Lawrence Berkeley National Laboratory
Recent Work

**Title**
A FORTRAN IV PROGRAM FOR SOLVING THE N/D STRIP-APPROXIMATION EQUATIONS

**Permalink**
https://escholarship.org/uc/item/8v0042mt

**Authors**
Teplitz, Doris C.
Teplitz, Vigdor L.

**Publication Date**
1964-10-02

eScholarship.org Powered by the California Digital Library
University of California

UCRL-11696

# University of California

# Ernest O. Lawrence Radiation Laboratory

TWO-WEEK LOAN COPY

*This is a Library Circulating Copy which may be borrowed for two weeks. For a personal retention copy, call Tech. Info. Division, Ext. 5545*

A FORTRAN IV PROGRAM FOR SOLVING THE N/D STRIP-APPROXIMATION EQUATIONS

Berkeley, California

**DISCLAIMER**

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

UCRL-11696

UNIVERSITY OF CALIFORNIA
Lawrence Radiation Laboratory
Berkeley, California

AEC Contract No. W-7405-eng-48

# A FORTRAN IV PROGRAM FOR SOLVING THE N/D STRIP-APPROXIMATION EQUATIONS

Doris C. Teplitz and Vigdor L. Teplitz

October 2, 1964

# A FORTRAN IV PROGRAM FOR SOLVING THE N/D STRIP-APPROXIMATION EQUATIONS

Doris C. Teplitz* and Vigdor L. Teplitz†

Lawrence Radiation Laboratory
University of California
Berkeley, California

October 2, 1964

ABSTRACT

This paper contains an account of a FORTRAN program for solving numerically the pion-pion N/D equations in the new form of the strip approximation. It incorporates the Wiener-Hopf technique for coping with the singularity at the strip boundary, and enables one to calculate the trajectories produced by any chosen Born-term input.

## I. INTRODUCTION AND MAIN PROGRAM

A block diagram of the program is given in Fig. 1.

In the first section we introduce the equations that are to be solved, and describe the main program. In subsequent sections the subroutines and function routines called by the main program are discussed, with an explanation of the meaning of the various symbols and the method of calculation. A glossary of the variables is given at the end, and is followed by a series of appendices containing listings of the various routines in FORTRAN IV.

This main program calculates the numerical solutions of the pion-pion strip-approximation N/D equations.[1] The solutions involve Eqs. (1) through (5) of reference 1; these equations are

$$D_\ell(s) = 1 - \pi^{-1} \int_4^{s_1} \frac{ds' \rho_\ell(s')\, N_\ell(s')}{(s' - s)} \tag{1a}$$

$$N_\ell(s) = B_\ell^V(s) + \frac{1}{\pi} \int_4^{s_1} \frac{B_\ell^V(s') - B_\ell^V(s)}{s' - s}\, \rho_\ell(s')\, N_\ell(s'). \tag{1b}$$

Equation (1b) is not Fredholm but has been put by Chew[2] into the following form:

$$N_\ell^{(s)} = \int_4^{s_1} ds'\, O_\ell(s, s')\, N_\ell^0(s') \tag{2}$$

$$N_\ell^0(s) = B_\ell^V(s) + \int_4^{s_1} ds'\, K_\ell'(s, s')\, N_\ell^0(s') . \tag{3}$$

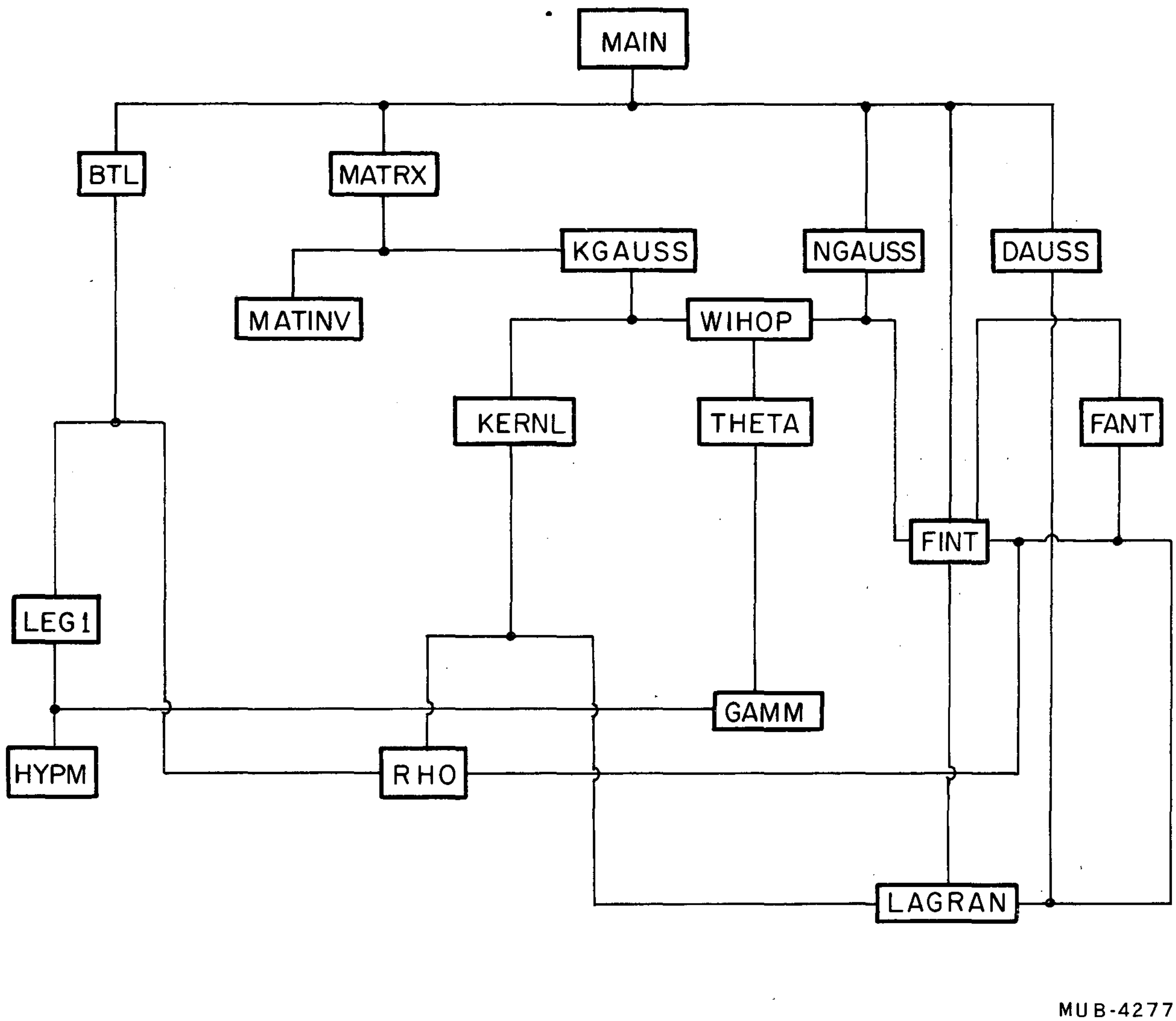



Fig. 1. Block diagram of the Program.

$$K_\ell'(s, s') = \int_4^{s_1} ds'' \, K_\ell(s, s'') \, O_\ell(s'', s') \tag{4}$$

$$K_\ell(s, s') = [\pi(s'-s)]^{-1} \left\{ [B_\ell^V(s') - B_\ell^V(s)] \, \rho_\ell(s') + (\lambda_\ell/\pi)[\ell n(s_1 - s') - \ell n(s_1 - s)] \right\} , \tag{5}$$

where $\lambda_\ell = \sin^2 \pi a_\ell$, $\pi a_\ell = \delta_\ell(s_1)$ and the Wiener-Hopf resolvent kernel $O_\ell$ is defined in Sec. X (WIHOP). Equation (3), unlike Eq. (1b), is Fredholm.

For each $\ell$, the program finds $N_\ell(s)$ for $4 < s < s_1$, $D_\ell(s)$ for arbitrary s, and the zeroes of $D_\ell(s)$. Half of the zeroes of D are resonances. Since for each $\ell$ the point SR at which D(SR) is zero is known, the trajectory $\alpha(SR) = \ell$ is known. The quantity

$$\frac{N_\ell(SR)}{D_\ell'(SR)} = - \frac{\gamma(SR)}{\alpha'(SR)} \tag{6}$$

is calculated. Here $\gamma$ is the reduced residue. For SR greater than 4, the width of the resonance is also found, and is

$$W = -\rho_\ell(SR)\gamma(SR)/\alpha'(SR) \, 2\sqrt{SR} \ . \tag{7}$$

For SR less than 4, the program computes $N_\ell(SR)$ by the following method. From the non-Fredholm integral equation (1b), and equation (1a) for D(s), the relation is found to be

$$N_\ell(s) = B_\ell^V(s) \, D_\ell(s) + \pi^{-1} \int_{s_0}^{s_1} ds' \, B_\ell^V(s') \, \rho_\ell(s')/(s' - s) \ .$$

For $s = SR$, $D(SR) = 0$, and we get the equation

$$N_\ell(SR) = \int_{s_0}^{s_1} ds' B_\ell^V(s') \rho_\ell(s') N_\ell(s')/(s' - SR) \tag{8}$$

from which the program calculates $N(SR)$ for $SR < 4$. Note that the integrand in Eq. (8) differs from that in Eq. (1a) for D only by a factor of $B_\ell^V(s')$.

In brief, the MAIN program calls subroutine MATRX, which returns a one-dimensional array YANS that approximates the solution of the Fredholm integral Eq. (3). Calling the subroutine NGAUSS, the MAIN program obtains an array of values, GAUS, which approximates $N_\ell$ at a set of points from 4 to $s_1$. The MAIN program then calls DGAUSS for another set of points and obtains a new array of elements GAUS, which approximates $D_\ell(s)$ of Eq. (1a). DGAUSS also returns the zeroes of D, and the slopes of D at the zeroes. The MAIN program then calculates Eqs. (6) and (7). In a second call statement DGAUSS returns $N_\ell(SR)$ by means of Eq. (8) when necessary.

Equations (4) and (5) are calculated by subroutines KGAUSS and KERNL, respectively. Subroutine MATRX calls KGAUSS and KGAUSS calls KERNL.

Note that the choice of $B_\ell^V$ may be varied by changing the function subroutine BTL. The function used in the calculation BTL(s) in Ref. 1 is discussed in Sec. II (BTL). The subroutine BTL(s) may be omitted entirely (since it is used only at a finite number of values of s from 4 to $s_1$) if $B_\ell^V(s)$ is specified at these points. This freedom is in order that the input to the program may be a numerical calculation of the generalized potential $B_\ell^V$. Note, further, that the only restriction to the equal-mass case is in the definition of $\rho(s)$. Simple changes thus allow application to a wide variety of problems.

These variables are read from data cards:

NN X H IST JST ICA JCA NOEROR NAL S1 DAL E AL XL RWIDR RL KNOW. (These quantities are defined where used and in the Glossary.)

Variables

| | |
|---|---|
| XA | = 4, the threshold of the two-pion system. |
| XB | = S1. |
| A | lower limit of integration in the transformed system found by using the "square-root trick." |
| B | upper limit of integration in the transformed system found by using the "square-root trick." |
| I IP TA C D T YT SP | explained in the discussion of the method of Gaussian quadratures. |
| YANS(I, 1) | solution of the Fredholm equation. |
| SONE(I) | points at which the solutions to the Fredholm equation for $N^0$ are evaluated. |
| NUMBER | = I×IP number of points at which $B_\ell^V$ is evaluated by the function subroutine BTL. |
| NNN | number of SONE points. |
| NNS | number of points at which $N_\ell(s)$ is evaluated. |
| SVAL(I) | name of array of points; used by NGAUSS as points at which $N_\ell(s)$ is evaluated; used by DGAUSS as a different set of points at which $D_\ell(s)$ is evaluated. |

| | |
|---|---|
| GAUS(I) | as returned by subroutine NGAUSS, $N_\ell(s)$; as returned by the first calling statement of DGAUSS, $D_\ell(s)$; as returned by the second calling statement of DGAUSS, $[1-N_\ell(SR)]$. |
| SR | point at which D is zero. |
| NZ | number of zeroes of D. |
| ZERO(NZ) | array of zeroes of D. |
| REDR | $N_\ell(SR)/D'(SR)$. |
| WIDTH | width of resonance [in DGAUSS, however, there is a quantity called WIDTH that is equal to $D_\ell'(s_R)$]. |
| PAIR(I) | This is an array of points and values of functions that has as even-numbered elements the values of the functions, and as odd-numbered elements the points at which the next-highest even-numbered element is evaluated. The same name PAIR is used for different arrays to save space in core of the machine; the definition depends on where the array occurs in the program. In the MAIN program it will be an array involving $B_\ell^V(s)$ or $N_\ell^0(s)$ or $N_\ell(s)$. |
| PEER | array PAIR involving $B_\ell^V$ but with different index than the PAIR array; present through block COMMON in subroutine FANT where it is called PAIR. |

## Notation

Lines in the listing of the program are referred to by the statement number or by the line number. The line number is the number on the FORTRAN card.

## The Method of Gaussian Quadratures and the Square-Root Trick

The integrations done in this program are of the form

$$\mathrm{INT} = \int_{XA}^{XB} F(s')/(s'-s) = \int_{XA}^{XB} f(s')\,ds' ,$$

where the function f generally has an infinite-type singularity [not worse than $(s_1 - s)^{-1/2}$] at $s_1$ but not at $s = 4$. [For $\ell < 1/2$, $\rho_\ell \sim (s-4)^{\ell + 1/2}$ introduces an infinite-type singularity of f at 4 amenable to the same methods applied at S1.] The singularity at S1 is avoided by the following transformation, which we refer to as the square-root trick.

$$\mathrm{INT} = 2 \int_0^{\sqrt{XB-XA}} Y \cdot f(XB - Y^2)\,dY, \qquad (9)$$

where $Y = \sqrt{s_1 - s}$.

The integrations are performed by the method of Gaussian quadratures.[3] For a polynomial f(X) of degree 2m - 1, the following equation

$$\int_{-1}^{1} f(X)dX = \sum_{i=1}^{n} W_i f(X_i) \qquad (10)$$

is exact where the $W_i$ are Gaussian weights and the $X_i$ are Gaussian points. The point $X_i$ is the ith zero of $P_m(X)$, the mth Legendre polynomial. Hildebrand gives the formula for the wieghts in terms of the Legendre polynomial, and tabulates a number of weights and points.[3]

To use the method of Gaussian quadratures, we break the region $(0, \sqrt{XB - XA})$ into intervals and apply Eq. (10) to each interval. For a number of intervals I, the length of the interval C is (B - A)/I, and half this length D is C/2. Now by transforming the region of integration and using the notation of the program, we obtain

$$INT = 2 \sum_{i=1}^{I} \int_{-1}^{1} D \cdot YT \cdot f(S1 - YT^2) dT$$

where YT = D(1 + T) in the first interval. In the second interval YT = C + D(1 + T); and in general YT = TA(i) + D(1+T) where TA(i) is the beginning of the ith interval. Equation (10) then gives for the integral INT

$$INT = 2D \sum_{i=1}^{I} \sum_{i_p=1}^{IP} YT \cdot H(IP, i_p) \cdot f(S1 - YT^2),$$

where IP is the number of Gaussian points per interval--IP=IDO(ICA)=ICA+1. To test this approximation, INT may be evaluated until two successive values lie within a preassigned difference, as discussed in Sec. VI (KGAUSS).

