Internal Report EMF 75-1

Sept. 4, 1975

Numerical computation of electro-magnetic scattering from rotational symmetric configurations.†

by

Bo Peterson

†This project was supported by the Swedish Institute of Applied Mathematics and The research institute of the national defence.

Institute of Theoretical Physics

Fack

S-402 20 GÖTEBORG 5

Sweden

Introduction

In Ref. [1] P.C. Waterman describes the numerical solution of a certain class of electromagnetic scattering problems, using the T matrix formalism developed by him in Ref. [2]. Later new theoretical results have been obtained which makes it possible to consider more general bodies and configurations of such bodies, Refs. [3] and [4]. The purpose of the present work is to document a system of computer programs for the calculation of scattering from rotational symmetric configurations of bodies. In section I the relevant results from the T matrix formalism are given. In section IIa we describe the various Mainroutines in the computer programs. In section IIb we describe the various subroutines and functions called by the Mainroutines. In Appendix I we give some lengthy formulas. In Appendix II we give a complete list of the routines in Fortran.

From
Bo Peterson
Mekanik
Chalmers
Hörsalsv 5
412 96 Göteborg
Sweden

I. Theory

Consider the problem of scattering of an electromagnetic wave by a system of obstacles. The problem is described by a matrix T in the following way.

Incoming wave $$I\!E^i = \sum_{in} a_{in} \, \mathrm{Re}\, \vec{\psi}_{in} \equiv \mathrm{Re}\, \vec{\psi}^{\,t} \vec{a} \qquad (1.1)$$

scattered wave $$I\!E^s = \sum_{in} f_{in} \vec{\psi}_{in} \equiv \vec{\psi}^{\,t} \vec{f} \qquad (1.2)$$

with $$f_{in} = \sum_{in'} T_{in,in'} a_{in'} \qquad (1.3)$$

where $T_{mn'}$ is the T matrix. A time dependence e^{-iwt} is suppressed. The expression (1.2) for the scattered field is valid outside the smallest sphere circumscribing the body or bodies and with centre at origin. The total field $I\!E$ is the sum of the incoming and scattered fields.

$$I\!E = I\!E^i + I\!E^s \qquad (1.4)$$

As basis functions we use the spherical wave solutions $\vec{\psi}$ to

$$\nabla \times (\nabla \times \vec{\psi}) - k^2 \vec{\psi} = 0 \quad \text{with} \quad \nabla \cdot \vec{\psi} = 0 \qquad (1.5)$$

which form a complete set:

$$\vec{\psi}_{in} \equiv \vec{\psi}_{\tau\sigma mn}(k\vec{r}) \equiv \gamma_{mn}^{1/2} (k^{-1}\nabla\times)^{\tau} \left[k\vec{r}\, Y_{\sigma mn}(\hat{r}) h_n^{(1)}(kr) \right] \qquad (1.6)$$

where $\tau = 1, 2$, $\sigma = e, o$ ("even" or "odd"), n = 1,2,3..., m = 0,1,2,... n, and

$$\gamma_{mn} = \frac{\varepsilon_m (2n+1)(n-m)!}{4\pi\, n(n+1)(n+m)!} \qquad (1.7)$$

$$\varepsilon_0 = 1, \quad \varepsilon_m = 2 \quad m \neq 0$$

$h_n^{(1)}(kr)$ is a spherical Hankelfunction and

$$Y_{emn}(\hat{r}) = P_n^m(\cos\theta)\cos m\varphi \qquad (1.8)$$

$$Y_{omn}(\hat{r}) = P_n^m(\cos\theta)\sin m\varphi \tag{1.9}$$

where P_n^m is an associated Legendre function. $\mathrm{Re}\vec{\psi}_{1m}$ stands for the regular part of $\vec{\psi}_{1m}$ i.e., the expression (1.6) with $h_n^{(1)}(kr)$ replaced by $j_n(kr)$, a spherical Bessel function. The explicit expression for $\vec{\psi}_{1m}$ is given in Appendix I. Because of the rotational symmetry of the configurations we shall treat several simplifications. We choose the z-axis as axis of rotational symmetry and this implies that all matrices involved in the formalism are diagonal in the indices m,m'. We define the polarization to be orthogonal or parallel according to whether the electric vector is parallel or orthogonal to the plane spanned by the z-axis and the wave vector. More specifically for the incoming wave we choose the wave vector to be in the x-z plane. In this case we have wave vector

$$\vec{k} = k(\sin\theta, 0, \cos\theta) \tag{1.10}$$

and the orthogonal respectively parallel polarization unit vectors

$$\hat{e}_{\perp} = \hat{y} \tag{1.11}$$

and

$$\hat{e}_{\parallel} = (\cos\theta, 0, -\sin\theta) \tag{1.12}$$

During the scattering these two polarizations do not mix. This is reflected in the fact that all matrices involved (in case of rotational symmetry) fulfill the relations:

$$\begin{aligned} M_{1emn,1emn'} &= M_{1omn,1omn'} \\ M_{2emn,2emn'} &= M_{2omn,2omn'} \\ M_{1emn,2omn'} &= -M_{1omn,2emn'} \\ M_{2omn,1emn'} &= -M_{2emn,1omn'} \end{aligned} \tag{1.13}$$

with all other combinations of $\gamma\sigma, \gamma'\sigma'$ giving zero.

The orthogonal polarization is expressed by the four combinations of the $1e$ and $2o$ in the indices $\tau\sigma, \tau'\sigma'$ and the parallell polarization by the four combinations of $1o$ and $2e$. It is also clear that the two polarizations can be treated separately and from the T matrix for one polarization we can easily transform to the T matrix for the other polarization. We treat the orthogonal polarization as the normal case and make the following definitions for the matrix M:

$$
\begin{aligned}
M_{1emn,1emn'} &= M^1(m)_{nn'} \\
M_{1emn,2omn'} &= M^2(m)_{nn'} \\
M_{2omn,1emn'} &= M^3(m)_{nn'} \\
M_{2omn,2omn'} &= M^4(m)_{nn'}
\end{aligned}
\tag{1.14}
$$

After rearranging the matrices we get the T matrices

$$
T^{ort}(m) \equiv \begin{pmatrix} T^1 & T^2 \\ T^3 & T^4 \end{pmatrix} \quad \text{and} \quad T^{par}(m) \equiv \begin{pmatrix} T^1 & -T^2 \\ -T^3 & T^4 \end{pmatrix}
$$

for the orthogonal and parallell polarizations respectively. In order to reduce the number of matrix indices we make the following transformation.

$$
\begin{aligned}
M^1(m)_{nn'} &= \bar{M}(m)_{2n-1,2n'-1} \\
M^2(m)_{nn'} &= \bar{M}(m)_{2n-1,2n'} \\
M^3(m)_{nn'} &= \bar{M}(m)_{2n,2n'-1} \\
M^4(m)_{nn'} &= \bar{M}(m)_{2n,2n'}
\end{aligned}
\tag{1.15}
$$

We now turn to the case of infinitely conducting bodies. The T matrix is expressed by Q matrices as

$$
T = -Q(Re,Re)\,Q(Out,Re)^{-1} \tag{1.16}
$$

where

$$Q(Out,Re)_{mnn'} = k\int_S d\vec{s}\cdot\left(\vec{\psi}_{mn}(k\vec{r})\times\left[\nabla\times Re\,\vec{\psi}_{mn'}(k\vec{r})\right]\right) \qquad (1.17)$$

S is the surface of the body and the first argument in Q tells whether the function associated with the left index is irregular or regular ($\vec{\psi}_{mn}$ or $Re\,\vec{\psi}_{mn}$) and similarly for the second argument and the right index. Explicit calculations show that

$$Q^4 = \mathbb{I}i - Q^1 \qquad (1.18)$$

and

$$Q^2 = Q^3 \qquad (1.19)$$

The explicit expressions for Q^1 and Q^2 are given in Appendix I. In general Q(Re,Re) is symmetric. Further, if the z-y plane is a plane of symmetry we have $Q^1_{nn'} = 0$ when n+n' is odd and $Q^2_{nn'} = 0$ when n+n' is even. The spheroid shaped bodies have symmetric Q(Out,Re) and mirror symmetry if the spheroid axes intersect at the origin. The reciprocity and the fact that infinitely conducting bodies are lossless is reflected in the following two relations for the T matrix.

$$T = T^t \qquad \text{(t is transposition)} \qquad (1.20)$$

$$T^\dagger T = -Re\,T \qquad \text{(† is Hermite conjugation)} \qquad (1.21)$$

These relations can be used as subsidiary conditions by which a direct inversion of Q(Out,Re) can be avoided. The procedure which is called orthogonalization is described in Ref. [1] and is very important because the procedure with direct inversion of Q(Out,Re) has to be performed with double dimension compared to the other one in order to obtain the same numerical accuracy. In the numerical truncations of $\bar{Q}$ we need a dimension $N \geq 2ka$, where a is the radius of the smallest sphere circumscribing the body, in

the orthogonalization procedure. The homogeneous dielectric bodies can be treated in the same way as the infinitely conducting bodies, although one then has a more complicated Q matrix. We now have the wave vectors k_1 and k_2 outside respectively inside the surface S_1 of the body. The relative permeability is μ_1 outside and μ_2 inside the surface. The matrix Q is given by:

$$Q(Out,Re)_{mm'} = k_1 \int_{S_1} d\vec{s} \cdot \left[(\nabla \times \vec{\psi}_{1m}(k_1\vec{r})) \times Re\, \vec{\psi}_{1m'}(k_2\vec{r}) + \right.$$
$$\left. + \frac{\mu_1}{\mu_2} \vec{\psi}_{1m}(k_1\vec{r}) \times (\nabla \times Re\, \vec{\psi}_{1m'}(k_2\vec{r})) \right] \qquad (1.22)$$

By keeping $\varepsilon_1\mu_1 = \varepsilon_2\mu_2$ (thus $k_1 = k_2$), where the ε:s are the relative dielectric constants, and taking $\lim_{\varepsilon_2 \to \infty} \frac{\mu_2}{\mu_1} Q(out,Re)_{mm'}$ we get the Q matrix for the infinitely conducting body bounded by the surface S_1. In order to avoid duplicate operations we partition the matrices Q^i in the following way

$$Q^1_{nn'} = -A_{nn'} + \frac{\mu_1}{\mu_2} D_{nn'}$$
$$Q^2_{nn'} = \frac{k_1}{k_2} C_{nn'} + \frac{\mu_1 k_2}{\mu_2 k_1} B_{nn'}$$
$$Q^3_{nn'} = -B_{nn'} - \frac{\mu_1}{\mu_2} C_{nn'}$$
$$Q^4_{nn'} = \frac{k_1}{k_2} D_{nn'} - \frac{\mu_1 k_2}{\mu_2 k_1} A_{nn'} \qquad (1.23)$$

The explicit expressions for A, B, C and D are given in Appendix I. When $k_1 = k_2$ and $\mu_1 = \mu_2$ the Q matrices have to fulfill the following relations:

$$Q(Re,Re) = Q(Out,Out) = 0 \qquad (1.24)$$

$$Q(Re,Out) = -Q(Out,Re) = \mathbb{I}\, i \qquad (1.25)$$

For real k_i the matrices Q(Re, Re), Q(out,Re) and Q(Re, out) have the same real part while Q(out, out) and Q(out, Re) + + Q(Re, out) have the same imaginary part. The orthogonalization procedure is also well suited for lossless (i.e. real k_i) homogeneous dielectric bodies. By a multilayered scatterer we mean a body consisting of several layers, each of which has constant electric and magnetic properties and which consecutively enclose each other. Consider a body with N surfaces, as depicted in Fig. 1, separating layers with constant properties. If the surface S_i outside the surface S_{i-1} can be separated by a spherical shell, with centre at origin, we can apply the following iteration formula for the different T matrices T(i) for the layered object whose outer surface is S_i. (The T matrix T(i) is defined as a relation between the coefficients of the regular and irregular functions.)

$$T(i-1) = -\left[Q^{i-1}(Re,Re) + Q^{i-1}(Re,Out)\,T(i)\right] \times \times \left[Q^{i-1}(Out,Re) + Q^{i-1}(Out,Out)\,T(i)\right]^{-1} \tag{1.26}$$

where Q^i is the Q matrix of the surface S_i. Thus starting with the T matrix T(N) for the homogeneous innermost body (which also can be infinitely conducting) inside the surface S_N and the Q matrices of the surface S_{N-1} applying the formula (1.26) N-1 times we finally obtain the T matrix T(1) of the whole multilayered body. The orthogonalization procedure can be used to obtain the T matrices T(i) as long as the layers inside S_{i-2} are lossless. Consider a configuration of two bodies as depicted in Fig. 2. The bodies have their T matrices T_i associated to the local coordinate systems with centre at O_i within the bodies. The coordinate systems with centre at O_i are pure trans-

lations a distance $\vec{a}_i$ of the original one with center at 0. If the radius vector $\vec{r}_i''$ from 0_i to the surface S_i fulfill the relation $r_i'' < |\vec{a}_j - \vec{a}_i|$, $1,2 = i \neq j = 1,2$ we get the following formula for the total T matrix of the configuration.

$$T = \sum_{1,2=k\neq i=1,2} R(\vec{a}_i)\, T_i \left(\mathbb{I} - \nabla(-\vec{a}_i+\vec{a}_k)\, T_k\, \nabla(-\vec{a}_k+\vec{a}_i)\, T_i\right)^{-1} \times \left(\mathbb{I} + \nabla(-\vec{a}_i+\vec{a}_k)\, T_k\, R(\vec{a}_i-\vec{a}_k)\right) R(\vec{a}_i)^t \tag{1.27}$$

where R and ∇ are the translation matrices for the basis functions as defined by:

$$\mathrm{Re}\,\vec{\psi}_{lm}(\vec{r}+\vec{c}) = \sum_{lm'} R_{lmlm'}(\vec{c})\, \mathrm{Re}\,\vec{\psi}_{lm'}(\vec{r}) \quad \text{all } r_i c \tag{1.28}$$

$$\vec{\psi}_{lm}(\vec{r}+\vec{c}) = \sum_{lm'} \nabla_{lmlm'}(\vec{c})\, \mathrm{Re}\,\vec{\psi}_{lm'}(\vec{r}) \quad \text{for } r<c \tag{1.29}$$

we also have

$$\vec{\psi}_{lm}(\vec{r}+\vec{c}) = \sum_{lm'} R_{lmlm'}(\vec{c})\, \vec{\psi}_{lm'}(\vec{r}) \quad \text{for } r>c \tag{1.30}$$

The matrices R and ∇ are given, for translations along the z-axis, in the Appendix I. The preceeding results can be combined to the case of an arbitrary number of bodies.

If $T_2 \to 0$ then $T \to R(\vec{a}_1)\, T_1\, R(\vec{a}_1)^t$ (1.31)

This is just the formula for the transformation of T_1 which is referred to origin 0_1, to the corresponding matrix T, which is referred to origin 0. If r_1^{max} is the radius of the smallest sphere , with centre at origin 0_1, circumscribing the body 1 (body 2 is removed c.f. Fig. 3), we have an expansion about 0_1 as in (1.2) valid for $r_1 > r_1^{max}$. Here $f_{lm}^{(1)}$ are given in terms of $a_{lm}^{(1)}$, both referred to 0_1 by T_1. The T matrix T gives another

expansion , with coefficients f_{lm} and a_{lm} about 0 valid for $r > r^{max}$ where r^{max} is the radius of the smallest sphere, with centre at 0, circumscribing the body 1. It is clear that by such transformations we can obtain the field at any point outside the convex part of the surface S_1. In order to obtain the field at points just outside the concave part of the body we have to make an expansion in terms of regular functions about an origin 0 outside the body

$$\vec{\psi}^{S} = \sum_{lm} f_{lm}^{reg} \, Re\, \vec{\psi}_{lm}(k\vec{r}) \equiv Re\, \vec{\psi}^{t} \vec{f}^{reg} \qquad (1.32)$$

This expansion is valid for $r < r^{min}$ (see Fig. 4) where r^{min} now is the radius of the greatest sphere, with centre at 0, and not intersecting the interior of the body. However now f_{lm}^{reg} is given by:

$$\vec{f}^{reg} = \Gamma(\vec{a}_1)\, T_1\, R(\vec{a}_1)^{t}\, \vec{a} \quad \text{for } a_1 > r'' \qquad (1.33)$$

(Instead of $\vec{f} = R(\vec{a}_1)\, T_1\, R(\vec{a}_1)^{t} \vec{a}$ as in the other case.)

In the numerical treatment of (1.27) it is important to notice that it is not only the radius r_i^{max} of the two smallest spheres, circumscribing the resp. bodies and with centre at 0_i, which determine the dimension needed. The distance between the bodies also plays an essential role and the dimension N of the matrix $\bar{T}$ is given by

$$N \geq 2\left[k\,|\vec{a}_1 - \vec{a}_2| + \max_i r_i^{max}\right].$$

The formula (1.27) consists of two terms, i = 1 and 2 where the term i = 1 corresponds to all those waves which are scattered in all possible combinations between the bodies and then scattered the last time from body no 1, and similarly for i = 2. The matrix $\Gamma_{lm,l'm'}(\vec{c}) \to c^{-1} F_{lm,l'm'}(\hat{c})$ for large c $(\vec{c} = c\hat{c})$. This makes the von Neuman series expansion of the inverses in (1.27) rapidly convergent. The limit value of T becomes $R(\vec{a}_1)\, T_1\, R(a_1)^{t} + R(\vec{a}_2)\, T_2\, R(\vec{a}_2)^{t}$ for large separation. (i.e. large $|\vec{a}_1 - \vec{a}_2|$).

For intermediate separations it can be preferable to use only a few terms in the von Neuman series. This means that the important contribution comes from the first few multiple reflections between the bodies. In some case it may be preferable to use the result of the translation operators operation on the basis functions and the plane wave coefficients. One such case is when one wants to calculate the far field for plane wave scattering using the T matrix $T_0 = R(\vec{c})T_1R(\vec{c})^t$. The asymptotic behaviour of the basis functions gives the following relation:

$$R^{op}(\vec{c})\,\vec{\psi}_{in}(k\vec{r}) \approx e^{ik\vec{c}\cdot\vec{r}}\,\vec{\psi}_{in}(k\vec{r}) \quad \text{for } r>>c \tag{1.34}$$

The plane wave $IE^i = \vec{A}\, e^{i\vec{k}\cdot\vec{r}}$ where $\vec{A}$ is a constant vector defining the polarization, has the expansion $IE^i = \sum_{in} a_{in}\, Re\,\vec{\psi}_{in}$

This gives the relation ($\vec{A}$ is unaffected by translation and $e^{i\vec{k}\cdot\vec{r}} = e^{i\vec{k}\cdot\vec{r}_1}\, e^{i\vec{k}\cdot\vec{c}}$, $\vec{r} = \vec{r}_1 + \vec{c}$):

$$R^{op}(\vec{c})\,\vec{a} = e^{-i\vec{k}\cdot\vec{c}}\,\vec{a} \tag{1.35}$$

Used together these two simplifications give

$$IE^s = \vec{\psi}^t T_0\,\vec{a} \approx e^{i(\vec{k}-k\hat{r})\cdot\vec{c}}\,\vec{\psi}^t T_1\,\vec{a} \quad \text{for } r>>c \tag{1.36}$$

which of course is of great importance in the numerical computation of scattering amplitudes.

II. Description of the computer program

Included in the system are seven Mainroutines which we associate with the following symbols.

T-00 computes the T matrix of an infinitely conducting body.

Q-D computes Q matrices for a dielectric body.

Q-T computes, for given Q matrices, the T matrix for a homogeneous dielectric body.

