A COMPUTER PROGRAM

TO CALCULATE ZEROES, EXTREMA, AND INTERVAL INTEGRALS

FOR THE ASSOCIATED LEGENDRE FUNCTIONS

Mary H. Payne

N 73-19
Contract No. NAS 5-21787
April 1973

ANALYTICAL MECHANICS ASSOCIATES, INC.
60 JERICHO TURNPIKE
JERICHO, N. Y. 11753
A computer program is described for the calculation of the zeroes of the associated Legendre functions, \( P_{nm} \), and their derivatives, for the calculation of the extrema of \( P_{nm} \) and also the integral between pairs of successive zeroes. The program has been run for all \( n,m \) from \((0,0)\) to \((20,20)\) and selected cases beyond that for \( n \) up to 40. Up to \((20,20)\), the program (written in double precision) retains nearly full accuracy, and indications are that up to \((40,40)\) there is still sufficient precision (4-5 decimal digits for a 54-bit mantissa) for estimation of various bounds and errors involved in geopotential modeling, the purpose for which the program was written.
TABLE OF CONTENTS

Section | Page
---|---
I | Introduction ........................................ 1
II | Input, Initialization, and Output .................... 5
III | Calculation of the Coefficients ....................... 12
IV | The Graeffe Root Squaring Method .................... 17
V | Calculation of the Extrema and Interval Integrals .... 22
References .............................................. 24
Appendix A - Listing of the Main Program .............. 25
Appendix B - Listing of the Coefficient Subroutines .... 32
Appendix C - Listing of Subroutine GRAEFF .............. 36
Appendix D - Listing of GAUSS and the Function Evaluation... 38

Subroutines
I. INTRODUCTION

This report describes a computer program for the calculation of data on the associated Legendre functions of the first kind. These data are useful in the estimation of bounds for truncation error in the spherical harmonic expansion of the geopotential, and also for the estimation of bounds on the coefficients in such an expansion. The application of the results of this calculation to these estimation problems is discussed in References 1 and 2. The accuracy requirements for estimation purposes are not very stringent, a few significant digits should be adequate. The program can operate up to degree and order 100; this limitation is imposed by the dimensioning of various arrays and would be easy to change. The program has been run from 0, 0 through 20, 20 and appears to have accuracy of 8 or 9 significant digits for this range of degrees and orders. Runs for degrees 30 and 40 with order zero indicate that one can probably run it to 40, 40 with an accuracy of four significant digits. The accuracy can probably be significantly increased by implementing one or another of the suggested modifications to the subroutine for finding roots.

In constructing the program, two formulations for the associated Legendre functions were implemented. In one, \( z = \cos \theta \), where \( \theta \) is the polar angle of spherical coordinates, is the independent variable. In the other, \( x = \sin^2 \theta/2 \) is the independent variable. These two variables are related by

\[
z = 1 - 2x
\]

and the corresponding associated Legendre functions are given by

\[
P_{nm}(z) = \begin{cases} 
\frac{m/2}{(1-z^2)} & \text{polynomial of degree } (n-m)/2 \ln z^2 \\
& \text{for } n-m \text{ even} \\
\frac{m!/2}{(1-z^2)} & \text{z \cdot polynomial of degree } (n-m-1)/2 \\
& \ln z^2 \text{ for } n-m \text{ odd}
\end{cases}
\]
\[ P_{nm}(x) = [x(1-x)]^{m/2} \text{ polynomial of degree } (n-m) \text{ in } x \quad (1.2) \]

From Eqs. (1.2), it would at first appear that the calculation must accommodate three cases; actually there are six cases, since the extrema of \( P_{nm} \) are found from the zeroes of the derivative of \( P_{nm} \) with respect to its independent variable, and the derivative must be handled in different ways for \( m = 0 \) and \( m > 0 \). In addition, there are seven special cases that must be handled separately (e.g., one of these is \( P_{00} = \text{constant} \), for which there are no zeroes, extrema, or interval integrals.

The "interval integrals," mentioned above and in the title, are the integrals between successive zeroes of \( P_{nm} \), with respect to its independent variable. From Eq. (1.1)

\[ dz = -2dx \]

\[ x = 0 \quad \text{corresponds to } z = 1 \quad (1.3) \]

\[ x = 1 \quad \text{corresponds to } z = -1 \]

The final print-out of the full set of calculations lists the zeroes of \( P_{nm} \) and its derivative in adjacent columns and in increasing order relative to the variable used. The associated extrema and interval integrals appear in the third and fourth columns. Because of the correspondence of the endpoints of the interval of definition of \( P_{nm} \) indicated in Eq. (1.3), the results read from top to bottom in the \( z \)-formulation correspond to those read from bottom to top in the \( x \)-formulation. The magnitudes of the zeroes are related by Eq. (1.1). The extrema should be identical. The interval integrals are related by a factor \( 2 \) which comes from Eq. (1.3) \((\text{not } -2, \text{ since the minus sign is compensated by an interchange in limits of integration as one transforms from one formulation to the other})\).
In any particular run of the program, one formulation or the other is selected by an input switch. The two formulations were implemented because it seemed likely that they might well complement one another and, as we shall see, this is indeed the case. In addition, check-out of the program was greatly facilitated.

Several output options are available through another input switch. The general flow of the main program is as follows:

1. Input and initialization, including selection of the formulation to be used.

2. Calculate and print the coefficients of the polynomial parts of $P_{nm}$ and $P'_{nm}$.
   Option: Terminate the program at this point and go to more input at 1

3. Calculate the zeroes of $P_{nm}$ and $P'_{nm}$.
   Option: Print these zeroes and go to 1
   Option: No print; bypass 4 and go to 5

4. Calculate extrema of $P_{nm}$ by evaluation at the zeroes of $P'_{nm}$.
   Option: Print zeroes and extrema and go to 1

5. Calculate the interval integrals using the zeroes of $P_{nm}$.
   Option (only if 4 is bypassed):
   Print zeroes and interval integral and go to 1

6. Print zeroes of $P_{nm}$ and $P'_{nm}$, extrema of $P_{nm}$, and interval integrals in tabular form.

7. Go to 1 (with exit if no more data available).

Listings of the main program and all the subroutines are provided in the appendices. The remaining sections of the report describe the steps listed above.
in greater detail, with references to line and statement numbers appearing in the listings.

Section II contains a list of the input parameters and a discussion of their various functions. The branching involved in the six cases mentioned earlier is also described in Section II. This is followed, in Section III, by the recursion formulas used to obtain the coefficients of the polynomial parts of $P_{nm}$ and $P'_{nm}$, and a discussion of the subroutines in which they are implemented.

The zeroes of the polynomial parts of $P_{nm}$ and $P'_{nm}$ are calculated by Graeffe's root squaring method, implemented in subroutine GRAEFF. Some interesting problems were encountered, and these problems and their resolution are described in Section IV. This subroutine presently limits the accuracy of the program, and hence the size of degree and order to which it can be applied. The results of a few test runs are presented, and several possibilities for improvement of the accuracy are discussed briefly.

The extrema of $P_{nm}$ are found by direct substitution of the zeroes of $P'_{nm}$ into $P_{nm}$ and this is accomplished by subroutines FUNCT and EVAL, which are straightforward and easily followed from the listing. The interval integrals are calculated by Gaussian quadrature in subroutine GAUSS, which is also straightforward. A few comments on these three subroutines appear in Section V.
II. INPUT, INITIALIZATION, AND OUTPUT

The major portion of the Main Program is taken up by input, initialization, and output. The calculations are all done in subroutines, called by the Main Program. A listing of the Main Program is given in Appendix A. The references to symbols, statement numbers, and line numbers in this section apply to the Main Program. The output section is located between Statements 600 and 800. It follows the flow indicated in the Introduction with the indicated options implemented in Lines 20800, 24600, 24800, 31300, and 31800.