## Operation

Through statement 1234 the program reads variables from data cards and prints quantities of interest.

Through statement 901 arrays PEER and PAIR are constructed. They differ only in indexing--

$$PEER(I) = PAIR(2I + 2).$$

The function evaluated for the even-indexed PAIR elements is $B_\ell^V$. These values are given here by the function subroutine BTL(SP), but could be taken from another calculation. The points SP are chosen to give a greater density of points near S1 where there is a singularity in BTL.

Main calls subroutine MATRX for values of E, IST, ICA, JST, and JCA that have been found to give good convergence [as discussed in Sec. VI (KGAUSS)]. This subroutine MATRX returns the solution $N_\ell^0$ to the Fredholm equation. The solutions are the elements of a one-dimensional array YANS(I, 1), whose elements are evaluated at the points SONE(I). The values of SONE are determined in MATRX.

Through statement 445 the array PAIR is redefined. The odd-numbered elements are equal to the points of SONE, and the even to the values of YANS. The number of elements is twice NNN where NNN is defined in MATRX.

Through statement 311 the array of points SVAL(KZ) is constructed; these are the points at which we want to know $N_\ell(s)$. This array is made with points clustered near S1, since there is a singularity in the function $N_\ell$ at S1. Notice that K2 in line 0800 is defined in such a way that the highest index corresponds to the highest value of SVAL.

The MAIN program then calls subroutine NGAUSS. The function is in FINT so that the calling statement NGAUSS will return the quantity

$$N_\ell(s) = \int_4^{S1} ds' \, O_\ell(s, s') \, \text{FINT}(s').$$

At this point in the program, FINT is the function $N_\ell^0$, since FINT uses the PAIR values formed from YANS; but we have the freedom of using other functions without having to change subroutine NGAUSS [for example, forming PAIR from $B_\ell^V$ in order to test the approximation $N_\ell^0 \approx B_\ell^V$].

Between statements 3330 and 444, PAIR is again redefined. The points of PAIR are the values SVAL(KY), and the function is $N_\ell$ returned by NGAUSS. A new array of points of SVAL is then constructed for the points at which we wish to know D. Depending on the problem the choice of these points may be anywhere from $-\infty$ to $+\infty$. The MAIN program calls DGAUSS, which returns an array of points ZERO. These are the points at which D is zero. It also returns the slopes of D at the zeros. The DO loop ending in statement 104 finds for each zero point SR, the quantity REDR. The IF statement 106 separates zeroes less than 4 from those greater than or equal to 4. For the latter, N(SR) can be found by calling FINT. For $SR < 4$, $N_\ell$ is found by calling DGAUSS again, for a point SVAL(1) = SR. The quantity GAUS returned is $1 - N_\ell(SR)$, so REDR = (1 - GAUS)/D'. The first call statement for DGAUSS returns values of GAUS equal to values of D, because the function FINT in the calling statement is $N_\ell$; the value of GAUS given by the second call statement, because of the function FANT in the calling statement, is $B_\ell \overset{V}{N}_\ell$. For SR greater than or equal to 4, the second statement after statement 106 calculates the width of the resonance.

Another loop of the over-all DO loop beginning in line 0340 is now begun with a new value of either (a) the angular momentum or (b) the phase shift at the strip boundary, the choice depending on the value of the indicator KNOW.

At the finish of the DO loop a new set of data is read and the program repeated.

## II. BTL

We give here the program for the generalized potential used in reference 1.

$$B_\ell^V(s) = B_\ell^P(s) + B_\ell^\rho(s) .$$

We calculate

$$BTL = BPRL - BPPO .$$

The term-BPPO corresponds to Eq. (10) of reference 1.

$$B_\ell^P(s) = -[\sin^2 \pi a_\ell / \pi \rho_\ell(s_1)] \; \ell n[(s_1 - s)/s_1] .$$

The term BPRL corresponds to Eq. (9) of reference 1 for the case T = 1,

$$B_\ell^\rho(s) = (1/2)(3\Gamma_\rho \sqrt{t_\rho} / q_s^{2\ell+2})(1 + s/2q_\rho^2) \, Q_\ell(1 + t\rho/2q_s^2),$$

where $\Gamma_\rho$ is the full width of the exchanged $\rho$ in pion masses for the direct T = 1 channel, and is one-half the width for the direct T = 0 channel. The function $Q_\ell$ is the Legendre function of the second kind.

### Variables

| | |
|---|---|
| XR | $=\sqrt{t_\rho}$ mass of the $\rho$. |
| QS2 | $q_s^2$. |
| Z | argument of $Q_\ell$. |
| RWIDR | $\Gamma_\rho$ read from data cards in the MAIN program. |
| Q1FN | $Q_\ell$. |

## Operation

After defining variables we can calculate the term BPRL from line 0100 through the line after 0220.

The two IF statements, 0100 and 0110, distinguish three cases for evaluating Q1FN according to the value of XL: $\ell$ near one, $\ell$ near zero, and $\ell$ noninteger.

For the case $\ell = 1$, statement 3 gives $Q_1$; for $\ell = 0$, statement 1 gives $Q_0$. To evaluate $Q_\ell$ for noninteger $\ell$, BTL calls LEGP1, which returns the function F1. The Legendre function is

$$Q_\ell(z) = (\pi/\tan \pi\ell)\, F1. \qquad (11)$$

Function subroutine RHO is called in evaluating BPPQ.

## III. LEGP1

This SHARE routine[4] has been modified to give Legendre functions of the second kind, $Q_\ell(Z)$ for real $\ell$ and real Z.

To call LEGP1 (A1, Z, F1), the calling program must give $\ell$ (called A1 in the SHARE program) and must give Z (the argument).

The program calculates $F1 = F_{A1}(Z)$, which is related to $Q_\ell(z)$ by Eq. (11).

With $A1 = \ell$ and $|Z| > 1$, we have for $F_\ell$

$$F_\ell(Z) = \frac{\tan \ell\pi}{\sqrt{\pi}\,(2Z)^{\ell+1}} \frac{\Gamma(\ell+1)}{\Gamma(\ell+3/2)} \quad F\left(\frac{\ell}{2}+1, \frac{\ell+1}{2}, \ell+\frac{3}{2}, \frac{1}{Z^2}\right).$$

Subroutine LEGP1 calls subroutine GAMM to evaluate the gamma function and HYPM to evaluate the hypergeometric function.

## IV. HYPM

This routine is SHARE routine C3EO HYPR modified to find the hypergeometric function with only real degree. To call HYPM (A1, B1, C1, Z, EP, F1) the calling program must give

Z = $1/Z^2$.

EP = 0.000001 = desired accuracy.

## V. MATRX

This subroutine solves the Fredholm integral Eq. (3). A Fredholm integral equation may be solved by approximating it by a matrix equation. Thus,

$$N_\ell^0(s_i) = B_\ell^V(s_i) + \sum_{j=1}^{N} H_j \, \mathrm{XMAT}\,(s_i, s_j) N_\ell^0(s_j),$$

where the points $s_i$ are chosen to approximate the integral in Eq. (3) and the $H_j$ are the Gaussian weights; note XMAT = $K_\ell'(s_i, s_j)$

$$\mathrm{XMAT}\,(s_i, s_j) = \delta_{ij} - \sum_j H_j \, \mathrm{XMAT}\,(s_i, s_j). \qquad (12)$$

## Variables

| | |
|---|---|
| XL, E, IST, ICA, JST, JCA | quantities read from data cards in the calling program, MAIN. |
| NNN | size of the matrix XMAT(I, J) is NNN×NNN. |
| SONE | points at which $N^0$ is evaluated. |
| XMAT | name of the matrix approximating $K_\ell'(s, s')$, name of the matrix to be inverted, and name of the inverted matrix. |
| BPL | vector whose elements are values of $B_\ell^V$ equal to the array BPL before inversion, and the solution of the Fredholm equation after inversion. |

## Operation

Through statement 7, the one-dimensional array of points SONE is constructed with a concentration of points near S1, and a distribution appropriate to applying Gaussian quadratures to the integral in Eq. (3), with $\sqrt{s_1 - s}$ for the variable of integration. This array is in COMMON and is used by KGAUSS and KERNL.

KGAUSS evaluates $K_\ell'(s, s')$ by approximating it by a matrix XMAT(KX, IY). SONE(IY) corresponds in Eq. (14) to the value of $s'$ in $O_\ell(s'', s')$; SONE(KX) corresponds to the value of $s$ in $K_\ell(s, s'')$. The variables $s$ and $s'$ then range over the same set of points. Since $K_\ell'$ is the kernel of the integral equation (3), it is necessary to use the square-root trick, as explained in MAIN (Sec. I), because of the singularity at S1. The values of $\sqrt{s_1 - s_i}$ are picked as functions of Gaussian points because, in evaluating the integral equation, the integral is approximated by a summation over Gaussian points by means of weights.

In line 0420 MATRX calls KGAUSS, which returns XMAT. The size of this matrix is determined by NNN, the number of SONE points; this number is determined by IST and ICA. For 2% accuracy, NNN = 15 was found to be sufficient in the work of reference 1. It should be noted that a 15 by 15 matrix here should give an answer as accurate as a 30 by 30 matrix in a program using, e.g., Simpson's rule. The equation

$$I - \int ds' K_\ell' = I - 2D \int y\, dy\, K_\ell' \approx I - 2D \sum_j H_j\, YT_j\, XMAT(i,j)$$

is approximated by a matrix called XMAT (III, KK). The outer two DO loops calculate the Gaussian points T and the quantity YT. The inner DO loop constructs XMAT (III, KK). KK is associated with the variable of integration s'. For each KK and combination of I and IP--i.e., each YT and H--XMAT is evaluated for each s corresponding to the index III. Notice the highest KK value is associated with the lowest YT (and therefore the highest s'), since $YT = \sqrt{S1 - s'}$. The minus sign in line 0480 is due to the one in Eq. (12). Statement 8000 calculates the diagonal elements. The DO loop ending in statement 9999 makes the array YANS from the vector BPL calculated in KERNL. MATRX then calls MATINV, which inverts the matrix and returns the solutions to the Fredholm equation also called YANS. Several different quantities are called XMAT in the course of the above in order that room for only one large matrix need be set aside in core storage.

## VI. KGAUSS

This subroutine calculates $K_\ell'$ of Eq. (4). We approximate $K_\ell'$ by a matrix

$$XMAT(I, J) = \int_4^{s_1} K_\ell(s_i, s'') \, O_\ell(s'', s_j') \, ds'' .$$

Each element is an integral over $s''$. We perform all integrations simultaneously.

### Variables

| | |
|---|---|
| N | called NNN in MATRX. |
| IST | called JST in MAIN = number of intervals; used in first evaluation of above integral. |
| ICA | called JCA in MAIN, = number of points per interval in first evaluation. |
| FXK | the function WIHOP, which is $O_\ell(s'', s')$ minus the delta function. |
| XJERB | $K_\ell(s, s')$. |
| XMAT (KX, IY) | $K_\ell'(s, s')$. |
| SVAL | SONE as calculated in MATRX corresponds to $s'$ in $K_\ell'(s, s')$ and is associated with index IY in XMAT(KY, IY). |
| KX | index associated with s in $K_\ell(s, s'')$. |
| N2 | $= 2 \times N$ |
| E | = 0.01, the relative error in testing convergence (read from data cards in MAIN). |
| NE | number of times convergence test has been made. |
| NOEROR | upper limit of NE read from data cards in MAIN. |
| MOK | index of XMAT element that fails to converge. |

## Operation

Through statement 5009 this subroutine calculates the K' integral minus the delta function by means of Gaussian quadratures, as explained in MAIN (Sec. I). In line 0230 the limits of the integral are tested to ensure that the lower limit XA is less than the upper limit XB. For each SP, KGAUSS calls subroutine KERNL, which returns XKERN. FXK is the Wiener-Hopf resolvent kernel $O_\ell(s'', s')$, and is evaluated in function subroutine WIHOP for each SP and SVAL.

The integration is done by four nested DO loops ending at statements 103, 117, 6, and 7. The outer two allow the variable of integration, SP, to range over the IP points of each of the I intervals into which the integration range is divided. For each SP, the next-to-innermost loop ending in statement 117 finds FXK for successive values of SVAL. For each SVAL = s' the innermost DO loop does the sum for each value of s, i.e., for each XKERN(KX). Thus it is the outer two loops that do the Gaussian sum, and the inner two that give the array of integrals. Note that the loop ending in statement 33 has initialized XMAT to zero. Statement 5009 multiplies the final sum by D because of the transformation dy → Ddt. The factor of 2 from the square-root trick is taken into account in line 1010. Statements 110 through 874 add the delta-function contribution of $O_\ell$ to each element of XMAT. (This is not included in WIHOP, as discussed in Sec. X.) We add the delta term here

$$\int_4^{S1} ds'' \, K_\ell(s_j, s'') \, \delta(s'' - \mathrm{SVAL}) = K_\ell(s_j, \mathrm{SVAL}).$$

A test for the convergence of the integral is done as follows. Each time I or IP is changed, the program returns to statement 5 and finds a new matrix XMAT. Just after statement 5 the array S(KX), KX = 1, 2N is made. Terms in this array correspond to elements of XMAT for the previous I and IP. When KGAUSS is first called, all elements of course, are zero. It is not desirable, when N is large, to test each term of XMAT, so we test the elements XMAT (1, KX) and XMAT (N, KX). The element converges if

$$\left| \frac{\text{element - (element of previous I and IP)}}{\text{element}} \right| < E.$$

If this test fails for any element, the index of the SVAL value of this element is stored in MOK. NE gives the number of times the integration has been done for each element. NOEROR is the number of times the integration is allowed to be done. If it is zero no test is made.

Each time the integral is repeated it is done with a large number of points SP. This number is increased by increasing the number, IP, of points per interval, by increasing L, in statement 109, in steps of two (where IP = L + 1 for successive integrals) until the number of points in each interval is 9. We do not wish to approximate the function by a polynomial of too high a degree; hence the number of intervals is then increased by three, with the number of Gaussian points per interval returned to the original number read in from subroutine MATRX.

## VII. KERNL

This subroutine calculates $K_\ell(s, s')$, which is given by Eq. (5).

For each value of $s'$ given to this subroutine by subroutine KGAUSS, we calculate a one-dimensional array XKERN(KX), the elements of which are values of $K_\ell(s, s')$ for $s$ = SONE(KX). The quantity $K_\ell(s, s')$ is approximate to the extent that $B_\ell^V$ is.