Q,T-T computes, for given Q and T matrices, T matrices for layered bodies by means of the iteration procedure in (1.26)

T1,T2-T computes, for given single body T matrices, the total T matrix of a two body scatterer.

T-T computes, for given T matrix, the translated T matrix.

PSIS computes the scattered field by means of T matrix and coefficients of incoming wave.

These routines can be used together as follows by Fig. 5. Routines called by the Mainroutines are:

BESSFL computes the spherical Bessel function for real argument.

CBESS computes the spherical Bessel function for complex argument.

BN computes the spherical Neuman function for real argument.

CBN computes the spherical Neuman function for complex argument.

LEG computes the associated Legendre polynom.

TRIXJ computes the three-j symbol for m_3 equal to zero.

VPSI computes the basis functions.

VKOEF computes the plane wave coefficients.

VR computes the translation matrices.

The following routines perform some matrix operations.

MCNV inverts a complex matrix

COND conditions Q matrices in the sense of Waterman.

ORTHO orthogonalizes Q matrices.

PERFT computes, by the orthogonalization procedure, T matrices.

The following routines compute $r(\theta)$ and $\frac{dr}{d\theta}$, used in the numerical integrations, for different sections of the body.

TRCIRC a circle with centre on the z-axis.

LINE a line

ELLIPS an ellips with centre at origin and one axis along the z-axis.

TRELLI an ellips with centre on and one axis parallell with the z-axis.

Sphere-cone-sphere is a combination of routines which generates the sphere-cone-sphere shape.

The following routine computes the endpoints for the integration in the various sections in composed figures.

SPCOSP for a sphere-cone-sphere with the z-axis as axis of rotational symmetry.

IIa. The Mainroutines

T-00 program

The Mainroutine T-00 which computes the T matrix for an infinitely conducting body has the following input parameters.

NRISK, NTRIX and NEND (integers) are parameters used for the factorials FCT (k+1) $\equiv$ k! (FCT real * 8) in order to avoid overflow, see TRIXJ.

NR (integer) is half the dimension of the $\bar{Q}$ matrices and is chosen as NR $\geq$ ka where k is the wave vector and a is the radius of the smallest around the body circumscribed sphere, with centre at origin.

NP (integer) is the polarization index which is 0 for orthogonal and 1 for parallell polarization.

ISYM (integer) is the symmetri parameter and

ISYM=0 for no symmetry (other than the rotational)

ISYM=1 for mirror symmetry in the x-y plane and

ISYM=2 for ellipsoid "symmetry". (i.e. for ellipsoids with center in origin).

K1 (real * 8) is the wavevector.

A, B, C, D, E, F and G (real * 8) are body size parameters which are given values according to the type of body (See the routines for body shape.).

NSECT (integer) is the number of sections in the numerical integration choosen according to the different sections which the body is constructed of.

NDLTH(I) (integer) is the number of integration intervals in the numerical integration in section number I. Observe that NDLTH(I) has to be divisible by four!

THETAI (real * 8), in radians, is the endpoint and starting point

respectively for the numerical integration in section I resp. I + 1.

After 41 continue the routine or system of routines which generates the body shape are called.

The various fields are dimensioned as follows

NDLTH and CDH are vectors of dimension NSECT.

PN is a vector of dimension NR + 1.

X and Y are vectors of dimension 2·NR.

AR, AI, BR and BI are matrices of dimension NR.

T, RE and IM are matrices of dimension 2·NR.

TN is a matrix of the the type (NR+1) x 2·NR x 2·NR.

The routine uses the subroutines BESSEL, BN and LEG to compute the transpose of Q^1 and Q^2 by means of the Simpson's formula, given in Appendix I, for the numerical integration. The subroutine PERFT calls the conditioning routine COND and orthogonalization routine ORTHO and computes the T matrix $\bar{\bar{T}}(m)$ called T in the program. The matrix TN in the program is the collection of all $m_{max} + 1$ $\bar{\bar{T}}(m)$ matrices. That is $TN(m + 1, I,J) = \bar{\bar{T}}(m)_{I,J}$. The routine separates the real and imaginary parts of all matrices except the matrices T and TN. Thus AR resp. AI are the real resp. imaginary part of $Q^1(out, Re)^t$ and BR resp. BI are the real resp. imaginary parts of $Q^2(out, Re)^t$. Furthermore RE resp. IM are the real resp. imaginary part of $\bar{\bar{Q}}(out, Re)^t$.

Q-D program

The Maincutine Q-D which computes the Q matrices for dielectric bodies has the following input parameters:

NRISK, NTRIX and NEND (integers) are parameters used for the factorials FCT (k+1) = k! (FCT real * 8) in order to avoid overflow, see TRIXJ.

NR (integer) is half the dimension of the $\bar{Q}$ matrices and is chosen as $NR \geq 2|k|a$ for a homogeneous body. Here k is the complex wave vector outside the surface and a is the radius of the smallest around the body circumscribed sphere, with centre at origin, when a direct inversion of $\bar{Q}$(out, Re) is assumed.

NP (integer) is the polarization index which is 0 for orthogonal and 1 for parallell polarization.

LAY (integer) is a parameter which is 0 for homogeneous bodies (only $\bar{Q}$(Re, Re) and $\bar{Q}$(out, Re) are computed) and 1 for layered bodies ($\bar{Q}$(Re, out) and $\bar{Q}$(out, out) are also computed).

PMY1 and PMY2 (real * 8) are the relative permeabilities outside resp. inside the surface.

K1 (complex * 16) is the complex wave vector outside the surface.

NIND and KAPPA (real * 8) are parameters which determine the complex wave vector K2 (complex * 16) inside the surface in the following way: K2 = NIND·(1 + i·KAPPA)·K1 (i is the imaginary unit with $i^2 = -1$). Observe that we always have ImK2 ≥ 0!

A, B, C, D, E, F and G (real * 8) are body size parameters which are given values according to body (see the routines for body shape).

NSECT (integer) is the number of sections in the numerical integration choosen according to the different sections of which the body is constructed.

NDLTH(I) (integer) is the number of integration intervalls in the numerical integration in section number I. Observe that NDLTH(I) has to be divisible by four!

THETAI(real * 8), in radians, is the endpoint and starting point respectively for the numerical integration in section I resp. I + 1.

After 41 continue the routine or system of routines which generates the body shape are called.

The various fields are dimensioned as follows.

NDLTH and CDH are vectors of dimension NSECT.

PN is a vector of dimension NR + 2.

X1, Y1, X2, Y2 are vectors of dimension NR + 1.

ARR, AOR, ARO, AOO, BRR, BOR, BRO, BOO, CRR, COR, CRO, COO, DRR, DOR, DRO, DOO are matrices of dimension NR.

Q is a matrix of dimension 2·NR.

The routine uses the subroutines CBESS, CBN and LEG to compute the matrices A, B, C and D, which are defined in (1.23) and called ARR, AOR etc. in the program (ARR = A(Re, Re), AOR = A(out, Re) etc.). Simpson's formula, which is given in Appendix I, is used for the numerical integration. These matrices are then used to compute the matrices $\bar{Q}(m)$ which are stored on a tape. If LAY = 0 the matrices $\bar{Q}$(Re, Re, m) and $\bar{Q}$(out, Re, m) are stored on one tape (21) in the following order:

$\bar{Q}$(Re, Re, 0), $\bar{Q}$(out, Re, 0), $\bar{Q}$(Re, Re, 1), $\bar{Q}$(out, Re, 1), $\bar{Q}$(Re, Re, 2), $\bar{Q}$(out, Re, 2),.....,$\bar{Q}$(out, Re, m_{max}). If LAY = 1 the matrices $\bar{Q}$(Re, out, m) and $\bar{Q}$(out, out, m) are stored on an

additional tape (22) in the following order:

$\bar{Q}$(Re, out, 0), $\bar{Q}$(out, out, 0), $\bar{Q}$(Re, out, 1), $\bar{Q}$(out, out, 1), $\bar{Q}$(Re, out, 2),.....,$\bar{Q}$(out, out, m_{max}). The matrix Q in the program is used for $\bar{Q}\left(\begin{Bmatrix}\text{Re}\\ \text{Out}\end{Bmatrix}, \begin{Bmatrix}\text{Re}\\ \text{Out}\end{Bmatrix}\right)$.

Q-T program

The Mainroutine Q-T which computes T matrices for homogeneous dielectric bodies has the input parameter

NR (integer) which is half the dimension of the $\bar{Q}$ matrices.

The various fields are dimensioned as follows.

LL and MM are vectors of dimension 2·NR.

QRR and QOR are matrices of dimension 2·NR.

TN is a matrix of the type $(NR+1) \times 2 \cdot NR \times 2 \cdot NR$.

The routine reads $\bar{Q}(Re, Re, m)$ and $\bar{Q}(out, Re, m)$ from a tape.

By means of the inversion procedure MCNV the collection of the $\bar{T}(m)$ matrices, which is called TN in the program, is computed.

That is $TN(m + 1, I, J) = \bar{T}(m)_{I,J}$.

Q, T-T program

The Mainroutine Q, T-T which computes T matrices for layered bodies has the input parameter:

NR(integer) which is half the dimension of the $\bar{\bar{Q}}$ and $\bar{\bar{T}}$ matrices.

The various fields are dimensioned as follows.

LL and MM are vectors of dimension 2·NR.

Q1, Q2, Q3 and T are matrices of dimension 2·NR.

TN is a matrix of the type (NR+1) x 2·NR x 2·NR.

The routine reads $\bar{\bar{Q}}$(Re, Re, m) and $\bar{\bar{Q}}$(Out, Re, m) from the first tape (21), $\bar{\bar{Q}}$(Re, out, m) and $\bar{\bar{Q}}$(out, out, m) from the second tape (22) and $\bar{\bar{T}}$(m), called T in the program, from the third tape (23). By means of the inversion procedure MCHV the resulting $\bar{\bar{T}}$(m) matrices, also called T in the program, are computed by formula (1.26) The collection of the resulting $\bar{\bar{T}}$(m) matrices is called TN in the program. That is TN(m+1, I, J) = $\bar{\bar{T}}(m)_{I,J}$.

T1, T2-T program

The Mainroutine T_1, T_2-T which computes the total T matrix for two bodies has the following input parameters.

NRISK, NTRIX and NEND (integers) are parameters used for the factorials FCT(k+1) ≡ K! (FCT real * 8) in order to avoid overflow, see TRIXJ.

ND (integer) is the dimension of the $\bar{T}_i(m)$ matrices.

NP (integer) is the polarization index which is 0 for orthogonal and 1 for parallell polarization.

SEP (real * 8) is the wave vector times the distance between the origins to which the T matrices of the two bodies are related. Body number one with T matrix T_1 is situated at z = $^{-SEP}$/wave vector on the negative z-axis and body number two with T matrix T_2 is situated at z = +SEP/wave vector on the positive z-axis.

The various fields are dimensioned as follows.

LL and MM are vectors of dimension ND.

X1, X2, Y1 and Y2 are vectors of dimension ND + 1.

R1, R2, R3, R4, AT1, AT2, BT1, BT2, T, RET, RETT , TRAN, T1 and T2 are matrices of dimension ND.

TN is a matrix of the type $\left(\frac{ND}{2} + 1\right)$ x ND x ND.

The T matrices $\bar{T}_1(m)$ resp. $\bar{T}_2(m)$ of bodies number 1 resp. 2, called T1 resp. T2 in the program, are read on tape (21) and tape (22) respectively. By means of the translation matrices computed by the routine VR and the inversion procedure MCNV the total T matrix $\bar{T}(m)$, called T in the program, is computed by formula (1.27).

The collection of the matrices $\bar{T}(m)$ is called TN in the program. That is TN(m+1, I,J) = $\bar{T}(m)_{I,J}$. Observe that the subroutine VR calls the subroutines BESSEL and BN as well as the function TRIXJ.

T-T program

The Mainroutine T-T which computes the translated T matrix for a given T matrix has the following input parameters.

NRISK, NTRIX and NEND (integers) are parameters used for the factorials FCT(K + 1) = K! (FCT real * 8) in order to avoid overflow, see TRIXJ.

ND (integer) is the dimension of the $\bar{T}(m)$ matrix.

NP (integer) is the polarization index which is 0 for parallell and 1 for orthogonal polarization.

TR (real * 8) is the wave vector times the translation (along the z-axis). TR (positive) is defined in such a way that the resultant T matrix refers to a coordinate system which is a parallell translation TR/wave vector in the negative $\hat{z}$-direction of the original one.

The various fields are dimensioned as follows.

X1, X2, Y1 and Y2 are vectors of dimension ND + 1.

T, R1, R2, R3 and R4 are matrices of dimension ND.

TN is a matrix of the type $\left(\frac{ND}{2} + 1\right)$ x ND x ND.

The routine reads the collection of the $\bar{T}(m)$ matrices, which is called TN in the program. That is $TN(m+1, I,J) = \bar{T}(m)_{I,J}$. By means of the regular translation matrices called by subroutine VR the translation of $\bar{T}(m)$, which is called T in the program, is computed. The collection of the resulting $\bar{T}(m)$ is also called TN in the program. Observe that the subroutine VR calls the subroutines BESSEL and BN as well as the function TRIXJ.

PSIS program

The Mainroutine PSIS which computes the scattered field has the following input parameters.

NRISK, NTRIX and NEND (integers) are parameters used for the factorials FCT(k+1) ≡ K! (FCT real * 8) in order to avoid overflow, see TRIXJ.

NP (integer) is the polarization index which is 0 for orthogonal and 1 for parallell polarization.

NPCHAN (integer) is a parameter which for the value 0 has no influence on the computation. If one has computed the T matrix for the other polarization (i.e. other than the before given value of NP) it is necessary to give NPCHAN the value 1. This will transform the T matrix used in the program to the one with the correct polarization.

ND (integer) is the dimension of the $\bar{T}(m)$ matrix.

NB (integer) is a parameter used in the subroutine VPSI which computes the basis functions.

If NB = 1 then $kr \cdot \exp(-ikr)\, \vec{\psi}_{lm}$ is computed.

If NB = 2 then Re $\vec{\psi}_{lm}$ is computed.

If NB = 3 then $\vec{\psi}_{lm}$ is computed.

KV (real * 8) is the wave vector.

BHETA (real * 8) is the polar angle, in radians, of the wave vector of the incoming plane wave.

BHETAD (real * 8) is the angle BHETA in degrees.

DIST (real * 8) is the wave vector times the radius to the observation point .

THETA (real * 8) is the polar angle, in radians, to the observation point.

THETAD (real * 8) is the angle THETA in degrees.

FHI (real * 8) is the azimut angle, in radians, to the observation point.

FHID (real * 8) is the angle FHI in degrees.

The various fields are dimensioned as follows.

PN, X and Y are vectors of dimension $\frac{ND}{2} + 2$.

AP, FEXP, PSIR, PSITH, and PSIFH are vectors of dimension ND.

TN is a matrix of the type $\left(\frac{ND}{2} + 1\right) \times ND \times ND$.

The program reads the collection of the matrices $\bar{\bar{T}}(m)$ which is called TN in the program. That is $TN(m + 1, I,J) = \bar{\bar{T}}(m)_{I,J}$.

After calling the subroutines VKOEF and VPSI the plane wave coefficients resp. basis functions are computed. FEXP correspond to the expansion coefficients $\vec{f}$ for the scattered wave. $FEXP(I) = \sum_J \bar{\bar{T}}(m)_{I,J}\ \bar{a}_J$ where $\bar{a}_J$ is the plane wave coefficient after having performed the same transformation as in the definition of $\bar{\bar{T}}(m)$. PSIR, PSITH resp. PSIFH are the $\hat{r}$, $\hat{\vartheta}$ resp. $\hat{\varphi}$ components of $\begin{cases} kr \cdot \exp(-ikr) \\ Re \\ 1 \end{cases} \vec{\psi}$ which also is transformed like the plane wave coefficients. P1 resp. P2 are the $\hat{\theta}$ resp. $\hat{\varphi}$ components of the field $\vec{\psi}^{\,s}$ multiplied by $kr \cdot \exp(-ikr)$ when NB=1. These two "amplitudes" defines the scattered far field completely and can be used for other computations as well.

IIb. The subroutines and functions

Subroutine BESSEL computes a spherical Bessel-function of given order and given real argument. The routine uses the series expansion

$$j_n(X) = \frac{X^n}{(2n+1)!!} \sum_{i=0}^{\infty} a_i \,, \quad a_0 = 1 \,, \quad a_i = \frac{-X^2 a_{i-1}}{2i(2n+(2i+1))}$$

This series is summed up at most to 101 terms or until the relative contribution is less than or equal to 10^{-20}. If after the addition of the 101:st term the relative contribution is greater than 10^{-20} the error parameter (integer) will be given the value 1, instead of 0, and ERROR IN SUM OF BESSEL will printed. The routine is used as follows: BESSEL (n, x, $j_n(x)$, error parameter). Subroutine BN computes two vectors where the elements are the spherical Bessel and Neuman functions respectively for given real argument and varying order up to one unit less than a given number. The routine calls subroutine BESSEL to obtain the two Bessel functions of highest order needed. The recurrence relation

$$f_{n-1}(X) + f_{n+1}(X) = (2n+1)\, X^{-1} f_n(X)$$

for the two kinds of spherical functions is used to generate the Bessel functions of lower order. The routine starts with $n_0(X) = -\frac{\cos X}{X}$ $n_1(X) = -\frac{\cos X}{X^2} - \frac{\sin X}{X}$ and uses the recurrence relation to generate the remaining Neuman functions up to the same order as the Bessel functions. From the Wronskian for the two kinds of spherical functions it is possible to construct two tests for the functions:

For Bessel functions: $j_1(X) n_0(X) - j_0(X) n_1(X) = X^{-2}$

For Neuman functions: $j_N(X) n_{N-1}(X) - j_{N-1}(X) n_N(X) = X^{-2}$

where N is the biggest order used. These two tests are used in the routines T-OO, VR and VPSI. Observe that the result fields of BN has the dimension N + 1 which also is an input parameter of BN. The routine is used as follows:

$$BN\{X, N+1, (j_0(X), j_1(X), \ldots\ j_N(X)), (n_0(X), n_1(X), \ldots n_N(X))\}$$

The subroutines CBESS resp. CBN are like the subroutines BESSEL resp. BN with the exception that the argument of the two kinds of Bessel functions is complex. The test for complex Bessel functions is performed in the routine Q-D.

Subroutine LEG computes a vector where the elements are the associated Legendre functions for given argument and azimut index and for varying order up to one unit less than a given number. The routine uses the formula:

$$P_m^m(x) = \frac{(2m)!\,(1-x^2)^{m/2}}{m!\;2^m} \quad , \quad P_i^m(x) \equiv 0 \text{ for } i<m$$

and the recurrence relation

$$P_n^m(x) = (n-m)^{-1}\left((2n-1)x\,P_{n-1}^m(x) - (n+m-1)\,P_{n-2}^m(x)\right)$$

Observe that the resulting field of LEG has the dimension $n_{max}+1$ which also is an input parameter. The routine is used as follows:

$$LEG\left(\Theta, m, n_{max}+1, \left(P_0^m(\cos\Theta), P_1^m(\cos\Theta), \ldots P_{n_{max}}^m(\cos\Theta)\right)\right)$$

Function TRIXJ computes the Wigner 3-j symbol for four parameters. The routine uses the Wigner formula. The connection to the 3-j symbol is given by:

$$TRIXJ(2j_1, 2j_2, 2j_3, 2m, FCT, N, L) = \begin{pmatrix} j_1 & j_2 & j_3 \\ m & -m & 0 \end{pmatrix}$$

where FCT is the factorials of integers stored as a vector (FCT real * 8).