A block of 20 integers, IN(20), is reserved for input parameters. A block of 100 integers, NUM(100), is also used for input under certain conditions. These blocks are in NAMELISTS IN1 and IN2. The output of the program is carried in the arrays

- \( C(101) \): coefficients for the polynomial part of \( P_{nm} \)
- \( CP(102) \): coefficients for the polynomial part of \( P'_{nm} \)
- \( Z(102) \): zeroes of \( P_{nm} \)
- \( ZP(101) \): zeroes of \( P'_{nm} \)
- \( EX(101) \): extrema of \( P_{nm} \)
- \( FIN(101) \): interval integrals

The first part of the initialization consists of identifying the input block, IN, with mnemonic names as follows:

- \( IN(1) = IND = 0 \) independent variable is \( z = \cos \theta \)
- \( 1 \) independent variable is \( x = \sin^2 \theta/2 \)
IN(2) = NOPT = 0  
a range of degrees equally spaced is desired;  
see IN(7), IN(8), and IN(9)  

> 0  
a list of NOPT degrees to be read into the  
block NUM, using NAMELIST IN2 for the input  

IN(3) = MOPT = -1  
process all orders consistent with each specified  
degree  

≥ 0  
process only order MOPT for the specified  
degrees  

IN(4) = INC: Print Options:  
0  
compute and print only C and CP  
1  
compute and print only C, CP, Z, and ZP  
2  
compute and print only C, CP, Z, ZP,  
and FIN  
3  
compute and print only C, CP, Z, ZP,  
and EX  
4  
compute and print C, CP, Z, ZP, EX,  
and FIN  

IN(5) = ITMAX  
maximum number of iterations allowed in  
GRAEFF for the calculation of Z and ZP  

IN(6) = NI  
use the zeroes and weight factors for  
P_{(Ni+1), 0}  
in GAUSS  

IN(7) = IMIN  
process a range of INX  
IN(8) = ISTEP  
degrees starting at IMIN and  
spaced at ISTEP intervals  
IN(9) = INX  

IN(10) = NTOL  
convergence criterion  

IN(11)  
SCALE = IN(11)**IN(12)  

IN(12)  
See Section IV on  
GRAEFF  

IN(13)  
TOL = 10**IN(13)  
See Section V on GAUSS  

IN(14) - IN(20)  
not used at present
A single error return is provided for several input conditions which might result in poor functioning of the program.

The second part of the initialization involves setting up the array NUM(I) in such a way that NUM(I) is the $I^{th}$ degree to be processed, with a total of INX degrees. This information goes into the main DO loop starting at Statement 44; DO 1000, I=1,INX followed by N1=NUM(I), where N1 is the degree currently being processed. For NOPT>$0$, NUM is filled from the second READ statement (Line 4400). The DO loops to 6, 8, and 10 rearrange the degrees read and restore them to NUM so that

$$\text{NUM}(I_{1}) > \text{NUM}(I_{2}) \text{ if and only if } I_{1} > I_{2}$$

This means that the degrees may be in any order in the data statement. For NOPT=$0$, Statements 20 and 30 construct NUM so that

$$\begin{align*}
\text{NUM}(1) &= \text{IMIN} \\
\text{NUM}(I) &= \text{NUM}(I-1) + \text{ISTEP} \\
\text{NUM}(\text{INX}) &= \text{IMIN} + \text{ISTEP} \times (\text{INX}-1)
\end{align*}$$

Note that the dimensions of 101 and 102 for C and CP imply that the degree N1 must not exceed 100. For direct input (NOPT>$0$) no test is made, but for NOPT=$0$, NUM(I) is not permitted to exceed 100 (see DO loop 30).

The third part of the initialization calls subroutine FNORM0 (Line 8600); this step, together with the call to FNORM in Statement 58, is better discussed in the next section dealing with the calculation of the coefficients in the polynomial parts of $P_{nm}$ and $P_{nm}'$.

The fourth part of the initialization sets up IMX and the array MUM, which do for orders what INX and NUM do for degrees. If MOPT>$0$, MUM(1)=MOPT and IMX, the number of orders to be processed is set to 1. If MOPT<$0$,
MUM and IMX are defined (inside the DO 1000 I=1,INX loop) to include all orders consistent with the current value of N1 by the DO 45 loop.

The final step in the initialization is perhaps the most complex; it starts at Line 10300 near the beginning of the DO 999 loop (which processes all orders specified for the current N1 value) and extends to Line 20400, just before CALL COEF. This step sets up the branching procedure for the six cases mentioned in the Introduction. A basic reason for the large number of cases was the desire to make use of the symmetry involved in the \( z = \cos \theta \) formulation to reduce computation time. In this formulation, the polynomial parts of \( P_{nm} \) and \( P'_{nm} \) are polynomials in \( \cos^2 \theta \), so that only their positive zeroes need be calculated and, from these, only the corresponding extrema and interval integrals need be calculated. The complete set is then obtained from multiplication of this set by + or -1. Further, there is little point in making GRAEFF find a zero root which is readily found by factoring.

The parameter KIND identifies the six cases, the special case for each, and the differences in their treatment. The various parameters listed with KIND are as follows:

\[
\begin{align*}
NR &= \text{number of zeroes of } P_{nm} \text{ to be found by GRAEFF} \\
NRP &= \text{number of zeroes of } P'_{nm} \text{ to be found by GRAEFF} \\
NC &= \text{number of coefficients in the polynomial part of } P_{nm} \\
NCP &= \text{number of coefficients in the polynomial part of } P'_{nm} \\
NP &= \text{number of zeroes of } P_{nm}, \text{ including endpoints and zero, if present} \\
NPP &= \text{number of zeroes of } P'_{nm}, \text{ including endpoints and zero, if present}
\end{align*}
\]

Parameters starting with K are used in the rearranging and augmentation processes listed for each value of KIND below.
The case \( n = m = 0 \) is very special; there are no roots, extrema, or interval integrals. A special printout is provided as soon as this can be detected, Line number 9200.

For \( m > 1 \), \( P'_{\text{nm}} \) has zeroes at \( \pm 1 \) in the \( \cos \theta \) formulation and at 0 and 1 in the \( \sin^2 \theta/2 \) formulation; these points correspond to zeroes of \( P_{\text{nm}} \) rather than to extrema, at least for the purposes of this report. These zeroes of \( P'_{\text{nm}} \) are ignored in the program and output.

For \( \text{IND} = 0 \) (\( \cos \theta \) formulation), most of the zeroes of \( P_{\text{nm}} \) and \( P'_{\text{nm}} \) are obtained by taking \( \pm \) the square root of the output of GRAEFF. This formulation consists of four cases, as follows:

**KIND = 1:** \( m = 0 \), \( n \) even, special case is \( n = 2 \)
- Set of zeroes of \( P'_{\text{nm}} \) must be augmented by \( ZP = 0 \)
- Extrema corresponding to zeroes of \( P'_{\text{nm}} \) are symmetric about \( Z = 0 \)
- Interval integrals are also symmetric about \( Z = 0 \)
- Set of interval integrals must be augmented by \( \int_{-1}^{1} \) first zero \( \int_{-1}^{1} \) last zero

**KIND = 2:** \( m = 0 \), \( n \) odd, special case is \( n = 1 \)
- Set of zeroes of \( P_{\text{nm}} \) must be augmented by \( Z = 0 \)
- Extrema and interval integrals are antisymmetric about \( Z = 0 \)
- Set of interval integrals must be augmented by endpoint integrals

**KIND = 3:** \( m > 0 \), \( n - m \) even, special case is \( n = m \)
- Set of zeroes for \( P_{\text{nm}} \) must be augmented by \( Z = \pm 1 \)
- Set of zeroes for \( P'_{\text{nm}} \) must be augmented by \( ZP = 0 \)
- Extrema and interval integrals are symmetric about \( Z = 0 \)
KIND = 4: \( m > 0, n - m \) odd, special case is \( n = m + 1 \)

- set of zeroes for \( P_{nm} \) must be augmented by \( Z = 0, \pm 1 \)
- extrema and interval integrals are antisymmetric about \( Z = 0 \)

For \( \text{IND} = 1 \) (\( \sin^2 \theta/2 \) formulation), subroutine GRAEFF gives all zeroes for \( P_{nm} \) and \( P'_{nm} \) except at the endpoints. The parity of \( n - m \) is not significant and we do not exploit the symmetry properties of \( P_{nm} \) and \( P'_{nm} \) about the point \( x = \frac{1}{2} \).

KIND = 5: \( m = 0 \), special case is \( n = 1 \)

- output of GRAEFF is used unchanged for zeroes and extrema
- set of interval integrals must be augmented by the endpoint integrals

KIND = 6: \( m > 0 \), special case is \( n = m \)

- set of zeroes of \( P_{nm} \) must be augmented by \( x = 0, 1 \)

Although not properly a part of initialization, we mention here that in Statements 140-220 the positive square roots of the output of GRAEFF are taken (for \( \text{KIND} = 1, 2, 3, 4 \)) and \( Z = 0, ZP = 0 \) are introduced where necessary. The remaining rearrangement of all roots, extrema, and interval integrals for output purposes is carried out in Statements 535-600.