### Variables

| | |
|---|---|
| SP | is $s'$, which is the variable of integration in KGAUSS. |
| NUMBER | known through COMMON is the number of PAIR points for the array PAIR defined initially in MAIN. |
| PAIR | array constructed in MAIN involving function $B_\ell^V$ and array elements added in KERNL. |
| LPARI | number of PAIR points in KERNL. |
| SONE | one-dimensional array of points constructed in MATRX, and corresponding to the values $s$ of XMAT(s, s'). |
| XPAR | previous value of PAIR(4). |
| XSON | previous value of SONE(1). |
| COF | $\sin^2 \pi a_\ell / \pi$. |
| POOR | defined below under Operation. |
| NPAIR | number of POOR points. |
| U | SONE(1) |
| APL | interpolated value of even indexed PAIR elements at a particular point U where $U \leq S1/2$. |
| BPL(II) | $B_\ell^V$ at the points SONE(II). |

| | |
|---|---|
| POO | interpolated value of even-indexed POOR element at particular point $U > S1/2$. |
| N2P | number of PAIR points minus highest one. |
| RHSP | $\rho_\ell(SP)$ evaluated from function subroutine RHO. |

## Operation

XKERN is given by

$$XKERN = [TRANS-SPOOR] \left\{ \pi [SP - SONE(KX)] \right\}^{-1},$$

where

$$POOR(J) = \rho_\ell(s)\, B_\ell^V(s) + \frac{\sin^2 \pi a_\ell}{\pi}\, \ell n(s_1 - s),$$

with $s = SONE(J)$. TANS is the interpolation of the array POOR to the point SP at which we want to know XKERN, and SPOOR is given by

$$SPOOR = B_\ell^V(s)\, \rho_\ell(s') + \frac{\sin^2 \pi a_\ell}{\pi}\, \ell n(s_1 - s).$$

Note that SPOOR depends on both s and s'. The array PAIR constructed in the MAIN program gives $B_\ell^V$ and the points at which it is evaluated for $4 < s < S1$. Lines 0180 through 0229 extrapolate $B_\ell^V$ to its value PAIR(1), by means of a trick discussed in FINT (Sec. XV), which allows an interpolation to continue to extrapolate. The quantity $B_\ell^V$ at S1 is singular, so PAIR(2NUMBER + 4) cannot be evaluated. The array POOR is constructed from corresponding PAIR values and PAIR points (0260-0320). POOR is not singular at S1 and so the extrapolation procedure can be used to evaluate POOR(S1). POOR is singular at 4 and so we find the quantity POOR in the region $S < S1/2$ from its definition in terms of PAIR, rather than from direct interpolation with the array POOR(I). Lines 0600 to 0660 calculate POOR at the given SP. For SP near S1 we find POOR(SP) from the array POOR. For lower values of $SP(\leq S1/2)$ we find POOR(SP) from PAIR in statement 3.

To calculate SPOOR we need to know $B_\ell^V(s)$ at all the values of U = SONE(KX). This calculation is done between line 0390 and 0470 where the array BPL(II) is formed. Again we distinguish $U > S1/2$ and $U \leq S1/2$ in line 0410. For $U \leq S1/2$ statement 644 finds BPL from the array PAIR. For $U > S1/2$, 446 finds POOR at the required point U and then line 0460 finds BPL.

The DO loop ending in statement 444 then finds XKERN(KX) for all Y = SONE(KX) and a given SP. A difficulty arises when the denominator (SP-Y) of XKERN is very small. Lines 0710 through 1100 deal with the case of (SP-Y) less than or equal to 0.01. We compute XKERN for SP + 0.05 and for SP - 0.05 and take an average of the two values. Again we distinguish $(SP \pm 0.05) < S1/2$ in calculating the TANS term, and proceed as above. In adding 0.05 to SP we might obtain (SP + 0.05) greater or equal to S1. Lines 0740 to 0800 deal with this. If SPP = (SP + 0.05) is equal to S1, POOR(S1) is already known and can be used. If SPP is greater than S1 we equate it to S1. As the subroutine now reads we actually exclude this last possibility from the average in line 1080 by means of the indicator XFIX.

## VIII. LAGRAN

This is a very slight modification of an interpolation SHARE routine.[5] However, any interpolation routine that will work for the arrays involved can be used.

To call LAGRAN an array PAIR must be constructed. The even-numbered elements of this array are values of the function; the odd-numbered elements are the values of the points at which the next-highest odd-numbered element is evaluated. The subroutine must also be given the number, NPAIR, of odd-numbered elements. NPAIR is equal to the number of points at which the function is known. Finally the calling program must give X, the point to which the function is interpolated. LAGRAN then calculates ANS, which is the function evaluated at X. X must be inside the range of given PAIR points. If it is not, statements 96, 97, 99, and 300 give the value of X for which the error is made and the array PAIR for which the interpolation is being made. The statement DORIS = SQRT (-1.0) causes an error trace to be printed in this case, so that the sequence leading to an X outside the proper range may be traced. For NPAIR less than 6, a linear interpolation is used; statement 205 does this. Note that the value PAIR(2I + 1) must be monotonically increasing with I.

## IX. RHO

This function subroutine calculates $\rho_\ell(s)$ = RHO(S, XL). RHO is the function

$$\rho_\ell(s) = R_\ell \cdot \left(\frac{s-4}{4}\right)^\ell \sqrt{\frac{s-4}{s}} \; .$$

## X. WIHOP

This function subroutine returns the Wiener-Hopf resolvent kernel minus the delta-function term. From Eq. (11) of reference 6 we see the kernel is given by

$$O_\ell(s_1 - s') = \theta_\ell [X(s), X(s')] / (s_1 - s'),$$

and using Eq. (21) of the same reference, we write the delta-function term explicitly as

$$O_\ell = \frac{\delta(X' - X)}{s_1 - s'} + \frac{\mathrm{THETA}}{s_1 - s'},$$

where THETA is the function defined in Sec. XI (THETA). We make the transformation $ds'/(s - s') = dx'$ and $x' = \log[(s_1 - 4)/(s_1 - s)]$ so that

$$\delta(X' - X)\,dx' = \frac{\delta(X' - X)}{s_1 - s'}\,ds' = \delta(s' - s)ds' .$$

The delta-function term is treated separately in two integrals that call WIHOP, i.e., KGAUSS and NGAUSS.

WIHOP makes the transformation from s and s' to x and x'.

### Variables

| | |
|---|---|
| S | when WIHOP is called by KGAUSS it is equal to the variable of integration SP in KGAUSS; when called by NGAUSS it is the quantity SVAL corresponding to the points s at which we evaluate $N_\ell(s)$. |

SPRIME — when called by KGAUSS it is the quantity SVAL and corresponds to the points $s'$ at which we evaluate $K_\ell'(s, s')$; when called by NGAUSS it is the variable of integration SP.

X — function of S.

XPRIME — function of SPRIME.

## Operation

WIHOP calls function subroutine THETA and calculates THETA/$(s_1 - s')$. Notice that the notation in THETA reverses the order of $x$ and $x'$, i.e.,

$$\theta(X, X') = \text{THETA(XPRIME, X)}.$$

## XI. THETA

This function subroutine calculates the sum

$$\tan \pi a_\ell \, \theta_A(X, X') + \tan^2 \pi a_\ell \, \theta_B(X, X') \tag{13}$$

of the Wiener-Hopf resolvent kernel as discussed in WIHOP (Sec. X). The expressions for $\theta_A$ and $\theta_B$ are given by Eqs. (19) and (20), respectively, of reference 6. Equation (19) of reference 6 is

$$\theta_A(x, x') = \pi^{-1} \sinh a_\ell(x'-x) \exp[-(x'-x)]/\left[1 - \exp[-(x'-x)]\right]. \tag{14}$$

By reordering $\theta_B$ to a form consistent with the program, we obtain

$$\theta_B(X, X') = -i(2\pi)^{-2} \sum_{m=1}^{\infty} \sum_{m=0}^{-\infty} \left\{ -\frac{\Phi_2(K^-m)}{\Phi_1(K^-n)} \frac{\exp[i(K^-_n X' - K^-_m X)]}{K^-_n - K^-_m} \right.$$

$$+ \frac{\Phi_2(K_m)}{\Phi_1(K_n^+)} \frac{\exp[i(K_n^+ X' - K_m^- X)]}{K_n^+ - K_m^-} + \frac{\Phi_2(K_m^+)}{\Phi_1(K_n^-)} \frac{\exp[i(K_n^- X' - K_m^+ X)]}{K_n^- - K_m^+}$$

$$\left. - \frac{\Phi_2(K_m^+)}{\Phi_1(K_n^+)} \frac{\exp[i(K_n^+ X' - K_m^+ X)]}{K_n^+ - K_m^+} \right\}. \qquad (15)$$

Equation (15) of reference 6 gives

$$\Phi_{1\ell}(K) = \Gamma(-iK + a_\ell)\Gamma(-iK - a_\ell)/\Gamma^2(-iK) \quad \Phi_{2\ell}(K) = 1/\Phi_{1\ell}(i - K). \qquad (16)$$

## Variables

| | |
|---|---|
| Y | $x' - x$. |
| GX | $e^{(x' - x)}$. |
| GAY | $\exp[a_\ell(x' - x)]$. |
| SINHAY | $\sinh a_\ell(x' - x)$. |
| TANAL | $(\tan \pi a_\ell)/\pi$. |
| THEA | $a_\ell$ TANAL. |
| THETAA | $\theta_A \pi$ TANAL. |
| PHIL(J) | $\Gamma^2(J + AL)/(J)\,\Gamma(J + 2AL)$. |
| PHIR(J) | $\Gamma^2(J - AL)/\Gamma(J)\,\Gamma(J - 2AL)$. |

## Operation

The sum of Eq. (13) is calculated. This sum in the notation of the program is

THETAA + THETAB

Quantities that depend only on AL are calculated before statement 18, and due to the IF statement in line 0210, they are not recalculated if THETA is called two or more times with the same value of $a_\ell$ (as we expect it to be).

THETAA is calculated by lines 0430 and 0440, which give TANAL and THEA, and by lines 0790, 0800, 0810, and 0820. To calculate THETAA, we write $\pi\theta_A$ for convenience in the form

$$1/2 \left[ e^{a_\ell (x' - x)} - 1/e^{a_\ell (x' - x)} \right] 1/(e^{(x' - x)} - 1) .$$

When $(x' - x)$ approaches zero this expression goes to AL. The IF statement of line 0790 separates the cases where $(x' - x)$ is zero and where it is not. In the former case line 0800 gives THETAA and in the latter case, line 0820.

The rest of the program is concerned with calculating THETAB and in particular the summations shown in Eq. (15). We may write these sums by the shorthand notation

$$F_{m,n} = F_{\pm\pm} = \sum_{m=0}^{-\infty} \sum_{n=1}^{\infty} \frac{\Phi_2(Km^{\pm})}{\Phi_1(K_n^{\pm})} \frac{\exp(-nX' + mX)}{K_n^{+} - K_m^{\pm}} . \qquad (17)$$

The notation of the program is then

$$SF1 = F_{--}$$

$$SF2 = F_{-+}$$

$$SF3 = F_{+-}$$

$$SF4 = F_{++}$$

where the first subsign on the F refers to $K_m^{\pm}$ and the second to $K_n^{\pm}$. Using the relation $K_j^{\pm} = i\,(\pm AL + J)$, we can rewrite THETAB as

$$i/4 \cdot \tan^2 \frac{\pi AL}{\pi^2}\left[SF1 \cdot e^{AL(X'-X)} - SF2 \cdot e^{-AL(X'+X)} - SF3 \cdot e^{AL(X'+X)} + SF4 \cdot e^{-AL(X'-X)}\right]. \tag{18}$$

The exponents in Eq. 18 are calculated in lines 0760, 0770, and 0780.

The sums are calculated as two factors, one that depends on x and x', and one that depends only on AL.

$$SF1 = \Sigma\Sigma\, F1(N, M) \cdot e^{-NX' + M'X}$$

$$SF2 = \Sigma\Sigma\, F2(N, M) \cdot e^{-NX' + M'X}$$

$$SF3 = \Sigma\Sigma\, F3(N, M) \cdot e^{-NX' + M'X}$$

$$SF4 = \Sigma\Sigma\, F4(N, M) \cdot e^{-NX' + M'X}$$

The number of terms needed in the sum depends on the exponent. For x or x' less than or equal to 0.2, more terms are needed. It has been determined that for small x or x', twenty terms are needed. For x and x' greater than 0.2, ten terms are enough. Lines 0110 through 0160 determine the number of terms, MAX, to be used.

MAXX is MAX of the previous run of THETA. XX and XXPRIM are the X and XPRIME of the previous run. If x or x' do not change, $e^{MX}$ or $e^{-NX'}$ do not need to be recalculated. The IF statements of lines 0570 and 0630 delete these calculations for unchanged x and x'. The IF statement of line 0170 tests MAXX, and, if it is less than MAX, sets XX and XXPRIM equal to -1.0, so that the IF statements of line 0570 and 0630 will not delete the calculation even if x or x' is unaltered.

We make the transformation

$$\sum_{N=1}^{\infty} \sum_{M'=0}^{-\infty} \to \sum_{N=1}^{\infty} \sum_{M=1}^{\infty}$$

where M = 1 - M'.

Statements 17 to 666 calculate the terms F1, F2, F3, and F4 that do not depend on x and x'. These terms are defined as products of PHIR and PHIL. For example,

$$F1 = \frac{\Gamma^2(M+AL)}{\Gamma(M)\Gamma(M+2AL)} \frac{\Gamma^2(N-AL)}{\Gamma(N)\Gamma(N-2AL)} \frac{1}{N+M-1} = \frac{PHIL(M)PHIR(N)}{N+M-1} .$$

Notice that PHIR and PHIL involve only five different gamma functions. These are

$$PHIL(I) = \Gamma^2(1+AL)/\Gamma(1)\Gamma(1+2AL) = V_3{}^2/V1\ V5$$

$$PHIR(I) = \Gamma^2(1-AL)/\Gamma(1)\Gamma(1-2AL) = V_3{}^2/V1\ V4.$$

Subroutine GAMM is called to evaluate V1, V2, V3, V4, and V5 in statements between statement 17 and line 0300. For index greater than one, the DO loop ending in statement 101 calculates PHIL and PHIR. The corresponding gamma functions are found in this loop from V1, V2, V3, V4, and

V5 by means of the relation $\Gamma(M + 1) = M\Gamma(M)$. The DO loop ending in statement 666 then finds F1, F2, F3, and F4.

The sums SF1, SF2, SF3, and SF4 are done in the DO loop ending in statement 11. The factors depending on AL have been calculated for MAX = 20 since the sums are calculated many times for each AL. The DO loop uses as many as needed. Consider the limit of $\theta_B$ as $a_\ell$ approaches 1/2. As explained in reference 6, there are cancellations between F3 and F1 and between F4 and F2.

## XII. GAMM

This SHARE routine (C3 EO GAMA provided by C3 EO LEGN) has been modified to calculate the gamma function only for real arguments. The argument must also be nonzero and positive. The calling program gives GAMM(A1, F1) the argument A1. The program calculates

$$F1 = \Gamma(A1).$$

## XIII. MATINV

This is a SHARE routine very slightly modified for use with this program.[7] For the matrix approximation of an integral equation

$$A^{-1} B = ANS,$$

this subroutine inverts the matrix and finds the solution ANS, and then calculates the determinant DETERM.