FCT (k + 1) = k! for $k + 1 \leq N$

FCT (k + 1) = k! x 10^{-L} for $k + 1 \geq N + 1$

The routine TRIXJ has two different error indications. One indication "Error in argument of 3-j" is given if the following is not satisfied:

1) the arguments are even. (The j-s and m-s).

2) $j_1 \geq |m|$, $j_2 \geq |m|$

The other indication "Factorials exceeded" is given if in the calculations of the requested 3-j symbol there is required a factorial which is not stored. In both these cases the 3-j symbol is given no value at all and the whole program will stop.

Subroutine VPSI which computes the basic functions has the following input parameters.

RAD (real * 8) is the wave vector times the radius vector to the observation point.

THETA (real * 8) is the polar angle, in radius, to the observation point.

FHI (real * 8) is the azimut angle, in radians, to the observation point.

NP (integer) is the polarization index which is 0 for orthogonal and 1 for parallell polarization.

NB (integer) is a parameter for choice of kind of basis function.

NB = 1 gives $RAD \cdot \exp(-i \cdot RAD)\ \vec{\vec{\Psi}}$.

NB = 2 gives Re $\vec{\vec{\Psi}}$.

NB = 3 gives $\vec{\vec{\Psi}}$

M (integer) is the azimutindex.

ND (integer) is the dimension of the $\bar{\bar{T}}(m)$ matrices.

The output PSIR, PSITH resp. PSIFH are the $\hat{r}$, $\hat{\theta}$ resp. $\hat{\varphi}$ components of the basis functions which of course are transformed in the same way as the $\bar{\bar{T}}(m)$ matrices.

PN, U and V are auxiliary fields for the storage of Legendre polynomials and spherical Bessel functions.

The various fields are dimensioned as follows.

PSIR, PSITH and PSIFH are vectors of dimension ND.

PN is a vector of dimension $\frac{ND}{2} + 2$.

U, V are vectors of dimension $\frac{ND}{2} + 1$.

The routines LEG, BESSEL and BN are called when NB = 2 or 3.

When NB = 1 only the routine LEG is called.

Subroutine VKOEF computes the plane wave coefficients and has the following input parameters.

BHETA (real * 8) is the polar angle, in radians, of the incoming wave vector.

NP (integer) is the polarization index which is 0 for orthogonal and 1 for parallell polarization.

M (integer) is the azimut index.

ND (integer) is the dimension of the $\bar{\bar{T}}(m)$ matrices.

The fields AP and PN have the dimension ND resp. $\frac{ND}{2} + 2$.

AP are the plane wave coefficients transformed in the same way as $\bar{\bar{T}}(m)$. The subroutine LEG is called by the routine and the Legendre polynomials are stored in PN.

Subroutine VR computes the translation matrices $\bar{R}(C\,\hat{z}, M)$, $\bar{\bar{E}}(C\,\hat{z}, M)$, $\bar{\bar{E}}(-C\,\hat{z}, M)$ resp. $\bar{R}(\frac{1}{2}\,C\,\hat{z}, M)$, which are called R1, R2, R3 resp. R4 in the routine.

AR which is an input parameter of the routine is the wave vector times the translation distance called C above (AR positive real $\neq 0$).

NP (integer) is the polarization parameter which is 0 for orthogonal and 1 for parallell polarization.

M (integer) is the azimut index.

ND is the dimension of the matrices.

X1, X2, Y1, Y2 are auxillary vectors of dimension ND + 1 where the various Bessel functions are stored.

The routine uses subroutines BESSEL and BN as well as the function TRIXJ.

Subroutine MCNV inverts a given complex matrix. The routine uses the standard Gauss-Jordan method. When the routine is called it destroys the given matrix and stores the elements of the inverse in the storage where the given matrix was stored before. The determinant is computed. One has to give the dimension of the matrix as well as to make area for two auxiliary vectors of the same dimension but with integer elements. The routine is used as follows. MXNV (matrix, dimension, determinant, auxiliary vector, auxiliary vector).

Subroutine COND which performs the conditioning of the real resp. imaginary parts of $\bar{Q}(out, Re)^t$ in the orthogonalization procedure has the following input parameters.

M (integer) is the azimut index.

ND (integer) is the dimension of the $\bar{Q}$ matrices.

The matrices RE and IM have dimension ND.

The matrices RE and IM (both real * 8) are the real resp. the imaginary parts of $\bar{Q}(out, Re)^t$ as input. The conditioned matrices are also called RE and IM in the routine. The routine works as described in Ref. [1].

Subroutine ORTHO which orthogonalizes the conditioned matrices the real and imaginary parts resp. of $\bar{Q}(Out, Re)^t$ has the following input parameters.

M (integer) is the azimut index.

ND (integer) is the dimension of the $\bar{Q}$ matrices.

The fields are dimensioned as follows

X and Y are auxiliary vectors of dimension ND.

RE and IM are matrices of dimension ND.

The matrices RE and IM are the conditioned real and imaginary parts resp. of $\bar{Q}(out, Re)^t$ as input. The orthogonalized matrices are also called RE and IM in the routine. The routine works as described in Ref. [1].

Subroutine PERFT which computes the matrix $\bar{T}(m)$ (complex * 16), after the conditioning and orthogonalization, has the following input parameters.

NP (integer) is the polarization index which is 0 for orthogonal and 1 for parallell polarization.

NR (integer) is half the dimension of the $\bar{Q}(m)$ matrices.

ND (integer) is 2·NR.

The fields are dimensioned as follows.

X and Y are auxiliary vectors of dimension ND.

AR, AI, BR and BI are matrices of dimension NR.

T, RE and IM are matrices of dimension ND.

The matrices AR, AI, BR and BI are the real part of $Q^1(\text{out, Re})^t$, the imaginary part of $Q^1(\text{out, Re})^t$, the real part of $Q^2(\text{out, Re})^t$ and the imaginary part of $Q^2(\text{Out, Re})^t$ as input.

These matrices are used to compute RE and IM (i.e. the real and imaginary parts resp. of $\bar{Q}(\text{out, Re})^t$). After this the RE and IM matrices are conditioned and orthogonalized. Finally the $\bar{T}(m)$ matrices (complex * 16), called T in the routine, are computed by RE amd IM (of course not the original one). Notice that we could have separated real and imaginary parts of $\bar{T}(m)$ also, thus not introducing complex numbers.

Subroutine LINE computes $r(\theta)$ and $\frac{dr}{d\theta}$ for a line. The angle θ is the polar angle (measured from the z-axis). The routine has the following input parameters.

THETA (real * 8) is the polar angle, in radians.

MIP (integer) is a parameter which has the values +1 and -1 and gives the value $-\frac{\frac{dz}{dx}}{\left|\frac{dz}{dx}\right|}$ of the line (considered as a line in the x-z plane).

C (real * 8) is the z-coordinate of the intersection of the z-axis and the line.

ALPHA (real * 8) is the smaller angle between the line and the z-axis, in radians.

The output R and DR (both real * 8) are the computed $r(\theta)$ resp. $\frac{dr}{d\theta}$.

Subroutine TRCTRC computes $r(\theta)$ and $\frac{dr}{d\theta}$ for a circle with the center translated from the origin. The angle θ is the polar angle (measured from the z-axis). The routine has the following input parameters.

STH and CTH are sin resp. cos of the angle θ. (both real * 8).

C (real * 8) is the z-coordinate of the center of the circle.

A (real * 8) is the radius of the circle.

The output R and DR (both real * 8) are the computed $r(\theta)$ resp. $\frac{dr}{d\theta}$.

Subroutine ELLIPS computes $r(\theta)$ and $\frac{dr}{d\theta}$ for an ellips with center at the origin. The angle θ is the polar angle (measured from the z-axis). The routine has the following input parameters.

STH and CTH are sin resp. cos of the angle θ (both real * 8).

AZ is the length of the ellips half axis which lies on the z-axis (real * 8).

BRA is the length of the other half axis (real * 8).

The output R and DR (both real * 8) are the computed $r(\theta)$ resp. $\frac{dr}{d\theta}$.

Subroutine TRELLI computes $r(\theta)$ and $\frac{dr}{d\theta}$ for an ellips with the center translated from the origin. The routine has the same in and output as ELLIPS except C (real * 8) which is the z-coordinate of the center of the ellips.

Sphere-cone-sphere.

The shape of a sphere-cone-sphere as in Fig. 6 is generated by the subroutines TRCIRC and LINE using subroutine SPCOSP to compute the end- and starting points of the three sections. The body shape used in Ref. [1] is well suited as a test surface to check the relations (1.24) and (1.25) numerically. We also want to give it in order to illustrate how to use a composite body shape. The body consists of a bigger sphere with radius a at the upper end of a conical section with half angle α and a smaller sphere with radius $b=\frac{a}{1+\sin\alpha}$ at the lower end of the conical section. The bigger sphere has its center at $z = \frac{a}{2}\ \frac{1-\sin\alpha}{1+\sin\alpha}$ and this section starts at $\theta = 0$. At $\theta=\theta_1=$ $\arctg\left(\frac{\sin\alpha\cdot\cos\alpha}{q-\sin^2\alpha}\right)$, where $q=\frac{\sin\alpha(1-\sin\alpha)}{2(1+\sin\alpha)}$, the conical section starts with continuous $\frac{dr}{d\theta}$. At $\theta=\theta_2=\arctg\left(\frac{2\cos\alpha}{1+3\sin\alpha}\right)$ the second spherical part, with radius $b\left(=\frac{a}{1+\sin\alpha}\right)$, starts also with continuous $\frac{dr}{d\theta}$. The shape of this body is not generated by a single subroutine because of the speed of the computation. Anyhow, the above procedure is easy to use because one only has to put in a few cards in the Mainprogram and define the five parameters needed. The first parameter A (real * 8) is the radius of the bigger sphere. The second parameter is ALPHA the half cone angle (real * 8), in radians. Then follows NDLTH(I) the number of integration intervals in the three sections, which all can be different but all has to be divisible by four. The cards (as shown in Appendix II) A =.... DO to CDH(3) =.... shall be inserted before the card N = NRISK in the programs T-00 or Q-p. The cards GO TO (1, 2, 3), ISECT to 4 CONTINUE shall be inserted after card 41 CONTINUE in the programs T-00 or Q-p.

Subroutine SPCOSP computes the endpoints of the integrations in the three sections in the sphere-cone-sphere. The routine is used together with the routines TRCIRC and LINE to generate the shape of a sphere-cone-sphere, see Fig. 6. Input parameters are as follows.

ALPHA (real * 8) is the half-angle of the conical section, in radians.

A (real * 8) is the radius of the bigger sphere, which always is situated above the smaller sphere (on the z-axis).

The out put THETA 1 (real * 8) is the endpoint, in radians, of the first integration (section 1) which is over a part of the bigger sphere with radius A.

THETA 1 is also starting point of the second integration (section 2) which is over the conical part with half cone angle ALPHA.

THETA 2 (real * 8) is the end point, in radians, of the second integration, (section 2) and starting point of the third integration (section 3) which is over a part of the smaller sphere.

References

1. P.C. Waterman, "Numerical Solution of Electromagnetic Scattering Problems" in Computer Techniques for Electromagnetics, R. Mittra, Ed. (Pergamon, Oxford, 1973).

2. P.C. Waterman, Phys. Rev. D3, 825 (1971).

3. B. Peterson, S. Ström, Phys. Rev. D8, 3661 (1973).

4. B. Peterson, S. Ström, Phys. Rev. D10, 2670 (1974).

5. A.R. Edmonds, Angular Momentum in Quantum Mechanics (Princeton Univ. Press, Princeton, New Jersey, 1957).

Appendix I

The explicit expression for the basis functions is.

$$\vec{\Psi}_{1\begin{Bmatrix}e\\o\end{Bmatrix}mn} = \gamma_{mn}^{1/2} h_n^{(1)}\left(\frac{m}{\sin\theta} P_n^m \begin{Bmatrix}-\sin m\varphi\\ \cos m\varphi\end{Bmatrix}\hat{\theta} - \frac{\partial}{\partial\theta} P_n^m \begin{Bmatrix}\cos m\varphi\\ \sin m\varphi\end{Bmatrix}\hat{\varphi}\right)$$

$$\vec{\Psi}_{2\begin{Bmatrix}e\\o\end{Bmatrix}mn} = \gamma_{mn}^{1/2}\left(n(n+1)\frac{h_n^{(1)}}{kr} P_n^m \begin{Bmatrix}\cos m\varphi\\ \sin m\varphi\end{Bmatrix}\hat{r} + \right.$$

$$\left. + \frac{1}{kr}\left(\frac{\partial}{\partial r}(r h_n^{(1)})\right)\left(\frac{\partial}{\partial\theta} P_n^m \begin{Bmatrix}\cos m\varphi\\ \sin m\varphi\end{Bmatrix}\hat{\theta} + \frac{m}{\sin\theta} P_n^m \begin{Bmatrix}-\sin m\varphi\\ \cos m\varphi\end{Bmatrix}\hat{\varphi}\right)\right.$$

Note the following relations.

$$Z_n'(x) = Z_{n-1}(x) - \frac{n+1}{x} Z_n(x) = \frac{n}{x} Z_n(x) - Z_{n+1}(x)$$

$$h_n^{(1)}(x) \approx (-i)^{n+1} \frac{e^{ix}}{x}, \quad x >> n$$

Here Z_n is a spherical Bessel-Neuman- or Hankel function or any linear combination of these.

$$\frac{\partial}{\partial\theta} P_n^m(\cos\theta) = -\frac{1}{\sin\theta}\left((n+1)\cos\theta P_n^m - (n-m+1) P_{n+1}^m\right)$$

Here P_n^m is the associated Legendre function as in Ref. [5].

The orthogonally and parallely polarized plane waves are

$$\vec{E}_{\perp} = \exp(i\vec{k}\cdot\vec{r})\,\hat{e}_{\perp}$$

resp.

$$\vec{E}_{\parallel} = \exp(i\vec{k}\cdot\vec{r})\,\hat{e}_{\parallel}$$

The wave vector $\vec{k}$ is always chosen as

$$\vec{k} = k(\sin\theta', 0, \cos\theta')$$

We use

$$\vec{r} = r(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)$$

The polarization vectors of the incoming wave are

$$\hat{e}_{\perp} = \hat{y}$$

and

$$\hat{e}_{\parallel} = (\cos\theta', 0, -\sin\theta') \qquad (\text{i.e. } \hat{e}_{\parallel} \times \hat{e}_{\perp} = \hat{k})$$

The plane wave can be expanded

$$\vec{E}_{\{\perp\}}^{\{\parallel\}} = \sum_{in} a_{in}^{\{\perp\}_{\{\parallel\}}} \operatorname{Re}\vec{\psi}_{in} \qquad \text{with}$$

$$a^{\perp}_{1emn} = -i^{n}\,4\pi\,\gamma_{mn}^{1/2}\,\frac{\partial}{\partial\theta'}\,P_n^m(\cos\theta')$$

$$a^{\perp}_{2omn} = -i^{n+1}\,4\pi\,\gamma_{mn}^{1/2}\,\frac{m}{\sin\theta'}\,P_n^m(\cos\theta')$$

$$a^{\perp}_{1omn} = a^{\perp}_{2emn} = 0$$

especially $a^{\perp}_{1emn} = -i\,a^{\perp}_{2omn} = -i^{n}\left[2\pi(2n+1)\right]^{1/2}\delta_{m,1}\,\theta'=0$

and

$$a^{\parallel}_{1omn} = i^{n}\,4\pi\,\gamma_{mn}^{1/2}\,\frac{m}{\sin\theta'}\,P_n^m(\cos\theta')$$

$$a^{\parallel}_{2emn} = -i^{n+1}\,4\pi\,\gamma_{mn}^{1/2}\,\frac{\partial}{\partial\theta'}\,P_n^m(\cos\theta')$$

$$a^{\parallel}_{1emn} = a^{\parallel}_{2omn} = 0$$

especially $a^{\parallel}_{1omn} = i\,a^{\parallel}_{2emn} = i^{n}\left[2\pi(2n+1)\right]^{1/2}\delta_{m,1}\,; \qquad \theta'=0$

The translation matrices for the basis function are given by

$$R(\pm\vec{d}, z_\lambda)_{2omn,2om'n'} = R(\pm\vec{d}, z_\lambda)_{1omn,1om'n'} =$$

$$= R(\pm\vec{d}, z_\lambda)_{1emn,1em'n'} = R(\pm\vec{d}, z_\lambda)_{2emn,2em'n'} =$$

$$= \frac{(-1)^m}{2} \sum_{\lambda=|n-n'|}^{n+n'} (\pm 1)^\lambda (-1)^{\frac{1}{2}(n'-n+\lambda)} \left[\frac{(2n+1)(2n'+1)}{n(n+1)n'(n+1)}\right]^{1/2} \times$$

$$\times (2\lambda+1)\left[n(n+1)+n'(n'+1)-\lambda(\lambda+1)\right]\begin{pmatrix} n & n' & \lambda \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} n & n' & \lambda \\ m & -m & 0 \end{pmatrix} z_\lambda(kd)\,\delta_{mm'}$$

$$R(\pm\vec{d}, z_\lambda)_{1emn,2om'n'} = R(\pm\vec{d}, z_\lambda)_{2emn,1om'n'} =$$

$$= -R(\pm\vec{d}, z_\lambda)_{2omn,1em'n'} = -R(\pm\vec{d}, z_\lambda)_{1omn,2em'n'} =$$

$$= \frac{(-1)^m}{2} \sum_{\lambda=|n-n'|+1}^{n+n'} (\pm 1)^\lambda (-1)^{\frac{1}{2}(n'-n+\lambda+1)} \left[\frac{(2n+1)(2n'+1)}{n(n+1)n'(n'+1)}\right]^{1/2} \times$$

$$\times (2\lambda+1)\left[(\lambda^2-(n-n')^2)((n+n'+1)^2-\lambda^2)\right]^{\frac{1}{2}}\begin{pmatrix} n & n' & \lambda-1 \\ 0 & 0 & 0 \end{pmatrix} \times$$

$$\times \begin{pmatrix} n & n' & \lambda \\ m & -m & 0 \end{pmatrix} z_\lambda(kd)\,\delta_{mm'}$$

where $R = R(\ ,j_\lambda)$, $\Gamma = R(\ ,h_\lambda^{(1)})$, $\vec{d} = d\hat{z}$

$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$ is the usual Wigner 3 j symbol as in Ref.[5].

