rasters were finer and the cathode-ray printers were vector plotters. Now the rasters are still finer and the latest cathode-ray printers again are dot plotters. The dots are arranged so as to simulate vectors. Any simulation of a smooth curve by a vector plotter necessarily is a polygonization. The simulation of a smooth curve by a polygonization requires an optimization of the lengths of the sides of the polygonization.

POLYNOMIAL ARITHMETIC

Analysis

Let the polynomials $A(z)$, $B(z)$, $C(z)$ be defined by the equations

$$A(z) = \sum_{k=0}^{L-1} a_k z^k \quad (1)$$

$$B(z) = \sum_{k=0}^{M-1} b_k z^k \quad (2)$$

$$C(z) = \sum_{k=0}^{N-1} c_k z^k \quad (3)$$

where a_k, b_k, c_k are real coefficients and L, M, N are the numbers of coefficients respectively. The argument z may be real or complex. The coefficients of each polynomial are stored in an array in ascending order. If the degree of the polynomial is less than

the size of its array, then the upper end of the array is filled with zeros. The dimensions of an array are given by a two-integer specification. The first integer is the interval between elements of the array, and the second integer is the number of elements in the array.

In the transfer of a polynomial, the coefficients of the new polynomial are related to the coefficients of the old polynomial in accordance with the equation

$$b_k = a_k \quad (4)$$

Transfer continues until the second array has been filled with coefficients from the first array, or with zeros if the first array is shorter than the second array.

In the addition of two polynomials, the coefficients of the new polynomial are related to the coefficients of the old polynomials in accordance with the equation

$$c_k = a_k + b_k \quad (5)$$

Addition continues until the new array is filled.

In the subtraction of two polynomials, the coefficients of the new polynomial are related to the coefficients of the old polynomials in accordance with the equation

$$c_k = a_k - b_k \quad (6)$$

Subtraction continues until the new array is filled.

In the multiplication of two polynomials, the coefficients of the new polynomial are related to the coefficients of the old polynomials in accordance with the equation

$$c_m = \sum_{k=0}^m a_{m-k} b_k \quad (7)$$

The indices $m-k$ and k are coordinates in a rectangle with dimensions equal to the lengths of the old polynomials. In the summation the indices advance diagonally across the rectangle. Initialization of the summation is at the left or bottom of the rectangle, and termination of the summation is at the top or right of the rectangle.

In the division of polynomials, fractions of the denominator are subtracted from the numerator until the numerator has been eliminated. The fractions which are applied to the denominator are the coefficients in the quotient. Elimination is accomplished with the aid of the substitutions

$$c_m \rightarrow \frac{a_m}{b_0} \quad (8)$$

$$a_{m+k} \rightarrow a_{m+k} - c_m b_k \quad (9)$$

where k is limited to the range $k < M$ and to the range $k + m < N$.

The quotient between two polynomials is a finite polynomial if all roots of the denominator coincide with roots of the numerator. Otherwise the quotient is an infinite series. The circle of convergence of the infinite series goes through that nearest root of the denominator which does not correspond to a root of the numerator.

Polynomial arithmetic with real coefficients can be adapted to polynomial arithmetic with complex coefficients. Let complex polynomials be given by the expressions

$$A + iB \quad C + iD \quad (10)$$

Then complex transfer is expressed by the substitutions

$$C \rightarrow A \quad D \rightarrow B \quad (11)$$

Complex addition is expressed by the equation

$$(A + iB) + (C + iD) = (A + C) + (B + D)i \quad (12)$$

Complex subtraction is expressed by the equation

$$(A + iB) - (C + iD) = (A - C) + (B - D)i \quad (13)$$

Complex multiplication is expressed by the equation

$$(A + iB)(C + iD) = (AC - BD) + (AD + BC)i \quad (14)$$

Complex division is expressed by the equation

$$\frac{A + iB}{C + iD} = \frac{(AC + BD) + (BC - AD)i}{CC + DD} \quad (15)$$

The polynomials A, B, C, D have real coefficients but complex arguments.

Programming

SUBROUTINE PLNM (MO, PA, NA, PB, NB, PC, NC)

 FORTRAN SUBROUTINE FOR POLYNOMIAL ARITHMETIC

The mode of operation is given in argument MO. The polynomials A, B, C are given or stored in the arrays PA, PB, PC. The specifications of the polynomials are given in the arrays NA, NB, NC. The specification for a polynomial PX consists of the array NX as given in the following table:

Address	Specification
NX(1)	Interval between coefficients
NX(2)	Number of coefficients

There are no compatibility requirements on the lengths of the polynomials. The repertory of operations and call lines is given in the following table:

Operation	Call Line
$B = A$	CALL PLNM (1, PA, NA, PB, NB)
$C = A + B$	CALL PLNM (2, PA, NA, PB, NB, PC, NC)
$C = A - B$	CALL PLNM (3, PA, NA, PB, NB, PC, NC)
$C = AB$	CALL PLNM (4, PA, NA, PB, NB, PC, NC)
$C = A / B$	CALL PLNM (5, PA, NA, PB, NB, PC, NC)

During polynomial division, leading zeros are shifted off to the left as far as possible in the numerator and the denominator.

MULTIPLEX POLYNOMIAL EVALUATION

Analysis

Let $\varphi(u)$ be a polynomial of $(n - 1)$ th degree in the argument u . Let the polynomial

be expressed in terms of its argument by the equation

$$\varphi(u) = \sum_{k=0}^{n-1} a_k u^k \quad (16)$$

where the coefficient a_k is the k th element of an array. The first derivative of the polynomial is given by the equation

$$\frac{d\varphi(u)}{du} = \sum_{k=0}^{n-1} k a_k u^{k-1} \quad (17)$$

and the second derivative of the polynomial is given by the equation

$$\frac{d^2\varphi(u)}{du^2} = \sum_{k=0}^{n-1} k(k-1) a_k u^{k-2} \quad (18)$$

Integration of the polynomial is expressed by the equation

$$\int_0^u \varphi(u) du = \sum_{k=0}^{n-1} \frac{a_k}{k+1} u^{k+1} \quad (19)$$

Whether the operation is integration, evaluation, first differentiation, or second differentiation is determined by a mode-of-operation parameter.

Programming

SUBROUTINE MPLNMV (MO, AU, NC, AC, FV)

FORTRAN SUBROUTINE FOR MULTIPLEX POLYNOMIAL EVALUATION

The mode of operation is integration if MO = -1, evaluation if MO = 0, first differentiation if MO = +1, and second differentiation if MO = +2. The argument u is given in the address AU. The number of coefficients n is given in the address NC. The coefficients $a_0, \dots, a_{n-1}$ are given in the n -array AC. The value of the function is stored in address FV.

LAGRANGE POLYNOMIAL EXPANSION

Analysis

Let $u_1, \dots, u_n$ be discrete values of the argument u , and let $\varphi_1(u), \dots, \varphi_n(u)$ be Lagrange polynomials. The m th polynomial $\varphi_m(u)$ is defined by the equation

$$\varphi_m(u) = \frac{(u - u_1) \cdots (u - u_{m-1})(u - u_{m+1}) \cdots (u - u_n)}{(u_m - u_1) \cdots (u_m - u_{m-1})(u_m - u_{m+1}) \cdots (u_m - u_n)} \quad (20)$$

The Lagrange polynomials have the property which is expressed by the equation

$$\varphi_m(u_k) = \delta_{mk} \quad (21)$$

where δ_{mk} is zero if $k \neq m$ but is unity if $k = m$. The Lagrange polynomials are expressed in terms of powers of the argument by the equation

$$\varphi_m(u) = \sum_{k=0}^{n-1} a_{mk} u^k \quad (22)$$

where the coefficient a_{mk} is the mk th element of a matrix of coefficients.

Let the partial polynomial $F_m(u)$ be defined by the equation

$$F_m(u) = (u - u_1) \cdots (u - u_m) \quad (23)$$

and be expressed in terms of powers of the argument by the equation

$$F_m(u) = \sum_{k=0}^m c_{mk} u^k \quad (24)$$

The coefficients of the partial polynomials are generated with the aid of the recurrence equations

$$c_{mk} = c_{m-1,k-1} - u_m c_{m-1,k} \quad (25)$$

and

$$c_{mm} = 1 \quad (26)$$

The coefficients of the partial polynomials are stored temporarily in the matrix of coefficients.

Cycling of the recurrence leads to the complete polynomial $F(u)$ which is defined by the equation

$$F(u) = (u - u_1) \cdots (u - u_n) \quad (27)$$

and is expressed in terms of powers of the argument by the equation

$$F(u) = \sum_{k=0}^n c_k u^k \quad (28)$$

The coefficients of the complete polynomial are stored in the $(n+1)$ th row of the matrix of coefficients.

The Lagrange polynomial $\varphi_m(u)$ is recovered from the complete polynomial $F(u)$ through division by the binomial $u - u_m$. The algorithm for division is just the nested method for polynomial evaluation. The coefficients of the Lagrange polynomials are given initially by the recurrence equation

$$a_{mk} = c_{k+1} + u_m a_{m,k+1} \quad (29)$$

and then they are normalized through division by the product

$$(u_m - u_1) \cdots (u_m - u_{m-1})(u_m - u_{m+1}) \cdots (u_m - u_n) \quad (30)$$

All coefficients are stored in the computer in ascending order.

Inasmuch as the m th Lagrange polynomial is unity at the m th argument and zero for other discrete arguments, an arbitrary function $f(u)$ is expressed by the equation

$$f(u) = \sum_{m=1}^n f(u_m) \varphi_m(u) \quad (31)$$

The polynomial approximation of an arbitrary function is derived from a set of values of the function at the discrete arguments through vector-matrix multiplication.

Programming

SUBROUTINE LGRNGX (AU, NA, AC)

 FORTRAN SUBROUTINE FOR LAGRANGIAN POLYNOMIAL EXPANSION

The arguments $u_1, \dots, u_n$ are given in the n -array AU. The order n is given in the argument NA. The coefficients of the polynomials $\varphi_1(u), \dots, \varphi_n(u)$ are stored in the $(n+1) \times n$ array AC. The coefficients of the polynomial $F(u)$ are stored in the $(n+1)$ th row of the array AC.

LAGRANGE POLYNOMIAL EVALUATION

Analysis

If $u \neq u_k$ for any k , then the m th Lagrange polynomial $\varphi_m(u)$ is evaluated by the equation

$$\varphi_m(u) = \frac{(u - u_1) \cdots (u - u_{m-1})(u - u_{m+1}) \cdots (u - u_n)}{(u_m - u_1) \cdots (u_m - u_{m-1})(u_m - u_{m+1}) \cdots (u_m - u_n)} \quad (32)$$

The normalization divisor of the m th polynomial is given by the product

$$(u_m - u_1) \cdots (u_m - u_{m-1})(u_m - u_{m+1}) \cdots (u_m - u_n) \quad (33)$$

which is computed initially from the discrete data. The function $F(u)$ in the equation

$$F(u) = (u - u_1) \cdots (u - u_n) \quad (34)$$

is computed from the given argument u , is divided by each difference $u - u_m$ and by the normalization divisor to give the values of the successive polynomials $\varphi_m(u)$.

The first derivative of the polynomial $\varphi_m(u)$ is given by the equation

$$\frac{d}{du} \varphi_m(u) = + \left[\frac{1}{u - u_1} + \cdots + \frac{1}{u - u_{m-1}} + \frac{1}{u - u_{m+1}} + \cdots + \frac{1}{u - u_n} \right] \varphi_m(u) \quad (35)$$

The sums of the reciprocals of the first $m-1$ differences $u - u_{m-1}$ are accumulated and are stored in an array.

The second derivative of the polynomial $\varphi_m(u)$ is given by the equation

$$\begin{aligned} \frac{d^2}{du^2} \varphi_m(u) = & - \left[\frac{1}{(u - u_1)^2} + \cdots + \frac{1}{(u - u_{m-1})^2} + \frac{1}{(u - u_{m+1})^2} + \cdots + \frac{1}{(u - u_n)^2} \right] \varphi_m(u) \\ & + \left[\frac{1}{u - u_1} + \cdots + \frac{1}{u - u_{m-1}} + \frac{1}{u - u_{m+1}} + \cdots + \frac{1}{u - u_n} \right]^2 \varphi_m(u) \end{aligned} \quad (36)$$

Polynomial expansion and cancellation of squares of reciprocals of differences reduce the second derivative to the summation in the equation

$$\frac{d^2}{du^2} \varphi_m(u) = \sum_{i=1}^n \sum_{j=1}^n \frac{1}{(u - u_i)(u - u_j)} \varphi_m(u) \quad (i \neq j) \quad (37)$$

where the summation does not include any terms for which $i = m$, $j = m$, or $i = j$.

The terms in the summation may be arranged in the form of a matrix such that

the i, j th term occupies the i, j th position in the matrix. The m th row, the m th column, and the diagonal of the matrix contain zeros. There is a matrix for each value of m . The sum of elements in the upper left quadrant is twice the sum of products of the reciprocal of the difference $u - u_{m-1}$ and the sum of the first $m - 2$ reciprocals of the differences $u - u_{m-2}$. The sums of the elements in the upper left quadrant are computed and are stored in an array.

The computation of the second derivative is completed with further accumulations of the products of the reciprocals of differences. The sum of elements in the lower right quadrant is twice the sum of products of the reciprocal of the difference $u - u_{m+1}$ and the sum of the last $n - m - 1$ reciprocals of the differences $u - u_{m+1}$. The sum of elements in the lower left quadrant and in the upper right quadrant is twice the product of the sum of the first $m - 1$ reciprocals of differences and the sum of the last $n - m$ reciprocals of differences.

The computation of the first derivative is completed with further accumulations of the reciprocals of differences. The sum of the first $m - 1$ reciprocals of differences is added to the sum of the last $n - m$ reciprocals of differences to complete the summation.

If $u \equiv u_k$ for any k , then the reciprocal of the difference $u - u_k$ is omitted throughout the computation, and the values of the polynomials $\varphi_m(u)$ are replaced by zero when $m \neq k$ and by unity when $m = k$.

Programming

SUBROUTINE LGRNGN (AU, NA, AN)

 FORTRAN SUBROUTINE FOR LAGRANGIAN NORMALIZATION DIVISORS

The arguments $u_1, \dots, u_n$ are given in the n -array AU. The number of arguments is given in the argument NA. The normalization divisors are stored in the n -array AN.

SUBROUTINE LGRNGV (MO, NA, QU, AU, AN, FF, DF, SF)

 FORTRAN SUBROUTINE FOR LAGRANGIAN POLYNOMIAL EVALUATION

The mode of operation MO is 0 for functions, 1 for first derivatives, and 2 for second derivatives. The number of discrete arguments n is given in argument NA. The variable argument u is given in argument QU. The discrete arguments $u_1, \dots, u_n$ are given in the n -array AU, and the normalization divisors are given in the n -array AN. The functions are stored in the n -array FF, the first derivatives are stored in the n -array DF, and the second derivatives are stored in the n -array SF.