The calling program must give to MATINV(N, B, M, DETERM) three quantities: (a) the quantity N, where the size of the matrix is N by N; (b) the array B(J, M), and (c) the quantity M, which is the second index of B and is used as an indicator. When M is zero or negative, only the determinant is inverted by the subroutine. For our purposes M always equals one but is written explicitly to fit the notation of the SHARE routine.

The matrix A to be inverted is in common in the calling and called routines.

To conserve space in core, the quantity $A^{-1}$ is stored at A and ANS at B.

The running time is proportional to $N^3$.

## XIV. NGAUSS

This subroutine calculates the function $N_\ell(s)$ where

$$N_\ell(s_i) = \int_{XA}^{XB} ds' \; O_\ell(s_i, s_j') \; FNOL(s_j')$$

where $FNOL(s_j') = N_\ell^0(s_j')$. The function $N_\ell(s)$ is returned as a one-dimensional array GAUS(KX), the elements of which are integrals.

<u>Variables</u>

| | |
|---|---|
| N | NNS determined in MAIN; the number of PTS points. |
| PTS | the array of points at which we evaluate N; called SVAL and MAIN. |
| GAUS | $N_\ell$ |
| IST | called JST in MAIN. |
| ICA | called JCA in MAIN. |

FNOL function subroutine FINT.

FYN FNOL(s).

FXN WIHOP, which is $O_\ell$ minus the delta-function term.

SP s'.

## Operation

The operation is the same as that discussed in subroutine KGAUSS, with the following changes. The array XMAT (KX, IY) is replaced by GAUS (KX), so that we are doing N, rather than $N^2$, integrals. Notice that now it is the second parameter in WIHOP that is the variable of integration. These changes are noted in WIHOP.

## XV. FINT

Function subroutine FINT(S) simply interpolates the array PAIR to a point S.

## Variables

PAIR array, defined in Glossary, which consists of values of the function and points at which it is evaluated; the array is in COMMON and represents $N_\ell^0(s)$ or $N_\ell(s)$ according to the point in the program at which FINT is called.

NPAIR number of PAIR points in common.

LPAIR number of PAIR points in FINT.

N number of PAIR elements in FINT.

## Operation

The array PAIR is not defined for S = 4 and S = S1. The function FINT(S) may be required for any point of S within these limits, and is found at a point S by interpolation. In order that the interpolation routine can be used, S must lie between the highest and the lowest points in PAIR.

In order to find FINT(s), for $4 < s <$ PAIR(1) and PAIR(2N -1) $< s < 1$, FINT extrapolates by adding, to the array PAIR, two new points at $s = \pm 100 s_1$. As long as the values of the associated even-numbered PAIR elements are taken not greatly different from the other even-numbered PAIR values in the array, LAGRAN returns an answer independent of the values taken for these two elements. This is, of course, because LAGRAN weights the six points used in the interpolation inversely with their distance from the point to which they are being interpolated.

In the DO loop ending in statement 3, the PAIR index is redefined so the new points at each end can be added. The process is

$$\text{PAIR}(N - 4) = \text{PAIR}(N - 6)$$

$$\text{PAIR}(3) = \text{PAIR}(1).$$

Then PAIR (2) is set temporarily equal to PAIR(4). LAGRAN is called and given the value of the function at 4 so now [ PAIR(2) = ANS and PAIR(1)] is returned to the value 4.

Statement 2 excludes S greater than or equal to S1.

Statement 8 interpolates PAIR to the point S. LAGRAN returns FINT, which is the value of FINT(S) returned to the calling program.

Operations up to statements 2 need not be performed if the preceding time FINT was called the array PAIR was the same. The initial IF statement tests for a change in PAIR.

## XVI. DGAUSS

This subroutine calculates the function

$$D = 1 - \pi^{-1} \int_4^{S1} \frac{\rho_\ell(s') \, XNUM(s')}{s' - s} \, ds' . \qquad (19)$$

The MAIN program calls DGAUSS in two separate calling statements, which refer to as Call(A) and Call(B).

Call(A) calculates

$$D_\ell^{(s)} = 1 - \pi^{-1} \int_4^{S1} \rho_\ell(s') \, N_\ell(s')/(s - s')$$

and finds the zeroes of D and the derivative of D at the zeroes.

Call(B) calculates for the zero point SR the quantity

$$1 - \pi^{-1} \int_4^{S1} \rho_\ell(s') \, B_\ell^V(s') \, N_\ell(s')/(s' - SR).$$

### Variables

| | |
|---|---|
| SVA | quantity SVAL(J) defined in MAIN. |
| N | defined in MAIN; for call(B) equal to 1, for call(A) the number of SVAL values. |
| SUBTR | discussed below. |
| IST | JSD defined in MAIN. |

| | |
|---|---|
| GAUS(KX) | $D(s_i)$. |
| FX | XNUM × RHO; for Call(A) is RHO × FINT; for Call (B) is RHO × FANT. |
| NZ | number of zeroes of D. |
| SR | points where D is zero. |
| ZERO(NZ) | array of points SR. |
| WIDTH | slope of D at zero (not the resonance width). |

## Operation

Consider the integral

$$I = P \int_4^{S1} ds' \; \rho_\ell(s') \; XNUM(s')/(s' - SVA).$$

The quantity SVA may extend from $-\infty$ to $+\infty$. Outside of the integration range 4 to SI, the denominator cannot vanish. For SVA in the range of integration, we do a "subtraction trick" by adding and subtracting the term

$$SUBTR = \rho_\ell(SVA) \; XNUM(SVA),$$

so that the integral becomes

$$I = \int_4^{SI} ds' \; \frac{\rho_\ell(s') XNUM(s') - \rho_\ell(SVA) \; XNUM(SVA)}{S' - SVA}$$

$$+ \rho_\ell(SVA) \, XNUM(SVA) \int_4^{SI} \frac{ds'}{s'-SVA} = \int_4^{SI} ds' \; \frac{\rho_\ell(s') XNUM(s') - \rho_\ell(SVA) XNUM(SVA)}{s' - SVA}$$

$$+ \log \left| \frac{SI - SVA}{SVA - 4} \right| \; \rho_\ell(SVA) \, XNUM(SVA) = 2I1 + I2.$$

We have taken the absolute value of the log since D is the principal value integral. The IF statement in line 0270 tests to determine whether SVA(J) is in the range of integration. If it is outside this range, SUBTR(J) = 0. Statement 12 evaluates SUBTR(J) for SVA in the range of integration.

From statement 11 to statement 110 the integration proceeds exactly as in NGAUSS and KGAUSS, according to the method of Gaussian quadratures, except, of course, there is no delta-function term. The integral is approximated by the one-dimensional array GAUS(J). The DO loop ending in statement 874 does the subtraction

$$D = 1 - (2/\pi)\, I1,$$

where the factor of 2 comes from the square-root trick. Statement 877 is, for the element where SUBTR = 0,

$$D = 1 - (2/\pi)\, I1.$$

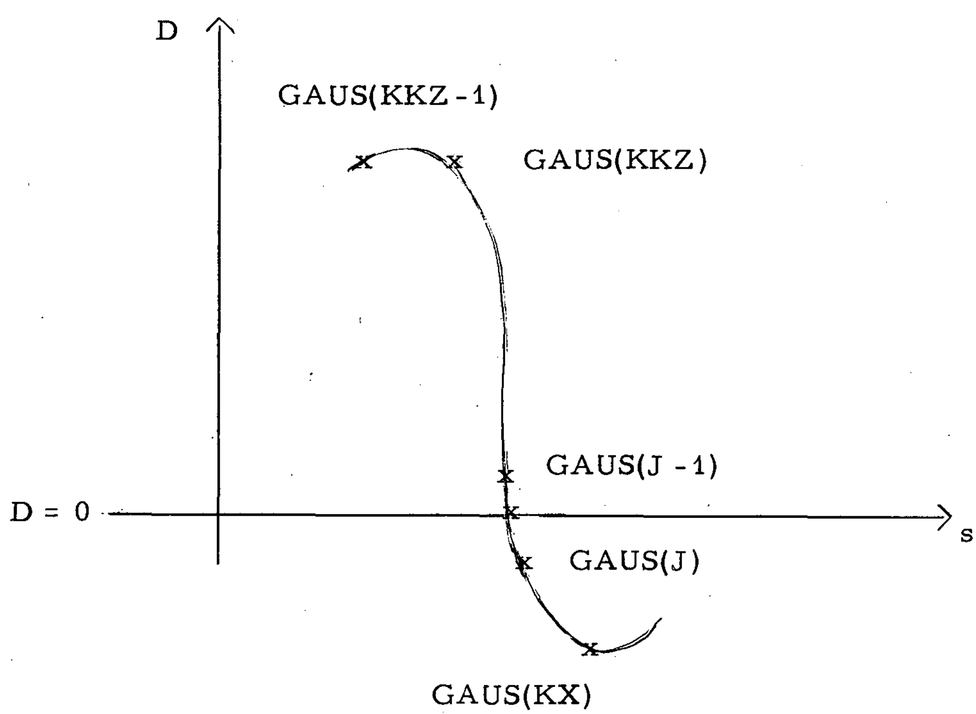
The statement in line 0950 gives GAUS(J) for SVA(J) in the range of integration

$$D = 1 - (2/\pi)\,[I1 + 1/2\ \ I2].$$

There is no square-root trick involved in I2, so I2 is multiplied by one half.

Through statement 874 CALLA and CALLB are the same. The IF statement just before statement 888 separates the two. CALLB uses DGAUSS only to this point. Since, for CALLB, SVA is outside the range of integration, for this case SUBTR = 0.

The rest of the subroutine is used by CALLA. The DO loop ending in statement 601 finds the zeroes of D, and D' at the zeroes. Through statement 660 DGAUSS finds the zeroes. An array is formed with the odd-numbered elements equal to the values of the function GAUS. The even-numbered elements are the corresponding points from the array SVAL. Then, interpolating to the 'point' GAUS = 0, we find the value of the 'function,' which is the value of S at which D is zero. To construct the array, we need to know the slope of the function, since the points of the array (odd-numbered elements) given to LAGRAN must be consecutive. The values of GAUS must then be monotonic. The points GAUS must of course include GAUS = 0. The IF statement in line 0990 for each loop (of the DO loop ending in statement 601) examines sets of two consecutive GAUS values in order to place the zero. If the product is negative the zero is between them. The index J is then the value of the index of the first GAUS element after GAUS = 0. Each time a zero is found, NZ is increased by one (statement 602). The next IF statement tests to see if the slope is negative or positive. If the slope is positive the program skips to statement 606 if negative to 605.



For the case of negative slope, the DO loop ending in statement 607 tests GAUS(K2) from index J to N to see how far the region of negative slope extends. When the slope changes, the IF statement sends the program out of the loop. The index of the last value of GAUS before the slope turns up is KZ. The DO loop ending in statement 609 constructs the array ZPAIR with values of GAUS for points

ZPAIR(1) = GAUS(K2)

ZPAIR(2) = SVAL(K2).

The construction is continued with consecutively higher values of GAUS for the points (odd-numbered elements). The IF statement (1120) tests for the change in slope sign to the left of the zero and sends the program out of the loop when the change is found.

Then the interpolation subroutine LAGRAN is called and returns the point ANS at which D = 0. The value is the $NZ^{th}$ element of the array ZERO. The DO loop ending in statement 650 rearranges the array ZPAIR so that the odd-numbered elements are the points SVAL and even-numbered elements the values of D. The program then goes to statement 660 to find D'.

If the IF statement in line 1010 found the slope through the zero was positive, then lines 1230 through 1420 find the zero of D in the same way.

Lines 1430 through 1490 find D' by taking the average of the slope from the zero point SR to SR + 0.5 and the slope from SR to SR-0.5. GAUS at these points is found by calling LAGRAN.

# XVII. FANT

This function subroutine finds FANT(S) by evaluating the function of PEER at a point S.

## Variable

PAIR — is defined in MAIN where it is called PEER, and is known in FANT through block COMMON. PEER has values of the function $B_\ell^V$ as even-numbered elements.

## Operation

Through statement 8 FANT is identical to FINT.

It differs in the statement after 8, which defines FANT to be the product of FANT(S) found in this subroutine, and FINT(S) found in FINT. FINT will now necessarily involve the same array PAIR as PEER does.

For example, in the present program when DGAUSS calls FANT, it returns

$$FANT(s) = B_\ell^V(s)\ N_\ell(s).$$

# GLOSSARY

| | |
|---|---|
| AL | is the quantity $a_\ell = \delta(s_1)/\pi$. |
| BPL | array formed in KERNL of values of $B_\ell^V$ at the points SONE. |
| BTL | the function $B_\ell^V$ evaluated in BTL for a certain set of points determined in MAIN. |
| C | length of one of the I intervals used in method of Gaussian quadratures. |
| D | C/2 |
| DAL | increment by which XL or AL is increased in successive runs of the program. |
| DERIV | derivative of $D(s_0)$ at points where $D(s_0) = 0$; called WIDTH in DGAUSS. |
| E | relative error in testing convergence. |
| FANT | function subroutine that interpolates the array PEER. |
| FINT | function subroutine that interpolates the array PAIR. |
| GAUS | name of arrays returned by NGAUSS and DGAUSS. For the former it corresponds to values of $N_\ell(s)$, for the latter to $D_\ell(s)$ or $1 - N_\ell(SR)$. |
| H | array of Gaussian weights. |
| I | number of intervals the region of integration is divided into in KGAUSS, NGAUSS, and DGAUSS. |
| ICA | indicator of the initial number of Gaussian points per interval in KGAUSS, NGAUSS, and DGAUSS. |
| IP | number of Gaussian points = (1 + ICA) initially. |
| IST | initial number of intervals. |
| KNOW | indicator in MAIN that chooses XL and AL for the next run in the over-all DO loop. |
| MOK | index of element that did not converge. |

| | |
|---|---|
| NAL | number of loops of entire program. |
| NE | counter of convergence tries. |
| NN | degree m of Legendre polynomial $P_m$ by which we approximate the integrand in the method of Gaussian quadratures. |
| NOEROR | upper limit of NE. |
| NZ | number of zeroes of D(s). |
| PAIR | see discussion in MAIN array of points and values of functions of the form X1, f(X1), X2, f(X2), X3, f(X3), $\cdots$. |
| PEER | array (formed in MAIN) equivalent to PAIR with $f(x) = B_\ell^V(x)$. |
| POOR | array defined in KERNL by removal from $B_\ell^V$ of the log singularity at S = S1. |
| RHO | $\rho_\ell(s)$. |
| RL | inelasticity factor in $\rho_\ell$. |
| RWIDR | $\Gamma_\rho$. |
| S1 | strip boundary, upper branch point in the N and D functions, used throughout the program, usually as upper limit of integration. |
| SONE | array of points constructed in MATRX and used by KGAUSS and KERNL. The quantity $K_\ell'(s, s')$ is evaluated at the points (s and s'). |
| SVAL | first the array of points at which $N_\ell(s)$ is evaluated, and then the array at which $D_\ell(s)$ is evaluated. |
| T | Gaussian points. |
| TA | translation from lower limit of integration. |
| WIDTH | D'(SR, called DERIV in MAIN. |
| WIHOP | Wiener-Hopf resolvent kernel minus $\delta(s'-s)$. |
| XA | lower limit of integration. |

| | |
|---|---|
| XB | S1. |
| XL | angular momentum, in center of mass of two-pion system ($\ell$). |
| XMAT | name of matrix operator approximating $\int ds' K_\ell'(s, s')$. name of inverted matrix operator $[1-\int ds' K_\ell'(s, s')]^{-1}$. |
| YANS | equal to array BPL in communicating with subroutinte MATINV. |
| YT | equal to $\sqrt{S1 - SP}$. |
| ZERO | array of points S at which D(s) = 0. |
| ZPAIR | array like PAIR involving the function D(s) in DGAUSS. |

## ACKNOWLEDGMENTS

We are grateful to Professor Geoffrey F. Chew for many helpful conversations, and to Dr. Peter Collins for a variety of assistance including an independent check of the program. One of us (VLT) wishes to thank Dr. David L. Judd for his hospitality at the Lawrence Radiation Laboratory.