The special cases, identified by \( \text{ISP} = 1 \), together with \( \text{KIND} \), are given special treatment in Statements 800-910.

A word should be said about values to be used for some of the input parameters. The principal reason for including \( \text{ITMAX} \) in GRAEFF was to avoid being trapped in a loop, in case convergence fails. The test cases run indicate that a reasonable value for \( \text{ITMAX} \) is
ITMAX = 20

since iterations in excess of 20 appear to have no significance. NTOL and SCALE are defined by IN(10), IN(11), and IN(12). The values used in testing the program were

\[
\begin{align*}
\text{IN}(10) &= \text{NTOL} = 14 \\
\text{IN}(11) &= 10 \\
\text{IN}(12) &= 1 \\
\end{align*}
\]

implying \(\text{SCALE} = 10\)

Utilization of a hexadecimal basis for SCALE with proper adjustment of NTOL might have computational advantages on the IBM 360.

In all the tests carried out, we set

\[\text{NI} = 9\]

Some experimentation might show that a lower value could be used, particularly for small values of \(N\), without sacrificing accuracy. Since there are \(\text{NI} + 1\) evaluations of the integrand for each entry into GAUSS, some saving of machine time could be achieved if lower values of \(\text{NI}\) yield acceptable results. In the tests on the program, we set

\[\text{IN}(13) = -12\]

implying \(\text{TOL} = 10^{-12}\)

This parameter is probably not significant for the analysis of \(P_{nm}\); it was introduced so that GAUSS would be a self-contained subroutine, available for any program in which a Gaussian quadrature would be of use.
III. CALCULATION OF THE COEFFICIENTS

The coefficients for the polynomial parts of $P_{nm}$ and $P'_{nm}$ are calculated in three steps for the $z = \cos \theta$ formulation, using subroutines FNORM0, FNORM, and COEF (listings given in Appendix B). For the $x = \sin^2 \theta/2$ formulation, only FNORM and COEF are required. We start by writing $P_{nm}$ in the two formulations as

$$P_{nm}(z) = (1 - z^2)^{m/2} \sum_{k=0}^{(n-m)/2} C_{nm}(k) z^{(n-m-2k)} ; \text{ IND } = 0$$

$$P_{nm}(x) = (x(1-x))^{m/2} \sum_{k=0}^{n-m} \bar{C}_{nm}(k) x^k ; \text{ IND } = 1$$

with

$$C_{nm}(0) = A_{nm} , \quad \bar{C}_{nm}(0) = \bar{A}_{nm}$$

$$C_{nm}(k+1) = -\frac{(n-m-2k)(n-m-2k-1)}{2(k+1)(2n-2k-1)} C_{nm}(k), \quad k = 1, 2, \ldots, \left[\frac{n-m-2}{2}\right]$$

$$\bar{C}_{nm}(k+1) = -\frac{(n+m+k+1)(n-m-k)}{(k+1)(m+k+1)} \bar{C}_{nm}(k), \quad k = 1, 2, \ldots, (n-m)$$

and

$$A_{nm} = \frac{(2n)!}{2^n \cdot n!} \sqrt{\frac{(2-\delta_{m0})(2n+1)}{(n-m)!(n+m)!}}$$

$$\bar{A}_{nm} = \frac{1}{m!} \sqrt{\frac{(2-\delta_{m0})(2n+1)(n+m)!}{(n-m)!}}$$

$$\delta_{m0} = \begin{cases} 1 & m = 0 \\ 0 & m > 0 \end{cases} , \quad [p] = \text{largest integer } \leq p$$
The $C_{nm}(k)$ appearing here are, of course, not geopotential coefficients. The factors $A_{nm}$ and $\overline{A}_{nm}$ include the factor

$$\sqrt{(2-\delta_{m0})(2n+1) \frac{(n-m)!}{(n+m)!}}$$ \hspace{1cm} (3.4)

which converts conventional associated Legendre functions into the fully normalized form used by geodesists. The derivatives for the two formulations of $P_{nm}$ take the form, for $m > 0$:

$$\frac{dP_{nm}}{dz} = (1-z^2)^{(m/2)-1} \left[ \sum_{k=0}^{n-m+1} C_{nm}(k) z^{(n-m+1-2k)} \right]$$ \hspace{1cm} (3.5)

$$\frac{dP_{nm}}{dx} = (x(1-x))^{(m/2)-1} \sum_{k=0}^{(m+2k)(n^2+n) - m(m+k)(m+k-1)} C_{nm}(k) x^k$$

with

$$C_{nm}(0) = B_{nm} = -n A_{nm}$$

$$\overline{C}_{nm}(0) = \overline{B}_{nm} = \frac{m}{2} \overline{A}_{nm}$$

$$C_{nm}(k) = C_{nm}(k) - (n^2 - m^2) C_{n-1,m}(k-1)/(n(2n-1))$$

$$k = 1, 2, \ldots, \left[ \frac{n-m+1}{2} \right]$$ \hspace{1cm} (3.6)

$$\overline{C}_{nm}(k) = -\frac{\overline{C}_{nm}(k-1)}{mk(m+k)} \left[ (m+2k)(n^2+n) - m(m+k)(m+k-1) \right]$$

$$k = 1, 2, \ldots, (n-m+1)$$

Verification of these formulas is tedious, but straightforward. For $m = 0$, things are simpler:
\[
\frac{dP}{dz}_{n0} = \sum_{k=0}^{n-m-1} \frac{[n-m-2k]}{2} CP_{n0}(k) z^{n-m-2k}
\]

\[
\frac{dP}{dx}_{n0} = \sum_{k=0}^{n-m-1} \frac{n-m-2k}{n} CP_{n0}(k) z^k
\]

with

\[
C_{n0}(0) = B_{n0} = n A_{n0}
\]

\[
\overline{C}_{n0}(0) = \overline{B}_{n0} = \overline{A}_{n0} = \sqrt{2n+1}
\]

\[
CP_{n0}(k) = \frac{n-2k}{n} C_{n0}(k)
\]

\[
\overline{CP}_{n0}(k) = (k+1) C_{n0}(k+1)
\]

The first step in the calculation of the C's and CP's for the \( z = \cos \theta \) formulation is to calculate \( B_{n0} \). This is done in subroutine FNORM0 by setting

\[
B_{00} = 0
\]

\[
B_{10} = \sqrt{3}
\]

and using the recursion relationship

\[
B_{k0} = \frac{\sqrt{4k^2 - 1}}{k} B_{k-1, 0}
\]

FNORM0 is called only once during a run, and computes and stores \( B_{n0} \) up to and including the maximum value of \( n \) to be processed. \( \overline{B}_{n0} \) is so simple that it is calculated when needed in subroutine FNORM.
The second step in calculating the coefficients is carried out in subroutine FNORM, which computes $A_{nm}$, $B_{nm}$ or $\bar{A}_{nm}$, $\bar{B}_{nm}$, depending upon the formulation selected. For $m = 0$, of course, the calculation is trivial. For $m > 0$, the following recursion formulas are implemented in FNORM.

\[
A_{nm} = \sqrt{\frac{n-m+1}{n+m}} A_{n,m-1} \quad m > 1
\]

\[
A_{n1} = \sqrt{\frac{2n}{n+1}} A_{n0} \quad ; \quad A_{n0} = B_{n0}/n
\]

\[
B_{nm} = -n A_{nm}
\]

\[
\bar{A}_{n,m+1} = \frac{\sqrt{(n+m+1)(n-m)}}{m+1} \bar{A}_{nm} \quad m > 1
\]

\[
\bar{A}_{n1} = \sqrt{\frac{2(2n+1)(n^2+n)}{2}} \quad ; \quad \bar{A}_{n0} = \sqrt{2n+1}
\]

\[
\bar{B}_{nm} = \frac{m}{2} \bar{A}_{nm}
\]

The factor $(2 - \delta_{m0})$ in $A_{nm}$ and $\bar{A}_{nm}$ necessitates starting the recursion from $A_{n1}$ and $\bar{A}_{n1}$, rather than from $A_{n0}$ and $\bar{A}_{n0}$.

Finally, subroutine COEF, using the output of FNORM, implements the recursion formulas given in Eqs. (3.2), (3.6), and (3.8) to obtain the $C$'s and $CP$'s or $\bar{C}$'s and $\bar{CP}$'s, depending upon the formulation desired.