ORTHONORMAL POLYNOMIAL SYNTHESIS

Analysis

Let $u_1, \dots, u_n$ be the discrete values of the argument u at n stations, and let $\varphi_0(u), \dots, \varphi_{n-1}(u)$ be n polynomials of progressively increasing degree. The polynomials

are orthonormal if they satisfy the condition

$$\sum_{i=1}^n \varphi_k(u_i) \varphi_m(u_i) = \delta_{km} \quad (38)$$

where δ_{km} is zero if $k \neq m$ and is unity if $k = m$. Any polynomial of degree m can be expanded in terms of the orthonormal polynomials of degrees 0 through m . The coefficients in the expansion are given by the solution of a sequence of equations which start at the highest degree and compare terms of the same degree. Let $\varphi_m(u)$ be expressed by the equation

$$\varphi_m(u) = \gamma_m u \varphi_{m-1}(u) + \sum_{\nu=0}^{m-1} \gamma_\nu \varphi_\nu(u) \quad (39)$$

where the γ_ν are constant coefficients. Multiplication throughout by $\varphi_k(u)$ and summation isolates γ_k in accordance with the equation

$$\sum_{i=1}^n \varphi_k(u_i) \varphi_m(u_i) = \gamma_m \sum_{i=1}^n u_i \varphi_k(u_i) \varphi_{m-1}(u_i) + \gamma_k = 0 \quad (40)$$

The polynomial $u \varphi_k(u)$ may be expressed by the equation

$$u \varphi_k(u) = \sum_{\nu=0}^{k+1} \epsilon_\nu \varphi_\nu(u) \quad (41)$$

where the ϵ_ν are constant coefficients. All terms of the summation are orthogonal to $\varphi_{m-1}(u)$ and the coefficient γ_k therefore is zero when $k \leq m-3$. The orthonormal polynomials can be generated by a three-term recurrence equation.

The polynomial of zero degree is given by the equation

$$\varphi_0(u) = \frac{1}{\sqrt{n}} \quad (42)$$

and the polynomial of first degree is given by the equation

$$\varphi_1(u) = \frac{1}{\alpha_0} (u - \beta_0) \varphi_0(u) \quad (43)$$

where the constants α_0 and β_0 are defined by the equations

$$\alpha_0 = \frac{1}{\sqrt{n}} \sum_{i=1}^n u_i \varphi_1(u_i) \quad (44)$$

and

$$\beta_0 = \frac{1}{n} \sum_{i=1}^n u_i \quad (45)$$

The subsequent polynomials are generated by the three-term recurrence equation

$$\varphi_m(u) = \frac{1}{\alpha_{m-1}} \left\{ u \varphi_{m-1}(u) - \beta_{m-1} \varphi_{m-1}(u) - \alpha_{m-2} \varphi_{m-2}(u) \right\} \quad (46)$$

where the constants α_{m-1} and β_{m-1} are defined by the equations

$$\alpha_{m-1} = \sum_{i=1}^n u_i \varphi_{m-1}(u_i) \varphi_m(u_i) \quad (47)$$

and

$$\beta_{m-1} = \sum_{i=1}^n u_i \varphi_{m-1}(u_i) \varphi_{m-1}(u_i) \quad (48)$$

The coefficients α_{m-1} , β_{m-1} are stored in a row array and the values $\varphi_m(u_i)$ are stored in a matrix array.

Let a function $f(u)$ be expanded as a series in the orthonormal polynomials as expressed by the equation

$$f(u) = \sum_{m=0}^{n-1} c_m \varphi_m(u) \quad (49)$$

The square of the standard deviation σ^2 is given by the equation

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n \left\{ f(u_i) - \sum_{k=0}^{n-1} c_k \varphi_k(u_i) \right\}^2 \quad (50)$$

Differentiation with respect to a coefficient gives the equation

$$-\frac{\partial \sigma^2}{\partial c_m} = \frac{2}{n} \sum_{i=1}^n \left\{ f(u_i) - \sum_{k=0}^{n-1} c_k \varphi_k(u_i) \right\} \varphi_m(u_i) \quad (51)$$

For least squares the coefficients are given by the equation

$$c_m = \sum_{i=1}^n f(u_i) \varphi_m(u_i) \quad (52)$$

The square of the standard deviation is given by the equation

$$\sigma^2 = \frac{1}{n} \left\{ \sum_{i=1}^n f^2(u_i) - \sum_{m=0}^{n-1} c_m^2 \right\} \quad (53)$$

For a complete expansion in terms of all n orthonormal polynomials the function is equal to its series representation at every discrete argument and the standard deviation is zero.

Programming

SUBROUTINE ORTHOS (AU, MA, AA, NA, AR)

 FORTRAN SUBROUTINE FOR ORTHONORMAL POLYNOMIAL SYNTHESIS

The arguments $u_1, \dots, u_n$ are given in the $1 \times n$ array AU. The number of polynomials m is given in argument MA. The values of the polynomials $\varphi_0(u), \dots, \varphi_{m-1}(u)$ are stored in the $m \times n$ array AA. The number of arguments n is given in argument NA. The recurrence coefficients $\alpha_0, \beta_0, \dots, \alpha_{m-2}, \beta_{m-2}$ are stored in the $2m-2$ array AR.

ORTHONORMAL POLYNOMIAL EXPANSION

Analysis

Let the m th orthonormal polynomial $\varphi_m(u)$ be expressed by the equation

$$\varphi_m(u) = \sum_{k=0}^{m-1} a_{mk} u^k \quad (54)$$

The polynomial of zero degree is given by the equation

$$\varphi_0(u) = \frac{1}{\sqrt{n}} \quad (55)$$

for which all coefficients are zero except

$$a_{00} = \frac{1}{\sqrt{n}} \quad (56)$$

The polynomial of first degree is given by the equation

$$\varphi_1(u) = \frac{1}{\alpha_0} (u - \beta_0) \varphi_0(u) \quad (57)$$

for which all coefficients are zero except

$$a_{10} = -\frac{\beta_0}{\sqrt{n}\alpha_0} \quad a_{11} = +\frac{1}{\sqrt{n}\alpha_0} \quad (58)$$

Thereafter the polynomials of progressively increasing degree are given by the equation

$$\varphi_m(u) = \frac{1}{\alpha_{m-1}} \left\{ u\varphi_{m-1}(u) - \beta_{m-1}\varphi_{m-1}(u) - \alpha_{m-2}\varphi_{m-2}(u) \right\} \quad (59)$$

for which the coefficients are generated with the aid of the recurrence equation

$$a_{mk} = \frac{1}{\alpha_{m-1}} \left\{ a_{m-1,k-1} - \beta_{m-1}a_{m-1,k} - \alpha_{m-2}a_{m-2,k} \right\} \quad (60)$$

The constants α_{m-1} , β_{m-1} are given in a row array and the coefficients a_{mk} are stored in a matrix array.

If a function $f(u)$ is represented by a series of the orthonormal polynomials, then the function $f(u)$ is converted into a series of the powers of the argument u by a vector-matrix multiplication.

Programming

SUBROUTINE ORTHOX (NA, AR, NC, AC)

 FORTRAN SUBROUTINE FOR ORTHONORMAL POLYNOMIAL EXPANSION

The number of arguments n is given in the argument NA. The recurrence coefficients $\alpha_0, \beta_0, \dots, \alpha_{m-2}, \beta_{m-2}$ are given in the $2m-2$ array AR. The number of coefficients m is given in the argument NC. The coefficients of the polynomials $\varphi_0(u), \dots, \varphi_{m-1}(u)$ are stored in the $m \times m$ array AC.

ORTHONORMAL POLYNOMIAL EVALUATION

Analysis

The polynomial of zero degree is given by the equation

$$\varphi_0(u) = \frac{1}{\sqrt{n}} \quad (61)$$

and the polynomial of first degree is given by the equation

$$\varphi_1(u) = \frac{1}{\alpha_0}(u - \beta_0)\varphi_0(u) \quad (62)$$

The polynomials of progressively higher degree are generated by the three-term recurrence equation

$$\varphi_m(u) = \frac{1}{\alpha_{m-1}} \left\{ u\varphi_{m-1}(u) - \beta_{m-1}\varphi_{m-1}(u) - \alpha_{m-2}\varphi_{m-2}(u) \right\} \quad (63)$$

where constants α_{m-1} and β_{m-1} are given in a $2m-2$ array.

The first derivative of the polynomial of zero degree is given by the equation

$$\frac{d}{du} \varphi_0(u) = 0 \quad (64)$$

and the first derivative of the polynomial of first degree is given by the equation

$$\frac{d}{du} \varphi_1(u) = \frac{1}{\alpha_0} \varphi_0(u) \quad (65)$$

The first derivatives of the polynomials of progressively higher degree are generated by the three-term recurrence equation

$$\frac{d}{du} \varphi_m(u) = \frac{1}{\alpha_{m-1}} \left\{ \varphi_{m-1}(u) + u\varphi'_{m-1}(u) - \beta_{m-1}\varphi'_{m-1}(u) - \alpha_{m-2}\varphi'_{m-2}(u) \right\} \quad (66)$$

where $\varphi'_m(u)$ is the first derivative of $\varphi_m(u)$.

The second derivative of the polynomial of zero degree is given by the equation

$$\frac{d^2}{du^2} \varphi_0(u) = 0 \quad (67)$$

and the second derivative of the polynomial of first degree is given by the equation

$$\frac{d^2}{du^2} \varphi_1(u) = 0 \quad (68)$$

The second derivatives of the polynomials of progressively higher degree are generated by the three-term recurrence equation

$$\frac{d^2}{du^2} \varphi_m(u) = \frac{1}{\alpha_{m-1}} \left\{ 2\varphi'_{m-1}(u) + u\varphi''_{m-1}(u) - \beta_{m-1}\varphi''_{m-1}(u) - \alpha_{m-2}\varphi''_{m-2}(u) \right\} \quad (69)$$

where $\varphi''_m(u)$ is the second derivative of $\varphi_m(u)$.

Programming

SUBROUTINE ORTHOV (MO, NA, AU, AR, NF, FF, DF, SF)

 FORTRAN SUBROUTINE FOR ORTHONORMAL POLYNOMIAL EVALUATION

The mode of operation MO is 0 for functions, 1 for first derivatives, and 2 for second derivatives. The number of discrete arguments n is given in argument NA. The variable argument u is given in argument AU. The recurrence coefficients are given in the $2m - 2$ array AR. The number of functions m is given in argument NF. The functions are stored in the m -array FF, the first derivatives are stored in the m -array DF, and the second derivatives are stored in the m -array SF.

ISOMETRIC REPRESENTATION

Analysis

Let x, y be functions of u such that u is proportional to the length of the curve which connects x, y . Approximate values of u are derived from the perimeter of a polygon which is inscribed within the curve. Let u_i be the i th discrete value of u , and let x_i, y_i be the i th discrete values of x, y . Let the values of u be limited to the range

$$-1 \leq u \leq +1 \quad (70)$$

The values of u are derived from the equation

$$u_{i+1} - u_i = \frac{2 \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{\sum \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \quad (71)$$

Accuracy of the simulation requires closely spaced data.

Let x, y be approximated as power polynomials in u by the equations

$$x = \sum_{i=1}^n x_i \varphi_i(u) \quad y = \sum_{i=1}^n y_i \varphi_i(u) \quad (72)$$

where $\varphi_i(u)$ is the i th Lagrange polynomial. The coefficients of the polynomial approximations of x, y are the sums of products of the discrete values of x, y and the columns of the matrix of coefficients for the functions $\varphi_i(u)$.

The polynomial approximation for x, y defines a curve for which the metric l is defined by the equation

$$\frac{dl}{du} = \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2} \quad (73)$$

The coefficients of a polynomial approximation for l are the sums of products of the discrete values of l and the columns of the matrix of coefficients for the functions $\varphi_i(u)$. Improved isometry is achieved if u is replaced in accordance with the iteration

$$u \rightarrow -1 + 2 \frac{\int_{-1}^u \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2} du}{\int_{-1}^{+1} \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2} du} \quad (74)$$

Stability of the iteration requires closely spaced data.

Programming

SUBROUTINE ISMTRS (NA, AX, AY, SU, AU)

FORTRAN SUBROUTINE FOR ISOMETRIC POLYGONAL SIMULATION

The number n of data is given in argument NA. The coordinates x_i are given in the n -array AX, and the coordinates y_i are given in the n -array AY. The perimeter of the polygon is stored in address SU. The coordinates u_i are stored in the n -array AU.

SUBROUTINE LPLNMA (NA, AU, AX, AY, CX, CY, AC)

FORTRAN SUBROUTINE FOR LAGRANGIAN POLYNOMIAL APPROXIMATION

The number n of data is given in the address NA. The arguments $u_1, \dots, u_n$ are given in the n -array AU, the arguments $x_1, \dots, x_n$ are given in the n -array AX, and the arguments $y_1, \dots, y_n$ are given in the n -array AY. The coefficients of the polynomial for x are stored in the n -array CX, and the coefficients of the polynomial for y are stored in the n -array CY. The coefficients of the Lagrange polynomials are stored in the $(n+1) \times n$ array AC. A call is made to Subroutine LGRNGX.

SUBROUTINE OPLNMA (NA, AU, AX, AY, NC, CX, CY, AA, AR, AC)

FORTRAN SUBROUTINE FOR ORTHONORMAL POLYNOMIAL APPROXIMATION

The number n of data is given in the address NA. The arguments $u_1, \dots, u_n$ are given in the n -array AU, the arguments $x_1, \dots, x_n$ are given in the n -array AX, and the arguments $y_1, \dots, y_n$ are given in the n -array AY. The number m of coefficients is given in the address NC. The coefficients of the polynomial for x are stored in the m -array CX, and the coefficients of the polynomial for y are stored in the m -array CY. Values of orthonormal polynomials are stored in the $m \times n$ array AA, the recurrence coefficients are stored in the $2m-2$ array AR, and the coefficients of the polynomials are stored in the $m \times m$ array AC. Calls are made to Subroutine ORTHOS and to Subroutine ORTHOX.

SUBROUTINE ISMTRL (NA, AU, AL, CX, CY, CL, AC)

FORTRAN SUBROUTINE FOR ISOMETRIC LAGRANGIAN ITERATION

The number of arguments n is given in the argument NA. The coordinates $u_1, \dots, u_n$ are given in the n -array AU. The values of the l -metric are stored in the n -array AL. The coefficients for the x -coordinate are given in the n -array CX, and the coefficients for the y -coordinate are given in the n -array CY. The coefficients of the l -metric are stored in the n -array CL. The matrix of Lagrange polynomials is given in the $(n+1) \times n$ array AC. Improved coordinates $u_1, \dots, u_n$ are returned to the n -array AU. Calls are made to Subroutine MPLNMV.

SUBROUTINE ISMTRO (NA, AU, AL, NC, CX, CY, CL, AA, AR, AC)

 FORTRAN SUBROUTINE FOR ISOMETRIC ORTHONORMAL ITERATION

The number of arguments n is given in the argument NA. The coordinates $u_1, \dots, u_n$ are given in the n -array AU. The values of the l -metric are stored in the n -array AL. The number of coefficients m is given in the argument NC. The coefficients for the x -coordinate are given in the m -array CX, and the coefficients for the y -coordinate are given in the m -array CY. The coefficients of the l -metric are stored in the m -array CL. Values of orthonormal polynomials are given in the $m \times n$ array AA, the recurrence coefficients are given in the $2m - 2$ array AR, and the matrix of coefficients is given in the $m \times m$ array AC. Improved coordinates $u_1, \dots, u_n$ are returned to the n -array AU. Calls are made to subroutine MPLNMV.