This work was performed under the auspices of the U. S. Atomic Energy Commission.

# APPENDICES

## A. MAIN

```
$IBFTC PIND
      EXTERNAL  FINT,FANT
      COMMON /IN/RL
      COMMON /R1/ RWIDR
      DIMENSIONXMAT(50,50)
      DIMENSION PAIR(200),BPL(100),GAUS(100)
      DIMENSION SONE(100),YANS(50,1)
      DIMENSION SVAL(100)
      DIMENSION ZERO(10),DERIV(10)
      DIMENSION IDO(10),H(10,10),X(10,10)
      COMMON  S1      , NUMBER , XL      , AL      , PAIR    , BPL
      COMMON  NNN     , SONE   , NOEROR , YANS    , IDO     , H
      COMMON  X       , XMAT   , NPAIR
      COMMON /P2/ PEER,NPEER
      DIMENSION PEER(200)
      DO 32 M=1,8
      READ  (2,12)NN,(X(J,NN),H(J,NN),J=1,NN)
   12 FORMAT(I10,(6F10.6))
      WRITE (3,13)NN,(X(J,NN),H(J,NN),J=1,NN)
   13 FORMAT(20H0 GAUSS1 INPUT DATA I10,(1P6E15.7))
      IDO(M)=NN
   32 CONTINUE
   66 READ  (2,60)IST,JST,ICA,JCA,NOEROR,NAL,S1,DAL,E,AL,XL,RWIDR
     X,RL
      WRITE (3,60)IST,JST,ICA,JCA,NOEROR,NAL,S1,DAL,E,AL,XL,RWIDR
     X,RL
   60 FORMAT(6I5/7F10.5)
      READ (2,1236) KNOW
 1236 FORMAT(I5)
      DO 555  LLL=1,NAL
      WRITE (3,1212)AL,XL,S1
 1212 FORMAT( 1HA        20X,   62HSOLUTION OF THE  PI PI N/D EQUATIONS FO
     1R AL, L, S1, EQUAL      , 3F10.4/   /)
      WRITE (3,1235)RL
 1235 FORMAT(20X,9HRL EQUALS,F10.5)
      WRITE (3,1234) RWIDR
 1234 FORMAT (20X, 70HWIDTH OF THE EXCHANGED RHO (IN THE DIRECT CHANNEL
     XT=1 STATE) IS                     ,F6.2////)
      XA=4.0
      XB=S1
      A=0.0
      B=SQRT(XB-XA)
      I=8
      IP=IDO(ICA)
      NUMBER=I*IP
    5 TA=A
      FI=I
      C=(B-A)/FI
      D=C*0.5
      PAIR(1)=4.0
      II=NUMBER
      DO 7 K=1,I
      DO 6 J=1,IP
      T=X(J,IP)
      YT=TA+D*(T+1.0)
      SP=XB-YT**2
```

```
      PAIR(2*II+1)=SP
      PAIR(2*II+2)=BTL(SP)
      II=II-1
6     CONTINUE
      TA=TA+C
7     CONTINUE
      NPEER=NUMBER
      DO 901  JJ=1,NUMBER
      PEER(2*JJ)=PAIR(2*JJ+2)
      PEER(2*JJ-1)=PAIR(2*JJ+1)
  901 CONTINUE
      CALL CLOCKT(T1)
      CALL MATRX(XL,E,        IST,ICA,JST,JCA)
      CALL CLOCKT(T2)
      TAA=T2-T1
      WRITE (3,111)TAA
      NPAIR=NNN
      DO 445  I=1,NNN
      PAIR(2*I)=YANS(I,1)
      PAIR(2*I-1)=SONE(I)
  445 CONTINUE
      NNS=25
      XNNS=FLOAT(NNS)
      XYX=SQRT(S1-4.0)
      XYY=XYX/(XNNS+1.0)
      XYX=XYY
      XXX=XYX
      DO 311 IJ=1,NNS
      K2=NNS+1-IJ
      SVAL(K2)=S1-XXX**2
  311 XXX=XXX+XYX
      CALL NGAUSS(4.0,S1,E,NNS,XL,GAUS,JST,JCA,FINT,SVAL,AL)
      N2S=2*NNN+2
      WRITE(3,3330) (PAIR(KY),KY=1,N2S)
 3330 FORMAT(////50X,16HN-SUBZERO VALUES///
     X(1X,2HS=,E13.5,4X,2HN=,E13.5,9X,2HS=,E13.5,4X,2HN=,E13.5,9X,2HS=,
     XE13.5,4X,2HN=,E13.5))
      DO 444  KY=1,NNS
      PAIR(2*KY-1)=SVAL(KY)
      PAIR(2*KY)=GAUS(KY)
  444 CONTINUE
      N2S=2*NNS
      WRITE(3,3331) (PAIR(KY),KY=1,N2S)
 3331 FORMAT(////50X,8HN-VALUES///
     X(1X,2HS=,E13.5,4X,2HN=,E13.5,9X,2HS=,E13.5,4X,2HN=,E13.5,9X,2HS=,
     XE13.5,4X,2HN=,E13.5))
      XN=N
      NPAIR=NNS
      JSD=8
      N=100
      STS=S1
      DS=5.
      DO 101 I=1,N
      II=N-I+1
      STS=STS-DS
 1101 SVAL(II)=STS
  101 CONTINUE
```

```
      CALL        DGAUSS(XA,XB,E,N,XL,SVAL,GAUS,JSD,ICA,FINT,ZERO,DERIV,      0990
     1NZ)                                                                     1000
      WRITE (3,103)(SVAL(I),GAUS(I),I=1,N )                                   1030
  103 FORMAT(/////50X,11HVALUES OF D///
     X(1X,2HS=,E13.5,4X,2HD=,E13.5,9X,2HS=,E13.5,4X,2HD=,E13.5,9X,2HS=,
     XE13.5,4X,2HD=,E13.5))
      DO 104 III=1,NZ
      SR=ZERO(III)
      IF(SR-4.) 105,106,106
  106 CONTINUE
      REDR=FINT(SR)/DERIV(III)
      WIDTH=RHO(SR,XL)*REDR/(2.*SQRT(SR))
      WRITE (3,107) SR,REDR,WIDTH
  107 FORMAT(//10X,16HRESONANCE ENERGY,E15.5,10X,15HREDUCED RESIDUE,E
     X15.5,10X,5HWIDTH,E15.5)
      GO TO 104
  105 CONTINUE
      N=1
      SVAL(1)=SR
      CALL        DGAUSS(XA,XB,E,N,XL,SVAL,GAUS,JSD,ICA,FANT,ZERO,DERIV,      0990
     1NZ)
      REDR=(1.-GAUS(1))/DERIV(III)
      WRITE (3,108)SR,REDR
  108 FORMAT(//10X,16HRESONANCE ENERGY,E15.5,10X,15HREDUCED RESIDUE,E15
     X.5)
  104 CONTINUE
      CALL CLOCKT(T3)                                                         1050
      TBA=T3-T2                                                               1060
      WRITE (3,111)TBA                                                        1070
  111 FORMAT(////20X,5HTIME=,E10.4///)                                        1080
      IF (KNOW) 1241,1242,1241
 1241 CONTINUE
      AL=AL+DAL
      GO TO 555
 1242 CONTINUE
      XL=XL+DAL
  555 CONTINUE                                                                1090
      GO TO 66                                                                1100
      END                                                                     1110
                         *** 'END-OF-FILE' CARD ***
```

## B. BTL

```
$IBFTC BTL
      FUNCTION BTL(S)
C     LABEL
      COMMON /R1/ RWIDR
C THE FOLLOWING STATEMENT(S) HAVE BEEN MANUFACTURED BY THE TRANSLATOR TO
C   COMPENSATE FOR THE FACT THAT EQUIVALENCE DOES NOT REORDER COMMON---
      COMMON  S1      , NUMBER , XL      , AL
C     COMMON S1, NUMBER, XL, AL
      XR=SQRT(28.0)
      QS2=(S-4.0)/4.0
      Z=1.0+28.0/(2.0*QS2)
      ZR=ABS((1.+Z)/(1.-Z))
      IF(ABS(XL)-.001) 1,2,2
    2 IF(ABS(XL-1.0)-.001) 3,4,4
    3 Q1FN=Z/2.0*ALOG(ZR)-1.0
      GO TO 7
    1 Q1FN=.5*ALOG(ZR)
      GO TO 7
    4 CONTINUE
      PXL=3.14159*XL
      CALL LEGP1(XL,Z,F1)
      Q1FN=3.14159*COS(PXL)/SIN(PXL)*F1
    7 CONTINUE
      XXL=XL+1.0
      BPRL=3.0*XR*(1.0+S/12.0)*Q1FN/QS2**XXL
      BPRL=.5*BPRL*RWIDR
      BPPQ=(SIN(3.14159*AL))**2*ALOG(2.0*(S1-S)/(3.0*S1))/(3.14159*RHO(S
     11,XL))
      BTL=BPRL-BPPQ
      RETURN
      END
                  *** 'END-OF-FILE' CARD ***
```

## C. LEGP1

```
$IBFTC LEGP1
      SUBROUTINE LEGP1(A1,Z,F1)
C     LABEL
 10   IF((ABS(Z))**(2.0*A1+1.0)-1000000.0)2,11,11
 11   F1=0.0
      GO TO 3
 2    CALL HYPM(A1/2.0+1.0,A1/2.0+.5,A1+1.5,1.0/Z**2,.000001,F1)
      IF(ALAST-A1)5,4,5
5     CALL GAMM(A1+1.0,F3)
      CALL GAMM(A1+1.5,F5)
4     F7=SIN(3.14159265*A1)/COS(3.14159265*A1)
      X1=ALOG(2.0*ABS(Z))
      X2=EXP(-(A1+1.0)*X1)*0.5641895
      F1=X2*F1*F3*F7/F5
      ALAST=A1
 3    RETURN
      END
                 *** 'END-OF-FILE' CARD ***
```

## D. HYPM

```
$IBFTC HYPM
      SUBROUTINE HYPM(A1,B1,C1,Z,EP,F1)
C     LABEL
      IF(ABS(Z)-1.0)1,2,2
 2    WRITE (3,3)
 3    FORMAT(///47H ARGUMENT OF HYPFN GREATER THAN OR EQUAL TO ONE///)
      F1=0.0
      GO TO 5
 1    I=0
      F1=1.0
      A3=A1
      B3=B1
      C3=C1
      S1=1.0
      A=1.0
    4 S3=S1*A3
      S1=S3*B3
      S3=S1/C3
      S1=S3*Z/A
      F1=F1+S1
      EP1=ABS(S1/F1)
      IF(EP1-EP)5,6,6
 6    I=I+1
      A3=A3+1.0
      B3=B3+1.0
      C3=C3+1.0
      A=A+1.0
      IF(I-30)4,9,9
 9    IF(EP1-0.01)5,7,7
  7   EP2=ABS(S1)
      WRITE (3,8)Z,A1,B1,C1,EP1,EP2
 8    FORMAT(    44H ERROR IN HYPFN ROUTINE FOR ARGUMENT EQUALS E10.4
     1,10X,13HINDICES EQUAL   3E14.4/   23H RELATIVE ERROR EQUALS E15.7,
     2   23H ABSOLUTE ERROR EQUALS E25.7)
 5    RETURN
      END
                   *** 'END-OF-FILE' CARD ***
```

## E. MATRX

```
$IBFTC MATRX
      SUBROUTINE MATRX(XL,E,        IST,ICA,JST,JCA)
C     LABEL
      DIMENSION PAIR(200),BPL(100),GAUS(100)
      DIMENSION IDO(10),H(10,10),X(10,10)
      DIMENSION XMAT(50,50),YANS(50,1)
      DIMENSION SONE(100)
C THE FOLLOWING STATEMENT(S) HAVE BEEN MANUFACTURED BY THE TRANSLATOR TO
C   COMPENSATE FOR THE FACT THAT EQUIVALENCE DOES NOT REORDER COMMON---
      COMMON  S1      , NUMBER , XL     , AL     , PAIR   , BPL
      COMMON  NNN     , SONE   , NOEROR , YANS   , IDO    , H
      COMMON  X       , XMAT
C     COMMON S1,NUMBER,XL,AL,PAIR,BPL,NNN,SONE,NOEROR,YANS,IDO,H,X,XMAT
C     DIMENSION PAIR(200),BPL(100),GAUS(100)
C     DIMENSION IDO(10),H(10,10),X(10,10)
C     DIMENSION XMAT(50,50),YANS(50,1)
C     DIMENSION SONE(100)
      XA=4.0
      XB=S1
      M=1
      A=0.0
      B=SQRT(XB-XA)
      I=IST
      IP=IDO(ICA)
      NNN=I*IP
    5 TA=A
      FI=I
      C=(B-A)/FI
      D=C*0.5
      II=NNN
      DO 7 K=1,I
      DO 6 J=1,IP
      T=X(J,IP)
      YT=TA+D*(T+1.0)
      SP=XB-YT**2
      SONE(II)=SP
      II=II-1
6     CONTINUE
      TA=TA+C
7     CONTINUE
      TA=A
      KK=NNN
      CALL KGAUSS(XA,XB,E,NNN   ,XL,          JST,JCA)
      DO 77  K=1,I
      DO 66  J=1,IP
      T=X(J,IP)
      YT=TA+D*(T+1.0)
      DO 103    III=1,NNN
      XMAT(III,KK)=-2.0*YT*H(J,IP)*XMAT(III,KK)*D
      IF (III-KK) 103,8000,103
 8000 XMAT(III,KK)=1.0+XMAT(III,KK)
103   CONTINUE
      KK=KK-1
   66 CONTINUE
      TA=TA+C
   77 CONTINUE
      DO 9999  JJ=1,NNN
```

```
9999 YANS(JJ,M)=BPL(JJ)
     CALL MATINV(NNN,YANS,M,DETERM)
     RETURN
     END
                   *** 'END-OF-FILE' CARD ***
```