The Q matrix for infinitely conducting bodies, with the z-axis as axis of rotational symmetry, is given by (c.f. formula (1.14), (1.18) and (1.19)):

$$Q^1(Out,Re,m)_{nn'} = \frac{2\pi}{\varepsilon_m}\left[\gamma_{mn}\gamma_{mn'}\right]^{1/2}\int_0^{\pi} d\theta \sin\theta \times$$

$$\times\Big\{kr\, h_n^{(1)}(kr)\left[\frac{\partial}{\partial r}(r j_{n'}(kr))\right]\left[\left(\frac{\partial}{\partial\theta}P_n^m(\cos\theta)\right)\left(\frac{\partial}{\partial\theta}P_{n'}^m(\cos\theta)\right)+\right.$$

$$\left.+\frac{m^2}{\sin^2\theta}P_n^m(\cos\theta)\,P_{n'}^m(\cos\theta)\right]+$$

$$+n'(n'+1)\left(\frac{\partial kr}{\partial\theta}\right)h_n^{(1)}(kr)\,j_{n'}(kr)\left(\frac{\partial}{\partial\theta}P_n^m(\cos\theta)\right)P_{n'}^m(\cos\theta)\Big\}$$

$$Q^2(Out,Re,m)_{nn'} = \pi m\left[\gamma_{mn}\gamma_{mn'}\right]^{1/2}\int_0^{\pi} d\theta\,(kr)^2 \times$$

$$\times h_n^{(1)}(kr)\,j_{n'}(kr)\,\frac{\partial}{\partial\theta}\left(P_n^m(\cos\theta)\,P_{n'}^m(\cos\theta)\right)$$

The Q matrix for dielectric (lossy) bodies, with the z-axis as axis of rotational symmetry, is given by (c.f. formula (1.14) and (1.23)):

$$A(Out,Re,m)_{nn'} = \frac{2\pi}{\varepsilon_m}\left[\gamma_{mn}\gamma_{mn'}\right]^{1/2}\int_0^{\pi} d\theta \sin\theta \times$$

$$\times\Big\{ k_1 r\Big[\frac{\partial}{\partial r}\left(r h_n^{(1)}(k_1 r)\right)\Big] j_{n'}(k_2 r)\Big[\Big(\frac{\partial}{\partial\theta} P_n^m(\cos\theta)\Big)\frac{\partial}{\partial\theta} P_{n'}^m(\cos\theta) +$$

$$+ \frac{m^2}{\sin^2\theta} P_n^m(\cos\theta) P_{n'}^m(\cos\theta)\Big] +$$

$$+ n(n+1)\Big(\frac{\partial k_1 r}{\partial\theta}\Big) h_n^{(1)}(k_1 r) j_{n'}(k_2 r) P_n^m(\cos\theta)\frac{\partial}{\partial\theta} P_{n'}^m(\cos\theta)\Big\}.$$

$$B(Out,Re,m)_{nn'} = \pi m\left[\gamma_{mn}\gamma_{mn'}\right]^{1/2}\int_0^{\pi} d\theta (k_1 r)^2 \times$$

$$\times h_n^{(1)}(k_1 r) j_{n'}(k_2 r)\frac{\partial}{\partial\theta}\left(P_n^m(\cos\theta) P_{n'}^m(\cos\theta)\right).$$

$$C(Out,Re,m)_{nn'} = \pi m\left[\gamma_{mn}\gamma_{mn'}\right]^{1/2}\int_0^{\pi} d\theta \times$$

$$\times\Big\{\Big[\frac{\partial}{\partial r}\left(r h_n^{(1)}(k_1 r)\right)\Big]\Big[\frac{\partial}{\partial r}\left(r j_{n'}(k_2 r)\right)\Big]\frac{\partial}{\partial\theta}\left(P_n^m(\cos\theta) P_{n'}^m(\cos\theta)\right) +$$

$$+ \frac{1}{r}\Big(\frac{\partial r}{\partial\theta}\Big)\Big[n(n+1) h_n^{(1)}(k_1 r)\Big[\frac{\partial}{\partial r}\left(r j_{n'}(k_2 r)\right)\Big] +$$

$$+ n'(n'+1)\Big[\frac{\partial}{\partial r}\left(r h_n^{(1)}(k_1 r)\right)\Big] j_{n'}(k_2 r)\Big] P_n^m(\cos\theta) P_{n'}^m(\cos\theta)\Big\}.$$

$$D(Out, Re, m)_{nn'} = \frac{2\pi}{\varepsilon_m}\left[\gamma_{mn}\gamma_{mn'}\right]^{1/2}\int_0^{\pi} d\theta \sin\theta \cdot$$

$$\cdot\Big\{k_1 r h_n^{(1)}(k_1 r)\left[\frac{\partial}{\partial r}(r j_{n'}(k_2 r))\right]\left[\left(\frac{\partial}{\partial\theta}P_n^m(\cos\theta)\right)\frac{\partial}{\partial\theta}P_{n'}^m(\cos\theta) + \right.$$

$$+ \left.\frac{m^2}{\sin^2\theta}P_n^m(\cos\theta)\,P_{n'}^m(\cos\theta)\right] +$$

$$+ n'(n'+1)\,\frac{\partial k_1 r}{\partial\theta}\,h_n^{(1)}(k_1 r)\,j_{n'}(k_2 r)\left(\frac{\partial}{\partial\theta}P_n^m(\cos\theta)\right)P_{n'}^m(\cos\theta)\Big\}.$$

For the numerical integrations we use the Simpson formula:

$$\int_a^b f(x)\,dx = \frac{h}{22.5}\left(7y_0 + 32y_1 + 12y_2 + 32y_3 + \right.$$

$$+ 14y_4 + 32y_5 + 12y_6 + 32y_7 + 14y_8 + \dots$$

$$\left. + \dots 32y_{N-3} + 12y_{N-2} + 32y_{N-1} + 7y_N \right) -$$

$$- (b-a)\frac{2h^6}{945} f^{(6)}(\xi)$$

where $h = \frac{b-a}{N}$, $N = 4M$ is the number of intervalls (divisible by four), $y_i = f(a+ih)$ and $\xi \in (a,b)$.

The equation of a line, as in Fig. 7, is given by:

$$r = \frac{c \sin\alpha}{\sin(\theta \pm \alpha)}$$

$$\frac{dr}{d\theta} = - \frac{c \sin\alpha \cos(\theta \pm \alpha)}{\sin^2(\theta \pm \alpha)}$$

where θ is the polar angle (measured from the z-axis). c is the z-coordinate of the intersection between the line and the z-axis. α is the smaller angle between the line and the z-axis. The + sign is for negative slope and the -sign is for positive slope (as curve in x-z plane). The equation of an ellips, with centre on the z-axis, as in Fig. 8, is given by:

$$r = \frac{b^2 c \cos\theta + ab\left[a^2\sin^2\theta + b^2\cos^2\theta - c^2\sin^2\theta\right]^{1/2}}{a^2\sin^2\theta + b^2\cos^2\theta}$$

$$\frac{dr}{d\theta} = \left(-2r\sin\theta\cos\theta(a^2-b^2) - b^2 c \sin\theta + \right.$$

$$\left. + \frac{\sin\theta\cos\theta\, ab\,(a^2-b^2-c^2)}{\left[a^2\sin^2\theta + b^2\cos^2\theta - c^2\sin^2\theta\right]^{1/2}}\right)\left[a^2\sin^2\theta + b^2\cos^2\theta\right]^{-1}$$

where θ is the polar angle (measured from the z-axis). c is the z-coordinate of the centre of the ellips. a is the half axes, along the z-axis. b is the half axes, orthogonal to the z-axis.

Fig.1

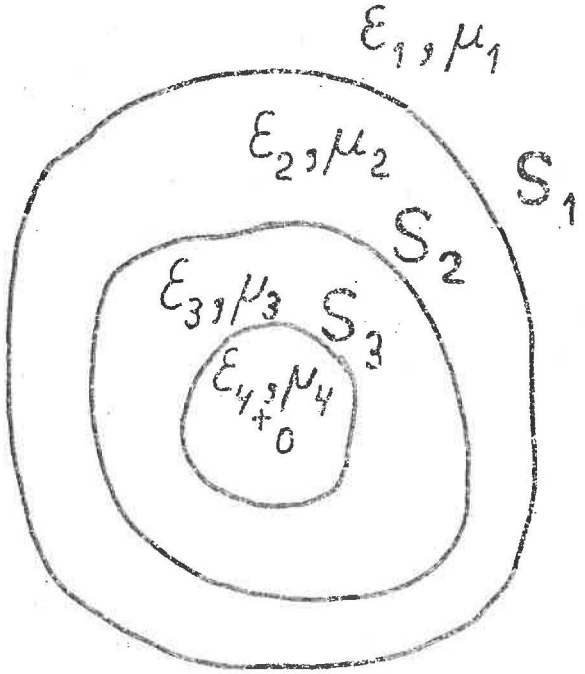

ε_1, μ_1
ε_2, μ_2
S_1
S_2
ε_3, μ_3
S_3
ε_4, μ_4
0

Fig.2

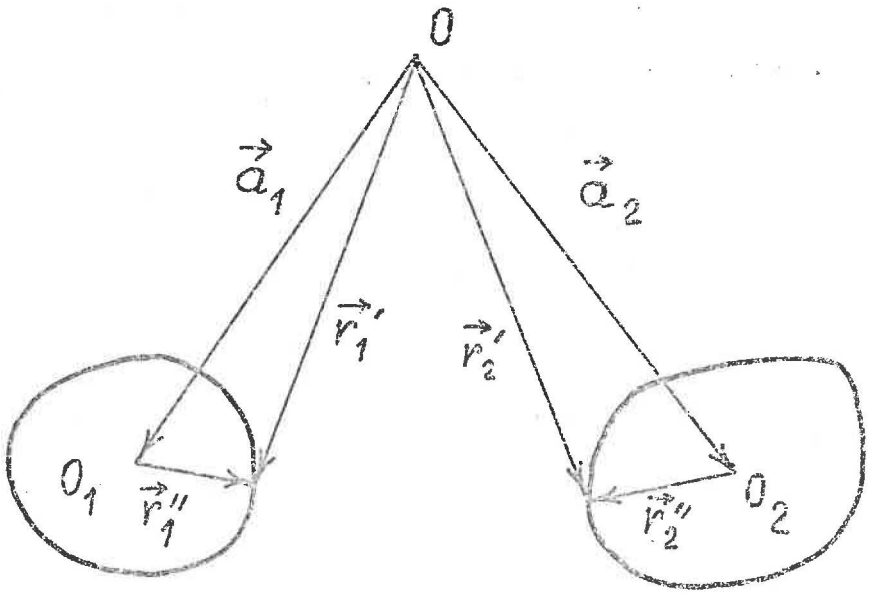

0
$\vec{a}_1$
$\vec{a}_2$
$\vec{r}_1'$
$\vec{r}_2'$
O_1
$\vec{r}_1''$
$\vec{r}_2''$
O_2