LAGRANGE INTEGRATION

Analysis

Let $u_1, \dots, u_n$ be arbitrary coordinates and let $f(u_1), \dots, f(u_n)$ be values of an arbitrary function. Lagrange interpolation approximates the function in accordance with the equation

$$f(u) = \sum_{i=1}^n f(u_i) \varphi_i(u) \quad (75)$$

where $\varphi_i(u)$ is the i th Lagrange polynomial. Expansion of each Lagrange polynomial in powers of the argument and integration lead to the integration multipliers

$$m_i = \int_{-1}^{+1} \varphi_i(u) du \quad (76)$$

Then the integral of an arbitrary function is given by the equation

$$\int_{-1}^{+1} f(u) du = \sum_{i=1}^n m_i f(u_i) \quad (77)$$

Expansion and integration is achieved by reference to auxiliary subroutines.

Programming

SUBROUTINE LGRMLT (NA, AU, AM, CP)

 FORTRAN SUBROUTINE FOR LAGRANGE INTEGRATION MULTIPLIERS

The number of coordinates n is given in argument NA. The coordinates $u_1, \dots, u_n$ are given in the n -array AU. The integration multipliers $m_1, \dots, m_n$ are stored in the n -array AM. The coefficients of the Lagrange polynomials are stored in the $(n + 1) \times n$ array CP. Calls are made to Subroutine LGRNGX and Subroutine MPLNMV.

ORTHONORMAL INTEGRATION

Analysis

Let $u_1, \dots, u_n$ be arbitrary coordinates and let $f(u_1), \dots, f(u_n)$ be values of an arbitrary function. Orthonormal interpolation approximates the function in accordance with the equation

$$f(u) = \sum_{k=0}^{n-1} \sum_{i=1}^n f(u_i) \varphi_k(u_i) \varphi_k(u) \quad (78)$$

where $\varphi_k(u)$ is the k th orthonormal polynomial. Expansion of each orthonormal polynomial in powers of the argument and integration lead to the integration multipliers

$$m_i = \sum_{k=0}^{n-1} \varphi_k(u_i) \int_{-1}^{+1} \varphi_k(u) du \quad (79)$$

Then the integral of an arbitrary function is given by the equation

$$\int_{-1}^{+1} f(u) du = \sum_{i=1}^n m_i f(u_i) \quad (80)$$

Expansion and integration is achieved by reference to auxiliary subroutines.

Programming

SUBROUTINE OPLMLT (NA, AU, AM, AA, AR, AC)

 FORTRAN SUBROUTINE FOR ORTHONORMAL POLYNOMIAL INTEGRATION MULTIPLIERS

The number of coordinates n is given in argument NA. The coordinates $u_1, \dots, u_n$ are given in the n -array AU. The integration multipliers $m_1, \dots, m_n$ are stored in the n -array AM. The values of orthonormal polynomials are stored in the $n \times n$ array AA, the recurrence coefficients are stored in the $2n-2$ array AR, and the coefficients of expanded polynomials are stored in the $n \times n$ array AC. Calls are made to Subroutine ORTHOS, Subroutine ORTHOX, and Subroutine MPLNMV.

TRAPEZOIDAL INTEGRATION

Analysis

Let $\theta_1, \dots, \theta_n$ be equally spaced values of an angle θ such that the i th value is defined by the equation

$$\theta_i = (i - \frac{1}{2}) \frac{2\pi}{n} \quad (81)$$

The Euler theorem and the rule of summation for a geometric series are used in the summation of the products of the trigonometric functions

$$\cos k\theta \qquad \qquad \sin m\theta \quad (82)$$

The summations over the angles θ_i are given by the equations

$$\sum_{i=1}^n \cos k\theta_i \cos m\theta_i = \frac{\sin(k-m)2\pi}{4 \sin(k-m)\frac{\pi}{n}} + \frac{\sin(k+m)2\pi}{4 \sin(k+m)\frac{\pi}{n}} \quad (k \pm m \neq jn) \quad (83)$$

$$\sum_{i=1}^n \cos k\theta_i \sin m\theta_i \equiv 0 \quad (84)$$

$$\sum_{i=1}^n \sin k\theta_i \sin m\theta_i = \frac{\sin(k-m)2\pi}{4 \sin(k-m)\frac{\pi}{n}} - \frac{\sin(k+m)2\pi}{4 \sin(k+m)\frac{\pi}{n}} \quad (k \pm m \neq jn) \quad (85)$$

and by the equations

$$\sum_{i=1}^n \cos^2 k\theta_i = n \quad (k = 0) \quad (86)$$

$$\sum_{i=1}^n \cos^2 k\theta_i = \frac{1}{2}n \quad (0 < k < \frac{1}{2}n) \quad (87)$$

$$\sum_{i=1}^n \sin^2 k\theta_i = \frac{1}{2}n \quad (0 < k \leq \frac{1}{2}n) \quad (88)$$

The functions are orthogonal if k, m satisfy the inequality

$$0 < |k \pm m| < n \quad (89)$$

No term with $k = \frac{1}{2}n$ can be included in the cosine series because $\cos k\theta_i$ is zero for all i when $k = \frac{1}{2}n$. An arbitrary function $f(\theta)$ is approximated by the trigonometric polynomial

$$f(\theta) = \frac{1}{2}a_0 + \sum_{m=1}^{m < \frac{1}{2}n} a_m \cos m\theta + \sum_{m=1}^{m \leq \frac{1}{2}n} b_m \sin m\theta \quad (90)$$

The coefficients in the approximation are recovered with the equations

$$a_m = \frac{2}{n} \sum_{i=1}^n f(\theta_i) \cos m\theta_i \quad (m < \frac{1}{2}n) \quad (91)$$

$$b_m = \frac{2}{n} \sum_{i=1}^n f(\theta_i) \sin m\theta_i \quad (m \leq \frac{1}{2}n) \quad (92)$$

Only the constant a_0 survives integration through the angle 2π as expressed by the equation

$$\int_0^{2\pi} f(\theta) d\theta = \frac{2\pi}{n} \sum_{i=1}^n f(\theta_i) \quad (93)$$

which is the trapezoidal rule of integration over a complete cycle.

Programming

SUBROUTINE TPZMLT (NA, AQ, AM)

 FORTRAN SUBROUTINE FOR TRAPEZOIDAL INTEGRATION MULTIPLIERS

The number n of coordinates is given in argument NA. The equally spaced coordinates

The summations over the angles θ_i are given by the equations

$$\sum_{i=1}^n \cos k\theta_i \cos m\theta_i = \frac{\sin(k-m)2\pi}{4 \sin(k-m)\frac{\pi}{n}} + \frac{\sin(k+m)2\pi}{4 \sin(k+m)\frac{\pi}{n}} \quad (k \pm m \neq jn) \quad (83)$$

$$\sum_{i=1}^n \cos k\theta_i \sin m\theta_i = 0 \quad (84)$$

$$\sum_{i=1}^n \sin k\theta_i \sin m\theta_i = \frac{\sin(k-m)2\pi}{4 \sin(k-m)\frac{\pi}{n}} - \frac{\sin(k+m)2\pi}{4 \sin(k+m)\frac{\pi}{n}} \quad (k \pm m \neq jn) \quad (85)$$

and by the equations

$$\sum_{i=1}^n \cos^2 k\theta_i = n \quad (k = 0) \quad (86)$$

$$\sum_{i=1}^n \cos^2 k\theta_i = \frac{1}{2}n \quad (0 < k < \frac{1}{2}n) \quad (87)$$

$$\sum_{i=1}^n \sin^2 k\theta_i = \frac{1}{2}n \quad (0 < k \leq \frac{1}{2}n) \quad (88)$$

The functions are orthogonal if k, m satisfy the inequality

$$0 < |k \pm m| < n \quad (89)$$

No term with $k = \frac{1}{2}n$ can be included in the cosine series because $\cos k\theta_i$ is zero for all i when $k = \frac{1}{2}n$. An arbitrary function $f(\theta)$ is approximated by the trigonometric polynomial

$$f(\theta) = \frac{1}{2}a_0 + \sum_{m=1}^{m < \frac{1}{2}n} a_m \cos m\theta + \sum_{m=1}^{m \leq \frac{1}{2}n} b_m \sin m\theta \quad (90)$$

The coefficients in the approximation are recovered with the equations

$$a_m = \frac{2}{n} \sum_{i=1}^n f(\theta_i) \cos m\theta_i \quad (m < \frac{1}{2}n) \quad (91)$$

$$b_m = \frac{2}{n} \sum_{i=1}^n f(\theta_i) \sin m\theta_i \quad (m \leq \frac{1}{2}n) \quad (92)$$

Only the constant a_0 survives integration through the angle 2π as expressed by the equation

$$\int_0^{2\pi} f(\theta) d\theta = \frac{2\pi}{n} \sum_{i=1}^n f(\theta_i) \quad (93)$$

which is the trapezoidal rule of integration over a complete cycle.

Programming

SUBROUTINE TPZMLT (NA, AQ, AM)

 FORTRAN SUBROUTINE FOR TRAPEZOIDAL INTEGRATION MULTIPLIERS

The number n of coordinates is given in argument NA. The equally spaced coordinates

$\theta_1, \dots, \theta_n$ are stored in the n -array AQ. The integration multipliers $m_1, \dots, m_n$ for trapezoidal integration are stored in the n -array AM.

CHEBYSHEV INTEGRATION

Analysis

Let $\theta_1, \dots, \theta_n$ be equally spaced values of the angle θ such that the i th value is defined by the equation

$$\theta_i = (i - \frac{1}{2}) \frac{\pi}{n} \quad (94)$$

The Euler theorem and the rule of summation for a geometric series are used in the summation of the products of the trigonometric functions

$$\cos k\theta \quad \cos m\theta \quad (95)$$

The summations over the angles θ_i are given by the equations

$$\sum_{i=1}^n \cos k\theta_i \cos m\theta_i = \frac{\sin(k-m)\pi}{4 \sin(k-m)\frac{\pi}{2n}} + \frac{\sin(k+m)\pi}{4 \sin(k+m)\frac{\pi}{2n}} \quad (k \pm m \neq 2jn) \quad (96)$$

and by the equations

$$\sum_{i=1}^n \cos^2 k\theta_i = n \quad (k = 0) \quad (97)$$

$$\sum_{i=1}^n \cos^2 k\theta_i = \frac{1}{2}n \quad (0 < k < n) \quad (98)$$

The functions are orthogonal if k, m satisfy the inequality

$$0 < |k \pm m| < 2n \quad (99)$$

An arbitrary function $f(\theta)$ is approximated by the trigonometric polynomial

$$f(\theta) = \frac{1}{2}a_0 + \sum_{m=1}^{n-1} a_m \cos m\theta \quad (100)$$

The coefficients in the approximation are recovered with the equations

$$a_m = \frac{2}{n} \sum_{i=1}^n f(\theta_i) \cos m\theta_i \quad (101)$$

Only the constant a_0 survives integration through the angle π as expressed by the equation

$$\int_0^\pi f(\theta) d\theta = \frac{\pi}{n} \sum_{i=1}^n f(\theta_i) \quad (102)$$

which is the trapezoidal rule of integration over a half cycle.

Let the argument x be defined in terms of the angle θ by the equation

$$x = -\cos \theta \quad (103)$$

Integration with respect to x is expressed by the equation

$$\int_{-1}^{+1} f(x) dx = \int_0^\pi f(\theta) \sin \theta d\theta \quad (104)$$

Term by term integration of the trigonometric polynomial is achieved with the aid of the indefinite integral

$$\int \cos k\theta \sin \theta d\theta = \frac{\cos(k-1)\theta}{2(k-1)} - \frac{\cos(k+1)\theta}{2(k+1)} \quad (105)$$

Evaluation at the limits of integration leads to the equations

$$\int_0^\pi \cos k\theta \sin \theta d\theta = 0 \quad (\text{odd } k) \quad (106)$$

$$\int_0^\pi \cos k\theta \sin \theta d\theta = -\frac{2}{(k^2-1)} \quad (\text{even } k) \quad (107)$$

Integration multipliers are defined by the equation

$$m_i = \frac{2}{n} - \frac{4}{n} \sum_{k=1}^{k < \frac{1}{2}n} \frac{\cos 2k\theta_i}{4k^2-1} \quad (108)$$

Then integration with respect to x is evaluated by the equation

$$\int_{-1}^{+1} f(x) dx = \sum_{i=1}^n m_i f(x_i) \quad (109)$$

where the m_i are the integration multipliers for Chebyshev integration.

Programming

SUBROUTINE CHVMLT (NA, AU, AM)

 FORTRAN SUBROUTINE FOR CHEBYSHEV INTEGRATION MULTIPLIERS

The number n of coordinates is given in argument NA. The cosines $u_1, \dots, u_n$ of equally spaced angles are stored in the n -array AU. The integration multipliers $m_1, \dots, m_n$ for Chebyshev integration are stored in the n -array AM.

HIGH-ACCURACY QUADRATURE

Analysis

Christoffel-Darboux Identity

Let an arbitrary function $f(x)$ be given at the abscissae $x_1, \dots, x_n$. Let the polynomial $p(x)$ be equal to the values of $f(x)$ at the abscissae as expressed by the equation

$$p(x_i) = f(x_i) \quad (110)$$

Let the function $F(x)$ be defined by the equation

$$F(x) = (x - x_1) \cdots (x - x_n) \quad (111)$$

The function $F(x)$ is a power polynomial of the n th degree and is zero at each abscissa. Let the function $q(x)$ be a power series which satisfies the equation

$$f(x) = p(x) + q(x)F(x) \quad (112)$$

If the function $f(x)$ itself were expressed by a power series, then the coefficients of $q(x)$ could be determined one by one by an algorithm which starts at the highest power of x and compares terms with the same power of x . The abscissae are so selected as to satisfy the equation

$$\int_a^b q(x)F(x)w(x) dx = 0 \quad (113)$$

where $w(x)$ is a weighting function, and $q(x)$ is any polynomial of the $(n-1)$ th degree. Insofar as $f(x)$ can be approximated by a polynomial of the $(2n-1)$ th degree, the integral of $f(x)$ is given accurately by the equation

$$\int_a^b f(x)w(x) dx = \int_a^b p(x)w(x) dx \quad (114)$$

The function $q(x)$ can be expressed as a linear combination of orthonormal polynomials of which the n th polynomial is the function $F(x)$.

Let $\varphi_0(x), \dots, \varphi_{n-1}(x)$ be n polynomials of progressively increasing degree. The polynomials are orthonormal if they satisfy the condition

$$\int_a^b \varphi_k(x)\varphi_m(x)w(x) dx = \delta_{km} \quad (115)$$

where δ_{km} is zero if $k \neq m$ and is unity if $k = m$. Any polynomial of degree m can be expanded in terms of the orthonormal polynomials of degrees 0 through m . The coefficients in the expansion are given by the solution of a sequence of equations which start at the highest degree and compare terms of the same degree. Let $\varphi_m(x)$ be expressed by the equation

$$\varphi_m(x) = \gamma_m x \varphi_{m-1}(x) + \sum_{n=0}^{m-1} \gamma_n \varphi_n(x) \quad (116)$$

where the γ_n are constant coefficients. Multiplication throughout by $\varphi_k(x)$ and integration isolates γ_k in accordance with the equation

$$\int_a^b \varphi_k(x)\varphi_m(x)w(x) dx = \gamma_m \int_a^b x \varphi_k(x)\varphi_{m-1}(x)w(x) dx + \gamma_k = 0 \quad (117)$$

The polynomial $x\varphi_k(x)$ may be expressed by the equation

$$x\varphi_k(x) = \sum_{\nu=0}^{k+1} \epsilon_\nu \varphi_\nu(x) \quad (118)$$

where the ϵ_ν are constant coefficients. All terms of the summation are orthogonal to $\varphi_{m-1}(x)$ and the coefficient γ_k is zero for all $k \leq m-3$. The orthonormal polynomials can be generated by a three-term recurrence equation.