## F. KGAUSS

```
$IBFTC KGAUSS
      SUBROUTINE KGAUSS(XA,XB,E,N,XL,                 IST,ICA)
C     LABEL
      DIMENSION XMAT(50,50)
      DIMENSION PAIR(200),BPL(100),GAUS(100)
      DIMENSION IDO(10),H(10,10),X(10,10)
      DIMENSION S(100),SONE(100),XKERN(100)
      DIMENSION               YANS(50,1)
C THE FOLLOWING STATEMENT(S) HAVE BEEN MANUFACTURED BY THE TRANSLATOR TO
C   COMPENSATE FOR THE FACT THAT EQUIVALENCE DOES NOT REORDER COMMON---
      COMMON  S1      , NUMBER , XL      , AL      , PAIR    , BPL
      COMMON  NNN     , SONE    , NOEROR , YANS    , IDO     , H
      COMMON  X       , XMAT
C     COMMON S1,NUMBER,XL,AL,PAIR,BPL,NNN,SONE,NOEROR,YANS,IDO,H,X,XMAT
C     DIMENSION XMAT(50,50)
C     DIMENSION PAIR(200),BPL(100),GAUS(100)
C     DIMENSION IDO(10),H(10,10),X(10,10)
C     DIMENSION S(100),SONE(100),XKERN(100)
C     DIMENSION               YANS(50,1)
      N2=2*N
      A=0.0
      B=SQRT(XB-XA)
      NE=0
      IF(XA-XB) 34,24,24
   34 CONTINUE
      I=IST
      IP=IDO(ICA)
      L=ICA
    5 TA=A
      FI=I
      C=(B-A)/FI
      D=C*0.5
      DO 100 KX=1,N
      S(KX)=XMAT(1,KX)
      KXN=KX+N
      S(KX N)=XMAT(N,KX)
  100 CONTINUE
      DO 33 KX=1,N
      DO 33 IY=1,N
      XMAT(IY,KX)=0.0
   33 CONTINUE
      DO 7 K=1,I
      DO 6 J=1,IP
      T=X(J,IP)
      YT=TA+D*(T+1.0)
      SP=XB-YT**2
      CALL KERNL(  SP,      XKERN)
      DO 117  IY=1,N
      SVAL=SONE(IY)
      FXK=WIHOP(SP,SVAL,S1,AL)
      DO 103 KX=1,N
      XMAT(KX,IY)=XMAT(KX,IY)+H(J,IP)*YT*FXK*XKERN(KX)
103   CONTINUE
  117 CONTINUE
6     CONTINUE
      TA=TA+C
7     CONTINUE
```

```
      DO 5009 KX=1,N
      DO 5009  IY=1,N
 5009 XMAT(KX,IY)=XMAT(KX,IY)*D
      IF(NE) 107,107,5002
 5002 DO 927 KX=1,N
      S(KX)=(XMAT(1,KX)-S(KX))/XMAT(1,KX)
      KXN=KX+N
      S(KX   N)=(XMAT(N,KX)-S(KX   N))/XMAT(N,KX)
  927 CONTINUE
      DO 106  KX=1,N2
      IF (ABS(S(KX))-E) 106,106,804
  804 MOK=KX
      GO TO 107
106   CONTINUE
      GO TO 110
107   CONTINUE
      WRITE (3,875)NE,I,L,MOK
  875 FORMAT(1X,3HKE=,I4,5X,2HI=,I4,5X,2HL=,I4,5X,4HMOK=,I4)
      IF (NE-NOERCR) 109,110,110
109   L=L+2
      L=L+2
      NE=NE+1
      IF(L-8)28,28,900
28    IP=IDO(L)
      GO TO 5
  900 I=I+1
      I=I+2
      L=ICA
      IP=IDO(L)
      GO TO 5
24    CONTINUE
      WRITE (3,3)
    3 FORMAT(86HO**REJECTED**LOWER LIMIT OF INTEGRATION GREATER THAN OR
     1EQUAL TO UPPER LIMIT IN KGAUSS)
      GO TO 871
110   CONTINUE
      DO 874  K=1,N
      SVAL=SONE(K)
      CALL KERNL(SVAL,XKERN)
      DO 874 J=1,N
      XMAT(J,K)=2.0*XMAT(J,K)+XKERN(J)
  874 CONTINUE
871   RETURN
      END
                   *** 'END-OF-FILE' CARD ***
```

G. KERNL

```
$IBFTC KERNL
      SUBROUTINE KERNL(SP,XKERN)
      DIMENSION PAIR(200),BPL(100),SONE(100),XKERN(100)
      DIMENSION POOR(200)
C THE FOLLOWING STATEMENT(S) HAVE BEEN MANUFACTURED BY THE TRANSLATOR TO
C   COMPENSATE FOR THE FACT THAT EQUIVALENCE DOES NOT REORDER COMMON---
      COMMON  S1      , NUMBER , XL      , AL      , PAIR    , BPL
      COMMON  NNN     , SONE
C     COMMON S1,NUMBER,XL,AL,PAIR,BPL,NNN,SONE
C     DIMENSION PAIR(200),BPL(100),SONE(100),XKERN(100)
C     DIMENSION POOR(200)
      IF(PAIR(4)-XPAR) 15,14,15
   14 IF(SONE(1)-XSON) 15,12,15
   15 CONTINUE
      LPAIR=NUMBER+2
      PAIR(2*NUMBER+3)=10.0*S1
      PAIR(2*NUMBER+4)=PAIR(2*NUMBER+2)
      PAIR(1)=-10.0*S1
      PAIR(2)=PAIR(4)
      CALL LAGRAN(PAIR,LPAIR,4.0,ANS)
      PAIR(1)=4.0
      PAIR(2)=ANS
      XPAR=PAIR(4)
      XSON=SONE(1)
   11 XXL=XL
      POOR(1)=4.0
      COF=(SIN(3.14159*AL))**2/3.14159
      POOR(2)=ALOG(S1-4.0)*COF
      DO 3  J=1,NUMBER
      POOR(2*J+1)=PAIR(2*J+1)
      S=PAIR(2*J+1)
    3 POOR(2*J+2)=RHO(S,XL)*PAIR(2*J+2)+COF*ALOG(S1-S)
      POOR(2*NUMBER+3)=10.0*S1
      POOR(2*NUMBER+4)=POOR(2*NUMBER+2)
      NPAIR=NUMBER+2
      CALL LAGRAN(POOR,NPAIR,S1,ANS)
      POOR(2*NUMBER+3)=S1
      POOR(2*NUMBER+4)=ANS
      DO 555    II=1,NNN
      U=SONE(II)
      IF(U-S1/2.0) 644,644,446
  644 CALL LAGRAN(PAIR,LPAIR,U,APL)
      BPL(II)=APL
      GO TO 555
  446 CALL LAGRAN(POOR,NPAIR,U,POO)
      BPL(II)=(POO-COF*ALOG(S1-U))/RHO(U,XL)
  555 CONTINUE
      N2P=2*LPAIR-2
      WRITE (3,104)(PAIR(J),J=1,N2P)
  104 FORMAT(///50X,11HPAIR VALUES///
     X(1X,2HS=,E13.5,4X,2HB=,E13.5,9X,2HS=,E13.5,4X,2HB=,E13.5,9X,2HS=,
     XE13.5,4X,2HB=,E13.5))
      N2P=2*NPAIR
   12 CONTINUE
    1 CONTINUE
      RHSP=RHO(SP    ,XL)
      IF(SP-S1/2.0) 560,560,570
```

```
 560 CALL LAGRAN(PAIR,LPAIR,SP ,PANS)
     ANS=RHSP*PANS+COF*ALOG(S1-SP)
     GO TO 580
 570 CONTINUE
     CALL LAGRAN(POOR,NPAIR,SP,    ANS)
 580 CONTINUE
     DO 444   KX=1,NNN
     TANS=ANS
     Y=SONE(KX)
     TEST=SP-Y
     XFIX=0.0
     IF(ABS(TEST)-.01)  8,8,88
   8 SPP=SP+.05
     TRHSP=RHO(SPP,XL)
     IF(SPP-S1) 9,9999,99
  99 SPP=S1
     XFIX=1.0
9999 TANS1=POOR(2*NUMBER+4)
     GO TO 999
   9 CONTINUE
     IF(SPP-S1/2.0) 660,660,670
 660 CALL LAGRAN(PAIR,LPAIR,SPP,PANS)
     TANS1=TRHSP*PANS+COF*ALOG(S1-SPP)
     GO TO 680
 670 CONTINUE
     CALL LAGRAN(POOR,NPAIR,SPP,  SANS)
     TANS1=SANS
 680 CONTINUE
 999 CONTINUE
     TEST1=SPP-Y
     SPOOR=BPL(KX)*TRHSP+COF*ALOG(S1-Y)
     SPOOR1=SPOOR
     XKERN1    =(TANS1-SPOOR)/(3.14159*TEST1)
     SPP=SP-.05
     TEST2=SPP-Y
     TRHSP=RHO(SPP,XL)
     IF(SPP-S1/2.0) 760,760,770
 760 CALL LAGRAN(PAIR,LPAIR,SPP,PANS)
     TANS2=TRHSP*PANS+COF*ALOG(S1-SPP)
     GO TO 780
 770 CONTINUE
     CALL LAGRAN(POOR,NPAIR,SPP,SANSS)
     TANS2=SANSS
 780 CONTINUE
     SPOOR=BPL(KX)*TRHSP+COF*ALOG(S1-Y)
     XKERN2    =(TANS2-SPOOR)/(3.14159*TEST2)
     XKERN(KX)=(XKERN1    +XKERN2    )/2.0+XFIX*(XKERN2-XKERN1)/2.0
     KX=KX
     GO TO 444
  88 CONTINUE
     SPOOR=BPL(KX)*RHSP+COF*ALOG(S1-Y)
     XKERN(KX)=(TANS-SPOOR)/(3.14159*TEST)
 444 CONTINUE
     RETURN
     END
                   *** 'END-OF-FILE' CARD ***
```

H. LAGRAN

```
$IBFTC LAGRAN
 0005 SUBROUTINE LAGRAN(PAIR,NPAIR,X,ANS)
C      LABEL
       DIMENSION PAIR(500),PAIR1(12)
C      DIMENSION PAIR(500),PAIR1(12)
       XX=1.0
       K=(NPAIR*2)-1
       K3=1
  100 IF (K3-K)110,110,95
  110 IF (X-PAIR(K3))10,9,130
    9 K1=K3+1
       ANS=PAIR(K1)
       GO TO 55
00010 IF(K3-1)200,200,203
  200 WRITE (3,201)
  201 FORMAT(26H  SUBROUTINE LAGRAN--ERROR)
       WRITE (3,202)
00202 FORMAT(94H  VALUE DETERMINED IN PROBLEM FOR INDEPENDENT VARIABLE I
     1S LESS THAN FIRST VALUE GIVEN IN ARRAY)
       GO TO 98
  203 CONTINUE
       IF(NPAIR-6) 205,205,204
  205 ANS=PAIR(K3-1)+(PAIR(K3+1)-PAIR(K3-1))*(X-PAIR(K3-2))/(PAIR(K3)-
     XPAIR(K3-2))
       GO TO 55
  204 CONTINUE
       IF (K3-5)14,11,50
 0011 DO 12 J=1,10
       PAIR1(J)=PAIR(J)/XX
 0012 CONTINUE
   71 A=((X-PAIR1(3))*(X-PAIR1(5))*(X-PAIR1(7))*(X-PAIR1(9)))/
     1((PAIR1(1)-PAIR1(3))*(PAIR1(1)-PAIR1(5))*(PAIR1(1)-PAIR1(7))
     2*(PAIR1(1)-PAIR1(9)))*PAIR1(2)
       A1=((X-PAIR1(1))*(X-PAIR1(5))*(X-PAIR1(7))*(X-PAIR1(9)))/
     1((PAIR1(3)-PAIR1(1))*(PAIR1(3)-PAIR1(5))*(PAIR1(3)-PAIR1(7))
     2*(PAIR1(3)-PAIR1(9)))*PAIR1(4)
       A2=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(7))*(X-PAIR1(9)))/
     1((PAIR1(5)-PAIR1(1))*(PAIR1(5)-PAIR1(3))*(PAIR1(5)-PAIR1(7))
     2*(PAIR1(5)-PAIR1(9)))*PAIR1(6)
       A3=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(5))*(X-PAIR1(9)))/
     1((PAIR1(7)-PAIR1(1))*(PAIR1(7)-PAIR1(3))*(PAIR1(7)-PAIR1(5))
     2*(PAIR1(7)-PAIR1(9)))*PAIR1(8)
       A4=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(5))*(X-PAIR1(7)))/
     1((PAIR1(9)-PAIR1(1))*(PAIR1(9)-PAIR1(3))*(PAIR1(9)-PAIR1(5))
     2*(PAIR1(9)-PAIR1(7)))*PAIR1(10)
       CALL OVERFL(KOOOFX)
        GO TO(62,61),KOOOFX
 0061 A6=A+A1+A2+A3+A4
       ANS=A6*XX
       GO TO 55
 0014 DO 15 J=1,8
       PAIR1(J)=PAIR(J)/XX
 0015 CONTINUE
   72 A=((X-PAIR1(3))*(X-PAIR1(5))*(X-PAIR1(7)))/((PAIR1(1)-
     1PAIR1(3))*(PAIR1(1)-PAIR1(5))*(PAIR1(1)-PAIR1(7)))*PAIR1(2)
       A1=((X-PAIR1(1))*(X-PAIR1(5))*(X-PAIR1(7)))/((PAIR1(3)-
     1PAIR1(1))*(PAIR1(3)-PAIR1(5))*(PAIR1(3)-PAIR1(7)))*PAIR1(4)
```

```
      A2=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(7)))/((PAIR1(5)-
     1PAIR1(1))*(PAIR1(5)-PAIR1(3))*(PAIR1(5)-PAIR1(7)))*PAIR1(6)
      A3=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(5)))/((PAIR1(7)-
     1PAIR1(1))*(PAIR1(7)-PAIR1(3))*(PAIR1(7)-PAIR1(5)))*PAIR1(8)
      CALL OVERFL(KOOOFX)
       GO TO(62,63),KOOOFX
 0063 A6=A+A1+A2+A3
      ANS=A6*XX
      GO TO 55
   50 K2=K-2
      IF (K3-K2)16,51,53
 0016 DO 17 J=1,12
      JJ=K3-7+J
      PAIR1(J)=PAIR(JJ)/XX
 0017 CONTINUE
      A=((X-PAIR1(3))*(X-PAIR1(5))*(X-PAIR1(7))*(X-PAIR1(9))
     1*(X-PAIR1(11)))/((PAIR1(1)-PAIR1(3))*(PAIR1(1)-PAIR1(5))
     2*(PAIR1(1)-PAIR1(7))*(PAIR1(1)-PAIR1(9))*(PAIR1(1)-PAIR1(11)
     3))*PAIR1(2)
      A1=((X-PAIR1(1))*(X-PAIR1(5))*(X-PAIR1(7))*(X-PAIR1(9))
     1*(X-PAIR1(11)))/((PAIR1(3)-PAIR1(1))*(PAIR1(3)-PAIR1(5))
     2*(PAIR1(3)-PAIR1(7))*(PAIR1(3)-PAIR1(9))*(PAIR1(3)-PAIR1(11)
     3))*PAIR1(4)
      A2=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(7))*(X-PAIR1(9))
     1*(X-PAIR1(11)))/((PAIR1(5)-PAIR1(1))*(PAIR1(5)-PAIR1(3))
     2*(PAIR1(5)-PAIR1(7))*(PAIR1(5)-PAIR1(9))*(PAIR1(5)-PAIR1(11)
     3))*PAIR1(6)
      A3=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(5))*(X-PAIR1(9))
     1*(X-PAIR1(11)))/((PAIR1(7)-PAIR1(1))*(PAIR1(7)-PAIR1(3))
     2*(PAIR1(7)-PAIR1(5))*(PAIR1(7)-PAIR1(9))*(PAIR1(7)-PAIR1(11)
     3))*PAIR1(8)
      A4=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(5))*(X-PAIR1(7))
     1*(X-PAIR1(11)))/((PAIR1(9)-PAIR1(1))*(PAIR1(9)-PAIR1(3))
     2*(PAIR1(9)-PAIR1(5))*(PAIR1(9)-PAIR1(7))*(PAIR1(9)-PAIR1(11)
     3))*PAIR1(10)
      A5=((X-PAIR1(1))*(X-PAIR1(3))*(X-PAIR1(5))*(X-PAIR1(7))
     1*(X-PAIR1(9)))/((PAIR1(11)-PAIR1(1))*(PAIR1(11)-PAIR1(3)
     2)*(PAIR1(11)-PAIR1(5))*(PAIR1(11)-PAIR1(7))*(PAIR1(11)-
     3PAIR1(9)))*PAIR1(12)
      CALL OVERFL(KOOOFX)
       GO TO(62,64),KOOOFX
 0064 A6=A+A1+A2+A3+A4+A5
      ANS=A6*XX
      GO TO 55
 0051 DO 52 J=1,10
      JJ=K3-7+J
      PAIR1(J)=PAIR(JJ)/XX
 0052 CONTINUE
      GO TO 71
 0053 DO 54 J=1,8
      JJ=K3-7+J
      PAIR1(J)=PAIR(JJ)/XX
 0054 CONTINUE
      GO TO 72
 0062 XX=XX*10.0
      GO TO 10
  130 K3=K3+2
```

```
      GO TO 100
   95 WRITE (3,96)
   96 FORMAT(26H  SUBROUTINE LAGRAN--ERROR)
      WRITE (3,97)
00097 FORMAT(96H  VALUE DETERMINED IN PROBLEM FOR INDEPENDENT VARIABLE I
     1S GREATER THAN LAST VALUE GIVEN IN ARRAY)
   98 WRITE (3,99)X
00099 FORMAT(56H  VALUE DETERMINED IN PROBLEM FOR INDEPENDENT VARIABLE =
     1F14.4)
      NUMBER=2*NPAIR
      WRITE (3,300)(PAIR(I),I=1,NUMBER)
 0300 FORMAT(1X,10E13.4)
      DORIS=SQRT(-1.0)
 0055 RETURN
      END
```