Fig. 3

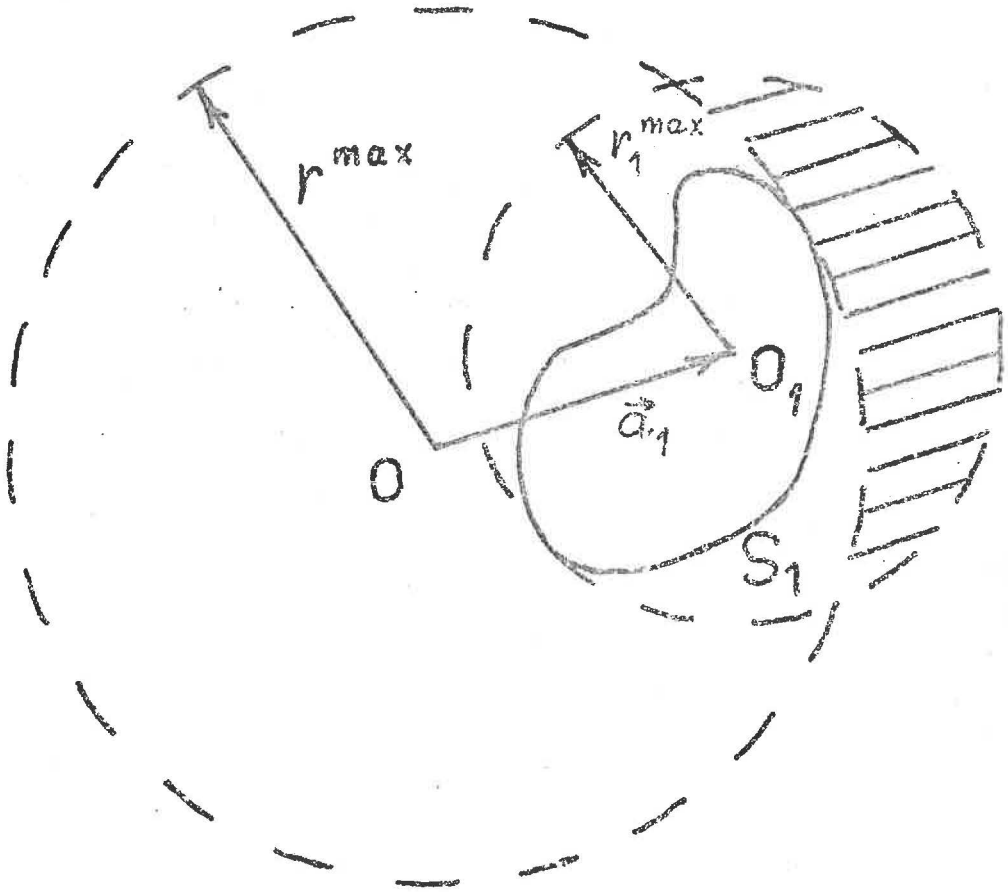


Fig. 4

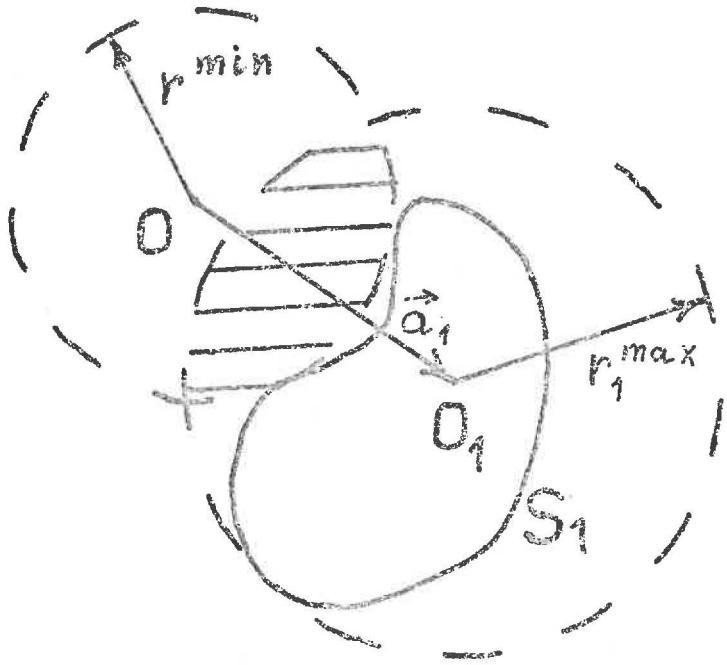


Fig. 5

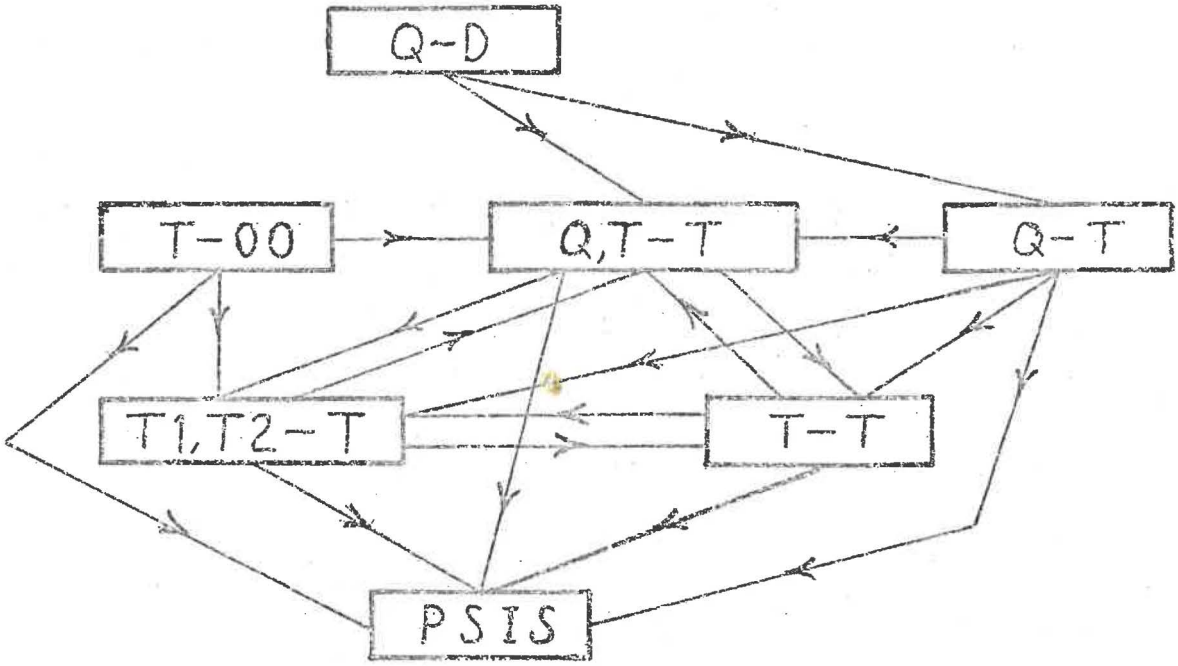

Q-D
T-00
Q,T-T
Q-T
T1,T2-T
T-T
PSIS

Fig.6

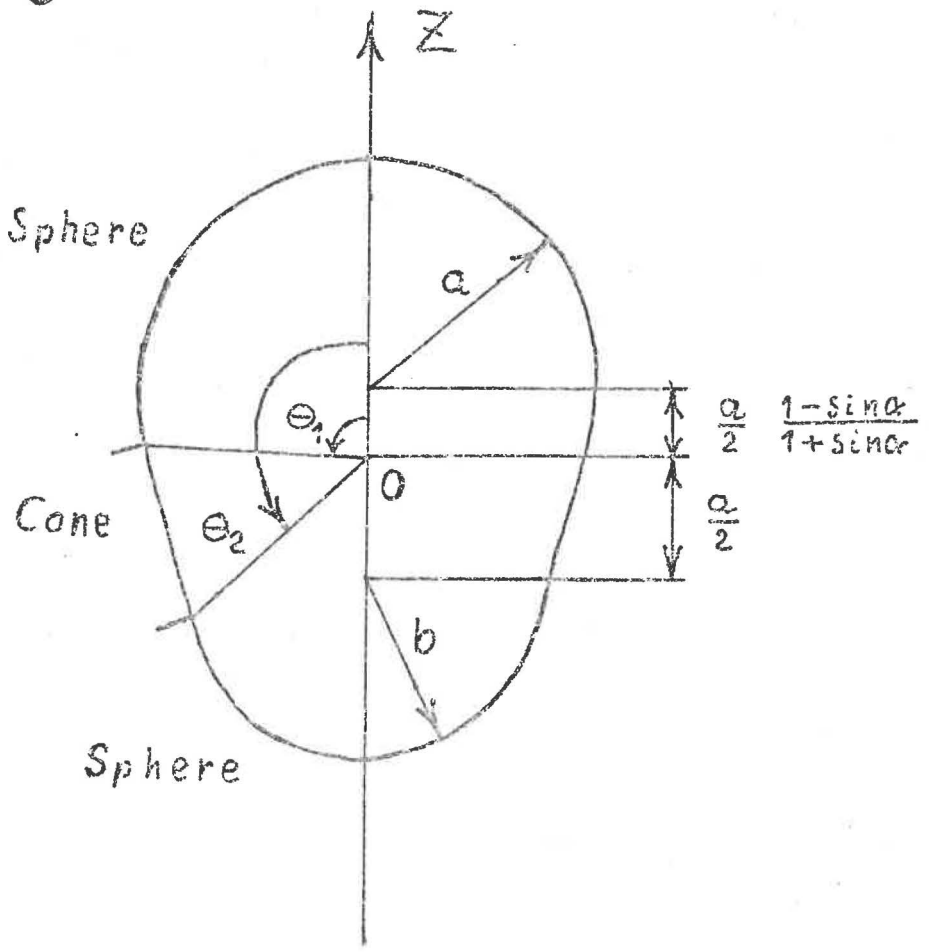

Z
Sphere
a
θ1
a/2 (1-sinα)/(1+sinα)
Cone
θ2
0
a/2
b
Sphere

Fig. 7

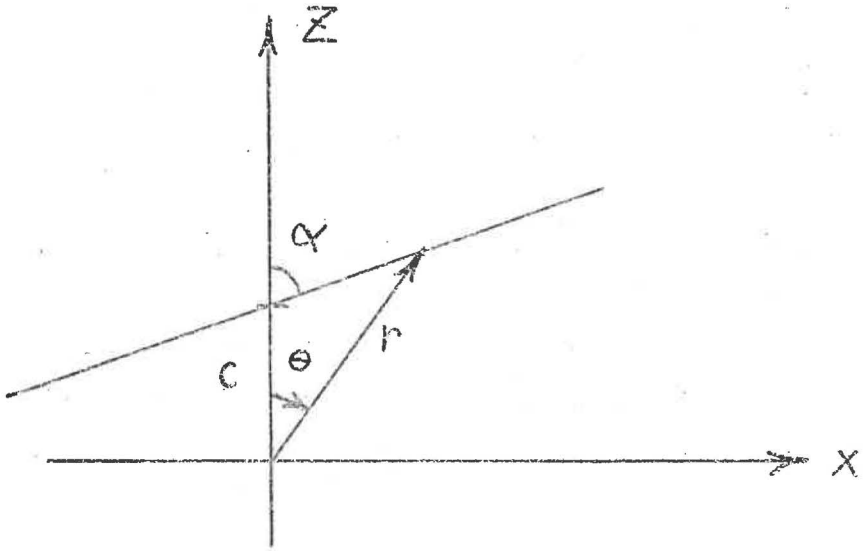
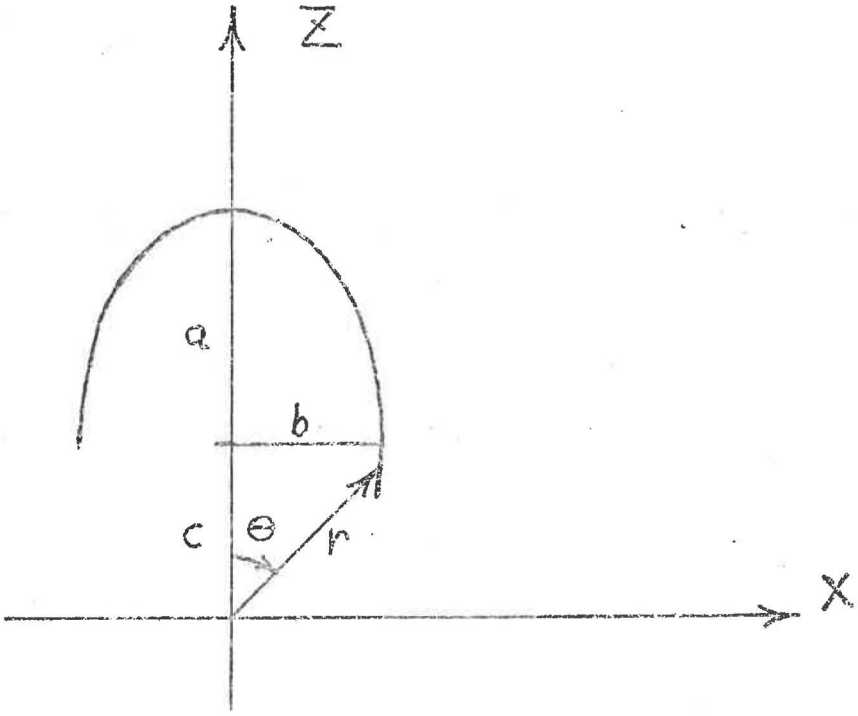