If $\varphi_{-1}(x)$ is zero and $\varphi_0(x)$ is constant, then the three-term recurrence equation is

$$\varphi_m(x) = \frac{1}{\alpha_{m-1}} \left\{ x\varphi_{m-1}(x) - \beta_{m-1}\varphi_{m-1}(x) - \alpha_{m-2}\varphi_{m-2}(x) \right\} \quad (119)$$

where the constants α_{m-1} and β_{m-1} are defined by the equations

$$\alpha_{m-1} = \int_a^b x\varphi_{m-1}(x)\varphi_m(x)w(x)dx \quad (120)$$

$$\beta_{m-1} = \int_a^b x\varphi_{m-1}(x)\varphi_{m-1}(x)w(x)dx \quad (121)$$

Multiplication of the three-term recurrence equation throughout by $\varphi_{m-1}(y)$, interchange of x and y , and subtraction gives the equation

$$(x-y)\varphi_{m-1}(x)\varphi_{m-1}(y) = \alpha_{m-1} \{ \varphi_m(x)\varphi_{m-1}(y) - \varphi_{m-1}(x)\varphi_m(y) \} - \alpha_{m-2} \{ \varphi_{m-1}(x)\varphi_{m-2}(y) - \varphi_{m-2}(x)\varphi_{m-1}(y) \} \quad (122)$$

Summation with respect to m gives the Christoffel-Darboux identity

$$\alpha_{n-1} \{ \varphi_n(x)\varphi_{n-1}(y) - \varphi_{n-1}(x)\varphi_n(y) \} = (x-y) \sum_{k=0}^{n-1} \varphi_k(x)\varphi_k(y) \quad (123)$$

Rearrangement and integration gives the equation

$$\int_a^b \varphi_0(x) \left\{ \frac{\varphi_n(x)\varphi_{n-1}(y) - \varphi_{n-1}(x)\varphi_n(y)}{x-y} \right\} w(x)dx = \frac{\varphi_0(y)}{\alpha_{n-1}} \quad (124)$$

which is fundamental to Lagrange integration when y is set equal to an abscissa x_i .

Chebyshev Quadrature

The Chebyshev polynomial $T_n(x)$ is defined by the equation

$$T_n(x) = \cos(n \cos^{-1} x) \quad (125)$$

The Chebyshev polynomial for $0 < n$ is given by the equation

$$T_n(x) = 2^{n-1} \sum_{m=0}^{n-1} \frac{(-1)^m n(n-m)! x^{n-2m}}{2^{2m}(n-m)(n-2m)!m!} \quad (0 < n) \quad (126)$$

The functions of lowest order are given by the equations

$$T_0(x) = 1 \quad T_1(x) = x \quad (127)$$

and the functions satisfy the recurrence equation

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \quad (128)$$

The first derivatives of lowest order are given by the equations

$$T'_0(x) = 0 \quad T'_1(x) = 1 \quad (129)$$

and the first derivatives satisfy the recurrence equation

$$T'_n(x) = nT_{n-1}(x) + \frac{n}{n-1}xT'_{n-1}(x) \quad (130)$$

The second derivatives of lowest order are given by the equations

$$T_0''(x) = 0 \quad T_1''(x) = 0 \quad (131)$$

and the second derivatives satisfy the recurrence equation

$$T_n''(x) = \frac{n^2}{n-1} T_{n-1}'(x) + \frac{n}{n-1} x T_{n-1}''(x) \quad (132)$$

The Chebyshev functions satisfy the orthogonality equations

$$\int_{-1}^{+1} \frac{T_k(x) T_m(x)}{\sqrt{1-x^2}} dx = 0 \quad (k \neq m) \quad (133)$$

$$\int_{-1}^{+1} \frac{T_n^2(x)}{\sqrt{1-x^2}} dx = \begin{cases} \pi & (n=0) \\ \frac{1}{2}\pi & (0 < n) \end{cases} \quad (134)$$

The limits of integration and the weighting function are given by the equations

$$a = -1 \quad w(x) = \frac{1}{\sqrt{1-x^2}} \quad b = +1 \quad (135)$$

The orthonormal functions and the Chebyshev functions are related by the equation

$$\varphi_n(x) = \sqrt{\frac{2}{\pi}} T_n(x) \quad (136)$$

A comparison between the three-term recurrence equations leads to the equations

$$\alpha_{n+1} = \frac{1}{2} \quad \beta_{n+1} = 0 \quad (137)$$

The function $F(x)$ is given by the equation

$$F(x) = \frac{1}{2^{n-1}} T_n(x) \quad (138)$$

and its derivative is given by the equation

$$F'(x) = \frac{1}{2^{n-1}} T_n'(x) \quad (139)$$

The abscissae are given by the equation

$$x_i = \cos\left(i - \frac{1}{2}\right) \frac{\pi}{n} \quad (140)$$

The Lagrangian integration multipliers are given by the equation

$$m_i = \int_{-1}^{+1} \frac{F(x)}{F'(x_i) (x - x_i)} dx \quad (141)$$

Substitutions in the Christoffel Darboux identity lead to the equation

$$m_i = \frac{\pi}{T_{n-1}(x_i) T_n'(x_i)} \quad (142)$$

Evaluations of the Chebyshev functions at the abscissae lead to the equation

$$m_i = \frac{\pi}{n} \quad (143)$$

The integral of an arbitrary function is given by the equation

$$\int_{-1}^{+1} \frac{f(x)}{\sqrt{1-x^2}} dx = \sum_{i=1}^n m_i f(x_i) \quad (144)$$

where the m_i are the multipliers for the trapezoidal rule.

Gauss Quadrature

The Legendre function of n th order is given by the equation

$$P_n(x) = \sum_{m=0}^{\lfloor \frac{n}{2} \rfloor} \frac{(-1)^m (2n-2m)! x^{n-2m}}{2^n (n-2m)! (n-m)! m!} \quad (145)$$

The functions of lowest order are given by the equations

$$P_0(x) = 1 \quad P_1(x) = x \quad (146)$$

and the functions satisfy the recurrence equation

$$nP_n(x) = (2n-1)xP_{n-1}(x) - (n-1)P_{n-2}(x) \quad (147)$$

The first derivatives of lowest order are given by the equations

$$P'_0(x) = 0 \quad P'_1(x) = 1 \quad (148)$$

and the first derivatives satisfy the recurrence equation

$$P'_n(x) = nP_{n-1}(x) + xP'_{n-1}(x) \quad (149)$$

The second derivatives of lowest order are given by the equations

$$P''_0(x) = 0 \quad P''_1(x) = 0 \quad (150)$$

and the second derivatives satisfy the recurrence equation

$$P''_n(x) = (n+1)P'_{n-1}(x) + xP''_{n-1}(x) \quad (151)$$

The Legendre functions satisfy the orthogonality equations

$$\int_{-1}^{+1} P_k(x)P_m(x) dx = 0 \quad (k \neq m) \quad (152)$$

and

$$\int_{-1}^{+1} P_n^2(x) dx = \frac{2}{2n+1} \quad (153)$$

The limits of integration and the weighting function are given by the equations

$$a = -1 \quad w(x) = 1 \quad b = +1 \quad (154)$$

The orthonormal functions and the Legendre functions are related by the equation

$$\varphi_n(x) = \sqrt{\frac{2n+1}{2}} P_n(x) \quad (155)$$

A comparison between the three-term recurrence equations leads to the equations

$$\alpha_{n+1} = \frac{n}{\sqrt{(2n-1)(2n+1)}} \quad \beta_{n+1} = 0 \quad (156)$$

The function $F(x)$ is given by the equation

$$F(x) = \frac{2^n (n!)^2}{(2n)!} P_n(x) \quad (157)$$

and its derivative is given by the equation

$$F'(x) = \frac{2^n (n!)^2}{(2n)!} P'_n(x) \quad (158)$$

The Lagrangian integration multipliers are given by the equation

$$m_i = \int_{-1}^{+1} \frac{F(x)}{F'(x_i)} \frac{dx}{(x - x_i)} \quad (159)$$

Substitutions in the Christoffel-Darboux identity lead to the equation

$$m_i = \frac{2}{nP_{n-1}(x_i)P'_n(x_i)} \quad (160)$$

The recurrence equation

$$(1 - x^2)P'_n(x) = n(P_{n-1}(x) - xP_n(x)) \quad (161)$$

leads further to the equation

$$m_i = \frac{2}{(1 - x_i^2)\{P'_n(x_i)\}^2} \quad (162)$$

The integral of an arbitrary function is given by the equation

$$\int_{-1}^{+1} f(x) dx = \sum_{i=1}^n m_i f(x_i) \quad (163)$$

where the m_i are the multipliers for Gaussian integration.

Let θ_i be an angle which is defined by the equation

$$\theta_i = \frac{i - \frac{1}{2}}{n + \frac{1}{2}} \pi \quad (164)$$

Then an initial approximation for x_i is given by the equation

$$x_i \sim \pm \cos \theta_i \quad (165)$$

The approximation for x_i is refined by Newton-Raphson iteration until

$$P_n(x_i) = 0 \quad (166)$$

to within rounding error. The Legendre function and its derivatives are computed by recurrence equations.

Programming

SUBROUTINE GSSMLT (NA, AU, AM)

FORTRAN SUBROUTINE FOR GAUSS INTEGRATION MULTIPLIERS

The number n of coordinates is given in argument NA. The roots $u_1, \dots, u_n$ of the n th degree Legendre polynomial are stored in the n -array AU. The integration multipliers $m_1, \dots, m_n$ are stored in the n -array AM.

COMPLEX POWER POLYNOMIAL EVALUATION

Analysis

Let $f(z)$ be a polynomial of $(n - 1)$ th degree in the argument z . Let the polynomial be expressed in terms of its argument by the equation

$$f(z) = \sum_{k=0}^{n-1} a_k z^k \quad (167)$$

where the coefficients a_k are complex. The coefficients are arranged in ascending order in an array with real parts in alternate addresses and with imaginary parts in the next higher addresses. The first derivative of the polynomial is given by the equation

$$\frac{df(z)}{dz} = \sum_{k=0}^{n-1} k a_k z^{k-1} \quad (168)$$

and the second derivative of the polynomial is given by the equation

$$\frac{d^2 f(z)}{dz^2} = \sum_{k=0}^{n-1} k(k-1) a_k z^{k-2} \quad (169)$$

Whether the operation is evaluation, first differentiation, or second differentiation is determined by a mode-of-operation parameter. The polynomials are evaluated by the nested method of polynomial evaluation.

Programming

SUBROUTINE CPWRV (MO, AZ, NC, AC, FV)

FORTRAN SUBROUTINE FOR COMPLEX POWER POLYNOMIAL EVALUATION

The mode of operation is evaluation if MO = 0, first differentiation if MO = 1, and second differentiation if MO = 2. The argument z is given in the 2-array AZ. The number of coefficients n is given in the address NC. The coefficients $a_0, \dots, a_{n-1}$ are given in the $2n$ -array AC. The value of the function is stored in the 2-array FV.

COMPLEX FOURIER SERIES EXPANSION

Analysis

Let $\theta_1, \dots, \theta_n$ be a set of angles such that

$$\theta_k = (k - \frac{1}{2}) \frac{2\pi}{n} \quad (170)$$

Each angle is the midpoint of one of n intervals into which 2π is divided. Let a function $f(\theta)$ be expressed in terms of its argument θ by the direct transform

$$f(\theta) = \sum_{k=0}^{n-1} A_k e^{ik\theta} \quad (171)$$

where the coefficients A_k are complex. The coefficients are recovered from values of the function by the inverse transform

$$A_m = \frac{1}{n} \sum_{k=1}^n f(\theta_k) e^{-im\theta_k} \quad (172)$$

The weighting factors for the transform are stored in an $n \times 2m$ matrix of coefficients. The weighting factors are the same for all values of $m(k - \frac{1}{2})$ which differ by n . It is sufficient to compute the $2n$ pairs of trigonometric functions for which $k = 1$ and $k = \frac{1}{2}$ are less than n . The trigonometric functions are computed and are stored in the first two columns of the matrix. The remainder of the matrix is filled by transfers from the first two columns. Then the first column is replaced by constants.

The coefficients for a particular set of values of the function are obtained from a vector-matrix multiplication. From the Fourier transform for one set of angles may be derived the Fourier transform for another set of angles. If the angles are displaced by a constant ϵ , then the coefficients of the Fourier transform are multiplied by powers of $e^{i\epsilon}$.

Programming

SUBROUTINE CFOURX (NA, NC, AC)

 FORTRAN SUBROUTINE FOR COMPLEX FOURIER SERIES EXPANSION

The number n of data is given in argument NA, and the number m of weighting factors is given in argument NC. The matrix of weighting factors is stored in the $n \times 2m$ array AC.

COMPLEX FOURIER SERIES EVALUATION

Analysis

Let $f(\theta)$ be a polynomial of $(n-1)$ th order with angular argument θ . Let the polynomial be expressed in terms of its argument by the equation

$$f(\theta) = \sum_{m=0}^{n-1} A_m e^{im\theta} \quad (173)$$

where the coefficients A_m are complex. The coefficients are arranged in ascending

order in an array with real parts in alternate addresses and with imaginary parts in the next higher addresses. The first derivative of the polynomial is given by the equation

$$\frac{df(\theta)}{d\theta} = \sum_{m=0}^{n-1} imA_m e^{im\theta} \quad (174)$$

and the second derivative of the polynomial is given by the equation

$$\frac{d^2f(\theta)}{d\theta^2} = - \sum_{m=0}^{n-1} m^2 A_m e^{im\theta} \quad (175)$$

Whether the operation is evaluation, first differentiation, or second differentiation is determined by a mode-of-operation parameter. The polynomials are evaluated by the nested method of polynomial evaluation.

Programming

SUBROUTINE CFOURV (MO, AQ, NA, AA, FV)

 FORTRAN SUBROUTINE FOR COMPLEX FOURIER SERIES EVALUATION

The mode of operation is evaluation if MO = 0, first differentiation if MO = 1, and second differentiation if MO = 2. The argument θ is given in the address AQ. The number of coefficients n is given in the address NA. The coefficients $A_0, \dots, A_{n-1}$ are given in the $2n$ -array AA. The value of the function is stored in the 2-array FV.