## I. RHO

```
$IBFTC RHO
      FUNCTION RHO(S,XL)
      COMMON /IN/RL
      RHO=((S-4.0)/4.0)**XL*SQRT((S-4.0)/S)
      IF(S-118.)  1,1,2
    2 CONTINUE
      XLL=XL+.5
      RHO=RHO*(1.+RL*((S-112.)/112.)**XLL)
    1 CONTINUE
      RETURN
      END
                *** 'END-OF-FILE' CARD ***
```

## J. WIHOP

```
$IBFTC WIHOP
      FUNCTION WIHOP(S,SPRIME,S1,AL)
      X=ALOG((S1-4.0)/(S1-S))
      XPRIME=ALOG((S1-4.0)/(S1-SPRIME))
      WIHOP=THETA(AL,XPRIME,X)/(S1-SPRIME)
      RETURN
      END
                    *** 'END-OF-FILE' CARD ***
```

## K. THETA

```
$IBFTC THETA
      FUNCTION THETA(AL,XPRIME,X)
C     LABEL
      DIMENSION F1(20,20),F2(20,20),F3(20,20),F4(20,20),PHIL(20),PHIR(
     120)     ,SAVE1(20),SAVE2(20)
C     DIMENSION F1(20,20),F2(20,20),F3(20,20),F4(20,20),PHIL(20),PHIR(
C    120)     ,SAVE1(20),SAVE2(20)
      Y=XPRIME-X
      GX=EXP(Y)
      GAY=EXP(AL*Y)
      SINHAY=.5*(GAY-1.0/GAY)
      IF(XPRIME-.2)1701,1701,1702
 1702 IF(X-.2)1701,1701,1703
 1703 CONTINUE
      MAX=10
      GO TO 1704
 1701 MAX=20
 1704 CONTINUE
      IF(MAXX-MAX)88,89,89
   88 XX=-1.0
      XXPRIM=-1.0
   89 CONTINUE
      IF(XAL-AL) 17,18,17
   17 V1=1.0
      XV=1.0-AL
      CALL GAMM(XV,V2)
      XV=1.0+AL
      CALL GAMM(XV,V3)
      XV=1.0-2.0*AL
      CALL GAMM(XV,V4)
      XV=1.0+2.0*AL
      CALL GAMM(XV,V5)
      PHIL(1)=V3**2/(V1*V5)
      PHIR(1)=V2**2/(V1*V4)
      DO 101 M=1,19
      XM=M
      V1=XM*V1
      V2=(XM-AL)*V2
      V3=(XM+AL)*V3
      V4=(XM-2.0*AL)*V4
      V5=(XM+2.0*AL)*V5
      PHIL(M+1)=V3**2/(V1*V5)
  101 PHIR(M+1)=V2**2/(V1*V4)
      XAL=AL
      TANAL=SIN(3.1416*AL)/COS(3.1416*AL)/3.14159
      THEA=AL*TANAL
      DO 666 M=1,20
      XM=M-1
      DO 666 N=1,20
      XN=N
      F1(N,M)=PHIL(M)*PHIR(N)/(XN+XM)
      F2(N,M)=PHIL(M)*PHIL(N)/(XN+XM+2.0*AL)
      F3(N,M)=PHIR(M)*PHIR(N)/(XN+XM-2.0*AL)
  666 F4(N,M)=PHIR(M)*PHIL(N)/(XN+XM)
   18 SF1=0.0
      SF2=0.0
      SF3=0.0
```

```
      SF4=0.0
      IF(X-XX) 77,78,77
   77 SAVE1(1)=1.0
      SAVE1(2)=EXP(-X)
      DO 79  M=3,MAX
   79 SAVE1(M)=SAVE1(2)*SAVE1(M-1)
      XX=X
   78 IF(XPRIME-XXPRIM)  80,81,80
   80 SAVE2(1)=EXP(-XPRIME)
      DO 82  N=2,MAX
   82 SAVE2(N)=SAVE2(1)*SAVE2(N-1)
      XXPRIM=XPRIME
   81 CONTINUE
      DO 11 M=1,MAX
      DO 11 N=1,MAX
      GEX=SAVE1(M)*SAVE2(N)
      IF(M-MAX)111,112,112
  111 IF(N-MAX)113,114,114
  112 IF(N-MAX)115,116,116
  113 CONTINUE
      SF1=SF1+F1(N,M)*GEX
      SF2=SF2+F2(N,M)*GEX
      SF3=SF3+F3(N,M)*GEX
      SF4=SF4+F4(N,M)*GEX
      GO TO 11
  114 CONTINUE
      SF3=SF3+F3(N,M)*GEX
      SF1=SF1+F1(N,M)*GEX
      GO TO 11
  115 CONTINUE
      SF3=SF3+F3(N,M)*GEX
      SF4=SF4+F4(N,M)*GEX
      GO TO 11
  116 CONT+NUE
      SF3=SF3+F3(N,M)*GEX
   11 CONT+NUE
      YP=XPRIME+X
      GEYP=EXP(AL*YP)
      GEY=GAY
      IF (ABS(Y)-.001) 13,13,14
   13 THETAA=THEA
      GO TO 15
   14 THETAA=TANAL*SINHAY/(GX-1.0)
   15 THETAB=.25*TANAL**2*(-GEYP*SF3+GEY*SF1+SF4/GEY-SF2/GEYP)
      MAXX=MAX
      THETA=THETAA+THETAB
      RETURN
      END
                   *** 'END-OF-FILE' CARD ***
```

## L. GAMM

```
$IBFTC GAMM
      SUBROUTINE GAMM(A1,F1)
C     LABEL
      DIMENSION S(4)
C     DIMENSION S(4)
      F1=1.0
      F2=0.0
      Q1=1.7
      IF(Q1-Q2) 2,1,2
    2 S(1)=1.0/12.0
      Q2=Q1
      S(2)=1.0/288.0
      S(3)=-139.0/51840.0
      S(4)=-571.0/2488320.0
      A=SQRT(2.0*3.14159265)
    1 IF(A1-10.0)3,4,4
   3  C1=A1**2
      IF(C1)6,7,6
    7 WRITE (3,9)
    9 FORMAT(///39H GAMMA FUNCTION OF NEG. INTEGER OR ZERO///)
      GO TO 8
    6 C2=F1*A1/C1
      F1=C2
      A1=A1+1.0
      GO TO 1
   4  B1=1.0/A1
      C1=1.0
      C3=B1
      DO 5I=1,4
      C1=C1+S(I)*C3
      C5=C3*B1-C4*B2
    5 C3=C5
      C3=F1*C1
      F1=C3
      C2=EXP(-A1)
      C4=.5*ALOG(A1**2)
      C6=(A1-.5)*C4
      C7=(A1-.5)*C4
      C1=EXP(C6)
      C4=C1*A
      C1=F1*C4
      F1=C1*C2
    8 RETURN
      END
              *** 'END-OF-FILE' CARD ***
```

## M. MATINV

```
$IBFTC MATINV
      SUBROUTINE MATINV(  N,B,M,DETERM)
C     LABEL
      DIMENSION PAIR(200),BPL(100),GAUS(100)
      DIMENSION IDO(10),H(10,10),X(10,10)
      DIMENSION XMAT(50,50),YANS(50,1)
      DIMENSION SONE(100)
      DIMENSION IPIVOT(50), A(50,50), B(50,1), INDEX(50,2), PIVOT(50)
C THE FOLLOWING STATEMENT(S) HAVE BEEN MANUFACTURED BY THE TRANSLATOR TO
C   COMPENSATE FOR THE FACT THAT EQUIVALENCE DOES NOT REORDER COMMON---
      COMMON  S1      , NUMBER , XL      , AL      , PAIR    , BPL
      COMMON  NNN     , SONE    , NOEROR , YANS    , IDO     , H
      COMMON  X       , A
C     COMMON S1,NUMBER,XL,AL,PAIR,BPL,NNN,SONE,NOEROR,YANS,IDO,H,X,A
C     DIMENSION PAIR(200),BPL(100),GAUS(100)
C     DIMENSION IDO(10),H(10,10),X(10,10)
C     DIMENSION XMAT(50,50),YANS(50,1)
C     DIMENSION SONE(100)
C     DIMENSION IPIVOT(50), A(50,50), B(50,1), INDEX(50,2), PIVOT(50)
      EQUIVALENCE (IROW,JROW), (ICOLUM,JCOLUM), (AMAX, T, SWAP)
      IF(NNN-50) 10,10,11
   11 WRITE (3,12)
  12  FORMAT(15X,32HNNN GREATER THAN DIMENSION GIVEN)
      RETURN
   10 DETERM=1.0
   15 DO 20 J=1,N
   20 IPIVOT(J)=0
   30 DO 550 I=1,N
C
C     SEARCH FOR PIVOT ELEMENT
C
   40 AMAX=0.0
   45 DO 105 J=1,N
   50 IF (IPIVOT(J)-1) 60, 105, 60
   60 DO 100 K=1,N
   70 IF (IPIVOT(K)-1) 80, 100, 740
   80 IF (ABS(AMAX)-ABS(A(J,K))) 85, 100, 100
   85 IROW=J
   90 ICOLUM=K
   95 AMAX=A(J,K)
  100 CONTINUE
  105 CONTINUE
  110 IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1
C
C     INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL
C
  130 IF (IROW-ICOLUM) 140, 260, 140
  140 DETERM=-DETERM
  150 DO 200 L=1,N
  160 SWAP=A(IROW,L)
  170 A(IROW,L)=A(ICOLUM,L)
  200 A(ICOLUM,L)=SWAP
  205 IF(M) 260, 260, 210
  210 DO 250 L=1, M
  220 SWAP=B(IROW,L)
  230 B(IROW,L)=B(ICOLUM,L)
  250 B(ICOLUM,L)=SWAP
```

```
  260 INDEX(I,1)=IROW
  270 INDEX(I,2)=ICOLUM
  310 PIVOT(I)=A(ICOLUM,ICOLUM)
  320 DETERM=DETERM*PIVOT(I)
C
C     DIVIDE PIVOT ROW BY PIVOT ELEMENT
C
  330 A(ICOLUM,ICOLUM)=1.0
  340 DO 350 L=1,N
  350 A(ICOLUM,L)=A(ICOLUM,L)/PIVOT(I)
  355 IF(M) 380, 380, 360
  360 DO 370 L=1,M
  370 B(ICOLUM,L)=B(ICOLUM,L)/PIVOT(I)
C
C     REDUCE NON-PIVOT ROWS
C
  380 DO 550 L1=1,N
  390 IF(L1-ICOLUM) 400, 550, 400
  400 T=A(L1,ICOLUM)
  420 A(L1,ICOLUM)=0.0
  430 DO 450 L=1,N
  450 A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
  455 IF(M) 550, 550, 460
  460 DO 500 L=1,M
  500 B(L1,L)=B(L1,L)-B(ICOLUM,L)*T
  550 CONTINUE
C
C     INTERCHANGE COLUMNS
C
  600 DO 710 I=1,N
  610 L=N+1-I
  620 IF (INDEX(L,1)-INDEX(L,2)) 630, 710, 630
  630 JROW=INDEX(L,1)
  640 JCOLUM=INDEX(L,2)
  650 DO 705 K=1,N
  660 SWAP=A(K,JROW)
  670 A(K,JROW)=A(K,JCOLUM)
  700 A(K,JCOLUM)=SWAP
  705 CONTINUE
  710 CONTINUE
  740 RETURN
  750 END
                    *** 'END-OF-FILE' CARD ***
```