No study of the growth of error with the number of passes through these recursion formulas has been made. It has been noted by S. Pines (Ref. 3) that care must be exercised in the use of recursion formulas. It is possible that inaccuracies in the coefficients are responsible for the lack of precision in the
determination of the zeroes of $P_{nm}$ and $P'_{nm}$, although the way in which this occurs suggests that other effects dominate any inaccuracy in the coefficients. This matter is discussed further in the next section.

Note that slight variations appear between the formulas given in this section and their implementation in subroutines FNORM0, FNORM, and COEF, because DO loops cannot start from zero.
IV. THE GRAEFFE ROOT SQUARING METHOD

The subroutine for finding the zeroes of $P_n$ and $P'_n$ is GRAEFF (AA, N, Z, SCALE, NTOL, ITMAX, IND). The listing is in Appendix C. It calculates the zeroes of a polynomial of degree $N-1$, with a coefficient array AA of $M$ elements, associated with increasing or decreasing powers of the variable according as IND is 1 or 0. The Graeffe root squaring method is implemented in less than full generality: An implicit assumption is that the roots are real, positive, and distinct, a condition fulfilled by the polynomial parts of $P_n$ and $P'_n$, if $z$ is factored from those of odd degree in the cos $\theta$ formulation. The zeroes are stored in the array Z. The remaining entries in the calling sequence, SCALE, NTOL, and ITMAX will be discussed later.

First we outline the basic idea of the method; an excellent discussion is given by Lanczos (Ref. 4). We suppose that

$$x_1 > x_2 > \ldots > x_n > 0$$

are the zeroes in descending order of magnitude of the polynomial

$$\sum_{k=0}^{n} A_k x^k$$

Then

$$y_1 = x_1^2 > y_2 = x_2^2 > \ldots > y_n = x_n^2$$

are the zeroes in descending order of magnitude of the polynomial

$$\sum_{i=0}^{n} B_i y^i$$
where

\[
\begin{align*}
B_0 &= A_0^2 \\
B_1 &= A_1^2 + 2A_0 A_2 \\
B_2 &= A_2^2 - 2A_1 A_3 + 2A_0 A_4 \\
&\vdots \\
B_{n-1} &= (-1)^{n-1}(A_{n-1}^2 - 2A_{n-2} A_n) \\
B_n &= (-1)^n A_n^2
\end{align*}
\]  \hspace{1cm} (4.5)

As this process is iterated one obtains, on the \(K\)\(^{th}\) iterate, a polynomial with coefficients \(B^{[K]}\) and zeroes

\[
x_1^{(2^K)} > x_2^{(2^K)} > \ldots > x_n^{(2^K)}
\]  \hspace{1cm} (4.6)

such that the ratio of the \(i\)\(^{th}\) to the \((i-1)\)st zero becomes arbitrarily small for all \(i\) and sufficiently large \(K\). Using this fact, and the relationship between the coefficients \(B^{[K]}\) and sums of products of roots, it is easy to verify that

\[
\left(\frac{B^{[K]}}{B_i^{[K]}}\right)^{2^{-K}}
\]  \hspace{1cm} (4.7)

(or its reciprocal, depending on \(\text{IND}\)) converges to the zeroes of the given polynomial. As the iterates of the coefficients \(B_i\) are constructed, it becomes apparent that they become more and more widely separated in order of magnitude. Numerically, the method terminates when the separation of the coefficients becomes such that

\[
B_i^{[K+1]} = \left[\frac{B_i^{[K]}}{B_i}\right]^2 (-1)^i
\]  \hspace{1cm} (4.8)
because the remaining cross-product terms [see Eq. (4.5)] are beyond the word length of $[B_i^{(K)}]^2$. If the word length for the calculation is $L$ decimal digits, the criterion for termination is thus

$$2 B_{i+j}^{(K)} B_{i-j}^{(K)} < [B_i^{(K)}]^2 \cdot 10^{-L}$$

(4.9)

for all relevant values of $j$. This is essentially the criterion used in GRAEFF, and $L$ is given the name NTOL, an input quantity.

In this subroutine, the terms contributing to each $B_i$ are added on one at a time from left to right, as shown in Eq. (4.5). An array $K1(I)$ is defined to give the number of terms making up $B_i^{(K)}$ from the previous set of coefficients $B_i^{(K-1)}$. When the last term in this sum is beyond the word length of the $K1(I)-1$ terms already summed, $K1(I)$ is diminished by 1. When $K1(I) = 0$ for all $I$, the iteration terminates.

Since both round-off error and machine time can be expected to increase with the number of iterations, ITMAX, another input quantity, is also allowed to terminate the iteration, in which case the calculation of the zeroes proceeds on the basis of the $B_i$'s so far obtained. In this case, a message is written together with the array $K1(I)$, which indicates which of the $B_i$'s have failed to converge. An error message is written if any zero is negative, and the calculation proceeds with the absolute value of such a zero. A standard print states the number of iterations used on the current entry to the subroutine.

A significant problem in the implementation of Graeffe's method arose because the iterates of the coefficients grow very rapidly, and soon produce overflows. To avoid this problem, the parameter SCALE is used to convert all coefficients and their iterates to values less than SCALE and greater than or equal to 1. Then
additional arrays are introduced to carry the powers associated with the coefficients; i.e., for each $I$

$$1 \leq B(I) < \text{SCALE} \quad \text{NEXB}(I) = \text{power} \quad (4.10)$$

and the actual corresponding coefficient is given by

$$B(I) \times \text{SCALE}^{\text{NEXB}(I)} \quad (4.11)$$

The program has been run (in double precision) using $\text{SCALE} = 10$, $\text{NTOL} = 14$ for all orders and degrees of $P_{nm}$ from $0,0$ to $20,20$, on the DEC KA10, which has a mantissa of 54 bits. The indications are that the zeroes near zero hold 15 decimal digit precision for polynomials at least up to degree 20. The polynomial parts of $P_{nm}$ and $P'_{nm}$ have their largest zeroes near unity and for such a polynomial part of degree 10, the largest zeroes have 10-11 digit precision; for one of degree 20, the precision of the largest zeroes is only three or four digits. These data on precision were obtained by comparison of the zeroes of $P_{n0}$ tabulated by the National Bureau of Standards (Ref. 5), and by comparison of the output from the two formulations. In fact, the availability of the two formulations probably enables one to go to $40,40$ with 7-8 digits of precision. This is so because the small zeroes of the $\sin^2 \theta/2$ formulation can be transformed into the zeroes near unity of the $\cos \theta$ formulation, while the small zeroes of the $\cos \theta$ formulation are transformed into those near $x = \frac{1}{2}$ in the $\sin^2 \theta/2$ formulation. Thus, using the "good" zeroes from each of the two formulations and the symmetry properties, a set of zeroes good to 12 or 13 digits may easily be constructed for $20,20$. Selected cases up to $40,40$ have been run. The user of the program is cautioned that an ITMAX of 20 will be exceeded and that overflows may occur for $\text{IND} = 1$, if $n-m$ is appreciably greater than 20. Both conditions may be ignored since they affect only those zeroes for which significance is already lost; they must be found by running $\text{IND} = 0$. 

20
One would like, of course, to account for the lack of precision of the "large" zeroes and, if possible, improve the accuracy. An immediate thought might be that errors in the input coefficients (recall that these are computed by recursion formulas) are the primary cause. This does not seem likely, however, because for \( \text{IND} = 0 \) (cos \( \theta \) formulation) the most important coefficients for large zeroes are those which start the recursion calculation. Still, all coefficients do ultimately enter the iterates of \( B^{[K]} \) for the large zeroes, and the possibility cannot be eliminated without further testing. Another thought is that the round-off error produced by scaling is the culprit. This possibility has been tested and round-off, while present, is several orders of magnitude less than the discrepancies observed. The most plausible, but as yet untested, explanation is loss of significance in the subtractions implied by Eq. (4.5). It is possible that combining these terms starting with the smallest and ending with the largest (in magnitude) might help, at the expense of machine time spent in the sort. Probably the most practical method to improve the situation is to use the output of GRAEFF as the initial guess to a Newton procedure.
V. CALCULATION OF THE EXTREMA AND INTERVAL INTEGRALS

The subroutines for these calculations are straightforward and require little comment. To obtain the extrema, \( P_{nm} \) must be evaluated at the zeroes of \( P'_{nm} \). This calculation is carried out by subroutines FUNCT (X, F, L) and EVAL (A, N, M, X, P, IND) (listed in Appendix D). Subroutine EVAL simply evaluates a polynomial of degree \( N-1 \) with a coefficient array \( A \) (associated with ascending or descending powers of the variable, according as \( \text{IND} = 1 \) or \( 0 \)) at an array \( X \) of \( M \) points. These evaluations are returned in the array \( P \). Subroutine FUNCT, which accepts the array of \( L \) evaluation points \( X \), supplies whichever of the factors \((1-z^2)^{\frac{m}{2}}, z(1-z^2)^{\frac{m}{2}}, \) or \([x(1-x)]^{\frac{m}{2}}\) is applicable and returns the values of \( P_{nm} \) in the array \( F \). It appears that the extrema are relatively insensitive to errors in the zeroes of \( P'_{nm} \). This supports the opinion given in the previous section that errors in the coefficients \( C \) and \( CP \) are relatively unimportant.