FAST FOURIER TRANSFORM

Analysis

In the conventional discrete transform a cycle is divided into N intervals. A set of discrete arguments $\theta_0, \dots, \theta_{N-1}$ is defined by the equation

$$\theta_k = k \frac{2\pi}{N} \quad (176)$$

A function $f(\theta)$ is expressed in terms of the argument θ by the series in the equation

$$f(\theta) = \sum_{m=0}^{N-1} A_m e^{im\theta} \quad (177)$$

The values of the function at the discrete arguments are given by the discrete Fourier transform in the equation

$$f(\theta_k) = \sum_{m=0}^{N-1} A_m e^{km \frac{2\pi}{N}} \quad (178)$$

The formula for the summation of geometric series leads to an orthogonality relation as expressed by the equation

$$\sum_{k=0}^{N-1} e^{k(m-n) \frac{2\pi}{N}} = N \delta_{nm} \quad (179)$$

where δ_{nm} is zero if $m \neq n$ and is unity if $m = n$. The coefficients of the Fourier series are isolated by applications of the orthogonality relation. The coefficients are given by the equation

$$A_m = \frac{1}{N} \sum_{k=0}^{N-1} e^{-km \frac{2\pi}{N}} f(\theta_k) \quad (180)$$

In the summation with respect to k , there is redundancy when N is factorable.

If $N = 2^n$, then the indices k, m are expressed as binary numbers in accordance with the equations

$$k = k_{n-1}2^{n-1} + \dots + k_0 \cdot 1 \quad (181)$$

$$m = m_{n-1}2^{n-1} + \dots + m_0 \cdot 1 \quad (182)$$

where each binary bit has the values 0, 1. Single summation with respect to k is replaced by multiple summation with respect to $k_{n-1}, \dots, k_0$ and functions of k or m are replaced by functions of $k_{n-1}, \dots, k_0$ or $m_{n-1}, \dots, m_0$. Thus the Fourier transform is expressed by the equation

$$A(m_{n-1}, \dots, m_0) = \frac{1}{N} \sum_{k_{n-1}=0}^1 \dots \sum_{k_0=0}^1 f(k_{n-1}, \dots, k_0) w^{(k_{n-1}2^{n-1} + \dots + k_0)(m_{n-1}2^{n-1} + \dots + m_0)} \quad (183)$$

where the weight w is defined by the equation

$$w = e^{-\frac{2\pi}{N}} \quad (184)$$

The weight w satisfies the identities

$$w^N = 1 \quad w^{\frac{1}{2}N} = -1 \quad (185)$$

There is therefore a repetition of factors whenever km is incremented by a multiple of $\frac{1}{2}N$.

In the Cooley-Tukey version of the FFT, the binary expression for k is resolved into its individual terms. In the product

$$(m_{n-1}2^{n-1} + \dots + m_0)k_{n-1}2^{n-1} \quad (186)$$

all terms reduce w to unity except those in the product

$$(m_{n-\nu}2^{n-\nu} + \dots + m_0)k_{n-1}2^{n-1} \quad (187)$$

The weighting factor in the Fourier transform is given by the equation

$$w^{km} = w^{(m_0)k_{n-1}2^{n-1} + \dots + (m_{n-1}2^{n-1} + \dots + m_0)k_0} \quad (188)$$

The summations with respect to $k_{n-1}, \dots, k_0$ are nested. The summation with respect to $k_{n-\nu}$ is associated with $m_{n-\nu}2^{n-\nu} + \dots + m_0$. If the transformations of data are made in place, then new arrays of data are created with the aid of the recurrence equation

$$A^{(\nu)}(m_0, \dots, m_{\nu-1}, k_{n-\nu-1}, \dots, k_0) = \sum_{k_{n-\nu}=0}^1 A^{(\nu-1)}(m_0, \dots, m_{\nu-2}, k_{n-\nu}, \dots, k_0) w^{(m_{n-\nu}2^{n-\nu} + \dots + m_0)k_{n-\nu}2^{n-\nu}} \quad (189)$$

where the order of the bits $m_{n-1}, \dots, m_0$ becomes reversed. The original set of data is transformed into new sets of data until the final set is the set of coefficients. Initially, summation is for $k_{n-1} = 0, 1$ with all values of $k_{n-2}, \dots, k_0$ and with $m_0 = 0, 1$. Finally, summation is for $k_0 = 0, 1$ with all values of $m_0, \dots, m_{n-1}$. In each cycle of transformation

the number of sets of data is doubled while the number of data per set is halved.

In the Sande-Tukey version of the FFT, the binary expression for m is resolved into its individual terms. In the product

$$(k_{n-1}2^{n-1} + \dots + k_0)m_{\nu-1}2^{\nu-1} \quad (190)$$

all terms reduce w to unity except those in the product

$$(k_{n-\nu}2^{n-\nu} + \dots + k_0)m_{\nu-1}2^{\nu-1} \quad (191)$$

The weighting factor in the Fourier transform is given by the equation

$$w^{km} = w^{(k_0)m_{n-1}2^{n-1} + \dots + (k_{n-1}2^{n-1} + \dots + k_0)m_0} \quad (192)$$

The summations with respect to $k_{n-1}, \dots, k_0$ are nested with the summation with respect to $k_{n-\nu}$ associated with $m_{\nu-1}$. If the transformations are made in place, then new arrays of data are created with the aid of the recurrence equation

$$A^{(\nu)}(m_0, \dots, m_{\nu-1}, k_{n-\nu-1}, \dots, k_0) = \sum_{k_{n-\nu}=0}^1 A^{(\nu-1)}(m_0, \dots, m_{\nu-2}, k_{n-\nu}, \dots, k_0) w^{(k_{n-\nu}2^{n-\nu} + \dots + k_0)m_{\nu-1}2^{\nu-1}} \quad (193)$$

while the order of the bits $m_{n-1}, \dots, m_0$ becomes reversed. The original set of data is transformed into new sets of data until the final set is the set of coefficients. Initially, summation is for $k_{n-1} = 0, 1$ with all values of $k_{n-2}, \dots, k_0$ and with $m_0 = 0, 1$. Finally, summation is for $k_0 = 0, 1$ with all values of $m_0, \dots, m_{n-1}$. In each cycle of transformation the number of sets of data is doubled and the number of data per set is halved.

The difference between the two algorithms is in the powers of w which are associated with summations with respect to the binary bits. Both algorithms require $n \log_2 n$ cycles of transformation, and both are unscrambled in accordance with bit reversal if they are executed in place.

Programming

SUBROUTINE CFFT (MO, LN, AA)

.....
 FORTRAN SUBROUTINE FOR COMPLEX FAST FOURIER TRANSFORM

The mode of operation is given in argument MO. The transform is from data to coefficients if MO = 1, and from coefficients to data if MO = +1. The $\log_2 N$ is given in argument LN. The data or coefficients are given or stored in the 2N array AA.

SUBROUTINE BR (LN, AA)

.....
 FORTRAN SUBROUTINE FOR BIT REVERSAL

The logarithm $\log_2 N$ is given in the argument LN. The bit-reversed indices for a half circle are stored in the $\frac{1}{2}N$ array B.

SUBROUTINE WFACT (LN, AA)

.....
 FORTRAN SUBROUTINE FOR COMPLEX WEIGHT FACTORS

The logarithm $\log_2 N$ is given in the argument LN. The trigonometric functions of

multiples of $(1, N)2\pi$ for a quadrant are stored in the $\frac{1}{2}N + 2$ array WK.

SUBROUTINE MXFFT (MO, LN, IR, WK, IA, AA)

FORTRAN SUBROUTINE FOR FOURIER TRANSFORM

The mode of operation is given in argument MO. The transform is from data to coefficients if MO = -1, and from coefficients to data if MO = +1. The $\log_2 N$ is given in argument LN. The bit reverse indices are given in the $\frac{1}{2}N$ array IR, and the trigonometric functions are given in the $\frac{1}{2}N + 2$ array WK. The interval between the addresses of data or coefficients is given in argument IA. The data or coefficients are given or stored in the $2N$ array AA.

ROOTS OF POLYNOMIALS

The solution of the quadratic equation was known to the Arabs in the ninth century. The solution of the cubic equation can be credited to Tartaglia in 1530. His solution was published by Cardan in 1545. The solution of the quartic equation can be credited to Ferrari who was a pupil of Cardan's. He resolved the quartic polynomial into a pair of quadratic factors. Another solution of the quartic equation was given by Euler in 1770. He assumed that the roots could be expressed by the sums or differences of three radicals. A correlation of the methods of solution with their discoverers is to be found in a book by Burnside and Panton¹⁴.

Solutions of the quartic equation can be classified as two radical^{15,16} or three radical¹⁷. The two methods must be equivalent analytically, but there is a difference in their accuracy of computation.

In the Ferrari method, the roots of the quartic equation are computed with the most positive real root of a resolvent cubic equation, whereas in the Euler method, the roots of the quartic equation are computed with all three roots of the resolvent cubic. In a special case of closely spaced roots, rounding error might upset the choice of root for the Ferrari method, but merely would modify the three roots for the Euler method.

Subroutines have been available in the CDC 6700 library for the solution of quadratic, cubic, and quartic equations. However, these subroutines do not give roots in a format which is useful for further complex arithmetic, and they do not give roots with the full accuracy of the computer. New and improved subroutines have been prepared in the present investigation. The new subroutines force the product of the roots to be equal to the constant in order to reduce rounding error in the smallest root. Previous subroutines for the quartic equation have used the Ferrari method, whereas there are new subroutines now for both the Ferrari method and the Euler method. In the previous version of the Ferrari method, a choice between alternate formulae was made only to avoid division by zero, whereas in the present version of the Ferrari method, the choice is made to achieve optimum accuracy.

The function name SQRT(*) has been preempted in FORTRAN for the square root of *. The function name CBR(*) is used on the UNIVAC 1108 computer for the cube root of *. A new function routine for the cube root has been prepared for the CDC 6700 computer. It obviates the inefficiency of exponentiation.

COMPLEX ROOTS OF A QUADRATIC EQUATION

Analysis

Let a quadratic equation have real coefficients as expressed by the equation

$$az^2 + bz + c = 0 \quad (194)$$

The term in z is eliminated by the substitution

$$z = t - \frac{b}{2a} \quad (195)$$

Then t is a solution of the equation

$$at^2 - \frac{b^2}{4a} + c = 0 \quad (196)$$

and z is given by the well known quadratic formula

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (197)$$

or by the equivalent formula

$$z = \frac{2c}{-b \pm \sqrt{b^2 - 4ac}} \quad (198)$$

which gives better accuracy when c is small and the sign of $-b$ is opposite to the sign of the radical.

If the discriminant satisfies the inequality

$$b^2 - 4ac \geq 0 \quad (199)$$

the roots are real, but if the discriminant satisfies the inequality

$$b^2 - 4ac \leq 0 \quad (200)$$

the roots are complex conjugate.

The discriminant is exactly zero only for integer values of the coefficients. Otherwise there is a residual error ϵ either from inherent error in the coefficients or from rounding error in the evaluation of the discriminant. In either case there remains an error $\epsilon^{1/2}$ in the square root of the discriminant. It is not possible to distinguish two closely spaced roots from one double root unless they have a discriminant larger than ϵ .

Programming

SUBROUTINE QDCRT (AC, RZ)

 FORTRAN SUBROUTINE FOR ROOTS OF QUADRATIC POLYNOMIAL

The real coefficients of the polynomial are given in array AC in ascending order. The roots of the polynomial are computed with the quadratic formula. The real and imaginary parts of the roots are stored in alternate addresses of array RZ.

COMPLEX ROOTS OF A CUBIC EQUATION

Analysis

Let a cubic equation have real coefficients as expressed by the equation

$$z^3 + pz^2 + qz + r = 0 \quad (201)$$

The term in z^2 is eliminated by the substitution

$$z = t - \frac{p}{3} \quad (202)$$

Then t is a solution of the equation

$$t^3 + at + b = 0 \quad (203)$$

where the coefficients are defined by the equations

$$a = \frac{1}{3}(3q - p^2) \quad (204)$$

$$b = \frac{1}{27}(2p^3 - 9pq + 27r) \quad (205)$$

In Tartaglia's method, t is given by the equation

$$t = A + B \quad (206)$$

where the constants A, B are defined by the equations

$$A = \left[-\frac{b}{2} + \sqrt{\frac{b^2}{4} + \frac{a^3}{27}} \right]^{\frac{1}{3}} \quad (207)$$

$$B = \left[-\frac{b}{2} - \sqrt{\frac{b^2}{4} + \frac{a^3}{27}} \right]^{\frac{1}{3}} \quad (208)$$

Among the three complex cube roots for each parameter only those pairs are selected for which

$$AB = -\frac{a}{3} \quad (209)$$

If the discriminant satisfies the inequality

$$\frac{b^2}{4} + \frac{a^3}{27} \geq 0 \quad (210)$$

there are one real root and two complex conjugate roots. If the discriminant satisfies the inequality

$$\frac{b^2}{4} + \frac{a^3}{27} \leq 0 \quad (211)$$

there are three real roots. The constants A, B are given by the equations

$$A = \sqrt[3]{-\frac{a}{3}} e^{+\frac{1}{3}\phi i} \quad (212)$$

$$B = \sqrt[3]{-\frac{a}{3}} e^{-\frac{1}{3}\phi i} \quad (213)$$

where ϕ is defined by the equation

$$\phi = \tan^{-1} \frac{\sqrt{-\frac{b^2}{4} - \frac{a^3}{27}}}{-\frac{b}{2}} \quad (214)$$

For every pair of constants A, B the two other pairs are obtained through multiplication by the factor

$$e^{\pm \frac{2}{3}\pi i} \quad (215)$$

Since A and B are complex conjugate, their sum is real.

Improved accuracy is achieved when the root of smallest absolute magnitude is replaced by the quotient of the constant with reversed sign and the product of the two largest roots.

Programming

SUBROUTINE CBRT (AC, RZ)

 FORTRAN SUBROUTINE FOR ROOTS OF CUBIC POLYNOMIAL

The real coefficients of the polynomial are given in array AC in ascending order. The roots of the polynomial are computed with the Tartaglia method. References are made to function routine CBRT. The real and imaginary parts of the roots are stored in alternate addresses of array RZ.