## N. NGAUSS

```
$IBFTC NGAUSS
      SUBROUTINE NGAUSS(XA,XB,E,N,XL,          GAUS,IST,ICA,FNOL,PTS,AL)
C     LABEL
      DIMENSIONXMAT(50,50)
      DIMENSION PAIR(200),BPL(100),GAUS(100)
      DIMENSION SONE(100),YANS(50,1)
      DIMENSION IDO(10),H(10,10),X(10,10)
      DIMENSION S(100)
      DIMENSION PTS(100)
C THE FOLLOWING STATEMENT(S) HAVE BEEN MANUFACTURED BY THE TRANSLATOR TO
C   COMPENSATE FOR THE FACT THAT EQUIVALENCE DOES NOT REORDER COMMON---
      COMMON  S1      , NUMBER , XL      , AL      , PAIR    , BPL
      COMMON  NNN     , SONE   , NOEROR , YANS    , IDO     , H
      COMMON  X       , XMAT   , NPAIR
C     COMMON S1,NUMBER,XL,AL,PAIR,BPL,NNN,SONE,NOEROR,YANS,IDO,H,X
C    1,XMAT,NPAIR
C     DIMENSIONXMAT(50,50)
C     DIMENSION PAIR(200),BPL(100),GAUS(100)
C     DIMENSION SONE(100),YANS(50,1)
C     DIMENSION IDO(10),H(10,10),X(10,10)
C     DIMENSION S(100)
C     DIMENSION PTS(100)
      DO 1110 KX=1,N
 1110 GAUS(KX)=0.0
      A=0.0
      B=SQRT(XB-XA)
      NE=0
      IF(XA-XB) 34,24,24
   34 CONTINUE
      I=IST
      IP=IDO(ICA)
      L=ICA
    5 TA=A
      FI=I
      C=(B-A)/FI
      D=C*0.5
      DO 100 KX=1,N
      S(KX)=GAUS(KX)
  100 GAUS(KX)=0.0
      DO 7 K=1,I
      DO 6 J=1,IP
      T=X(J,IP)
      YT=TA+D*(T+1.0)
      SP=XB-YT**2
      FYN=FNOL(SP)
      DO 103 KX=1,N
      SVAL=PTS(KX)
      FXN=WIHOP(SVAL,SP,S1,AL)
      GAUS(KX)=GAUS(KX)+H(J,IP)*YT*FXN*FYN
103   CONTINUE
6     CONTINUE
      TA=TA+C
7     CONTINUE
      DO 5009 KX=1,N
 5009 GAUS(KX)=GAUS(KX)*D
      IF(NE) 107,107,5002
 5002 DO 927 KX=1,N
```

```
  927 S(KX)=(GAUS(KX)-S(KX))/GAUS(KX)
      DO 106 KX=1,N
      IF (ABS(S(KX))-E) 106,106,804
  804 MOK=KX
      GO TO 107
106   CONTINUE
      GO TO 110
107   CONTINUE
      WRITE (3,875)NE,I,L,MOK
  875 FORMAT(1X,3HNE=,I4,5X,2HI=,I4,5X,2HL=,I4,5X,4HMOK=,I4)
      IF (NE-NOERCR) 109,110,110
109   L=L+2
      L=L+2
      NE=NE+1
      IF(L-8)28,28,900
28    IP=IDO(L)
      GO TO 5
  900 I=I+1
      I=I+2
      L=ICA
      IP=IDO(L)
      GO TO 5
24    CONTINUE
      WRITE (3,3)
    3 FORMAT(86H0**REJECTED**LOWER LIMIT OF INTEGRATION GREATER THAN OR
     1EQUAL TO UPPER LIMIT IN NGAUSS)
      GO TO 871
110   CONTINUE
      DO 874 J=1,N
      SVAL=PTS(J)
      GAUS(J)=2.0*GAUS(J)+FNOL(SVAL)
  874 CONTINUE
871   RETURN
      END
                   *** 'END-OF-FILE' CARD ***
```

## O. FINT

```
$IBFTC FINT
      FUNCTION FINT(S)
C     LABEL
      DIMENSION PAIR(200),BPL(100)
      DIMENSION SONE(100),YANS(50,1)
      DIMENSION IDO(10),H(10,10),X(10,10)
      DIMENSIONXMAT(50,50)
C THE FOLLOWING STATEMENT(S) HAVE BEEN MANUFACTURED BY THE TRANSLATOR TO
C   COMPENSATE FOR THE FACT THAT EQUIVALENCE DOES NOT REORDER COMMON---
      COMMON  S1      , NUMBER , XL      , AL      , PAIR    , BPL
      COMMON  NNN     , SONE   , NOEROR  , YANS    , IDO     , H
      COMMON  X       , XMAT   , NPAIR
C     COMMON S1,NUMBER,XL,AL,PAIR,BPL,NNN,SONE,NOEROR,YANS,IDO,H,X
C    1,XMAT,NPAIR
C     DIMENSION PAIR(200),BPL(100)
C     DIMENSION SONE(100),YANS(50,1)
C     DIMENSION IDO(10),H(10,10),X(10,10)
C     DIMENSIONXMAT(50,50)
      IF(PAIR(2)-XPAR) 1,2,1
    1 CONTINUE
      LPAIR=NPAIR+2
      N=2*LPAIR
      NN=N-4
      DO 3 I=1,NN
      KK=N-1-I
      KL=KK-2
    3 PAIR(KK)=PAIR(KL)
      PAIR(1)=-100.0*S1
      PAIR(2)=PAIR(4)
      CALL LAGRAN(PAIR,LPAIR,4.0,ANS)
      PAIR(2)=ANS
      XPAR=ANS
      PAIR(N-1)=100.0*S1
      PAIR(N)=PAIR(N-2)
    2 IF(S1-S) 7,7,8
    7 WRITE (3,11)S,(PAIR(I),I=1,N)
   11 FORMAT(14HAIN FINT(S),S=,  E13.5/(1X,10E13.4))
      CALL EXIT
    8 CALL LAGRAN ( PAIR,LPAIR,S,FINT)
      RETURN
      END
                  *** 'END-OF-FILE' CARD ***
```

## P. DGAUSS

```
$IBFTC DGAUSS
      SUBROUTINE DGAUSS(XA,XB,E,N,XL,SVAL,GAUS,IST,ICA,XNUM,ZERO,WIDTH,
     1NZ)
C     LABEL
      DIMENSION GAUS(100),SUBTR(100),SVAL(100),S(100)
      DIMENSION ZERO(10),WIDTH(10),ZPAIR(200)
      DIMENSION IDO(10),H(10,10),X(10,10)
      DIMENSIONXMAT(50,50)
      DIMENSION SONE(100),YANS(50,1)
      DIMENSION PAIR(200),BPL(100)
C THE FOLLOWING STATEMENT(S) HAVE BEEN MANUFACTURED BY THE TRANSLATOR TO
C    COMPENSATE FOR THE FACT THAT EQUIVALENCE DOES NOT REORDER COMMON---
      COMMON  S1      , NUMBER , XL     , AL      , PAIR    , BPL
      COMMON  NNN     , SONE   , NOEROR , YANS    , IDO     , H
      COMMON  X       , XMAT   , NPAIR
C     DIMENSION GAUS(100),SUBTR(100),SVAL(100),S(100)
C     DIMENSION ZERO(10),WIDTH(10),ZPAIR(200)
C     DIMENSION IDO(10),H(10,10),X(10,10)
C     DIMENSIONXMAT(50,50)
C     DIMENSION SONE(100),YANS(50,1)
C     DIMENSION PAIR(200),BPL(100)
C     COMMON S1,NUMBER,XL,AL,PAIR,BPL,NNN,SONE,NOEROR,YANS,IDO,H,X
C    1,XMAT,NPAIR
      A=0.0
      B=SQRT(XB-XA)
      DO 11  J=1,N
      SVA=SVAL(J)
      IF((SVA-XA)*(XB-SVA)) 13,12,12
   13 SUBTR(J)=0.0
      GO TO 11
   12 SUBTR(J)=XNUM(SVA)    *RHO(SVA,XL)
   11 CONTINUE
      NE=0
      IF(A-B)34,24,24
   34 CONTINUE
      I=IST
      IP=IDO(ICA)
      L=ICA
    5 TA=A
      FI=I
      C=(B-A)/FI
      D=C*0.5
      DO 100 KX=1,N
      S(KX)=GAUS(KX)
  100 GAUS(KX)=0.0
      DO 7 K=1,I
      DO 6 J=1,IP
      T=X(J,IP)
      YT=TA+D*(T+1.0)
      SP=XB-YT**2
      FX=XNUM(SP)    *RHO(SP,XL)
      DO 103 KX=1,N
      SVA=SVAL(KX)
      GAUS(KX)=GAUS(KX)+H(J,IP)*YT*(FX-SUBTR(KX))/(SP-SVA)
103   CONTINUE
6     CONTINUE
```

```
      TA=TA+C
7     CONTINUE
      DO 5009 KX=1,N
 5009 GAUS(KX)=GAUS(KX)*D
      IF(NE) 107,107,5002
 5002 DO 927 KX=1,N
  927 S(KX)=(GAUS(KX)-S(KX))/GAUS(KX)
      DO 106 KX=1,N
      IF (ABS(S(KX))-E) 106,106,804
  804 MOK=KX
      GO TO 107
106   CONTINUE
      GO TO 110
107   CONTINUE
      WRITE (3,875)NE,I,L,MOK
  875 FORMAT(1X,3HDE=,I4,5X,2HI=,I4,5X,2HL=,I4,5X,4HMOK=,I4)
      IF (NE-NOEROR) 109,110,110
109   L=L+2
      L=L+2
      NE=NE+1
      IF(L-8)28,28,900
28    IP=IDO(L)
      GO TO 5
  900 I=I+1
      I=I+2
      L=ICA
      IP=IDO(L)
      GO TO 5
24    CONTINUE
      DO 1110 KX=1,N
 1110 GAUS(KX)=0.0
      WRITE (3,3)
3     FORMAT(90HO**REJECTED**   UPPER LIMIT OF INTEGRATION GREATER THAN
     1OR EQUAL TO LOWER LIMIT IN DGAUS                                  )
      GO TO 871
110   CONTINUE
      DO 874 J=1,N
      SVA=SVAL(J)
      IF(SUBTR(J)) 876,877,876
  877 GAUS(J)=1.-GAUS(J)/1.5708
      GO TO 874
  876 CONTINUE
      GAUS(J)=1.0-(GAUS(J)+SUBTR(J)*ALOG((XB-SVA)/(SVA-XA))/2.0)/1.5708
  874 CONTINUE
      NZ=0
      IF(N-10) 871,871,888
  888 CONTINUE
      DO 601  J=2,N
      IF(GAUS(J)*GAUS(J-1)) 602,602,601
  602 NZ=NZ+1
      IF(GAUS(J)-GAUS(J-1)) 605,605,606
  605 DO 607  K=J,N
      KZ=K
      IF(GAUS(K+1)-GAUS(K)) 607,608,608
  607 CONTINUE
  608 CONTINUE
      DO 609  L=1,KZ
```

```
      KKZ=KZ+1-L
      ZPAIR(2*L-1)=GAUS(KKZ)
      ZPAIR(2*L)=SVAL(KKZ)
      NZAIR=L
      IF(GAUS(KKZ-1)-GAUS(KKZ)) 610,610,609
  609 CONTINUE
  610 CONTINUE
      CALL LAGRAN(ZPAIR,NZAIR,0.0,ANS)
      ZERO(NZ)=ANS
      DO 650  I=1,NZAIR
      X=ZPAIR(I)
      II=2*NZAIR-I+1
      ZPAIR(I)=ZPAIR(II)
  650 ZPAIR(II)=X
      GO TO 660
  606 DO 621  M=1,J
      MZ=J-M+1
      IF(GAUS(MZ)-GAUS(MZ-1)) 622,622,621
  621 CONTINUE
  622 CONTINUE
      DO 623  LL=MZ,N
      L=LL+1-MZ
      ZPAIR(2*L-1)=GAUS(LL)
      ZPAIR(2*L)=SVAL(LL)
      NZAIR=L
      IF(GAUS(LL+1)-GAUS(LL)) 624,624,623
  623 CONTINUE
  624 CONTINUE
      CALL LAGRAN(ZPAIR,NZAIR,0.0,ANS)
      ZERO(NZ)=ANS
      DO 630  I=1,NZAIR
      X=ZPAIR(2*I-1)
      ZPAIR(2*I-1)=ZPAIR(2*I)
  630 ZPAIR(2*I)=X
  660 CONTINUE
      Z=ZERO(NZ)+.5
      CALL LAGRAN(ZPAIR,NZAIR,Z,ANS)
      Z=ZERO(NZ)-.5
      CALL LAGRAN(ZPAIR,NZAIR,Z,BANS)
      Z=ZERO(NZ)
      WIDTH(NZ)=ANS-BANS
  601 CONTINUE
871   RETURN
      END
                    *** 'END-OF-FILE' CARD ***
```

## O. FANT

```
$IBFTC FANT
      FUNCTION FANT(S)
      COMMON S1
      DIMENSION PAIR(200)
      COMMON /P2/ PAIR,NPAIR
      IF(PAIR(2)-XPAR) 1,2,1
    1 CONTINUE
      LPAIR=NPAIR+2
      N=2*LPAIR
      NN=N-4
      DO 3 I=1,NN
      KK=N-1-I
      KL=KK-2
    3 PAIR(KK)=PAIR(KL)
      PAIR(1)=-100.0*S1
      PAIR(2)=PAIR(4)
      CALL LAGRAN(PAIR,LPAIR,4.0,ANS)
      PAIR(2)=ANS
      XPAR=ANS
      PAIR(N-1)=100.0*S1
      PAIR(N)=PAIR(N-2)
    2 IF(S1-S) 7,7,8
    7 WRITE (3,11)S,(PAIR(I),I=1,N)
   11 FORMAT(14HAIN FINT(S),S=,  E13.5/(1X,10E13.4))
      CALL EXIT
    8 CALL LAGRAN ( PAIR,LPAIR,S,FANT)
      FANT=FANT*FINT(S)
      RETURN
      END
                  *** 'END-OF-FILE' CARD ***
```

## FOOTNOTES AND REFERENCES

*Present address: Geneva, Switzerland.

†Present address: CERN, Geneva 23, Switzerland.


1. D. C. Teplitz and V. L. Teplitz, Numerical Solution of the Pion-Pion Strip Approximation N/D Equation, UCRL-11591. (Submitted to Phys. Rev.)
2. G. F. Chew, Phys. Rev. 130, 1264 (1963).
3. F. B. Hildebrand, Introduction to Numerical Analysis, (McGraw-Hill Book Co., N. Y., N. Y., 1956).
4. P. G. Burke, Legendre Function of Complex Degree and Real Argument, IBM SHARE Program, C3EO LEGN.
5. E. M. Anderson, LAGRANGE INTERPOLATION, IBM SHARE Program, E14 SCF LAGR.
6. V. Teplitz, Solution of the N/D Equations in the Strip Approximation, UCRL-5555. (Submitted to Phys. Rev.)
7. B. S. Garbow, Matrix Inversion with Accompanying Solution of Linear Equations, IBM SHARE Program, FIAN F402.

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.