To obtain the interval integrals, subroutine GAUSS(A, B, NI, ABINT, TOL) implements the Gaussian quadrature procedure, which is well described by Lanczos (Ref. 4). Two input options are provided: The zeroes and weight factors for \( P_k^0 \), \( k = 2, 3, \ldots, 10 \), are stored in data statements. The parameter \( NI \) selects those for \( k = NI+1 \). A parameter \( TOL \) is introduced to avoid difficulties with small differences: If the limits \( A, B \) of the integral to be evaluated satisfy

\[
|A-B| < TOL \tag{5.1}
\]

the subroutine returns the value zero in the output parameter \( ABINT \), and prints out a message to this effect. Subroutine GAUSS calls FUNCT and then EVAL to evaluate the integrand where necessary.
The interval integrals are quite sensitive to inaccuracies in the zeroes of $P_{nm}$, as one might expect, since these inaccuracies will destroy the non-negative character of the integrand. However, it is felt that, using both formulations and symmetry considerations, the interval integrals have 8-10 digits of accuracy up to 20, 20 and will probably retain 3-4 digits perhaps up to 40, 40, which should be adequate for the estimation purposes discussed in Reference 2.

It should be mentioned that the program does not implement the construction of a single table for the zeroes, extrema, and interval integrals utilizing the output of the two formulations in such a way as to maximize accuracy. The necessary additions to the program would be easy to insert. Time, however, did not permit sufficiently detailed examination of the output to determine the points at which the switch between formulations should be made. These switch points are very likely functions of $m$ and $n$, though perhaps sensitive only to the difference $n-m$. 