COMPLEX ROOTS OF A QUARTIC EQUATION

Analysis

Let a quartic equation have real coefficients as expressed by the equation

$$z^4 + pz^3 + qz^2 + rz + s = 0 \quad (216)$$

In Ferrari's method, the quartic polynomial is factored into the product of two quadratic polynomials. The solutions of the quartic equation are solutions of the quadratic equations

$$z^2 + \frac{1}{2}pz + \frac{1}{2}t \pm (az + b) = 0 \quad (217)$$

Multiplication of the quadratic factors and comparison of coefficients shows that the coefficients are related by the equations

$$q = t + \frac{1}{4}p^2 - a^2 \quad (218)$$

$$r = \frac{1}{2}pt - 2ab \quad (219)$$

$$s = \frac{1}{4}t^2 - b^2 \quad (220)$$

Elimination of a and b leads to the resolvent cubic equation

$$t^3 - qt^2 + (pr - 4s)t + 4qs - p^2s - r^2 = 0 \quad (221)$$

whence a and b can be calculated either with the equations

$$a = \sqrt{\frac{1}{4}p^2 + t - q} \quad b = \frac{\frac{1}{2}pt - r}{2a} \quad (222)$$

or with the equations

$$a = \frac{\frac{1}{2}pt - r}{2b} \quad b = \sqrt{\frac{1}{4}t^2 - s} \quad (223)$$

Computation of a and b is made with whichever pair of these equations gives the least rounding error. The choice of equations is based upon the sign of the expression

$$\frac{1}{8}p^2t - \frac{1}{4}qt + s \quad (224)$$

If the coefficients of the quartic equation satisfy the relationship

$$q = \frac{1}{4}p^2 + \frac{2r}{p} \quad (225)$$

then a is zero for one of the roots of the cubic equation. Substitution for q its expression in terms of p and r converts the cubic equation into

$$t^3 - \left(\frac{1}{4}p^2 + \frac{2r}{p}\right)t^2 + (pr - 4s)t + \frac{8rs}{p} - r^2 = 0 \quad (226)$$

which can be factored into the product

$$\left(t - \frac{2r}{p}\right)(t^2 - \frac{1}{4}p^2t + \frac{1}{2}pr - 4s) = 0 \quad (227)$$

The roots of the quadratic factor are

$$\frac{1}{8}p^2 \pm \frac{1}{2}\sqrt{\frac{1}{16}p^4 - 2pr + 16s} \quad (228)$$

for which the square of b is given by the equation

$$\frac{1}{4}t^2 - s = \frac{1}{128}p^4 - \frac{1}{8}pr \pm \frac{1}{32}p^2\sqrt{\frac{1}{16}p^4 - 2pr + 16s} \quad (229)$$

If the coefficients are related by the equation

$$s = \frac{r^2}{p^2} \quad (230)$$

then the radicand is a perfect square. For larger values of s and a positive radical the square of b is positive. The roots of the quartic equation are computed with the highest real root of the cubic equation.

The term in z^3 is eliminated from the quartic equation by the substitution

$$z = t - \frac{p}{4} \quad (231)$$

Then t is a solution of the equation

$$t^4 + at^2 + bt + c = 0 \quad (232)$$

where the coefficients are defined by the equations

$$a = -\frac{3}{8}p^2 + q \quad (233)$$

$$b = +\frac{1}{8}p^3 - \frac{1}{2}pq + r \quad (234)$$

$$c = -\frac{3}{256}p^4 + \frac{1}{16}p^2q - \frac{1}{4}pr + s \quad (235)$$

In Euler's method, the quartic equation in t is factored into the product

$$(t + \sqrt{l} + \sqrt{m} + \sqrt{n})(t + \sqrt{l} - \sqrt{m} - \sqrt{n})(t - \sqrt{l} + \sqrt{m} - \sqrt{n})(t - \sqrt{l} - \sqrt{m} + \sqrt{n}) = 0 \quad (236)$$

Multiplication of the factors leads to the equation

$$t^4 - 2(l + m + n)t^2 + 8\sqrt{lmn}t + (l + m + n)^2 - 4(mn + nl + lm) = 0 \quad (237)$$

Comparison of coefficients shows that the constants l, m, n are solutions of the equations

$$l + m + n = -\frac{1}{2}a \quad (238)$$

$$mn + nl + lm = \frac{1}{16}(a^2 - 4c) \quad (239)$$

$$\sqrt{lmn} = \frac{1}{8}b \quad (240)$$

Thus the constants l, m, n are the roots of the equation

$$k^3 + \frac{1}{2}ak^2 + \frac{1}{16}(a^2 - 4c)k - \frac{1}{64}b^2 = 0 \quad (241)$$

The roots of the quartic equation are derived directly from the roots of the resolvent cubic equation by the extraction of complex square roots.

Programming

SUBROUTINE QICRT (AC, R,)

FORTRAN SUBROUTINE FOR ROOTS OF QUARTIC POLYNOMIAL

The real coefficients of the polynomial are given in the array AC in ascending order. The roots of the polynomial are computed with the Euler method. Calls are made to Subroutine CBCRT. The real and imaginary parts of the roots are stored in alternate addresses of array R,.

COMPLEX ROOTS OF AN ANALYTIC FUNCTION

Analysis

Let z and w be complex variables which are expressed in terms of their real and imaginary parts by the equations

$$z = x + iy \quad w = u + iv \quad (242)$$

Let w be an analytic function of z as expressed by the equation

$$w = f(z) \quad (243)$$

The derivative of w is given by the equation

$$\frac{dw}{dz} = \frac{\left(\frac{\partial u}{\partial x} + i \frac{\partial v}{\partial x}\right) dx + \left(\frac{\partial v}{\partial y} - i \frac{\partial u}{\partial y}\right) i dy}{dx + i dy} \quad (244)$$

The derivative is independent of the direction only if the variables satisfy the Cauchy-Riemann equations

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \quad (245)$$

The modulus of w is given by the equation

$$|w| = (ww^*)^{\frac{1}{2}} = \sqrt{u^2 + v^2} \quad (246)$$

and the gradient of $|w|$ is given by the equation

$$\nabla|w| = \frac{\partial|w|}{\partial x} + i \frac{\partial|w|}{\partial y} \quad (247)$$

Differentiation and substitution lead to the equation

$$\nabla|w| = \frac{|\nabla|w||^2}{|w|} \frac{w}{dz} \quad (248)$$

Double differentiation in the Cauchy-Riemann equations shows that u, v satisfy Laplace's equation as expressed by the equations

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = 0 \quad (249)$$

whence the Laplacian of $|w|$ is given by the equation

$$\frac{\partial^2|w|}{\partial x^2} + \frac{\partial^2|w|}{\partial y^2} = \frac{|\nabla|w||^2}{|w|} \quad (250)$$

The gradient of the gradient of $|w|$ has the matrix

$$\begin{vmatrix} \frac{\partial^2|w|}{\partial x^2} & \frac{\partial^2|w|}{\partial x\partial y} \\ \frac{\partial^2|w|}{\partial x\partial y} & \frac{\partial^2|w|}{\partial y^2} \end{vmatrix} \quad (251)$$

The product of the characteristic roots of the matrix is the determinant

$$\frac{\partial^2|w|}{\partial x^2} \frac{\partial^2|w|}{\partial y^2} - \left(\frac{\partial^2|w|}{\partial x\partial y}\right)^2 \quad (252)$$

At a point of zero gradient the determinant is zero or negative and the point is a saddle point unless it is a root. The only minima in $|w|$ occur at the roots of $|w|$. The value of $|w|$ in a finite region is a maximum only on the perimeter of the region. Other analyses of the minima and the maxima are to be found in various texts.

Once a root has been established it must be eliminated from further consideration. Let $p(z)$ be defined by the equation

$$p(z) = \prod_{i=1}^k (z - a_i) \quad (253)$$

where $a_1, \dots, a_k$ are the first k roots. If $w(z)$ is defined by the equation

$$w(z) = \frac{f(z)}{p(z)} \quad (254)$$

then the roots of $w(z)$ are only those roots of $f(z)$ which have not yet been computed. The first derivative is given by the equation

$$\frac{dw}{dz} = \frac{f'}{p} - \frac{f}{p} \left[\sum_{i=1}^k \frac{1}{(z - a_i)} \right] \quad (255)$$

and the second derivative is given by the equation

$$\frac{d^2w}{dz^2} = \frac{f''}{p} - 2\frac{f'}{p} \left[\sum_{i=1}^k \frac{1}{(z - a_i)} \right] + \frac{f}{p} \left[\sum_{i=1}^k \frac{1}{(z - a_i)} \right]^2 + \frac{f}{p} \left[\sum_{i=1}^k \frac{1}{(z - a_i)^2} \right] \quad (256)$$

where f' and f'' are the derivatives of f .

During hunting, the displacement to an intercept on the complex plane is computed in accordance with the equation

$$\Delta z'' = - \frac{w}{\frac{dw}{dz}} \quad (257)$$

and the position z' of the intercept is computed in accordance with the equation

$$z' = z + \Delta z'' \quad (258)$$

For each displacement Δz of the position z there is a displacement $\Delta z'$ of the intercept z' . Hunting is continued as long as $|\Delta z'|$ is more than $|\Delta z|$.

The local expansion of w in terms of z is given to second order by the equation

$$w + \Delta w = w + \frac{dw}{dz} (\Delta z) + \frac{1}{2} \frac{d^2w}{dz^2} (\Delta z)^2 \quad (259)$$

The positions of two nearest roots are computed when the equation

$$w + \Delta w = 0 \quad (260)$$

is solved with the aid of the quadratic formula. A displacement is computed with the aid of the equation

$$\Delta z = \frac{2w}{-\frac{dw}{dz} \pm \sqrt{\left(\frac{dw}{dz}\right)^2 - 2w \frac{d^2w}{dz^2}}} \quad (261)$$

where the $\pm$ sign is selected to be whichever sign makes $|\Delta z|$ the lesser. Then the displacement is reduced to a step of length δ in the same direction as expressed by

the substitution

$$\Delta z \rightarrow \delta \frac{\Delta z}{|\Delta z|} \quad (262)$$

The local representation of w by a quadratic approximation eliminates the risk of entrapment at a saddle point in $|w|$.

During homing, a displacement to an intercept on the complex plane is computed in accordance with the equation

$$\Delta z'' = - \frac{w}{\frac{dw}{dz}} \quad (263)$$

and z is replaced in accordance with the substitution

$$z \rightarrow z' = z + \Delta z'' \quad (264)$$

Homing is continued as long as $|\Delta z''|$ is less than $|\Delta z|$. A tolerance ϵ distinguishes between disturbance at a saddle point and rounding error at a root. The position z is interpreted as a saddle point if

$$|\Delta z''| \geq |\Delta z| > \epsilon \quad (265)$$

while the position z is interpreted as a root if

$$|\Delta z| \leq |\Delta z''| < \epsilon \quad (266)$$

Regardless of the value of ϵ , a root is determined to the full accuracy to which the functions can be computed.

When the point of computation is near enough to a multiple root, the displacement in each cycle is a constant fraction of the distance to go to the root. The distance to go is the sum of an infinite geometric series. Displacement to the root in a single cycle is achieved when the displacement to the intercept is extrapolated to the interval

$$\frac{|\Delta z|}{|\Delta z| - |\Delta z''|} \Delta z'' \quad (267)$$

which expresses the summation of the geometric series.

Programming

SUBROUTINE FNCIN (MO, AZ, FF, DF, SF)

EXTERNAL SUBROUTINE FOR FUNCTION AND DERIVATIVES

The mode of operation MO is 0 for functions, 1 for first derivatives, and 2 for second derivatives. The argument z is given in the 2-array AZ. The function $f(z)$ is stored in the 2-array FF, the first derivative df/dz is stored in the 2-array DF, and the second derivative d^2f/dz^2 is stored in the 2-array SF.

SUBROUTINE CXRT (CD, CE, AZ, NA, AA, FNCTN)

 FORTRAN SUBROUTINE FOR COMPLEX ROOT DETERMINATION

The step length δ for hunting is given in argument CD, and the tolerance ϵ for homing is given in argument CE. The initial position z is given in the 2-array AZ. The number of roots n is given in argument NA. The region of a root is located by a hunting procedure with steps of fixed length, then the root is determined by a homing procedure with Newton-Raphson iteration. References are made to the external subroutine FNCTN to obtain values for $f(z)$, $f'(z)$, $f''(z)$. The real and imaginary parts of the roots are stored in alternate addresses of the $2n$ -array AA.

CONTOURS AMONG DATA

The General Purpose Contouring Program²³ or GPCP of CalComp works with a random array of elevations. Through each datum among the random data is passed a plane with an orientation which is the weighted average of the directions toward the nearest neighbors among the data. The weight function for the average has a bell-shaped distribution. The random array is converted into a regular grid by an application of the weight function to the heights of planes from neighbors nearest to a grid point.

The elevation in the interior of a square is expressed as the sum of products of functions which conform to the elevation and gradient at the four corners of the square. For interpolation with respect to x the functions $\varphi(x)$ are listed for a unit square in the following table.

$\varphi(x)$	$\varphi(0)$	$\varphi(1)$	$\varphi'(0)$	$\varphi'(1)$
$(1-x)^2(1+2x)$	1	0	0	0
$+x^2(3-2x)$	0	1	0	0
$+x(1-x)^2$	0	0	1	0
$-x^2(1-x)$	0	0	0	1

There is an analogous table for interpolation with respect to y . The products of the first two of one set with all of the other set provide enough functions to meet the twelve conditions of elevation and gradient at the four corners. Continuity of elevation and slope across each side of the square is guaranteed insofar as the elevation and slope along the side are determined by the elevations and gradients at the ends of the side.

The square is subdivided into a subgrid, and elevations are computed at grid points with the cubic approximation for elevation in the interior. Contours are traced through the subgrid with linear interpolation within each square of the subgrid.

The ultimate contouring system for topographic applications has been developed by Junkins and Jancaitis^{24,25}. The Contouring Via the Surface Averaging Concept or CONSAC system works with a rectangular array of elevations over a uniformly spaced grid of rectangular coordinates. The elevation z is expressed at a local center of approximation by the equation

$$z = c_0 + c_1x + c_2y + c_3xy + f(x, y) \quad (268)$$

where x, y are Cartesian coordinates with origin at the center of approximation. The coefficients in the local approximation are derived by sequential least squares from a local array of data which is centered over the center of approximation. Rational weight factors are applied to the data during the preliminary fitting to establish the local approximation for each corner of a square whose lower left corner is at the origin.

The local approximations at the corners of the square are blended in the interior of the square. Rational weight factors are applied to the local approximations during the final fitting to establish the interior approximation for the whole square. For a center of approximation at the origin of coordinates the weight factor $w(x, y)$ is given by the equation

$$w(x, y) = \frac{(1-x)^2(1-y)^2}{\{x^2 + (1-x)^2\}\{y^2 + (1-y)^2\}} \quad (269)$$

while the partial derivatives of the weight factor are given by the equations

$$\frac{\partial w}{\partial x} = - \frac{2x(1-x)(1-y)^2}{\{x^2 + (1-x)^2\}^2\{y^2 + (1-y)^2\}} \quad (270)$$

$$\frac{\partial w}{\partial y} = - \frac{(1-x)^2 2y(1-y)}{\{x^2 + (1-x)^2\}\{y^2 + (1-y)^2\}^2} \quad (271)$$

Thus the value of w is unity at the origin but is zero at the other three corners of a unit square, and the derivatives of w are zero at all four corners of the unit square. The weight factor satisfies the equation

$$w(x, y) + w(1-x, y) + w(x, 1-y) + w(1-x, 1-y) = 1 \quad (272)$$

throughout the interior of the square.

The elevation in the interior of the square is given by the equation

$$z = w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4 = F(x, y) \quad (273)$$

where w_1, w_2, w_3, w_4 are the weight factors for each of the four corners and f_1, f_2, f_3, f_4 are the local approximations for each of the four corners. Since the weight factors have quadratic numerators and denominators while the local approximations are linear, the final approximation for the interior of the square is the ratio between a cubic polynomial and a quadratic polynomial.

Differentiation with respect to x along a line of constant y and reduction to a common denominator replaces the numerator of $F(x, y)$ by a polynomial which is quartic in x , while differentiation with respect to y along a line of constant x and reduction to a common denominator replaces the numerator of $F(x, y)$ by a polynomial which is quartic in y . Solution of these quartic equations gives the points where $F(x, y)$ has minima or maxima along a side or along a median of the square. From the elevations of the minima or the maxima are derived the range of contour levels which intercept the sides or the medians of the square.