Fig. 8



Appendix II

```
C       T-00 PROGRAM
        DIMENSION NDLTH(4),CDH(4),AR(5,5),AI(5,5),BR(5,5),BI(5,5),PN(6),
       1X(10),Y(10),T(10,10),RE(10,10),IM(10,10),TN(6,10,10)
        DOUBLE PRECISION THETA,CDH,DLTH,SRMUL,STH,CTH,KR,DKR,PN,FCT,X,Y,BT
       1,CT,W1,W2,W3,W4,W5,AR,AI,BR,BI,RE,IM,KI,R,DR,ALPHA,A,B,C,D,PI,
       1THETA1,THETA2,THETA3,THETA4,E,F,G
        COMPLEX*16 T,TN
        COMMON/FAC/FCT(100),NRISK,NTRIX
        NAMELIST/HEJ/AR,AI,BR,BI,RE,IM,T
        CALL UNSTAE
        PI = 4.0D0*DATAN(1.0D0)
        NRISK = 57
        NTRIX = 39
        NEND = 78
        NR = 5
        NP = 0
        ISYM = 0
        K1 = 0.5D0
        A = 1.0D0
        C = -0.1D0
        NSECT = 1
        NDLTH(1) = 192
        THETA1 = PI
        CDH(1) = THETA1/DFLOAT(NDLTH(1))
        N = NRISK
        L = NTRIX
        M = NEND-2
        N1 = N-1
        DO 5 K = 1,100
      5 FCT(K) = 0.0D0
        FCT(1) = 1.0D0
        DO 6 K = 1,N1
      6 FCT(K+1) = FCT(K)*DFLOAT(K)
        FCT(N+1) = 0.1D0**L*FCT(N)*DFLOAT(N)
        DO 7 K = N,M
      7 FCT(K+2) = FCT(K+1)*DFLOAT(K+1)
        ND = 2*NR
        NR1 = NR+1
        DO 150 M1 = 1,NR1
        M = M1-1
        MD = M
        IF(M.EQ.0) MD = 1
        DO 25 J = 1,NR
        DO 25 I = 1,NR
        AR(I,J) = 0.0D0
        AI(I,J) = 0.0D0
        BR(I,J) = 0.0D0
        BI(I,J) = 0.0D0
     25 CONTINUE
        THETA = 0.0D0
        DO 90 ISECT = 1,NSECT
        NTHETA = NDLTH(ISECT)+1
        DLTH = CDH(ISECT)
        NFIX = 1
        DO 89 K = 1,NTHETA
        IF(K-1)31,31,32
     31 CONTINUE
        SRMUL = DLTH*7.0D0/22.5D0
        IF(ISECT.EQ.1) GO TO 89
```

```
      GO TO 41
   32 CONTINUE
      IF(K-NTHETA)34,33,33
   33 CONTINUE
      SRMUL = DLTH*7.0D0/22.5D0
      IF(ISECT-NSECT)40,89,89
   34 CONTINUE
      GO TO (35,36,37,38),NFIX
   35 CONTINUE
      SRMUL = DLTH*32.0D0/22.5D0
      NFIX = 2


      GO TO 39
   36 CONTINUE
      SRMUL = DLTH*12.0D0/22.5D0
      NFIX = 3
      GO TO 39
   37 CONTINUE
      SRMUL = DLTH*32.0D0/22.5D0
      NFIX = 4
      GO TO 39
   38 CONTINUE
      SRMUL = DLTH*14.0D0/22.5D0
      NFIX = 1
   39 CONTINUE
   40 CONTINUE
      THETA = THETA+DLTH
      STH = DSIN(THETA)
      CTH = DCOS(THETA)
      CALL LEG(THETA,M,NR1,PN)
   41 CONTINUE
      CALL TRCIR(STH,CTH,C,A,R,DR)
      KR = K1*R
      DKR = K1*DR
      IF(K.EQ.1) GO TO 52
      CALL BN(KR,NR1,X,Y)
      BT= DABS(KR*KR*(X(2)*Y(1)-X(1)*Y(2))-1.0D0)
      CT= DABS(KR*KR*(X(NR1)*Y(NR)-X(NR)*Y(NR1))-1.0D0)
      IF(BT-1.D-10)48,48,45
   45 CONTINUE
      PRINT 46
   46 FORMAT('0 ERROR IN BESSEL TEST')
      PRINT 47, BT,ISECT,K
   47 FORMAT(D25.15,2I5)
   48 CONTINUE
      IF(CT-1.D-10)52,52,49
   49 CONTINUE
      PRINT 50
   50 FORMAT('0 ERROR IN NEUMAN TEST')
      PRINT 51, CT,ISECT,K
   51 FORMAT(D25.15,2I5)
   52 CONTINUE
      DO 88 I = MD,NR
      DO 88 J = MD,NR
      II = I+1
      JJ = J+1
      IF(M.EQ.0) GO TO 74
      IF(ISYM.GT.0.AND.MOD(I+J,2).EQ.0) GO TO 74
```

```
      W1 = DFLOAT(I+J)*CTH*PN(I1)*PN(J1)-DFLOAT(I+M)*PN(I)*PN(J1)-DFLOAT
     1(J+M)*PN(I1)*PN(J)
      W2 = KR*KR*X(I1)
      W3 = W1*W2
      IF(ISYM.NE.2) GO TO 71
      IF(J-I)72,71,71
   71 CONTINUE
      BI(I,J) = BI(I,J)+SRMUL*W3*Y(J1)/STH
   72 CONTINUE
      IF(J-I)74,73,73
   73 CONTINUE
      BR(I,J) = BR(I,J)+SRMUL*W3*X(J1)/STH
   74 CONTINUE
      IF(ISYM.GT.0.AND.MOD(I+J,2).NE.0) GO TO 84
      W4 = KR*(KR*X(I)-DFLOAT(I)*X(IL))
      W5 = DFLOAT(I1)*DKR*STH*X(I1)
      W1 = PN(I1)*PN(J1)*(W4*(DFLOAT(I*J)*(CTH**2)+DFLOAT(M*M))+DFLOAT
     1(I*J)*W5*CTH)
      W2 = DFLOAT(I*(J+M))*PN(I1)*PN(J)*(CTH*W4+W5)
      W3 = DFLOAT(I+M)*PN(I)*W4*(DFLOAT(J+M)*PN(J)-DFLOAT(J)*CTH*PN(J1))
      IF(ISYM.NE.2) GO TO 81
      IF(J-I)82,81,81
   81 CONTINUE
      AI(I,J) = AI(I,J)+SRMUL*(W1-W2+W3)*Y(J1)/STH


   82 CONTINUE
      IF(J-I)84,83,83
   83 CONTINUE
      AR(I,J) = AR(I,J)+SRMUL*(W1-W2+W3)*X(J1)/STH
   84 CONTINUE
   88 CONTINUE
   89 CONTINUE
   90 CONTINUE
      DO 92 I = 2,NR
      JEND = I-1
      DO 91 J = 1,JEND
      BR(I,J) = BR(J,I)
      AR(I,J) = AR(J,I)
      IF(ISYM.NE.2) GO TO 91
      BI(I,J) = BI(J,I)
      AI(I,J) = AI(J,I)
   91 CONTINUE
   92 CONTINUE
      DO 97 I = MD,NR
      DO 97 J = MD,NR
      I1 = I+1
      J1 = J+1
      W1 = -DSQRT(DFLOAT((2*I+1)*(2*J+1))/DFLOAT(I*I1*J*J1))/2.0D0
      IF(M)96,96,95
   95 CONTINUE
      W1 = W1*DSQRT(FCT(I1-M)*FCT(J1-M)/(FCT(I1+M)*FCT(J1+M)))
   96 CONTINUE
      AR(I,J) = W1*AR(I,J)
      AI(I,J) = W1*AI(I,J)
      BR(I,J) = DFLOAT(M)*W1*BR(I,J)
      BI(I,J) = DFLOAT(M)*W1*BI(I,J)
   97 CONTINUE
      CALL PERFT(NP,M,NR,ND,AR,AI,BR,BI,T,RG,IM,X,Y)
```

```
      DO 130 I = 1,ND
      DO 130 J = 1,NO
      TN(M1,I,J) = T(I,J)
130   CONTINUE
      IF(M.NE.1) GO TO 150
      WRITE(6,HEJ)
150   CONTINUE
      WRITE(11) TN
      PRINT 161
161   FORMAT('0 TN NOW HOPEFULLY REPLACED INTO DATA SET')
      STOP
      DEBUG SUBCHK
      END
```

```
C       Q-D PROGRAM
        DIMENSION NDLTH(4),CDH(4),ARR(5,5),AOR(5,5),ARO(5,5),AOO(5,5),
       1BRR(5,5),BOR(5,5),BRO(5,5),BOO(5,5),CRR(5,5),COR(5,5),CRO(5,5),
       1COO(5,5),DRR(5,5),DOR(5,5),DRO(5,5),DOO(5,5),PN(7),X1(6),X2(6),
       1Y1(6),Y2(6),Q(10,10)
        DOUBLE PRECISION THETA,CDH,DLTH,SRMUL,STH,CTH,PN,DPNI,DPNJ,FCT,B1,
       1B2,C1,C2,PMY1,PMY2,GA,R,DR,NIND,KAPPA,ALPHA,A,B,C,D,PI,THETA1,
       1THETA2,THETA3,THETA4,E,F,G
        COMPLEX*16 CI,K1R,K2R,PMY,DK1R,DK2R,X1,X2,Y1,Y2,Z1I,Z2J,DKRX1I,
       1DKRY1I,DKRZ1I,DKRX2J,DKRY2J,DKRZ2J,W1,W2,W3,W4,W5,W6,W7,W8,W9,
       1ARR,AOR,ARO,AOO,BRR,BOR,BRO,BOO,CRR,COR,CRO,COO,DRR,DOR,DRO,DOO,
       1K1,K2,Q,FC
        COMMON/FAC/FCT(100),NRISK,NTRIX
        NAMELIST/HEJ/ARR,AOR,ARO,AOO,BRR,BOR,BRO,BOO,CRR,COR,CRO,COO,DRR,
       1DOR,DRO,DOO/NEJ/Q
        CALL UNSTAE
        PI = 4.0D0*DATAN(1.0D0)
        NRISK = 57
        NTRIX = 39
        NEND = 78
        NR = 5
        NP = 0
        LAY = 1
        PMY1 = 1.0D0
        PMY2 = 1.0D0
        K1 = (0.75D0,0.0D0)
        NIND = 4.0D0/5.0D0
        KAPPA = 0.0D0
        C = -0.1D0
        A = 1.0D0
        NSECT = 1
        NDLTH(1) = 192
        THETA1 = PI
        CDH(1) = THETA1/DFLOAT(NDLTH(1))
        N = NRISK
        L = NTRIX
        M = NEND-2
        N1 = N-1
        DO 5 K = 1,100
      5 FCT(K) = 0.0D0
        FCT(1) = 1.0D0
        DO 6 K = 1,N1
      6 FCT(K+1) = FCT(K)*DFLOAT(K)
        FCT(N+1) = 0.1D0**L*FCT(N)*DFLOAT(N)
        DO 7 K = N,M
      7 FCT(K+2) = FCT(K+1)*DFLOAT(K+1)
        FC = (1.0D0,0.0D0)
        IF(NP.EQ.1) FC = -FC
        CI = (0.0D0,1.0D0)
        PMY = DCMPLX(PMY1/PMY2,0.0D0)
        K2 = DCMPLX(NIND,NIND*KAPPA)*K1
        ND = 2*NR
        NR1 = NR+1
        NR2 = NR+2
        DO 150 M1 = 1,NR1
        M = M1-1
        MD = M
        IF(M.EQ.0) MD = 1
        DO 25 J = 1,NR
```

```
      DO 25 I = 1,NR
      ARR(I,J) = (0.0D0,0.0D0)
      AOR(I,J) = (0.0D0,0.0D0)
      ARO(I,J) = (0.0D0,0.0D0)
      AOO(I,J) = (0.0D0,0.0D0)
      BRR(I,J) = (0.0D0,0.0D0)
      BOR(I,J) = (0.0D0,0.0D0)
      BRO(I,J) = (0.0D0,0.0D0)
      BOO(I,J) = (0.0D0,0.0D0)
      CRR(I,J) = (0.0D0,0.0D0)
      COR(I,J) = (0.0D0,0.0D0)


      CRO(I,J) = (0.0D0,0.0D0)
      COO(I,J) = (0.0D0,0.0D0)
      DRR(I,J) = (0.0D0,0.0D0)
      DOR(I,J) = (0.0D0,0.0D0)
      DRO(I,J) = (0.0D0,0.0D0)
      DOO(I,J) = (0.0D0,0.0D0)
   25 CONTINUE
      THETA = 0.0D0
      DO 90 ISECT = 1,NSECT
      NTHETA = NDLTH(ISECT)+1
      DLTH = CDH(ISECT)
      NFIX = 1
      DO 89 K = 1,NTHETA
      IF(K-1)31,31,32
   31 CONTINUE
      SRMUL = DLTH*7.0D0/22.5D0
      IF(ISECT.EQ.1) GO TO 89
      GO TO 41
   32 CONTINUE
      IF(K-NTHETA)34,33,33
   33 CONTINUE
      SRMUL = DLTH*7.0D0/22.5D0
      IF(ISECT-NSECT)40,89,89
   34 CONTINUE
      GO TO (35,36,37,38),NFIX
   35 CONTINUE
      SRMUL = DLTH*32.0D0/22.5D0
      NFIX = 2
      GO TO 39
   36 CONTINUE
      SRMUL = DLTH*12.0D0/22.5D0
      NFIX = 3
      GO TO 39
   37 CONTINUE
      SRMUL = DLTH*32.0D0/22.5D0
      NFIX = 4
      GO TO 39
   38 CONTINUE
      SRMUL = DLTH*14.0D0/22.5D0
      NFIX = 1
   39 CONTINUE
   40 CONTINUE
      THETA = THETA+DLTH
      STH = DSIN(THETA)
      CTH = DCOS(THETA)
      CALL LEG(THETA,M,NR2,PN)
```

```
41 CONTINUE
   CALL TRCIR(STH,CTH,C,A,R,DR)
   K1R = K1*DCMPLX(R,0.0D0)
   DK1R = K1*DCMPLX(DR,0.0D0)
   K2R = K2*DCMPLX(R,0.0D0)
   IF(K.EQ.1) GO TO 52
   CALL CBN(K1R,NR1,X1,Y1)
   CALL CBN(K2R,NR1,X2,Y2)
   B1 = CDABS(K1R*K1R*(X1(2)*Y1(1)-X1(1)*Y1(2))-DCMPLX(1.0D0,0.0D0))
   B2 = CDABS(K2R*K2R*(X2(2)*Y2(1)-X2(1)*Y2(2))-DCMPLX(1.0D0,0.0D0))
   C1 = CDABS(K1R*K1R*(X1(NR1)*Y1(NR)-X1(NR)*Y1(NR1))-DCMPLX(1.0D0,0.
  1D0))
   C2 = CDABS(K2R*K2R*(X2(NR1)*Y2(NR)-X2(NR)*Y2(NR1))-DCMPLX(1.0D0,0.
  1D0))
   IF(B1.LT.1.D-10.AND.C1.LT.1.D-10) GO TO 47
   PRINT 45
45 FORMAT('0 ERROR IN BESSEL-NEUMAN-1 TEST')
   PRINT 46, B1,C1,ISECT,K
46 FORMAT(2D25.15,2I5)
47 CONTINUE
   IF(B2.LT.1.D-10.AND.C2.LT.1.D-10) GO TO 52
   PRINT 48
48 FORMAT('0 ERROR IN BESSEL-NEUMAN-2 TEST')
49 FORMAT(2D25.15,2I5)


   PRINT 49, B2,C2,ISECT,K
52 CONTINUE
   DO 88 I = MD,NR
   DO 88 J = MD,NR
   I1 = I+1
   J1 = J+1
   I2 = I+2
   J2 = J+2
   DPNI = -(DFLOAT(I1)*CTH*PN(I1)-DFLOAT(I1-M)*PN(I2))/STH
   DPNJ = -(DFLOAT(J1)*CTH*PN(J1)-DFLOAT(J1-M)*PN(J2))/STH
   Z1I = X1(I1)+CI*Y1(I1)
   DKRX1I = K1R*X1(I)-DCMPLX(DFLOAT(I),0.0D0)*X1(I1)
   DKRY1I = K1R*Y1(I)-DCMPLX(DFLOAT(I),0.0D0)*Y1(I1)
   DKRZ1I = DKRX1I+CI*DKRY1I
   Z2J = X2(J1)+CI*Y2(J1)
   DKRX2J = K2R*X2(J)-DCMPLX(DFLOAT(J),0.0D0)*X2(J1)
   DKRY2J = K2R*Y2(J)-DCMPLX(DFLOAT(J),0.0D0)*Y2(J1)
   DKRZ2J = DKRX2J+CI*DKRY2J
   W1 = DCMPLX(SRMUL*STH*(DPNI*DPNJ+DFLOAT(M*M)*PN(I1)*PN(J1)/STH**2)
  1,0.0D0)
   W2 = DCMPLX(SRMUL*STH*DFLOAT(I*I1)*PN(I1)*DPNJ,0.0D0)
   W3 = DCMPLX(SRMUL*STH*DFLOAT(J*J1)*PN(J1)*DPNI,0.0D0)
   ARR(I,J) = ARR(I,J)+K1R*DKRX1I*X2(J1)*W1+DK1R*X1(I1)*X2(J1)*W2
   AOR(I,J) = AOR(I,J)+K1R*DKRZ1I*X2(J1)*W1+DK1R*Z1I*X2(J1)*W2
   DRR(I,J) = DRR(I,J)+K1R*X1(I1)*DKRX2J*W1+DK1R*X1(I1)*X2(J1)*W3
   DOR(I,J) = DOR(I,J)+K1R*Z1I*DKRX2J*W1+DK1R*Z1I*X2(J1)*W3
   IF(LAY.EQ.0) GO TO 53
   ARO(I,J) = ARO(I,J)+K1R*DKRX1I*Z2J*W1+DK1R*X1(I1)*Z2J*W2
   AOO(I,J) = AOO(I,J)+K1R*DKRZ1I*Z2J*W1+DK1R*Z1I*Z2J*W2
   DRO(I,J) = DRO(I,J)+K1R*X1(I1)*DKRZ2J*W1+DK1R*X1(I1)*Z2J*W3
   DOO(I,J) = DOO(I,J)+K1R*Z1I*DKRZ2J*W1+DK1R*Z1I*Z2J*W3
53 CONTINUE
   IF(M.EQ.0) GO TO 88
```

```
      W4 = DCMPLX(SRMUL*(DPNI*PN(J1)+PN(I1)*DPNJ),0.0D0)
      W5 = DCMPLX(SRMUL*PN(I1)*PN(J1),0.0D0)
      BRR(I,J) = BRR(I,J)+K1R*K1R*X1(I1)*X2(J1)*W4
      BOR(I,J) = BOR(I,J)+K1R*K1R*Z1I*X2(J1)*W4
      W6 = DK1R*(X1(I1)*DKRX2J*DCMPLX(DFLOAT(I*(I+1)),0.0D0)+DCMPLX(
     1DFLOAT(J*(J+1)),0.0D0)*DKRX1I*X2(J1))/K1R
      CRR(I,J) = CRR(I,J)+DKRX1I*DKRX2J*W4+W5*W6
      W6 = DK1R*(Z1I*DKRX2J*DCMPLX(DFLOAT(I*(I+1)),0.0D0)+DCMPLX(
     1DFLOAT(J*(J+1)),0.0D0)*DKRZ1I*X2(J1))/K1R
      COR(I,J) = COR(I,J)+DKRZ1I*DKRX2J*W4+W5*W6
      IF(LAY.EQ.0) GO TO 88
      BRO(I,J) = BRO(I,J)+K1R*K1R*X1(I1)*Z2J*W4
      BOO(I,J) = BOO(I,J)+K1R*K1R*Z1I*Z2J*W4
      W6 = DK1R*(X1(I1)*DKRZ2J*DCMPLX(DFLOAT(I*(I+1)),0.0D0)+DCMPLX(
     1DFLOAT(J*(J+1)),0.0D0)*DKRX1I*Z2J)/K1R
      CRO(I,J) = CRO(I,J)+DKRX1I*DKRZ2J*W4+W5*W6
      W6 = DK1R*(Z1I*DKRZ2J*DCMPLX(DFLOAT(I*(I+1)),0.0D0)+DCMPLX(
     1DFLOAT(J*(J+1)),0.0D0)*DKRZ1I*Z2J)/K1R
      COO(I,J) = COO(I,J)+DKRZ1I*DKRZ2J*W4+W5*W6
   88 CONTINUE
   89 CONTINUE
   90 CONTINUE
      DO 98 I = MD,NR
      DO 98 J = MD,NR
      I1 = I+1
      J1 = J+1
      GA =  DSQRT(DFLOAT((2*I+1)*(2*J+1))/DFLOAT(I*I1*J*J1))/2.0D0
      IF(M)96,96,95
   95 CONTINUE
      GA = GA*DSQRT(FCT(I1-M)*FCT(J1-M)/(FCT(I1+M)*FCT(J1+M)))
   96 CONTINUE
      W1 = DCMPLX(GA,0.0D0)
      ARR(I,J) = W1*ARR(I,J)
      AOR(I,J) = W1*AOR(I,J)
      DRR(I,J) = W1*DRR(I,J)
      DOR(I,J) = W1*DOR(I,J)
      IF(LAY.EQ.0) GO TO 97


      ARO(I,J) = W1*ARO(I,J)
      AOO(I,J) = W1*AOO(I,J)
      DRO(I,J) = W1*DRO(I,J)
      DOO(I,J) = W1*DOO(I,J)
   97 CONTINUE
      IF(M.EQ.0) GO TO 98
      W1 = DCMPLX(DFLOAT(M),0.0D0)*W1
      BRR(I,J) = W1*BRR(I,J)
      BOR(I,J) = W1*BOR(I,J)
      CRR(I,J) = W1*CRR(I,J)
      COR(I,J) = W1*COR(I,J)
      IF(LAY.EQ.0) GO TO 98
      BRO(I,J) = W1*BRO(I,J)
      BOO(I,J) = W1*BOO(I,J)
      CRO(I,J) = W1*CRO(I,J)
      COO(I,J) = W1*COO(I,J)
   98 CONTINUE
      IF(M.GT.3) GO TO 99
      WRITE(6,HEJ)
   99 CONTINUE
```

```
      LLAY = 2+LAY*2
      DO 120 LL = 1,LLAY
      DO 110 I = 1,NR
      DO 110 J = 1,NR
      GO TO (101,102,103,104), LL
  101 CONTINUE
      Q(2*I-1,2*J-1) = -ARR(I,J)+FMY*DRR(I,J)
      Q(2*I-1,2*J) = FC*(K1*CRR(I,J)/K2+K2*FMY*BRR(I,J)/K1)
      Q(2*I,2*J-1) = -FC*(BRR(I,J)+FMY*CRR(I,J))
      Q(2*I,2*J) = (K1*DRR(I,J)/K2-K2*FMY*ARR(I,J)/K1)
      GO TO 110
  102 CONTINUE
      Q(2*I-1,2*J-1) = -AOR(I,J)+FMY*DOR(I,J)
      Q(2*I-1,2*J) = FC*(K1*COR(I,J)/K2+K2*FMY*BOR(I,J)/K1)
      Q(2*I,2*J-1) = -FC*(BOR(I,J)+FMY*COR(I,J))
      Q(2*I,2*J) = (K1*DOR(I,J)/K2-K2*FMY*AOR(I,J)/K1)
      GO TO 110
  103 CONTINUE
      Q(2*I-1,2*J-1) = -ARO(I,J)+FMY*DRO(I,J)
      Q(2*I-1,2*J) = FC*(K1*CRO(I,J)/K2+K2*FMY*BRO(I,J)/K1)
      Q(2*I,2*J-1) = -FC*(BRO(I,J)+FMY*CRO(I,J))
      Q(2*I,2*J) = (K1*DRO(I,J)/K2-K2*FMY*ARO(I,J)/K1)
      GO TO 110
  104 CONTINUE
      Q(2*I-1,2*J-1) = -AOO(I,J)+FMY*DOO(I,J)
      Q(2*I-1,2*J) = FC*(K1*COO(I,J)/K2+K2*FMY*BOO(I,J)/K1)
      Q(2*I,2*J-1) = -FC*(BOO(I,J)+FMY*COO(I,J))
      Q(2*I,2*J) = (K1*DOO(I,J)/K2-K2*FMY*AOO(I,J)/K1)
  110 CONTINUE
      WRITE(6,NEJ)
      GO TO (111,111,112,112),LL
  111 CONTINUE
      WRITE (21) Q
      END FILE 21
      GO TO 120
  112 CONTINUE
      WRITE(22) Q
      END FILE 22
  120 CONTINUE
      PRINT 777, M1
  777 FORMAT(14)
  150 CONTINUE
      PRINT 199
  199 FORMAT('0 Q-MATRICES NOW HOPEFULLY PLACED INTO TAPE')
      STOP
      DEBUG SUBCHK
      END
```

```
C       Q-T PROGRAM
        DIMENSION QRR(10,10),QOR(10,10),LL(10),MM(10),TN(6,10,10)
        COMPLEX*16 QRR,QOR,TN,P,D
        CALL UNSTAE
        NR = 5
        NR1 = NR+1
        ND = 2*NR
        DO 30 M1 = 1,NR1
        READ(21,END = 8) QRR
        GO TO 9
     8  READ(21,END=999) QRR
     9  CONTINUE
        READ(21,END=12) QOR
        GO TO 13
    12  READ(21,END=999) QOR
    13  CONTINUE
        IF(M1.LT.3) GO TO 16
        MD1 = 2*M1-4
        DO 15 I = 1,MD1
        QOR(I,I) = (1.0D0,0.0D0)
    15  CONTINUE
    16  CONTINUE
        CALL MCNV(QOR,ND,D,LL,MM)
        DO 21 I = 1,ND
        DO 21 J = 1,ND
        P = (0.0D0,0.0D0)
        DO 20 K = 1,ND
        P = P-QRR(I,K)*QOR(K,J)
    20  CONTINUE
        TN(M1,I,J) = P
    21  CONTINUE
    30  CONTINUE
        WRITE (11) TN
        PRINT 50
    50  FORMAT('0 TN NOW HOPEFULLY REPLACED INTO DATASET')
   999  CONTINUE
        STOP
        DEBUG SUBCHK
        END
```

```
C       Q,T-T PROGRAM
        DIMENSION Q1(10,10),Q2(10,10),Q3(10,10),T(10,10),LL(10),MM(10),TN
       1(6,10,10)
        COMPLEX*16 Q1,Q2,Q3,T,P,D,TN
        CALL UNSTAE
        NR = 5
        NR1 = NR+1
        ND = 2*NR
        DO 50 M1 = 1,NR1
        READ(21,END=8) Q1
        GO TO 9
    8   READ(21,END=999) Q1
    9   CONTINUE
        READ(21,END=12) Q2
        GO TO 13
   12   READ(21,END=999) Q2
   13   CONTINUE
        READ(22,END=16) Q3
        GO TO 17
   16   READ(22,END=999) Q3
   17   CONTINUE
        READ(23,END=20) T
        GO TO 21
   20   READ(23,END=999) T
   21   CONTINUE
        DO 25 J = 1,ND
        DO 25 I = 1,ND
        P = (0.0D0,0.0D0)
        DO 24 K = 1,ND
        P = P+Q3(I,K)*T(K,J)
   24   CONTINUE
        Q1(I,J) = Q1(I,J)+P
   25   CONTINUE
        READ(22,END=30) Q3
        GO TO 31
   30   READ(22,END=999) Q3
   31   CONTINUE
        DO 36 J = 1,ND
        DO 36 I = 1,ND
        P = (0.0D0,0.0D0)
        DO 35 K = 1,ND
        P = P+Q3(I,K)*T(K,J)
   35   CONTINUE
        Q2(I,J) = Q2(I,J)+P
   36   CONTINUE
        IF(M1.LT.3) GO TO 41
        MD1 = 2*M1-4
        DO 40 I = 1,MD1
        Q2(I,I) = (1.0D0,0.0D0)
   40   CONTINUE
   41   CONTINUE
        CALL MCNV(Q2,ND,D,LL,MM)
        DO 46 J = 1,ND
        DO 46 I = 1,ND
        P = (0.0D0,0.0D0)
        DO 45 K = 1,ND
        P = P+Q1(I,K)*Q2(K,J)
   45   CONTINUE
        T(I,J) = -P
```

```
      TN(M1,I,J) = -P
   46 CONTINUE
   50 CONTINUE
      WRITE(11) TN
      PRINT 70
   70 FORMAT('0 TN NOW HOPEFULLY REPLACED INTO DATASET')
  999 CONTINUE
      STOP
      DEBUG SUBCHK
      END
```

```
C       TN-T PROGRAM FOR Q,T-T PROGRAM
        DIMENSION TN(6,10,10),T(10,10)
        COMPLEX*16 TN,T
        CALL UNSTAE
        ND = 10
        NS = ND/2+1
        READ(11) TN
        DO 20 M = 1,NS
        DO 10 I = 1,ND
        DO 10 J = 1,ND
        T(I,J) = TN(M,I,J)
   10   CONTINUE
        WRITE(21) T
        END FILE 21
   20   CONTINUE
        STOP
        DEBUG SUBCHK
        END
```

```
C       T1,T2-T PROGRAM
        DIMENSION R1(10,10),R2(10,10),R3(10,10),R4(10,10),AT1(10,10),AT2(
       110,10),BT1(10,10),BT2(10,10),T(10,10),RET(10,10),RETT(10,10),TRAN(
       110,10),T1(10,10),T2(10,10),X1(11),X2(11),Y1(11),Y2(11),LL(10),MM(
       110),TN(6,10,10)
        COMPLEX*16 R1,R2,R3,R4,AT1,AT2,BT1,BT2,T,RET,RETT,TRAN,TN,T1,T2,
       1W1,W2,W3,W4,S,DD
        DOUBLE PRECISION X1,X2,Y1,Y2,SEP,FCT
        COMMON/FAC/FCT(100),NRISK,NTRIX
        NAMELIST/HEJ/T,RET,RETT,TRAN
        CALL UNSTAE
        NRISK = 57
        NTRIX = 39
        NEND = 78
        ND = 10
        NP = 0
        SEP = 1.5D0
        N = NRISK
        L = NTRIX
        M = NEND-2
        N1 = N-1
        DO 5 K = 1,100
      5 FCT(K) = 0.0D0
        FCT(1) = 1.0D0
        DO 6 K = 1,N1
      6 FCT(K+1) = FCT(K)*DFLOAT(K)
        FCT(N+1) = 0.1D0**L*FCT(N)*DFLOAT(N)
        DO 7 K = N,M
      7 FCT(K+2) = FCT(K+1)*DFLOAT(K+1)
        NS = ND/2+1
        DO 200 M1 = 1,NS
        M = M1-1
        READ(21,END=8) T1
        GO TO 9
      8 READ(21,END=999) T1
      9 CONTINUE
        READ(22,END=12) T2
        GO TO 13
     12 READ(22,END=999) T2
     13 CONTINUE
        IF(M-1)20,20,21
     20 CONTINUE
        MD = 1
        GO TO 22
     21 CONTINUE
        MD = 2*M-1
     22 CONTINUE
        DO 34 I = 1,ND
        DO 34 J = 1,ND
        T(I,J) = (0.0D0,0.0D0)
        TN(M1,I,J) = (0.0D0,0.0D0)
        IF(I-J) 30,31,30
     30 CONTINUE
        AT1(I,J) = (0.0D0,0.0D0)
        AT2(I,J) = (0.0D0,0.0D0)
        BT1(I,J) = (0.0D0,0.0D0)
        BT2(I,J) = (0.0D0,0.0D0)
        GO TO 34
     31 CONTINUE
```

```
      IF(M-2)30,32,32
   32 CONTINUE
      MD1 = MD-1
      IF(MD1-J)30,33,33
   33 CONTINUE
      AT1(I,J) = (1.0D0,0.0D0)
      AT2(I,J) = (1.0D0,0.0D0)
      BT1(I,J) = (1.0D0,0.0D0)
      BT2(I,J) = (1.0D0,0.0D0)
   34 CONTINUE
      CALL VR(SEP,NP,M,ND,R1,R2,R3,R4,X1,X2,Y1,Y2)


      DO 41 I = MD,ND
      DO 41 J = MD,ND
      W1 = (0.0D0,0.0D0)
      W2 = (0.0D0,0.0D0)
      W3 = (0.0D0,0.0D0)
      W4 = (0.0D0,0.0D0)
      DO 40 K = MD,ND
      W1 = W1+R2(I,K)*T2(K,J)
      W2 = W2+R3(I,K)*T1(K,J)
      W3 = W3+R4(K,I)*T1(K,J)
      W4 = W4+R4(I,K)*T2(K,J)
   40 CONTINUE
      T(I,J) = W1
      RET(I,J) = W2
      RETT(I,J) = W3
      TRAN(I,J) = W4
   41 CONTINUE
      DO 45 I = MD,ND
      DO 45 J = MD,ND
      W1 = (0.0D0,0.0D0)
      W2 = (0.0D0,0.0D0)
      W3 = (0.0D0,0.0D0)
      W4 = (0.0D0,0.0D0)
      DO 42 K = MD,ND
      W1 = W1+T(I,K)*RET(K,J)
      W2 = W2+RET(I,K)*T(K,J)
      W3 = W3+T(I,K)*R1(J,K)
      W4 = W4+RET(I,K)*R1(K,J)
   42 CONTINUE
      IF(I-J)44,43,44
   43 CONTINUE
      AT1(I,J) = (1.0D0,0.0D0)-W1
      AT2(I,J) = (1.0D0,0.0D0)-W2
      BT1(I,J) = (1.0D0,0.0D0)+W3
      BT2(I,J) = (1.0D0,0.0D0)+W4
      GO TO 45
   44 CONTINUE
      AT1(I,J) = -W1
      AT2(I,J) = -W2
      BT1(I,J) = W3
      BT2(I,J) = W4
   45 CONTINUE
      CALL MCNV(AT1,ND,DD,LL,MM)
      CALL MCNV(AT2,ND,DD,LL,MM)
      DO 47 I = MD,ND
      DO 47 J = MD,ND
```

```
      W1 = (0.0D0,0.0D0)
      W2 = (0.0D0,0.0D0)
      W3 = (0.0D0,0.0D0)
      W4 = (0.0D0,0.0D0)
      DO 46 K = MD,ND
      W1 = W1+RETT(I,K)*AT1(K,J)
      W2 = W2+TRAN(I,K)*AT2(K,J)
      W3 = W3+BT1(I,K)*R4(K,J)
      W4 = W4+BT2(I,K)*R4(J,K)
   46 CONTINUE
      R1(I,J) = W1
      R2(I,J) = W2
      R3(I,J) = W3
      RET(I,J) = W4
   47 CONTINUE
      DO 70 I = MD,ND
      DO 70 J = MD,ND
      W1 = (0.0D0,0.0D0)
      W2 = (0.0D0,0.0D0)
      DO 49 K = MD,ND
      W1 = W1+R1(I,K)*R3(K,J)
      W2 = W2+R2(I,K)*RET(K,J)
   49 CONTINUE
      S = W1+W2


      T(I,J) = S
      TN(M1,I,J) = S
   70 CONTINUE
      IF((M-1).NE.0) GO TO 200
      DO 81 I = 1,ND
      DO 81 J = 1,ND
      S = (0.0D0,0.0D0)
      DO 80 K = MD,ND
      S = S-DCONJG(T(K,I))*T(K,J)
   80 CONTINUE
      RET(I,J) = S
      RETT(I,J) = T(I,J)-S
      TRAN(I,J) = T(I,J)-T(J,I)
   81 CONTINUE
      WRITE(6,HEJ)
  200 CONTINUE
      WRITE(11) TN
      PRINT 98
   98 FORMAT('0 TN NOW HOPEFULLY REPLACED INTO DATASET')
  999 CONTINUE
      STOP
      DEBUG SUBCHK
      END
```

```
C       TN-T PROGRAM NR 1 FOR T1,T2-T PROGRAM
        DIMENSION TN(6,10,10),T(10,10)
        COMPLEX*16 TN,T
        CALL UNSTAE
        ND = 10
        NS = ND/2+1
        READ(11) TN