REFERENCES


GO TO NEXT CASE!
GO TO 5
15 INX=INX+1
GO TO 40
20 INX=IN(7)
ISTEP=IN(8)
IF (ISTEP,LT,0) GO TO 2000
IF (ISTEP,LE,0) GO TO 2000
IF (INX,LE,0) GO TO 2000
DO 30 I=1, INX
II=I-1
NUM(I)=MIN(I)+I*ISTEP
IF (NUM(I),LE,100) GO TO 30
WRITE (6,25) I, NUM(I), II
25 FORMAT (I, 16, I6, IGREATERTHAN 100)
CONTINUE
30 CONTINUE
40 NUM(I)=MOPT
IMX=1
N=NUM(INX)
IF (IND,EQ,1) GO TO 44
N1=N+1
CALL FNORM0(N, IND)
WRITE (6, 42) N, (NUM(I), I=1, N1)
42 FORMAT (I, NORMALIZATION FACTORS FOR P00 TO PN0
WITH N = !, 13, ! ARE!X/(10X1P3025, 14)7)
DO 100 I=1, INX
DO 1000 I=1, INX
9100 N1=NUM(I)
9200 IF (N1,EQ,0) GO TO 765
9300 IF (MOPT,GE,0) GO TO 50
IMX=N1+1
DO 45 J=1, IMX
9450 MUM(J)=J=1
45 CONTINUE
9500 CALL FNORM (N1, MOPT, IND)
DO 999 J=1, IMX
10000 M1=MUM(J)
10100 WRITE (5,510) N1, M1, IND, A(J), B(J)
10200 N1=M1=N1+1
10300 MODNM=MOD(N1MM1, 2)
10400 IF (IND,EQ,1) MODNM=1
10500 ISP=0
10600 IF (M1,GT,0) GO TO 90
10700 IF (MODNM) 60, 70, 80
10800 60 NR=N1
10900 NRP=NR=1
11000 NP=NR
11100 KIND=5
11200 IF (N1MM1,EQ,1) ISP=1
11300 GO TO 130
11400 70 NR=N1MM1/2
11500 NRP=NR=1
11600 NP=2*NR
11700 KIND=1
11800 IF (N1MM1,EQ,2) ISP=1
11900 K7=2*NR
12000 K1=K7+1
18100 K7=K1
18200 K3=NR+1
18300 K6=K3
18400 K9=K3
18500 K11=K3
18600 K12=K3
18700 K2=K3+1
18800 K4=K2
18900 K5=K2
19000 K8=K2
19100 K10=K2
19200 Z(K1)=1,D0
19300 SI=-1,D0
19400 GO TO 130
19500 120 NR=N1*M1
19600 NRP=NR+1
19700 NP=NR+2
19800 KIND=6
19900 IF (N1,M1,EQ,0) ISP=1
20000 130 NC=NR+1
20100 NCP=NRP+1
20200 NPP=NP+1
20300 IF (M1,EQ,0) NFP=NP+1
20400 IF (M1,GT,0) NFP=NP+1
20500 CALL COEF
20600 WRITE (6,520)(C(K),K=1,NC)
20700 WRITE (6,530)(CP(K),K=1,NCP)
20800 IF (INC,EQ,0) GO TO 999
20900 IF (ISP,EQ,1) GO TO 800
21000 CALL GRAEFF(C,NC,Z,SCALE,NTOL,ITMAX,IND)
21100 CALL GRAEFF(CP,NCP,ZP,SCALE,NTOL,ITMAX,IND)
21200 GO TO (140,160,180,200,220,240) KIND
21300 140 "DO 150 K=1,NRP
21400 Z(K)=DSQRT(Z(K))
21500 KK=NRP+1-K
21600 ZP(KK+1)=DSQRT(ZP(KK))
21700 150 CONTINUE
21800 Z(NR)=DSQRT(Z(NR))
21900 ZP(1)=0,D0
22000 NRP=NRP+1
22100 GO TO 220
22200 160 "DO 170 K=1,NR
22300 KK=NR+1-K
22400 Z(KK+1)=DSQRT(Z(KK))
22500 ZP(K)=DSQRT(ZP(K))
22600 170 CONTINUE
22700 Z(1)=0,D0
22800 NR=NR+1
22900 GO TO 220
23000 180 "DO 190 K=1,NR
23100 KK=NR+1-K
23200 Z(K)=DSQRT(Z(K))
23300 ZP(KK+1)=DSQRT(ZP(KK))
23400 190 CONTINUE
23500 ZP(1)=0,D0
23600 NRP=NRP+1
23700 GO TO 220
23800 200 "DO 210 K=1,NR
23900 KK=NR+1-K
24000 Z(KK+1)=DSQRT(Z(KK))
24100 \[ \text{ZP}(K) = \text{DSORT}(\text{ZP}(K)) \]
24200 CONTINUE
24300 \[ \text{Z}(1) = 0, 0 \]
24400 \[ \text{FP}(NRP) = \text{DSORT}(\text{FP}(NRP)) \]
24500 \[ \text{NR} = \text{NR} + 1 \]
24600 IF (INC = 2) 535, 240, 230
24700 CALL FUNCT(\text{FP}, \text{EX}, NRP)
24800 IF (INC, EQ, 3) GO TO 535
24900 \[ \text{X}1 = 0, \text{D}0 \]
25000 IF (\text{MUNH}) 250, 250, 260
25100 \[ \text{KK}1 = 1 \]
25200 \[ \text{KK}2 = 0 \]
25300 GO TO 270
25400 \[ \text{KK}1 = 2 \]
25500 \[ \text{KK}2 = 1 \]
25600 \[ \text{DO} 300 \text{K} = \text{KK}1, \text{NR} \]
25700 \[ \text{X}2 = \text{Z}(K) \]
25800 CALL GAUSS(X1, X2, N1, ABINT, TOL)
25900 \[ \text{FIN}(\text{NR} + 1 + \text{KK}2) = \text{ABINT} \]
26000 CONTINUE
26100 \[ \text{X}2 = 1, \text{D}0 \]
26200 CALL GAUSS(X1, X2, N1, ABINT, TOL)
26300 \[ \text{FIN}(\text{NR} + 1 + \text{KK}2) = \text{ABINT} \]
26400 GO TO 535
26500 510 FORMAT \( ' N = 1, 13, 6 X, ' M = 1, 13, 6 X, \)
26600 C \( 1, \text{IND} = 1, 12 / \) NORM FACTORS: \( A(N, M) = 1 \)
26700 D, 1PD21, 14, 5X, \( B(N, M) = 1, 0211, 14 \)
26800 520 FORMAT \( ' C / (6 X, 1P3025, 14) \)
26900 \( \text{IF (KIND}, \text{EQ, 1)} \) GO TO 600
27000 \[ \text{IF (KIND}, \text{EQ, 2)} \) GO TO 580
27100 \[ \text{IF (M1, GT, 0) GO TO 537 \]
27200 KF1 = K7 + 2
27300 KF2 = K8 + 1
27400 KF4 = K10 + 1
27500 KF5 = K11 + 1
27600 GO TO 538
27700 537 KF1 = K7
27800 KF2 = K8
27900 KF4 = K10
28000 KF5 = K11
28100 538 DO 540 K = 1, K3
28200 \[ \text{FIN}(1) = 2 \cdot 08 \cdot \text{FIN}(1) \]
28300 \[ \text{Z}(K1 + K) = \text{Z}(K2 + K) \]
28400 CONTINUE
28500 540 DO 550 K = 1, K6
28600 \[ \text{Z}(K4 + K) = \text{Z}(K5 + K) \]
28700 CONTINUE
28800 550 DO 560 K = 1, K9
28900 \[ \text{FIN} (\text{KF}1 + K) = \text{FIN} (\text{KF}2 + K) \]
29000 CONTINUE
29100 \[ \text{IF (M1, EQ, 0) FIN(KF3) = FIN(1) \]
29200 \[ \text{DO 570 K} = 1, K12 \]
29300 \[ K1 = K10 + K \]
29400 \[ K2 = K8 + K \]
29500 \[ K3 = K7 + K \]
29600 \[ K4 = K6 + K \]
29700 \[ K5 = K5 + K \]
29800 \[ K6 = K4 + K \]
29900 \[ K7 = K3 + K \]
30000 \[ K8 = K2 + K \]
30100 \[ K9 = K1 + K \]
30200 \[ K10 = K0 + K \]
30300 \[ K11 = K9 + K \]
30400 \[ K12 = K8 + K \]
30100   K2=K11+K
30200   ZP(K1)=-ZP(K2)
30300   EX(K1)=SI*EX(K2)
30400   FIN(KF4+K)=SI*FIN(KF5+K)
30500   CONTINUE
30600   IF (M1,EQ,0) FIN(1)=SI*FIN(K7+1)
30700   GO TO 603
30800   DO 590  K=1,NR
30900   Z(NR+K+2)=Z(NR+K+1)
31000   CONTINUE
31100   Z(1)=3,00
31200   Z(NR+2)=1
31300   600 IF (INC,EQ,4) GO TO 700
31400   WRITE (6,612) (Z(K),K=1,NP)
31500   610 FORMAT ('ZEROES OF PNM ARE!!/(10X,1P3D25,14)/')
31600   WRITE (6,622) (EP(K),K=1,NPP)
31700   620 FORMAT ('ZEROES OF PNM PRIME ARE!!/(10X,1P3D25,14)/')
31800   IF (INC=2) 999,650,630
31900   630 WRITE (6,642) (EX(K),K=1,NPP)
32000   640 FORMAT ('EXTREMA OF PNM ARE!!/(10X,1P3D25,14)/')
32100   IF (INC,EQ,3) GO TO 999
32200   650 WRITE (6,662) (FIN(K),K=1,NFP)
32300   660 FORMAT ('INTERVAL INTEGRALS FOR PNM ARE!!/')
32400   C(1:8X,1P3D25,14)/
32500   GO TO 999
32600   700 WRITE (6,712)
32700   710 FORMAT ('ZEROES OF PNM AND PNM PRIME,'
32800   'EXTREMA OF PNM AND INTERVAL INTEGRALS FOR PNM FOLLOW!/'
32900   '9X,1P3D25,14,' 'ZEROES OF PNM PRIME,8X,1P3D25,14,'
33000   'EXTREMA OF PNM,8X,1P3D25,14,' 'INTERVAL INTEGRALS/')
33100   IF (M1,GT,0) GO TO 715
33200   IP=IND-1
33300   WRITE (6,712) IP,FIN(1)
33400   715 FORMAT (43X,'INTERGRAL FROM 1,12,1 TO FIRST ZERO'
33500   C(1:4X,1P3D25,14)
33600   WRITE (6,722) Z(1)
33700   720 FORMAT (3X,1P3D25,14)
33800   K1=NFP=1
33900   GO 750 K=1,K1
34000   KK=K
34100   IF(M1,EQ,0) KK=KK+1
34200   WRITE (6,730) ZP(K),EX(K),FIN(KK)
34300   730 FORMAT (28X,1P3D25,14)
34400   K2=K+1
34500   WRITE (6,740) Z(K2)
34600   740 FORMAT (3X,1P3D25,14)
34700   CONTINUE
34800   IF (M1,GT,0) GO TO 999
34900   WRITE (6,762) FIN(NP+1)
35000   760 FORMAT (43X,'INTERGRAL FROM LAST ZERO TO 1 IS'
35100   C(1:4X,1P3D25,14)
35200   GO TO 999
35300   765 WRITE (6,770)
35400   770 FORMAT ('PO0=1,0; PO0 PRIME =0; NO ROOTS,'
35500   'AND EXTREMA,NO INTERVAL INTEGRALS/')
35600   GO TO 1000
35700   800 GO TO (810,820,830,840,850,860) KIND
35800   810 Z(2)=DSRT1(-C(2)/C(1))
35900   ZP(1)=Z(2)
36000   30
CALL GAUSS(\(z(2), 1, D0, NI, FIN(3), TOL\))
CALL GAUSS(\(z(1), z(2), NI, FIN(2), TOL\))
FIN(1) = FIN(3)
GO TO 603

Z(1) = 0, D0
CALL GAUSS(\(z(2), D0, 1, D0, NI, FIN(2), TOL\))
FIN(1) = -FIN(2)
GO TO 903

Z(1) = -1, D0
Z(2) = 1, D0
ZP(1) = 0, D0
EX(1) = C(1)
CALL GAUSS(-1, D0, 1, D0, NI, FIN(1), TOL)
GO TO 603

Z(1) = -1, D0
Z(2) = 0, D0
Z(3) = 1, D0
ZP(2) = DSORT(\(C(2)/C(1)\))
ZP(1) = ZP(2)
CALL FUNCT(ZP, EX, 2)
CALL GAUSS(\(z(2), D0, z(1), NI, FIN(2), TOL\))
FIN(1) = -FIN(2)
GO TO 600

Z(1) = C(1)/C(2)
CALL GAUSS(0, D0, Z(1), NI, FIN(1), TOL)
FIN(2) = -FIN(1)
GO TO 903

Z(1) = 0, D0
Z(2) = 1, D0
ZP(1) = 1/2, D0
EX(1) = C(1)/2, D0
EXM1
CALL GAUSS(E, D0, 1, D0, NI, FIN(1), TOL)
GO TO 600

IF (KIND, EQ, 2) NX1 = 1
IF (KIND, EQ, 5) NX1 = 0
WRITE (6, 910) Z(1), NX1, FIN(1), FIN(2)
GO TO 910

FORMAT ('P10 HAS ONE ZERO AT', 1PD13, 2, ', AND NO
A EXTREMA; THE INTERVAL INTEGRALS ARE', 1PD13, 2, ', FROM
B, 13, TO', 1PD25, 14, ', 1PD25, 14, TO', 1PD13, 2, ', FROM Z(1)
GO TO 999