On a contour of level h the elevation is expressed by the equation

$$F(x, y) = h \quad (274)$$

Clearance of the denominator in this equation gives a cubic equation for the contour line. The cubic equation is solved for y with x set to a sequence of values or for x with y set to a sequence of values to obtain a sequence of points on the contour.

CONTOURS OF FUNCTIONS

Analysis

Let x, y be Cartesian coordinates and let z be a constant contour level. Let i, j be unit vectors in the directions of increasing x, y . Let $f(x, y)$ be a function of x, y . Then the equation

$$z = f(x, y) \quad (275)$$

defines implicitly a relation between x and y . A position vector r is defined by the equation

$$r = xi + yj \quad (276)$$

The differential change df in the function $f(x, y)$ for the differential change dr in the position vector r is given by the equation

$$df = \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial y} dy = \nabla f \cdot dr \quad (277)$$

and the gradient of $f(x, y)$ is defined by the equation

$$\nabla f = i \frac{\partial f}{\partial x} + j \frac{\partial f}{\partial y} \quad (278)$$

For each trial point there is an intercept in the x, y plane where the gradient is estimated to be zero. The displacement $\Delta x^*, \Delta y^*$ from the trial point to the point of zero gradient is a solution of the equations

$$\frac{\partial^2 f}{\partial x^2} \Delta x^* + \frac{\partial^2 f}{\partial x \partial y} \Delta y^* = - \frac{\partial f}{\partial x} \quad (279)$$

$$\frac{\partial^2 f}{\partial x \partial y} \Delta x^* + \frac{\partial^2 f}{\partial y^2} \Delta y^* = - \frac{\partial f}{\partial y} \quad (280)$$

The solution is expressed by the equations

$$\Delta x^* = - \frac{\frac{\partial f}{\partial x} \frac{\partial^2 f}{\partial y^2} - \frac{\partial f}{\partial y} \frac{\partial^2 f}{\partial x \partial y}}{\frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2} \quad (281)$$

$$\Delta y^* = - \frac{\frac{\partial f}{\partial y} \frac{\partial^2 f}{\partial x^2} - \frac{\partial f}{\partial x} \frac{\partial^2 f}{\partial x \partial y}}{\frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2} \quad (282)$$

The coordinates x^*, y^* of the point of zero gradient are given by the equations

$$x^* = x + \Delta x^* \quad y^* = y + \Delta y^* \quad (283)$$

The value f^* of the function at the coordinates x^*, y^* is given by the equation

$$f^* = f + \frac{\partial f}{\partial x} \Delta x^* + \frac{\partial f}{\partial y} \Delta y^* + \frac{1}{2} \frac{\partial^2 f}{\partial x^2} (\Delta x^*)^2 + \frac{\partial^2 f}{\partial x \partial y} \Delta x^* \Delta y^* + \frac{1}{2} \frac{\partial^2 f}{\partial y^2} (\Delta y^*)^2 \quad (284)$$

The matrix of the equations which determine the point of zero gradient is the matrix of the gradient of the gradient of f . It is symmetric, and it has real characteristic roots with orthogonal characteristic vectors. Its trace is the sum of its characteristic roots, and its determinant is the product of its characteristic roots.

Each of the characteristic roots μ, ν of the matrix is given by the equation

$$\mu, \nu = \pm \frac{1}{2} \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right) \pm \frac{1}{2} \sqrt{\left(\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \right)^2 + 4 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2} \quad (285)$$

where the choice of root is determined by the $\pm$ sign. The characteristic unit vectors $\mathbf{m}, \mathbf{n}$ are given by the equation

$$\begin{aligned} \mathbf{m}, \mathbf{n} = & \pm \frac{\left\{ \pm \frac{1}{2} \left(\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \right) + \frac{1}{2} \sqrt{\left(\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \right)^2 + 4 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2} \right\}^{\frac{1}{2}}}{\left\{ \left(\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \right)^2 + 4 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 \right\}^{\frac{1}{4}}} \mathbf{i} \\ & \pm \frac{\left\{ \pm \frac{1}{2} \left(\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \right) + \frac{1}{2} \sqrt{\left(\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \right)^2 + 4 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2} \right\}^{\frac{1}{2}}}{\left\{ \left(\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \right)^2 + 4 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 \right\}^{\frac{1}{4}}} \mathbf{j} \end{aligned} \quad (286)$$

where the choice of vector is determined by the $\pm$ sign. The gradient of the gradient of f is given by the equation

$$\nabla \nabla f = \mu \mathbf{m} \mathbf{m} + \nu \mathbf{n} \mathbf{n} \quad (287)$$

If the roots have opposite signs, the point of zero gradient is a saddle point, and the plane of z intersects the surface of f . If the roots have the same signs, the point of zero gradient is a minimum or a maximum. If the signs of the roots then are the same as the sign of $f - z$, the plane of z does not intersect the surface of f .

Let ρ be the position of the trial point relative to the point of zero gradient as expressed by the equation

$$\rho = \mathbf{r} - \mathbf{r}^* \quad (288)$$

Let u, v be Cartesian coordinates with unit vectors $\mathbf{m}, \mathbf{n}$ in the directions of increasing u, v . The position vector ρ of the trial point is given by the equation

$$\rho = u \mathbf{m} + v \mathbf{n} \quad (289)$$

The coordinates u, v are expressed in terms of the coordinates x, y by the equations

$$u = (\mathbf{x} - \mathbf{x}^*) \cdot \mathbf{m} + (\mathbf{y} - \mathbf{y}^*) \cdot \mathbf{n} \quad (290)$$

$$v = (\mathbf{x} - \mathbf{x}^*) \cdot \mathbf{n} + (\mathbf{y} - \mathbf{y}^*) \cdot \mathbf{m} \quad (291)$$

Let φ be the value of the function relative to the point of zero gradient as expressed by the equation

$$\varphi = f - f^* \quad (292)$$

The value of the function is given in terms of the coordinates by the equation

$$\varphi = \frac{1}{2} \mu u^2 + \frac{1}{2} \nu v^2 \quad (293)$$

If the characteristic roots have opposite signs the contours of constant φ are hyperbolas

and if the roots both have the same sign as φ the contours of constant φ are ellipses.

The trace of the matrix of the equations which determine the point of zero gradient is given by the expression

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (294)$$

and the determinant of the matrix is given by the expression

$$\frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 \quad (295)$$

If the determinant is zero there is no single point of zero gradient. The partial derivatives are related in accordance with the equation

$$\frac{\partial^2 f}{\partial x \partial y} = \pm \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}} \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}} \quad (296)$$

and the function is given for any $\Delta x^*, \Delta y^*$ by the equation

$$\begin{aligned} f^* = f - \frac{1}{2} & \left[\frac{\frac{\partial f}{\partial x} \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}} \pm \frac{\partial f}{\partial y} \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}}{\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}} \right]^2 \\ & + \frac{\frac{\partial f}{\partial x} \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}} \mp \frac{\partial f}{\partial y} \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}}}{\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}} \left[\left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}} \Delta x^* \mp \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}} \Delta y^* \right] \\ & + \frac{1}{2} \left[\frac{\frac{\partial f}{\partial x} \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}} \pm \frac{\partial f}{\partial y} \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}}{\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}} + \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}} \Delta x^* \pm \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}} \Delta y^* \right]^2 \end{aligned} \quad (297)$$

The contours of constant function are parabolas. The nearest point on the axis of the parabolas is reached when $\Delta x^*, \Delta y^*$ are given by the equations

$$\Delta x^* = - \frac{\frac{\partial f}{\partial x} \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}} \pm \frac{\partial f}{\partial y} \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}}{\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^2} \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}} \quad (298)$$

$$\Delta y^* = \mp \frac{\frac{\partial f}{\partial x} \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}} \pm \frac{\partial f}{\partial y} \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}}{\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^2} \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}} \quad (299)$$

Both terms of the trace have the same sign and their square roots are positive or

imaginary. Let λ, ν be defined by the equations

$$\lambda = \frac{\frac{\partial f}{\partial x} \left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}} + \frac{\partial f}{\partial y} \left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}}}{\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}} \quad (300)$$

$$\nu = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (301)$$

and let m, n be defined by the equations

$$m = + \frac{\left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}}{\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}} i + \frac{\left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}}}{\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}} j \quad (302)$$

$$n = + \frac{\left(\frac{\partial^2 f}{\partial x^2} \right)^{\frac{1}{2}}}{\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}} i + \frac{\left(\frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}}{\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^{\frac{1}{2}}} j \quad (303)$$

Let the coordinates u, v be derived from the coordinates x, y in accordance with the equations

$$u = (x - x^*)i \cdot m + (y - y^*)j \cdot m \quad (304)$$

$$v = (x - x^*)i \cdot n + (y - y^*)j \cdot n \quad (305)$$

Then the function φ is given by the equation

$$\varphi = \lambda u + \frac{1}{2} \nu v^2 \quad (306)$$

The contours of constant φ are parabolas with axis in the direction of m .

During hunting perpendicular to a contour a displacement to an intercept on the x, y plane is computed in accordance with the equation

$$\Delta r'' = - \frac{f}{|\nabla f|^2} \nabla f \quad (307)$$

and the position of the intercept is computed in accordance with the equations

$$x' = x + \Delta x'' \quad y' = y + \Delta y'' \quad (308)$$

The increments $\Delta x, \Delta y$ are replaced by a step of length δ in the direction of $\Delta r''$ as expressed by the equation

$$\Delta r = \delta \frac{\Delta r''}{|\Delta r''|} \quad (309)$$

unless a point of zero gradient is detected within that circle which is defined by the equation

$$(x^* - x)^2 + (y^* - y)^2 = \delta^2 \quad (310)$$

The characteristic roots and vectors of the matrix of $\nabla\nabla f$ are analysed to determine whether the plane at z can intersect the surface of f . Computation is terminated if there can be no intersection. Otherwise the trial point is displaced toward the nearest point of intersection. Hunting is continued as long as $|\Delta r'|$ is more than $|\Delta r|$, then homing is started when $|\Delta r'|$ becomes less than $|\Delta r|$.

During hunting parallel to a contour the steps of the trial point are expressed by quadratic curves which are osculatory to the contour.

If μ, ν have opposite signs, then the contour of constant φ is an hyperbola. Let the sign of μ be the same as the sign of φ so that u, v can be expressed in terms of a parameter η by the equations

$$u = \left(\frac{2\varphi}{\mu} \right)^{\frac{1}{2}} \cosh \eta \quad (311)$$

$$v = \left| \frac{2\varphi}{\nu} \right|^{\frac{1}{2}} \sinh \eta \quad (312)$$

Increments $\Delta u, \Delta v$ in the coordinates are given in terms of the increment $\Delta \eta$ in parameter by the equations

$$\Delta u = \left(\frac{2\varphi}{\mu} \right)^{\frac{1}{2}} \sinh \eta \Delta \eta \quad (313)$$

$$\Delta v = \left| \frac{2\varphi}{\nu} \right|^{\frac{1}{2}} \cosh \eta \Delta \eta \quad (314)$$

The hyperbolic functions may be expanded with the aid of the addition formula and may be expressed in terms of the coordinates to give the equations

$$u + \Delta u = u \cosh(\Delta \eta) + \left| \frac{\nu}{\mu} \right|^{\frac{1}{2}} v \sinh(\Delta \eta) \quad (315)$$

$$v + \Delta v = v \cosh(\Delta \eta) + \left| \frac{\mu}{\nu} \right|^{\frac{1}{2}} u \sinh(\Delta \eta) \quad (316)$$

which are valid for arbitrary increments, and the equations

$$\Delta u = \left| \frac{\nu}{\mu} \right|^{\frac{1}{2}} v \Delta \eta \quad (317)$$

$$\Delta v = \left| \frac{\mu}{\nu} \right|^{\frac{1}{2}} u \Delta \eta \quad (318)$$

which are valid for small increments. The increments $\Delta u, \Delta v$ are initially in the direction of the tangent, which can be derived directly from the gradient ∇f . The stability of the computation is controlled when the increments are subject to the limitation

$$\sqrt{(\Delta u)^2 + (\Delta v)^2} \leq \delta \quad (319)$$

where δ is the maximum length of step. Then the increment $\Delta \eta$ is determined with

sufficient accuracy by the equation

$$\Delta\eta = \frac{\delta}{\frac{\Delta v}{\sqrt{(\Delta u)^2 + (\Delta v)^2}} \left| \frac{\mu}{\nu} \right|^{\frac{1}{2}} u + \frac{\Delta u}{\sqrt{(\Delta u)^2 + (\Delta v)^2}} \left| \frac{\nu}{\mu} \right|^{\frac{1}{2}} v} \quad (320)$$

unless this would make $|\Delta\eta|$ more than $\frac{1}{2}$, in which case $\Delta\eta$ is reduced to $\pm \frac{1}{2}$.

If μ is zero, the contour of constant φ is a parabola. Let u, v be expressed in terms of a parameter σ by the equations

$$u = \frac{\varphi}{\lambda} - \frac{1}{2} \frac{\lambda}{\nu} \sigma^2 \quad (321)$$

$$v = \frac{\lambda}{\nu} \sigma \quad (322)$$

Increments $\Delta u, \Delta v$ in the coordinates are given in terms of the increment $\Delta\sigma$ in parameter by the equations

$$\Delta u = -\frac{\lambda}{\nu} \sigma \Delta\sigma \quad (323)$$

$$\Delta v = +\frac{\lambda}{\nu} \Delta\sigma \quad (324)$$

The coordinates are given by the equations

$$u + \Delta u = u - \frac{1}{2} \frac{\lambda}{\nu} (\Delta\sigma)^2 \quad (325)$$

$$v + \Delta v = v + \frac{\lambda}{\nu} \Delta\sigma \quad (326)$$

which are valid for arbitrary increments and the increments are given by the equations

$$\Delta u = -\frac{\lambda}{\nu} \sigma \Delta\sigma \quad (327)$$

$$\Delta v = +\frac{\lambda}{\nu} \Delta\sigma \quad (328)$$

which are valid for small increments. The increments $\Delta u, \Delta v$ are initially in the direction of the tangent, and the length of step is limited to δ . Then the increment $\Delta\sigma$ is determined with sufficient accuracy by the equation

$$\Delta\sigma = \frac{\delta}{\frac{\Delta v}{\sqrt{(\Delta u)^2 + (\Delta v)^2}} \frac{\lambda}{\nu} - \frac{\Delta u}{\sqrt{(\Delta u)^2 + (\Delta v)^2}} v} \quad (329)$$

unless this would make $|\Delta\sigma|$ more than $\frac{1}{2}$, in which case $\Delta\sigma$ is reduced to $\pm \frac{1}{2}$.

If μ, ν both have the same sign as φ , then the contour of constant φ is an ellipse.