        DO 20 M = 1,NS
        DO 10 I = 1,ND
        DO 10 J = 1,ND
        T(I,J) = TN(M,I,J)
   10   CONTINUE
        WRITE(21) T
        END FILE 21
   20   CONTINUE
        STOP
        DEBUG SUBCHK
        END
```

```
C       TN-T PROGRAM NR 2 FOR T1,T2-T PROGRAM
        DIMENSION TN(6,10,10),T(10,10)
        COMPLEX*16 TN,T
        CALL UNSTAE
        ND = 10
        NS = ND/2+1
        READ(11) TN
        DO 20 M = 1,NS
        DO 10 I = 1,ND
        DO 10 J = 1,ND
        T(I,J) = TN(M,I,J)
   10   CONTINUE
        WRITE(21) T
        END FILE 21
   20   CONTINUE
        STOP
        DEBUG SUBCHK
        END
```

```
C       T-T PROGRAM
        DIMENSION TN(6,10,10),T(10,10),R1(10,10),R2(10,10),R3(10,10),R4
       1(10,10),X1(11),X2(11),Y1(11),Y2(11)
        DOUBLE PRECISION TR,X1,X2,Y1,Y2,FCT
        COMPLEX*16 TN,T,R1,R2,R3,R4,P
        COMMON/FAC/FCT(100),NRISK,NTRIX
        NAMELISY/HEJ/T
        NRISK = 57
        NTRIX = 39
        NEND = 78
        ND = 10
        NP = 0
        TR = 0.05D0
        N = NRISK
        L = NTRIX
        M = NEND-2
        N1 = N-1
        DO 5 K = 1,100
      5 FCT(K) = 0.0D0
        FCT(1) = 1.0D0
        DO 6 K = 1,N1
      6 FCT(K+1) = FCT(K)*DFLOAT(K)
        FCT(N+1) = 0.1D0**L*FCT(N)*DFLOAT(N)
        DO 7 K = N,M
      7 FCT(K+2) = FCT(K+1)*DFLOAT(K+1)
        READ(11) TN
        NS = ND/2+1
        DO 20 M1 = 1,NS
        M = M1-1
        CALL VR(TR,NP,M,ND,R1,R2,R3,R4,X1,X2,Y1,Y2)
        DO 11 J = 1,ND
        DO 11 I = 1,ND
        P = (0.0D0,0.0D0)
        DO 10 K = 1,ND
        P = P+TN(M1,I,K)*R1(J,K)
     10 CONTINUE
        R4(I,J) = P
     11 CONTINUE
        DO 16 J = 1,ND
        DO 16 I = 1,ND
        P = (0.0D0,0.0D0)
        DO 15 K = 1,ND
        P = P+R1(I,K)*R4(K,J)
     15 CONTINUE
        T(I,J) = P
        TN(M1,I,J) = P
     16 CONTINUE
        WRITE(6,HEJ)
     20 CONTINUE
        WRITE(12) TN
        PRINT 70
     70 FORMAT('0 TN NOW HOPEFULLY REPLACED INTO DATASET')
        STOP
        DEBUG SUBCHK
        END
```

```
C       PSIS PROGRAM
        DIMENSION TN(6,10,10),AP(10),PN(7),FEXP(10),PSIR(10),PSITH(10),
       1PSIFH(10),X(7),Y(7)
        DOUBLE PRECISION BHETA,PN,DIST,THETA,FHI,X,Y,FCT,DIFSC,RAD,AMPL,
       1BHETAD,FHID,THETAD,PI,KV,AMPL2,A,B,C,D,E
        COMPLEX*16 TN,AP,PSIR,PSITH,PSIFH,FEXP,Q1,Q2,P1,P2,R1,R2,FC
        COMMON/FAC/FCT(100),NRISK,NTRIX
        NAMELIST/HEJ/Q1,Q2,P1,P2,R1,R2
        CALL UNSTAE
        PI = 4.0D0*DATAN(1.0D0)
        NRISK = 57
        NTRIX = 39
        NEND = 78
        NP = 1
        NPCHAN = 1
        ND = 10
        NB = 1
        KV = 2.0D0*PI/15.53333D0
        BHETA = PI/2.0D0
        BHETAD = BHETA*180.0D0/PI
        DIST = 1.0D0
        THETA = PI/2.0D0
        THETAD = THETA*180.0D0/PI
        FHI = 0.0D0
        FHID = FHI*180.0D0/PI
        N = NRISK
        L = NTRIX
        M = NEND-2
        N1 = N-1
        DO 5 K = 1,100
      5 FCT(K) = 0.0D0
        FCT(1) = 1.0D0
        DO 6 K = 1,N1
      6 FCT(K+1) = FCT(K)*DFLOAT(K)
        FCT(N+1) = 0.1D0**L*FCT(N)*DFLOAT(N)
        DO 7 K = N,M
      7 FCT(K+2) = FCT(K+1)*DFLOAT(K+1)
        NPC = NPCHAN+1
        GO TO (11,12), NPC
     11 CONTINUE
        FC = (1.0D0,0.0D0)
        GO TO 13
     12 CONTINUE
        FC = (-1.0D0,0.0D0)
     13 CONTINUE
        READ (11) TN
        NS = ND/2+1
        P1 = (0.0D0,0.0D0)
        P2 = (0.0D0,0.0D0)
        DO 23 M1 = 1,NS
        M = M1-1
        MD = 2*M-1
        IF((M-2).LT.0) MD = 1
        CALL VKOEF(BHETA,NP,M,ND,AP,PN)
        CALL VPSI(DIST,THETA,FHI,NP,NB,M,ND,PSIR,PSITH,PSIFH,PN,X,Y)
        DO 21 I = MD,ND
        Q1 = (0.0D0,0.0D0)
        DO 20 J = MD,ND
        Q1 = Q1+TN(M1,I,J)*AP(J)*FC**(I+J)
```

```
 20 CONTINUE
    FEXP(I) = Q1
 21 CONTINUE
    Q1 = (0.0D0,0.0D0)
    Q2 = (0.0D0,0.0D0)
    DO 22 I = MD,ND
    Q1 = Q1+PSITH(I)*FEXP(I)
    Q2 = Q2+PSIFH(I)*FEXP(I)
 22 CONTINUE
    P1 = P1+Q1
    P2 = P2+Q2


    R1 = P1/DCMPLX(KV,0.0D0)
    R2 = P2/DCMPLX(KV,0.0D0)
    WRITE(6,HEJ)
    AMPL2= CDABS(P1)**2+CDABS(P2)**2
    AMPL = DSQRT(AMPL2)
    A = THETAD
    B = FHID
    C = AMPL
    D = AMPL2
    E = AMPL/KV
    PRINT 40, A,B,C,D,E
 40 FORMAT(5D25.15)
 23 CONTINUE
100 CONTINUE
300 CONTINUE
999 CONTINUE
    STOP
    DEBUG SUBCHK
    END
```

```
C       T TEST PROGRAM
        DIMENSION TN(6,10,10),T(10,10),RET(10,10),RETT(10,10),TRAN(10,10)
        COMPLEX*16 TN,T,RET,RETT,TRAN,S
        NAMELIST/HEJ/T,RET,RETT,TRAN
        CALL UNSTAE
        ND = 10
        READ (11) TN
        NS = ND/2+1
        DO 20 M = 1,NS
        DO 10 I = 1,ND
        DO 10 J = 1,ND
        T(I,J) = TN(M,I,J)
     10 CONTINUE
        DO 12 I = 1,ND
        DO 12 J = 1,ND
        S = (0.0D0,0.0D0)
        DO 11 K = 1,ND
        S = S-DCONJG(T(K,I))*T(K,J)
     11 CONTINUE
        RET(I,J) = S
        RETT(I,J) = T(I,J)-S
        TRAN(I,J) = T(I,J)-T(J,I)
     12 CONTINUE
        WRITE(6,HEJ)
        IF(M.GT.3) GO TO 99
     20 CONTINUE
     99 CONTINUE
        STOP
        DEBUG SUBCHK
        END
```

```
      SUBROUTINE BESSEL(NORDER,ARGMNT,ANSWR,IERROR)
      DOUBLE PRECISION ARGMNT,ANSWR,X,SUM,APR,TOPR,CI,CNI,ACR,PROD,FACT
      IERROR = 0
      N = NORDER
      X = ARGMNT
      SUM = 1.0D0
      APR = 1.0D0
      TOPR = -0.5D0*X*X
      CI = 1.0D0
      CNI = DFLOAT(2*N+3)
      DO 60 I = 1,100
      ACR = TOPR*APR/(CI*CNI)
      SUM = SUM+ACR
      IF(DABS(ACR/SUM)-1.0D-20)100,100,40
40    APR = ACR
      CI = CI+1.0D0
      CNI = CNI+2.0D0
60    CONTINUE
      IERROR = 1
      PRINT 10
   10 FORMAT(24H0 ERROR IN SUM OF BESSEL)
      GO TO 200
100   PROD = DFLOAT(2*N+1)
      FACT = 1.0D0
      IF(N)160,160,120
120   DO 140 IFCT = 1,N
      FACT = FACT*X/PROD
      PROD = PROD-2.0D0
140   CONTINUE
160   ANSWR = FACT*SUM
200   RETURN
      END
```

```
      SUBROUTINE BN(PCKR,NRINK,BSSLSP,CNEUMN)
      DIMENSION BSSLSP(NRINK),CNEUMN(NRINK)
      DOUBLE PRECISION PCKR,ANSWR,ANSA,ANSB,ANSC,CONN,CMULN,SNSA,SNSB,
     1SNSC,BSSLSP,CNEUMN
      NRANKI = NRINK
      NRANK = NRANKI-1
      NVAL = NRANK-1
      DO 40 I = 1,4
      CALL BESSEL(NVAL,PCKR,ANSWR,IERROR)
      IF(IERROR)20,20,32
   20 ANSA = ANSWR
      NVAL = NVAL+1
      CALL BESSEL(NVAL,PCKR,ANSWR,IERROR)
      IF(IERROR)24,24,28
   24 ANSB = ANSWR
      GO TO 60
   28 NVAL = NVAL-1
   32 NVAL = NVAL+NRANK
   40 CONTINUE
      STOP
   60 IF(NVAL-NRANK)100,100,64
   64 IEND = NVAL-NRANK
      CONN = DFLOAT(2*(NVAL-1)+1)
      DO 72 IP = 1,IEND
      ANSC = CONN*ANSA/PCKR-ANSB
      CONN = CONN-2.0D0
      ANSB = ANSA
      ANSA = ANSC
   72 CONTINUE
  100 BSSLSP(NRANKI) = ANSB
      BSSLSP(NRANKI-1) = ANSA
      CONN = DFLOAT(NRANK+NRANK-1)
      IE = NRANKI-2
      JE = IE
      DO 120 JB = 1,JE
      ANSC = CONN*ANSA/PCKR-ANSB
      BSSLSP(IE) = ANSC
      ANSB = ANSA
      ANSA = ANSC
      IE = IE-1
      CONN = CONN-2.0D0
  120 CONTINUE
      CMULN = 3.0D0
      SNSA = -DCOS(PCKR)/PCKR
      SNSB = -DCOS(PCKR)/(PCKR*PCKR)-DSIN(PCKR)/PCKR
      CNEUMN(1) = SNSA
      CNEUMN(2) = SNSB
      DO 280 I = 3,NRANKI
      SNSC = CMULN*SNSB/PCKR-SNSA
      CNEUMN(I) = SNSC
      SNSA = SNSB
      SNSB = SNSC
      CMULN = CMULN+2.0D0
  280 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CBESS(NORDER,ARGMNT,ANSWR,IERROR)
      COMPLEX*16 ARGMNT,ANSWR,X,SUM,APR,TOPR,CI,CNI,ACR,PROD,FACT
      IERROR = 0
      N = NORDER
      X = ARGMNT
      SUM = (1.0D0,0.0D0)
      APR = (1.0D0,0.0D0)
      TOPR = -(0.5D0,0.0D0)*X*X
      CI = (1.0D0,0.0D0)
      CNI = DCMPLX(DFLOAT(2*N+3),0.0D0)
      DO 60 I = 1,100
      ACR = TOPR*APR/(CI*CNI)
      SUM = SUM+ACR
      IF(CDABS(ACR/SUM)-1.0D-20)100,100,40
   40 APR = ACR
      CI = CI+(1.0D0,0.0D0)
      CNI = CNI+(2.0D0,0.0D0)
   60 CONTINUE
      IERROR = 1
      PRINT 10
   10 FORMAT('0 ERROR IN SUM OF CBESS')
      GO TO 200
  100 PROD = DCMPLX(DFLOAT(2*N+1),0.0D0)
      FACT = (1.0D0,0.0D0)
      IF(N)160,160,120
  120 DO 140 IFCT = 1,N
      FACT = FACT*X/PROD
      PROD = PROD-(2.0D0,0.0D0)
  140 CONTINUE
  160 ANSWR = FACT*SUM
  200 RETURN
      END
```

```
      SUBROUTINE CBN(PCKR,NRINK,BSSLSP,CNEUMN)
      DIMENSION BSSLSP(NRINK),CNEUMN(NRINK)
      COMPLEX*16 PCKR,ANSWR,ANSA,ANSB,ANSC,CONN,CMULN,SNSA,SNSB,SNSC,
     1BSSLSP,CNEUMN
      NRANKI = NRINK
      NRANK = NRANKI-1
      NVAL = NRANK-1
      DO 40 I = 1,4
      CALL CBESS(NVAL,PCKR,ANSWR,IERROR)
      IF(IERROR)20,20,32
   20 ANSA = ANSWR
      NVAL = NVAL+1
      CALL CBESS(NVAL,PCKR,ANSWR,IERROR)
      IF(IERROR)24,24,28
   24 ANSB = ANSWR
      GO TO 60
   28 NVAL = NVAL-1
   32 NVAL = NVAL+NRANK
   40 CONTINUE
      STOP
   60 IF(NVAL-NRANK)100,100,64
   64 IEND = NVAL-NRANK
      CONN = DCMPLX(DFLOAT(2*(NVAL-1)+1),0.0D0)
      DO 72 IP = 1,IEND
      ANSC = CONN*ANSA/PCKR-ANSB
      CONN = CONN-(2.0D0,0.0D0)
      ANSB = ANSA
      ANSA = ANSC
   72 CONTINUE
  100 BSSLSP(NRANKI) = ANSB
      BSSLSP(NRANKI-1) = ANSA
      CONN = DCMPLX(DFLOAT(NRANK*NRANK-1),0.0D0)
      IE = NRANKI-2
      JE = IE
      DO 120 JB = 1,JE
      ANSC = CONN*ANSA/PCKR-ANSB
      BSSLSP(IE) = ANSC
      ANSB = ANSA
      ANSA = ANSC
      IE = IE-1
      CONN = CONN-(2.0D0,0.0D0)
  120 CONTINUE
      CMULN = (3.0D0,0.0D0)
      SNSA = -CDCOS(PCKR)/PCKR
      SNSB = -CDCOS(PCKR)/(PCKR*PCKR)-CDSIN(PCKR)/PCKR
      CNEUMN(1) = SNSA
      CNEUMN(2) = SNSB
      DO 280 I = 3,NRANKI
      SNSC = CMULN*SNSB/PCKR-SNSA
      CNEUMN(I) = SNSC
      SNSA = SNSB
      SNSB = SNSC
      CMULN = CMULN+(2.0D0,0.0D0)
  280 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LEG(THETA,M,NRJNK,PNMLLG)
      DIMENSION PNMLLG(NRJNK)
      DOUBLE PRECISION THETA,PLA,PLB,PLC,DTWM,CNM,CNMM,CNMUL,PRODM,
     1QUANM,PNMLLG
      NRANKI = NRJNK
      KMV = M
      DTWM = DFLOAT(2*M+1)
      IF(THETA)16,4,16
    4 IF(KMV)12,12,6
    6 DO 8 ILG = 1,NRANKI
      PNMLLG(ILG) = 0.0D0
    8 CONTINUE
      GO TO 88
   12 PNMLLG(1) = 1.0D0
      PLA = 1.0D0
      GO TO 48
   16 IF(KMV)20,20,40
   20 PLA = 1.0D0
      PLB = DCOS(THETA)*PLA
      PNMLLG(1) = PLA
      PNMLLG(2) = PLB
      IBEG = 3
      GO TO 60
   40 DO 44 ILG = 1,KMV
      PNMLLG(ILG) = 0.0D0
   44 CONTINUE
      PRODM = 1.0D0
      QUANM = DFLOAT(KMV)
      DO 52 IFCT = 1,KMV
      QUANM = QUANM+1.0D0
      PRODM = QUANM*PRODM/2.0D0
   52 CONTINUE
      PLA = PRODM*DSIN(THETA)**KMV
      PNMLLG(KMV+1) = PLA
   48 PLB = DTWM*DCOS(THETA)*PLA
      PNMLLG(KMV+2) = PLB
      IBEG = KMV+3
   60 CNMUL = DFLOAT(IBEG+IBEG-3)
      CNM = 2.0D0
      CNMM = DTWM
      DO 80 ILGR = IBEG,NRANKI
      PLC = (CNMUL*DCOS(THETA)*PLB-CNMM*PLA)/CNM
      PNMLLG(ILGR) = PLC
      PLA = PLB
      PLB = PLC
      CNMUL = CNMUL+2.0D0
      CNM = CNM+1.0D0
      CNMM = CNMM+1.0D0
   80 CONTINUE
   88 RETURN
      END
```

```
      FUNCTION TRIXJ(J1,J2,J,M,FCT,N,L)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION FCT(1)
      INTEGER Z,ZMIN,ZMAX
      Y=1.0D0
      CC=0.0D0
      JSUM=J1+J2+J
      JM1=J1-IABS(M)
      JM2=J2-IABS(N)
      IF((MOD(JSUM,2).NE.0).OR.(MOD(JM1,2).NE.0).OR.(MOD(JM2,2).NE.0)
     1.OR.(JM1.LT.0).OR.(JM2.LT.0))
     2GO TO 3
      IF((J.GT.J1+J2).OR.(J.LT.IABS(J1-J2))) GO TO 1
      ZMIN=0
      IF(J-J2+M.LT.0) ZMIN=-J+J2-M
      IF(J-J1+M+ZMIN.LT.0) ZMIN=-J+J1-M
      ZMAX=J1+J2-J
      IF(J2-M-ZMAX.LT.0) ZMAX=J2-M
      IF(J1-M-ZMAX.LT.0) ZMAX=J1-M
      JA=(J1+M)/2+1
      JB=JA-M
      JC=(J2-M)/2+1
      JD=JC+M
      JE=J/2+1
      JF=(J1+J2-J)/2+1
      JG=JA+JB-JF
      JH=JC+JD-JF
      JJ=2*JE+JF-1
      IF(JJ.GT.N) Y=0.1D0**L
      IF(FCT(JJ))7,5,7
    7 CONTINUE
      IA=ZMIN/2
      IB=JF-IA+1
      IC=JB-IA+1
      ID=JC-IA+1
      IE=JA-JF+IA
      IF=JD-JF+IA
      FASE=1.0D0
      IF(MOD(IA,2).EQ.0) FASE=-FASE
      Z=ZMIN
    2 IA=IA+1
      IB=IB-1
      IC=IC-1
      ID=ID-1
      IE=IE+1
      IF=IF+1
      FASE=-FASE
      CC=CC+FASE/(FCT(IA)*FCT(IB)*FCT(IC)*FCT(ID)*FCT(IE)*FCT(IF))
      Z=Z+2
      IF(Z.LE.ZMAX) GO TO 2
      FASE=-DSIGN(1.0D0,CC)
      IF(MOD(J1-J2,4).EQ.0) FASE=-FASE
      CC=FASE*DSQRT(Y*FCT(JB)*FCT(JC)*FCT(JE)*CC*FCT(JA)*FCT(JD)*FCT(JE)
     1*CC*FCT(JF)*FCT(JG)*FCT(JH)/FCT(JJ))
    1 TRIXJ=CC
      RETURN
    3 PRINT 4
    4 FORMAT('0 ERROR IN ARGUMENT OF TRIXJ')
      CALL EXIT
    5 PRINT 6
    6 FORMAT('0 ERROR FACTORIALS EXCEEDED IN TRIXJ')
      CALL EXIT
      END
```

```
      SUBROUTINE VPSI(RAD,THETA,FHI,NP,NB,M,ND,PSIR,PSITH,PSIFH,PN,U,V)
      DIMENSION PSIR(1),PSITH(1),PSIFH(1),PN(1),U(1),V(1)
      DOUBLE PRECISION RAD,R,THETA,T,FHI,F,PN,U,V,FCT,FR,FR1,FR2,B,C
      COMPLEX*16 FC,FC1,FC2,PSIR,PSITH,PSIFH
      COMMON/FAC/FCT(100),NRISK,NTRIX
      R = RAD
      T = THETA
      F = FHI
      NP1 = NP+1
      K = M
      N = ND
      L = N/2
      L1 = L+1
      L2 = L+2
      FC = (0.0D0,1.0D0)
      IF(NB.EQ.1) GO TO 5
      CALL BN(R,L1,U,V)
      B = DABS(R*R*(U(2)*V(1)-U(1)*V(2))-1.0D0)
      C = DABS(R*R*(U(L1)*V(L)-U(L)*V(L1))-1.0D0)
      PRINT 3
    3 FORMAT('0 BESSEL- NEUMAN- TEST FOR VPSI')
      PRINT 4, B,C
    4 FORMAT(2D25.15)
    5 CONTINUE
      CALL LEG(T,K,L2,PN)
      DO 42 I = L,L
      I1 = I+1
      I2 = I+2
      IF(I-K)6,7,7
    6 FR = 0.0D0
      GO TO 10
    7 IF(K)8,8,9
    8 FR = DSQRT(DFLOAT(2*I+1)/(DFLOAT(I*I1)*16.0D0*DATAN(1.0D0)))
      GO TO 10
    9 FR = DSQRT(DFLOAT(2*I+1)*FCT(I1-K)/(DFLOAT(I*I1)*FCT(I1+K)*8.0D0*
     1DATAN(1.0D0)))
   10 CONTINUE
      IF(T)16,16,19
   16 CONTINUE
      IF(K-1)18,17,18
   17 CONTINUE
      FR1 = DSQRT(DFLOAT(2*I+1)/(32.0D0*DATAN(1.0D0)))
      FR2 = DSQRT(DFLOAT(2*I+1)/(32.0D0*DATAN(1.0D0)))
      GO TO 20
   18 CONTINUE
      FR1 = 0.0D0
      FR2 = 0.0D0
      GO TO 20
   19 CONTINUE
      FR1 = FR*PN(I1)*DFLOAT(K)/DSIN(T)
      FR2 = -FR*(DFLOAT(I1)*DCOS(T)*PN(I1)-DFLOAT(I1-K)*PN(I2))/DSIN(T)
   20 CONTINUE
      GO TO (30,31,32),NB
   30 CONTINUE
      FC1 = (-FC)**I1
      FC2 = (-FC)**I
      GO TO 33
   31 CONTINUE
      FC1 = DCMPLX(U(I1),0.0D0)
```

```
      FC2 = DCMPLX(U(I)-DFLOAT(I)*U(I1)/R,0.0D0)
      GO TO 33
   32 CONTINUE
      FC1 = DCMPLX(U(I1),V(I1))
      FC2 = DCMPLX(U(I)-DFLOAT(I)*U(I1)/R,V(I)-DFLOAT(I)*V(I1)/R)
   33 CONTINUE
      PSIR(2*I-1) = (0.0D0,0.0D0)
      IF(NB-1)34,34,35
   34 CONTINUE
      PSIR(2*I) = (0.0D0,0.0D0)
   35 CONTINUE


      GO TO (36,39),NP1
   36 CONTINUE
      IF(NB-1)38,38,37
   37 CONTINUE
      PSIR(2*I) = FC1*DCMPLX(DFLOAT(I*I1)*FR*PN(I1)*DSIN(DFLOAT(K)*F)/R,
     10.0D0)
   38 CONTINUE
      PSITH(2*I-1) = -FC1*DCMPLX(FR1*DSIN(DFLOAT(K)*F),0.0D0)
      PSITH(2*I) = FC2*DCMPLX(FR2*DSIN(DFLOAT(K)*F),0.0D0)
      PSIFH(2*I-1) = -FC1*DCMPLX(FR2*DCOS(DFLOAT(K)*F),0.0D0)
      PSIFH(2*I) = FC2*DCMPLX(FR1*DCOS(DFLOAT(K)*F),0.0D0)
      GO TO 42
   39 CONTINUE
      IF(NB-1)41,41,40
   40 CONTINUE
      PSIR(2*I) = FC1*DCMPLX(DFLOAT(I*I1)*FR*PN(I1)*DCOS(DFLOAT(K)*F)/R,
     10.0D0)
   41 CONTINUE
      PSITH(2*I-1) = FC1*DCMPLX(FR1*DCOS(DFLOAT(K)*F),0.0D0)
      PSITH(2*I) = FC2*DCMPLX(FR2*DCOS(DFLOAT(K)*F),0.0D0)
      PSIFH(2*I-1) = -FC1*DCMPLX(FR2*DSIN(DFLOAT(K)*F),0.0D0)
      PSIFH(2*I) = -FC2*DCMPLX(FR1*DSIN(DFLOAT(K)*F),0.0D0)
   42 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VKOEF(BHETA,NP,M,ND,AP,PN)
      DIMENSION PN(1),AP(1)
      DOUBLE PRECISION BHETA,U,DI,PN,FR,FCT
      COMPLEX*16 FC1,FC2,AP
      COMMON/FAC/FCT(100),NRISK,NTRIX
      N = NP
      N1 = N+1
      K = M
      L = ND/2
      L2 = L+2
      U = BHETA
      DI = DATAN(1.0D0)
      FC1 = (0.0D0,1.0D0)
      CALL LEG(U,K,L2,PN)
      IF(U)10,10,20
   10 DO 100 I = 1,L
      IF(K-1)11,12,11
   11 AP(2*I-1) = (0.0D0,0.0D0)
      AP(2*I) = (0.0D0,0.0D0)
      GO TO 100
   12 FR = DSQRT(8.0D0*DI*DFLOAT(2*I+1))
      FC2 = DCMPLX(FR,0.0D0)
      GO TO (13,14),N1
   13 AP(2*I-1) = -(FC1**I)*FC2
      AP(2*I) = -(FC1**(I+1))*FC2
      GO TO 100
   14 AP(2*I-1) = (FC1**I)*FC2
      AP(2*I) = (FC1**(I+3))*FC2
  100 CONTINUE
      GO TO 300
   20 DO 200 I = 1,L
      I1 = I+1
      I2 = I+2
      IF(K)21,21,30
   21 FR = 4.0D0*DSQRT((DI*DFLOAT((2*I+1)*(I+1)))/DFLOAT(I))*(DCOS(U)*
     1PN(I1)-PN(I2))/DSIN(U)
      FC2 = DCMPLX(FR,0.0D0)
      GO TO (22,23),N1
   22 AP(2*I-1) = (FC1**I)*FC2
      AP(2*I) = (0.0D0,0.0D0)
      GO TO 200
   23 AP(2*I-1) = (0.0D0,0.0D0)
      AP(2*I) = (FC1**(I+1))*FC2
      GO TO 200
   30 IF(I-K)34,31,31
   31 FR = 4.0D0*DSQRT((2.0D0*DI*DFLOAT(2*I+1)*FCT(I-K+1))/(DFLOAT(I*(I
     1+1))*FCT(I+K+1)))
      FC2 = DCMPLX((FR*(DFLOAT(I+1)*DCOS(U)*PN(I1)-DFLOAT(I-K+1)*PN(I2)
     1))/DSIN(U),0.0D0)
      FC3 = DCMPLX((DFLOAT(K)*FR*PN(I1))/DSIN(U),0.0D0)
      GO TO (32,33),N1
   32 AP(2*I-1) = (FC1**I)*FC2
      AP(2*I) = -(FC1**(I+1))*FC3
      GO TO 200
   33 AP(2*I-1) = (FC1**I)*FC3
      AP(2*I) = (FC1**(I+1))*FC2
      GO TO 200
   34 AP(2*I-1) = (0.0D0,0.0D0)
      AP(2*I) = (0.0D0,0.0D0)
  200 CONTINUE
  300 RETURN
      END
```

```
      SUBROUTINE VR(AR,NP,M,ND,R1,R2,R3,R4,X1,X2,Y1,Y2)
      DIMENSION R1(ND,1),R2(ND,1),R3(ND,1),R4(ND,1),X1(1),X2(1),Y1(1),Y2
     1(1)
      DOUBLE PRECISION   AR,U,V,F3J1,F3J2,F3J3,FACT1,FACT2,FACT3,FACT4,
     1FACT5,FACT6,FACT7,FACT8,FACT9,FACT10,ACR1,ACR2,ACR3,ACR4,X1,X2,
     1Y1,Y2,B1,B2,CN1,CN2,FCT
      COMPLEX*16 R1,R2,R3,R4,W1,W2,W3,W4,W5,W6,W7,W8
      COMMON/FAC/FCT(100),NPISK,NTRIX
      NP1 = NP+1
      N = M
      NK = ND+1
      U = AR
      V = 0.5D0*AR
      CALL BN(U,NK,X1,Y1)
      CALL BN(V,NK,X2,Y2)
      L1 = NK
      L = NK-1
      B1 = DABS(U**2*(X1(2)*Y1(1)-X1(1)*Y1(2))-1.0D0)
      B2 = DABS(V**2*(X2(2)*Y2(1)-X2(1)*Y2(2))-1.0D0)
      CN1 = DABS(U**2*(X1(L1)*Y1(L)-X1(L)*Y1(L1))-1.0D0)
      CN2 = DABS(V**2*(X2(L1)*Y2(L)-X2(L)*Y2(L1))-1.0D0)
      PRINT 89
   89 FORMAT('0 BESSEL- NEUMAN- TEST FOR VR')
      PRINT 90, B1,CN1,B2,CN2
   90 FORMAT(4D25.15)
      DO 7 I = 1,L
      DO 7 J = 1,L
      R1(I,J) = (0.0D0,0.0D0)
      R2(I,J) = (0.0D0,0.0D0)
      R3(I,J) = (0.0D0,0.0D0)
      R4(I,J) = (0.0D0,0.0D0)
    7 CONTINUE
      NR = L/2
      DO 200 I = 1,NR
      DO 200 J = 1,NR
      IF(N-I)8,8,200
    8 IF(N-J)9,9,200
    9 L1 = I+J+1
      W1 = (0.0D0,0.0D0)
      W2 = (0.0D0,0.0D0)
      W3 = (0.0D0,0.0D0)
      W4 = (0.0D0,0.0D0)
      W5 = (0.0D0,0.0D0)
      W6 = (0.0D0,0.0D0)
      W7 = (0.0D0,0.0D0)
      W8 = (0.0D0,0.0D0)
      DO 100 L = 1,L1
      K = L-1
      IF(K-IABS(I-J))100,10,10
   10 CONTINUE
      IF(N)11,11,12
   11 IF(MOD((I+J+K),2).NE.0) GO TO 100
      FACT1 = 1.0D0
      GO TO 13
   12 FACT1 = DFLOAT((-1)**N)
   13 J1 = 2*I
      J2 = 2*J
      J3 = 2*K
      M1 = 2*N
```

```
      F3J1 = TRIXJ(J1,J2,J3,M1,FCT,NRISK,NTRIX)
      FACT2 = 0.5D0*DFLOAT(2*K+1)*F3J1*
     1DSQRT(DFLOAT((2*I+1)*(2*J+1))/DFLOAT(I*(I+1)*J*(J+1)))
      FACT3 = FACT1*FACT2
      IF(MOD((J-I+K),2).NE.0) GO TO 27
      IF(J-I+K)25,23,24
   23 FACT4 = 1.0D0
      GO TO 26
   24 FACT4 = DFLOAT((-1)**((J-I+K)/2))
      GO TO 26
   25 FACT4 = DFLOAT((-1)**((I-J-K)/2))


   26 J1 = 2*I
      J2 = 2*J
      J3 = 2*K
      M1 = 0
      F3J2 = TRIXJ(J1,J2,J3,M1,FCT,NRISK,NTRIX)
      FACT5 = DFLOAT(I*(I+1)+J*(J+1)-K*(K+1))*F3J2
      GO TO 28
   27 FACT6 = 0.0D0
      GO TO 29
   28 FACT6 = FACT4*FACT5
   29 IF(K-IABS(I-J)-1)44,30,30
   30 IF(MOD((J-I+K+1),2).NE.0) GO TO 44
      IF(J-I+K+1)33,31,32
   31 FACT7 = 1.0D0
      GO TO 41
   32 FACT7 = DFLOAT((-1)**((J-I+K+1)/2))
      GO TO 41
   33 FACT7 = DFLOAT((-1)**((I-J-K-1)/2))
   41 J1 = 2*I
      J2 = 2*J
      J3 = 2*(K-1)
      M1 = 0
      F3J3 = TRIXJ(J1,J2,J3,M1,FCT,NRISK,NTRIX)
      IF(IABS(I-J))42,42,43
   42 FACT8 = DFLOAT(K)*DSQRT(DFLOAT((I+J+1)**2-K**2))*F3J3
      GO TO 45
   43 FACT8 = DSQRT(DFLOAT((K**2-IABS(I-J)**2)*((I+J+1)**2-K**2)))*F3J3
      GO TO 45
   44 FACT9 = 0.0D0
      GO TO 50
   45 FACT9 = FACT7*FACT8
   50 CONTINUE
      IF(K)51,51,52
   51 FACT10 = 1.0D0
      GO TO 53
   52 FACT10 = DFLOAT((-1)**K)
   53 ACR1 = FACT3*FACT6
      ACR2 = FACT10*ACR1
      ACR3 = FACT3*FACT9
      ACR4 = FACT10*ACR3
      K1 = K+1
      W1 = W1+DCMPLX(ACR1*X1(K1),0.0D0)
      W2 = W2+DCMPLX(ACR1*X1(K1),ACR1*Y1(K1))
      W3 = W3+DCMPLX(ACR2*X1(K1),ACR2*Y1(K1))
      W4 = W4+DCMPLX(ACR1*X2(K1),0.0D0)
      IF(N)100,100,99
```

```
 99 CONTINUE
    W5 = W5+DCMPLX(ACR3*X1(K1),0.0D0)
    W6 = W6+DCMPLX(ACR3*X1(K1),ACR3*Y1(K1))
    W7 = W7+DCMPLX(ACR4*X1(K1),ACR4*Y1(K1))
    W8 = W8+DCMPLX(ACR3*X2(K1),0.0D0)
100 CONTINUE
    R1(2*I-1,2*J-1) = W1
    R2(2*I-1,2*J-1) = W2
    R3(2*I-1,2*J-1) = W3
    R4(2*I-1,2*J-1) = W4
    R1(2*I,2*J) = W1
    R2(2*I,2*J) = W2
    R3(2*I,2*J) = W3
    R4(2*I,2*J) = W4
    GO TO (101,103),NP1
101 CONTINUE
    IF(N)200,200,102
102 CONTINUE
    R1(2*I-1,2*J) = W5
    R2(2*I-1,2*J) = W6
    R3(2*I-1,2*J) = W7
    R4(2*I-1,2*J) = W8
    R1(2*I,2*J-1) = -W5
    R2(2*I,2*J-1) = -W6


    R3(2*I,2*J-1) = -W7
    R4(2*I,2*J-1) = -W8
    GO TO 200
103 CONTINUE
    IF(N)200,200,104
104 CONTINUE
    R1(2*I-1,2*J) = -W5
    R2(2*I-1,2*J) = -W6
    R3(2*I-1,2*J) = -W7
    R4(2*I-1,2*J) = -W8
    R1(2*I,2*J-1) = W5
    R2(2*I,2*J-1) = W6
    R3(2*I,2*J-1) = W7
    R4(2*I,2*J-1) = W8
200 CONTINUE
    RETURN
    END
```