GO TO 999
CONTINUE

GO TO 5
WRITE (6, 2010)
FORMAT ('INPUT INI DEFECTIVE; GO TO NEXT CASE')
GO TO 5
CONTINUE

END
SUBROUTINE FNORM0 (N,IND)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON /NORMO/B(101)
IF (IND = 1) 10, 100, 200
10 B(1) = 0.0D0
R(2) = DSQRT(3.0D0)
DO 20 1, 2, N
EI = 1
EISQ = EI * EI
ENUM = DSQRT(4.0D0 * EISQ + 1.0D0)
B(I+1) = ENUM * B(I) / (EI = 1.0D0)
CONTINUE
20 RETURN
RETURN
RETURN
END
SUBROUTINE FNORM(N,M,IND)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/NORM0/ B0(101)
COMMON/NORM/A(101), B(101)
EN = N
10 IF (IND EQ 1) 10,100,200
A(1) = B0(N+1)/EN
B(1) = B0(N+1)
10 IF (M,EQ,0) RETURN
IF (M,LT,0) NM = N
IF (M,GT,0) NM = M
C = 2.D0*EN/(EN+1,D0)
10 IF (M,EQ,1) GO TO 90
B(2) = B(1)*B(1)/C
DO 30 I = 2,NM
EI = I
D = (EN=EI+1,D0)/(EN+EI)
B(I+1) = B(I)*D
30 CONTINUE
20 IF (M,GT,0) GO TO 60
A(1) = B(1)/EN
20 NN = N+1
DO 50 I = 3,NN
B(I) = DSORT(B(I))
A(I) = B(I)/EN
50 CONTINUE
270 GO TO 90
230 B(M+1) = DSORT(B(M+1))
290 A(M+1) = B(M+1)/EN
300 RETURN
310 C = DSORT(C)
320 B(2) = B(1)*C
A(2) = B(2)/EN
340 RETURN
350 EN2PN = EN*(EN+1,D0)
360 C = (2,D0*EN+1,D0)
370 A(I) = DSORT(C)
380 B(1) = EN2PN*A(I)
390 IF (M,EQ,0) RETURN
400 A(2) = C*EN2PN = 2.D0
410 IF (M,LT,0) MM = N
420 IF (M,LT,0) MM = M
430 DO 130 I = 2,MM
EI = I
440 EI2M1 = EI*EI*EI
A(I+1) = A(I)*(EN2PN = EI2M1)/(EI*EI)
470 CONTINUE
480 EM = M
490 IF (MM,GT,M) GO TO 150
50 DO 160 I = 1,MM
510 EI = I
520 A(I+1) = DSORT(A(I+1))
530 R(I+1) = (A(I+1))*EI/2.D0
540 CONTINUE
550 RETURN
560 A(M+1) = DSORT(A(M+1))
570 B(M+1) = A(M+1)*EM/2.D0
580 RETURN
590 END
SUBROUTINE COEF
IMPLICIT REAL*6(A-H,O-Z)
C COMPUTE COEFFICIENTS OF POLYNOMIAL PARTS OF PNM, PNM PRIME
C INPUT: ORDER N, DEGREE M AND IND TO GIVE FORMULATION DESIRED
C OUTPUT: NUMBER NC OF COEFFICIENTS C OF PNM
C NUMBER NCP OF COEFFICIENTS CP OF PNM PRIME
C COEFFICIENTS C(I), CP(I)
C IND=0: PNM GIVEN IN POWERS OF (COS(THETA))/2 WITH
C NC=(N-M+2)/2, NCP=NC+1
C IND=1: PNM GIVEN IN POWERS OF (SIN(THETA)/2)/2 WITH
C NC=N*M, NCP=N*M+1
C IF M=0, NCP=NC=1 FOR BOTH FORMULATIONS
C IND GT 1 MAY BE USED FOR OTHER FORMULATIONS
COMMON/NORM/A(101), B(101)
COMMON/COEFF/C(101), CP(102), NC, NCP, N, M, IND
NM=N=M
ENM=NM
EN=N
MN=M
C(I)=A(M+1)
CP(I)=B(M+1)
EN2PN=EN=EN+EN
IF(IND=1)1100,400,700
100 ENH02=ENMM/2,00
NM02=ENM02
K1=NC-1
200 TWONP1=2,00*EN1+1,00
TW0NM1=TWONP1-2,00
TW0N=2,00*EM
ENMP1=ENMH1+1,00
T=ENM02*CP(1)/(EN*TW0NM1)
S=EN2PN=EN*EM+TW0M
300 DO 150 K1=1,K1
400 EK=K
500 TWOK=2,00*EK
C(K+1)=-C(K)*((ENMM+2,00*TWOK)*(ENMP1*TWOK))
1/TWOK*(TWONP1-TWOK)
600 CP(K+1)=T*S
700 T=T*(ENM-TWOK)*(ENMP1-TWOK)/(TWOK)
800 1+2,00*(TWONM1-TWOK)
900 S=S+TW0M
100 150 CONTINUE
110 IF(M.EQ.3) GO TO 200
120 IF(IND.EQ.1) GO TO 200
130 200 ENC=NC
140 CP(NCP)=C(NC)
150 RETURN
160 400 DO 420 K1=1,NMM
170 410 EK=K
180 TWOK=2,00*EK
190 EMPK=EM+EK
200 EMPK1=EMPK-1,00
210 EKMPK=EK*EMPK
220 T=(EN2PN=EMPKM1*EMPK)/EKMPK
230 C(K+1)=T*C(K)
240 420 CONTINUE
250 34
06100 \text{S} = (\text{EM} + \text{TWO}K) * \text{EN}ZP\text{N} = \text{EM} * \text{EM}PKM1 * \text{EM}PK
06200 \text{CP}(K+1) = \frac{\text{S} \cdot \text{C}(K)}{(2,04 * \text{E}K\text{MP}K)}
06300 \text{CO}N\text{T}I\text{NUE}
06400 \text{CP}(NCP) = \text{EN} \cdot \text{C}(NCP)
06500 \text{IF}(N, GT, 0) \text{R}E\text{T}U\text{R}N
06600 \text{DO} 450 \text{K} = 2, NCP
06700 \text{EK} = K
06800 \text{CP}(K) = \text{EK} \cdot \text{C}(K+1)
06900 \text{C}O\text{T}I\text{NUE}
07000 \text{R}E\text{T}U\text{R}N
07100 \text{R}E\text{T}U\text{R}N
07200 \text{END}
SURROUN.INE GRAEFF (AA,N.Z,SCALE,NTOL,ITMAX,IND)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION A(102),AA(102),Z(102),B(102),K1(102)
CMEXA(102),NEXA(102)
C Roots z(i) of a polynomial of degree N-1 by Graeff's method
C A(i) is coefficient of x**(N-I), I=1,2,...,N for IND=0
C " " " of x**(I-1), " " for IND.GT.0
C ITMAX is the maximum number of iterations
C ATOL terminates iteration on a convergence criterion
IF (N.EQ.1) GO TO 270
IF (N.EQ.2) GO TO 280
ITER = 1
EN=N
ENO2 = EN/2.0D0
ND2=ENO2
DO 10 I = 1,N
A(I)=AA(I)
SIG1=1.D0
IF (T.LT.NO2) K1(I)=I-1
IF (T.GT.NO2) K1(I)=N-1
NEX=0
TEST=A(I)
10 IF (TEST.LT.0.D0) SIG1=1.0D0
TEST=DABS(TEST)
2 IF (TEST.LT.SCALE) GO TO 4
TEST=TEST/SCALE
NEX=NEX+1
GO TO 2
10 IF (TEST.GE.1.0D0) GO TO 6
TEST=TEST*SCALE
NEX=NEX*2.
4 IF (TEST.LE.1.0D0) GO TO 6
TEST=TEST*SCALE
NEX=NEX-1
GO TO 2
6 NEXA(I)=NEX
AK(I)=SIG1*TEST
10 CONTINUE
DO 20 I = 1,N
SIG=-1.0D0
C=A(I)*A(I)
NEXC=NEXA(I)*2
KSUM=K1(I)
20 IF (K1(I).EQ.0) KSUM = 1
DO 30 K=1,KSUM
IF (K1(I).EQ.0) GO TO 75
TERM=A(I+K)*A(I-K)*2.0D0
NEXD=NEXC-NEXT
4 IF (IABS(NEXD).LT.NTOL) GO TO 45
IF (K.EQ.KSUM) K1(I)=KSUM-1
GO TO 94
30 IF (K1(I).EQ.0) GO TO 75
TERM=A(I+K)*A(I-K)*2.0D0
NEXD=NEXC-NEXT
4 IF (IABS(NEXD).LT.NTOL) GO TO 45
IF (K.EQ.KSUM) K1(I)=KSUM-1
GO TO 94
45 IF (NEXD.LT.0) GO TO 50
TERM=TERM*SCALE**(-NEXD)
C=C*TERM*SIG
GO TO 75
50 C=C*TERM*SIG
GO TO 55
55 C=C*SCALE**(NEXD)
C=C*TERM*SIG
NEXC=NEX
75 SIG1=1
56 IF (C.LT.0.D0) SIG1=-1
C=DABS(C)
59 IF (C.LT.SCALE) GO TO 85
82
06000 C = C/SCALE
06200 NEXC = NEXC + 1
06300 GO TO 80
06400 85 IF (C .GE. 1.00) GO TO 90
06500 C = C/SCALE
06600 NEXC = NEXC + 1
06700 GO TO 85
06800 90 C = C*SCALE
06900 IF (K1(1).EQ.0) GO TO 96
07000 94 SIG = -SIG
07100 95 CONTINUE
07200 96 B(I) = C
07300 120 CONTINUE
07400 DO 110 I = 1, N
07500 110 CONTINUE
07600 GO TO 200
07700 120 IF (ITER .GE. ITMAX) GO TO 180
07800 180 WRITE (6,190) (K1(I);I=1;N)
07900 190 FORMAT (' ITMAX EXCEEDED! K !S',I5/(3X,15It/))
08000 N3 = N - 1
08100 200 N4 = I
08200 210 Z1 = Z(I+1)/B(I)
08300 NEXZ = NEXB(I+1) - NEXB(I)
08400 GO TO 230
08500 220 Z1 = R(I+1)/B(I+1)
08600 NEXZ = NEXB(I) - NEXB(I+1)
08700 230 IF (Z1 .LT. 0) WRITE (6,240) N4, Z1
08800 240 FORMAT (' Z(',I5,) NEGATIVE AND EQUAL TO;E-8')
08900 N4 = I
09000 250 CONTINUE
09100 260 WRITE (6,260) ITER
09200 260 FORMAT (' POLYNOMIAL IS OF DEGREE 0; NO ROOTS')
09300 RETURN
09400 270 WRITE (6,275)
09500 275 FORMAT (' POLYNOMIAL IS LINEAR; Z(1) = ',I0D25.14)
09600 RETURN
09700 END
SUBROUTINE FUNCT (X, P, L)