Let u, v be expressed in terms of a parameter ϕ by the equations

$$u = \left(\frac{2\varphi}{\mu}\right)^{\frac{1}{2}} \cos \phi \quad (330)$$

$$v = \left(\frac{2\varphi}{\nu}\right)^{\frac{1}{2}} \sin \phi \quad (331)$$

Increments $\Delta u, \Delta v$ in the coordinates are given in terms of the increment $\Delta\phi$ in angle by the equations

$$\Delta u = - \left(\frac{2\varphi}{\mu}\right)^{\frac{1}{2}} \sin \phi \Delta\phi \quad (332)$$

$$\Delta v = + \left(\frac{2\varphi}{\nu}\right)^{\frac{1}{2}} \cos \phi \Delta\phi \quad (333)$$

The trigonometric functions may be expanded with the aid of the addition formula and may be expressed in terms of the coordinates to give the equations

$$u + \Delta u = u \cos(\Delta\phi) - \left|\frac{\nu}{\mu}\right|^{\frac{1}{2}} v \sin(\Delta\phi) \quad (334)$$

$$v + \Delta v = v \cos(\Delta\phi) + \left|\frac{\mu}{\nu}\right|^{\frac{1}{2}} u \sin(\Delta\phi) \quad (335)$$

which are valid for arbitrary increments, and the equations

$$\Delta u = - \left|\frac{\nu}{\mu}\right|^{\frac{1}{2}} v \Delta\phi \quad (336)$$

$$\Delta v = + \left|\frac{\mu}{\nu}\right|^{\frac{1}{2}} u \Delta\phi \quad (337)$$

which are valid for small increments. The increments $\Delta u, \Delta v$ are initially in the direction of the tangent, which can be derived directly from the gradient ∇f . The stability of the computation is controlled when the increments are subject to the limitation

$$\sqrt{(\Delta u)^2 + (\Delta v)^2} \leq \delta \quad (338)$$

where δ is the maximum length of step. Then the increment $\Delta\phi$ is determined with sufficient accuracy by the equation

$$\Delta\phi = \frac{\delta}{\frac{\Delta v}{\sqrt{(\Delta u)^2 + (\Delta v)^2}} \left|\frac{\mu}{\nu}\right|^{\frac{1}{2}} u - \frac{\Delta u}{\sqrt{(\Delta u)^2 + (\Delta v)^2}} \left|\frac{\nu}{\mu}\right|^{\frac{1}{2}} v} \quad (339)$$

unless this would make $|\Delta\phi|$ more than $\frac{1}{2}$, in which case $\Delta\phi$ is reduced to $\pm \frac{1}{2}$

After the increments $\Delta u, \Delta v$ have been determined, the increments $\Delta x, \Delta y$ are given by the equations

$$\Delta x = i \cdot m \Delta u + i \cdot n \Delta v \quad (340)$$

$$\Delta y = j \cdot m \Delta u + j \cdot n \Delta v \quad (341)$$

unless the initial point is detected ahead on the path of hunting. When the position x_0, y_0 of initiation lies within that zone which is defined by the criteria

$$(x_0 - x)^2 + (y_0 - y)^2 < (\Delta x)^2 + (\Delta y)^2 \quad (342)$$

$$|(x_0 - x)\Delta y - (y_0 - y)\Delta x| < \epsilon \sqrt{(\Delta x)^2 + (\Delta y)^2} \quad (343)$$

where ϵ is a specified tolerance, then the coordinates x, y are replaced in accordance with the substitutions

$$x \rightarrow x_0 \quad y \rightarrow y_0 \quad (344)$$

and the computation is terminated. Otherwise the hunting procedure is replaced by the homing procedure.

During homing the coordinates x, y are replaced in accordance with the substitutions

$$x \rightarrow x' \quad y \rightarrow y' \quad (345)$$

until $\Delta x, \Delta y$ satisfy the criterion

$$|\Delta r| < \epsilon \quad (346)$$

and the displacements of the trial point have been reduced to the level of rounding error.

If the trial point is stepped out of bounds, it is reset on the boundary. If x is outside the boundary it is reset onto the boundary and y is adjusted in linear proportion, or if y is outside the boundary it is reset onto the boundary and x is adjusted in linear proportion. The computation is terminated if the trial point is trapped in a corner. If x already is on a boundary, then partial derivatives with respect to x are set equal to zero, or if y already is on a boundary, then partial derivatives with respect to y are set equal to zero. Intersection of the contour with the boundary is established by iteration.

After Newton-Raphson iteration the function and its derivatives are known at each trial point. Two consecutive trial points are centers of expansion from each of which the contours are extended halfway to the other. Insofar as the extensions are accurate only to second order, they do not quite meet in the middle. The extensions are adjusted in proportion to the cube of the distance from their centers of expansion in order to bring them into coincidence midway between the trial points.

Programming

SUBROUTINE FNCIN (MO, AX, AY, FE, FX, FY, XX, XY, YY)

EXTERNAL SUBROUTINE FOR SURFACE ELEVATION AND DERIVATIVES

The mode of operation MO is 0 for functions, 1 for first derivatives, and 2 for second derivatives. The coordinates x, y are given in the arguments AX, AY. The elevation f is stored in the function FE. The first derivatives $\partial f / \partial x, \partial f / \partial y$ are stored in the functions FX, FY, the second derivatives $\partial^2 f / \partial x^2, \partial^2 f / \partial x \partial y, \partial^2 f / \partial y^2$ are stored in the functions XX, XY, YY.

SUBROUTINE CNTCRV (CD, CE, CF, CA, AX, AY, AZ, NO, FNCTN)

FORTRAN SUBROUTINE FOR CONTOUR CURVE CONSTRUCTION

The step length δ for hunting is given in argument CD, and the tolerance ϵ for homing is given in argument CE. The edges of the plotter field are given in the 4-array CF in the order left, right, upper, lower, and the edges of the contoured area are given in the 4-array CA in the order left, right, upper, lower. The initial coordinates x_0 and y_0 are given in the arguments AX and AY. The elevation of the contour is given in the argument AZ. Datum points on the contour are written on tape file NO. The elevation of the surface is computed by reference to the external subroutine FNCTN.

DISCUSSION

Experiments have been performed on the computer to determine the levels of rounding error in power polynomial approximation. Values of polynomials at nodes and at antinodes and coefficients of polynomials have been compared for a few orders. In all of the experiments the abscissae ranged from -1 to +1.

For equal spacing the values of the Lagrange polynomials are greatest for those polynomials which are unity at abscissae near the middle of the range of abscissae. The values at antinodes are larger than unity but less than the maximum values at endpoints.

For Chebyshev spacing the values of the Lagrange polynomials are greatest for those polynomials which are unity at abscissae near the ends of the range of abscissae. The values at antinodes are everywhere less than unity for all polynomials but the values at endpoints are greater than unity.

The increase in endpoint values with order is illustrated by the following table.

Endpoint Maxima			
Number of Data	Equal Spacing	Chebyshev Spacing	Degree
11	$8.6 \cdot 10^1$	1.27	10
17	$3.5 \cdot 10^3$	1.27	16
21	$4.5 \cdot 10^4$	1.27	20
25	$6.1 \cdot 10^5$	1.27	24

Thus any error in an ordinate near the middle of the range of abscissae is amplified into wild wiggles near the ends of the range when the abscissae are equally spaced.

The effect of rounding error is related to the disparities between the magnitudes of terms and the values of polynomials. The increase in the coefficients of terms of

Lagrange polynomials with order is illustrated by the following table.

Number of Data	Maximum Coefficients		Degree
	Equal Spacing	Chebyshev Spacing	
11	3.2×10^3	2.6×10^2	10
17	1.4×10^6	2.9×10^4	16
21	1.0×10^8	7.4×10^5	20
25	6.8×10^9	1.9×10^7	24

The increase in the coefficients of terms of orthonormal polynomials with order is illustrated by the following table.

Number of Data	Maximum Coefficients		Degree
	Equal Spacing	Chebyshev Spacing	
11	5.3×10^3	5.5×10^2	10
17	2.7×10^6	7.3×10^4	16
21	2.2×10^7	2.0×10^6	20
25	1.4×10^{10}	5.6×10^7	24

The rounding error for abscissae near the ends of the range of abscissae is on the order of the product of the maximum coefficient and 2^{-N} for N -bit arithmetic. Thus expansion into coefficients becomes rapidly more vulnerable to rounding error with increasing order.

If the abscissae are proportional to the cosines of equally spaced angles, then the trigonometric functions of multiples of the angles are orthogonal with respect to summation, and the orthonormal polynomials are proportional to the trigonometric functions. The three-term recurrence for the polynomials is just the addition theorem for the trigonometric functions. Experiments on the CDC 6600 with exact recurrence separate the effects of rounding error in the three-term recurrence from the effects of rounding error in the evaluation of coefficients. That the rounding error in the recurrence increases only very slowly with increasing order is indicated by the following table.

Number of Data	Maximum Error	Degree
11	1.3×10^{-13}	10
17	2.0×10^{-13}	16
21	3.2×10^{-13}	20
25	6.9×10^{-13}	24

where the maximum error is estimated from the residual at the n th abscissa after the $(n + 1)$ th cycle of recurrence.

There seem to be as many subroutines for the FFT as there are users. Only three of the existing subroutines will be considered in the present discussion. A subroutine NLOGN has been published by Robinson¹¹. It is used extensively. It is short and moderately

fast. A subroutine FFT has been published by Brigham¹². It is relatively straightforward. It is short, but quite slow. An improved version CXFFT has been prepared for the present program. One major subroutine is FFTP, which is distributed by IMSL. It is very long but very fast. It is not limited to orders which are a power of 2. It factors any order into prime factors and applies the original formulation of Cooley and Tukey. The ultimate subroutine is HARM, which is distributed by IBM. It is very long but very fast. It is three-dimensional. A set of three subroutines has been prepared for the present program. They can be used in computing loops to perform the same functions as HARM. The computing loops are only half as fast, however, because they do not take every possible shortcut in computation. The merit of each subroutine depends upon the length of program, the storage requirement for data, and the speed of computation. Any increase in speed of computation is achieved at the expense of an increase in storage requirement.

In the IMSL, subroutine ZANLYT applies the Muller method, and subroutine ZCPOLY applies the method of Jenkins and Traub. In the present system, CXRT follows the hunting and homing method. The length of ZANLYT is a few percent less than the length of CXRT, but the length of ZCPOLY is more than three times the lengths of the other two subroutines. Comparative tests with tenth-degree polynomials showed that ZCPOLY takes a few percent less time than the other two subroutines. For roots equally spaced along the real axis, all three subroutines gave full machine accuracy, but for roots equally spaced around a circle of unit radius, ZANLYT and ZCPOLY gave two to three digits less accuracy than CXRT. All three subroutines gave double roots with only half the accuracy of single roots.

CONCLUSION

Although Lagrange and orthonormal polynomials of high degree should not be expanded in coefficient form, they can be evaluated accurately to any order by recurrence relations. Insofar as computations of roots and contours by hunting and homing procedures are applied to unreduced functions, they give high accuracy determinations.

BIBLIOGRAPHY

1. *System 360 Scientific Subroutine Package.*
Programmer's Manual Number 360A-CM-03X, (International Business Machines Corporation, New York, 1970)
2. *IMSL Library 3 Reference Manual.*
(International Mathematical and Statistical Libraries, Inc., Houston, Texas, 1977)
3. *Numerical Mathematical Analysis.*
J. B. Scarborough, (Johns Hopkins Press, Baltimore, Maryland, 1966)
4. *Introduction to Numerical Analysis (Second Edition)*
F. B. Hildebrand, (McGraw-Hill, New York, 1974)
5. *A First Course in Numerical Analysis.*
A. Ralston, (McGraw-Hill, New York, 1974)
6. *Vorlesungen Über Numerisches Rechnen.*
C. Runge, and H. König, (Springer, Berlin, 1924)
7. *An Algorithm for the Machine Calculation of Complex Fourier Series.*
J. W. Cooley, and J. W. Tukey, *Mathematics of Computation*, 19, 297 (1965)
8. *Fast Fourier Transforms for Fun and Profit.*
W. M. Gentleman, and G. Sande, *AFIPS Conference Proceedings*, 29, 563 (1966)
9. *What is the Fast Fourier Transform.*
W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling, Jr., D. E. Nelson, C. M. Rader, P. D. Welch, *Proceedings of the IEEE*, 55, 1664 (1967)
10. *One Week Short Course on Modern Techniques of Filtering.*
H. F. Vanlandingham, (Virginia Polytechnic Institute, Blacksburg, Virginia, 1974)
11. *Multichannel Time Series Analysis with Digital Computer Programs.*
E. A. Robinson, (Holden-Day, San Francisco, 1967)
12. *The Fast Fourier Transform.*
E. O. Brigham, (Prentice Hall, Englewood Cliffs, New Jersey, 1974)
13. *A Generalization of the Fast Fourier Transform.*
J. A. Glassman, *IEEE Transactions on Computers*, C-19, 105 (1970)
14. *The Theory of Equations.*
W. S. Burnside, and A. W. Panton, (S. Chand and Co., Ram Nagar, New Delhi, 1963)
15. *Mathematics Dictionary.*
G. James, and R. C. James, (Van Nostrand Company, Princeton, New Jersey, 1959)
16. *Standard Mathematical Tables.*
S. M. Selby, and B. Girling, Editors, (The Chemical Rubber Company, Cleveland, Ohio, 1965)
17. *Handbook of Mathematical Tables and Formulas.*
R. S. Burington, (Handbook Publishers, Inc. Sandusky, Ohio, 1948)

18. *The "Pegasus" Method for Computing the Root of an Equation.*
M. Dowell, and P. Jarratt, BIT, 12, 503 (1972)
19. *A Machine Method for Solving Polynomial Equations.*
D. H. Lehmer, Journal of the Association for Computing Machinery, 8, 151 (1961)
20. *A Method for Solving Algebraic Equations Using an Automatic Computer.*
D. E. Muller, Mathematical Tables and Other Aids to Computation, 10, 208 (1956)
21. *The Use of Rational Functions in the Iterative Solution of Equations on a Digital Computer.*
P. Jarratt, and D. Nudds, The Computer Journal, 8, 62 (1965)
22. *A Three Stage Variable Shift Iteration for Polynomial Zeros and its Relation to Generalized Rayleigh Iteration.*
M. A. Jenkins, and J. P. Traub, Numerische Mathematik, 14, 252 (1970)
23. *CALCOMP Applications Software. GPCP. A General Purpose Contouring Program. User's Manual.*
(California Computer Products, Inc., Anaheim, California, 1973)
24. *Modeling and Contouring Irregular Surfaces Subject to Constraints.*
J. R. Jancaitis, U. S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia, Report Number ETL-CR-74-19 (January 1975)
25. *Mathematical Techniques for Automated Cartography.*
J. R. Jancaitis, and J. L. Junkins, U. S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia, Report Number ETL-CR-73-4 (February 1973)

APPENDIX A

DISTRIBUTION

DISTRIBUTION

Defense Documentation Center
Cameron Station
Alexandria, VA 22314

Defense Printing Service
Washington Navy Yard
Washington, DC 20374

Library of Congress
Washington, DC 20540
Attn: Gift and Exchange Division

C
D
E
F
G
K
K05H
K05S
K05D
K10
K20
K30
K40
K50
K60
K70
K80
R
U
V
X21
X220

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NSWC/DL TR-3688/	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) POLYNOMIAL ARITHMETIC AND CONTOUR CONSTRUCTION		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) A. V. Hershey		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Weapons Center (K05) Dahlgren, Virginia 22448		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NIF
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE October 1977
		13. NUMBER OF PAGES 61
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) polynomials differentiation approximation integration roots contours analysis programming computation		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Power polynomials and trigonometric polynomials are used for the approximation of general functions. Analysis and documentation are given for a set of subroutines which perform polynomial arithmetic, find roots of a function, or construct contours of constant function.		

DD FORM 1473
1 JAN 73EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)