IMPLICIT REAL*8 (A-H, O-Z)

C TO EVALUATE ASSOCIATED LEGENDRE FUNCTIONS PN M AT L POINTS X. OUTPUT IS L VALUES OF PNM, IND

C INDICATES THE FORMULATION USED;

COMMON/COEFF/C(101),CP(102),NC,NCP,N,M,IND

DIMENSION X(102),P(102),Y(102)

EN=M

EM02=EM/2,000

IF(IND-1) 10,100,200

10 DO 20 I=1,L

20 CONTINUE

CALL EVAL(C,NC,L,Y,P,IND)

IF (MOD((N-M),2),EG,0) GO TO 40

DO 30 I =1,L

30 CONTINUE

RETURN

20 DO 50 I=1,L

P(I)=P(I)*(1,0D0-X(I)**2)*EM02

50 CONTINUE

RETURN

40 DO 53 I=1,L

P(I)=P(I)*(1,0D0-X(I)**2)*EM02

53 CONTINUE

RETURN

CALL EVAL(C,NC,L,X,P,IND)

DO 110 I=1,L

110 CONTINUE

RETURN

CALL EVAL(C,NC,L,X,P,IND)

END
SUBROUTINE EVAL(A,N,M,X,P,IND)
C EVALUATE A POLYNOMIAL OF DEGREE N-1 AT M POINTS X(I)
C RETURN N VALUES P(I)
C A(I) IS THE COEFFICIENT OF X**(N-I) FOR IND = 0
C " " " X**(I-1) FOR IND,NE,0
DIMENSION A(102),X(102),P(102)
IF (N,GT,1) GO TO 10
DO 5 K=1,M
P(K)=A(1)
5 CONTINUE
RETURN
IF (IND,NE,1) GO TO 50
DO 40 I=1,M
T=X(I)
Y=A(1)*T+A(2)
IF (N,NE,2) GO TO 35
DO 30 K=3,N
Y=Y*T+A(K)
30 CONTINUE
P(I)=Y
40 CONTINUE
RETURN
DO 80 I=1,M
T=X(I)
Y=A(N)*T+A(N-I)
IF (N,NE,2) GO TO 75
DO 70 K=3,N
Y=Y*T+A(N+1-K)
70 CONTINUE
P(I)=Y
80 CONTINUE
RETURN
END
SUBROUTINE GAUSS(A,R,N,ABINT,TOL)
C SUBROUTINE FOR THE INTEGRAL FROM A TO B BY GAUSSIAN QUADRATURE
C Z(J,L) ARE THE ZEROS OF THE (L+1)ST LEGENDRE POLYNOMIAL
C W(J,L) ARE THE CORRESPONDING WEIGHTS
C CALLS SUBROUTINE FUNCT WHICH DEFINES THE INTEGRAND

IMPLICIT REAL *8 (A-H, O-Z)
DIMENSION Z(5,9), W(5,9), X(10), F(10)
DATA Z/'57735P26918962600, 4*0.D0,
1.000, 7.7459666924148300, 3*0.D0,
5.110, 3.3399148564856D0, ,86113631159405300, 3*0.D0,
1.220, 0.00, 5.846931010568300, 9.0617984593866400, 2*0.D0,
1.330, 4.2361986588319700, ,66120938646626500, 3*0.D0,
1.440, 5.00, 4.0584515137739700, 7.415311859939400,
1.550, 0.94910791234275900, ,3*0.D0,
1.660, 7.8343462495650D0, ,52553240991632900, 9.7666547741362700,
1.770, 9.00, 2.829865649753600, 3*0.D0,
1.880, 0.00, 5.2425342349386900, 6.133714372099000;
1.990, A.330110732663600, 9.6816023950762600,
2.000, B.14887433898163100, 4.3339539412924700, 6.7940956829902400,
2.110, C.86506336668898500, 7.390652831717200/
2.220, DATA W/1,00, 4*0.00,
2.330, 1.000, 8.8888888888888900, 5.5555555555555500, 3*0.D0,
2.440, 2.65245154858254600, ,34785484513745400, 3*0.D0,
2.550, 5.6868888888888800, 4.7862867049936600, 3*0.D0,
2.660, 0.00, 2.3492688505618900, 2*0.D0,
2.770, 5.4679133457269100, 3.6076157304813900,
2.880, 0.00, 7.132449237912700, 2*0.D0,
2.990, 4.1795918367346900, ,38183053505119000,
3.000, 0.00, 2.7977539146927700, 1.2948496616887000, 0.000,
3.110, 9.3626378337836200, 3.1370645877887000,
3.220, A.2223167054534700, 1.0122853629037600, 0.000,
3.330, B.3302393550212600, 3.1234777870403000, 2.6361069642935000,
3.440, C.18064861679485700, 0.01274388361575400, 2.9552422471475300,
3.550, D.2692671937999600, 2.1908636251598200,
3.660, E.14945134915058100, 0.6667134430868800/
3.770, FACM=(B-A)/2.00
3.880, IF (DABS(FACM)>TOL) GO TO 5
3.990, ABINT=A,0.0
4.000, WRITE (6,2)
4.110, 2 FORMAT (',(',B-A,)',,LT',TOL,ABINT SET TO ZERO!)
4.220, RETURN
4.330, 5 FACAP=(B+A)/2.00
4.440, ABINT=B,0.0
4.550, EN=N
4.660, ENO2=EN/2.00
4.770, NO2=ENO2
4.880, IF((N-2*NO2),EQ,0) K1=2
4.990, IF((N-2*NO2),GT,0) K1=1
5.000, 05 K2=NO2+1
5.110, 05 K3=2*K2+1
5.220, DO 10 K=K1,K2
5.330, TERM=FACM**2(K,N)
5.440, X(K)=FACP*TERM
5.550, X(K3,K)=FACP*TERM
5.660, CONTINUE
5.770, IF (K1,EQ,1) GO TO 15
5.880, X(K)=FACP
5.990, 10 L=N+1
6.000, CALL FUNCT(X,F,L)
06100  00 27  K=K1+K2
06200  ABINT=ABINT+(F(K)+F(K3-K))*W(K,N)
06300  CONTINUE
06400  IF(K1.EQ.1) GO TO 25
06500  ABINT=ABINT+F(1)*W(1;N)
06600  25  ABINT=ABINT*FACM
06700  RETURN