If A is really stored as a vector then variable dimension N can be used!


```
      SUBROUTINE MCNV(A,N,D,L,M)
      DIMENSION A(1),L(1),M(1)
      COMPLEX*16 A,D,BIGA,HOLD
      D = (1.0D0,0.0D0)
      NK=-N
      DO 80 K=1,N
      NK=NK+N
      L(K)=K
      M(K)=K
      KK=NK+K
      BIGA=A(KK)
      DO 20 J=K,N
      IZ=N*(J-1)
      DO 20 I=K,N
      IJ=IZ+I
   10 IF(CDABS(BIGA)-CDABS(A(IJ))) 15,20,20
   15 BIGA=A(IJ)
      L(K)=I
      M(K)=J
   20 CONTINUE
      J=L(K)
      IF(J-K) 35,35,25
   25 KI=K-N
      DO 30 I=1,N
      KI=KI+N
      HOLD=-A(KI)
      JI=KI-K+J
      A(KI)=A(JI)
   30 A(JI) =HOLD
   35 I=M(K)
      IF(I-K) 45,45,38
   38 JP=N*(I-1)
      DO 40 J=1,N
      JK=NK+J
      JI=JP+J
      HOLD=-A(JK)
      A(JK)=A(JI)
   40 A(JI) =HOLD
   45 IF(CDABS(BIGA)) 48,46,48
   46 D = (0.0D0,0.0D0)
      RETURN
   48 DO 55 I=1,N
      IF(I-K) 50,55,50
   50 IK=NK+I
      A(IK)=A(IK)/(-BIGA)
   55 CONTINUE
      DO 65 I=1,N
      IK=NK+I
      HOLD=A(IK)
      IJ=I-N
      DO 65 J=1,N
      IJ=IJ+N
      IF(I-K) 60,65,60
   60 IF(J-K) 62,65,62
   62 KJ=IJ-I+K
      A(IJ)=HOLD*A(KJ)+A(IJ)
   65 CONTINUE
      KJ=K-N
      DO 75 J=1,N
```

```
      KJ=KJ+N
      IF(J-K) 70,75,70
   70 A(KJ)=A(KJ)/BIGA
   75 CONTINUE
      D=D*BIGA
      A(KK) = (1.0D0,0.0D0)/BIGA
   80 CONTINUE
      K=N
  100 K=(K-1)
      IF(K) 150,150,105
  105 I=L(K)


      IF(I-K) 120,120,108
  108 JQ=N*(K-1)
      JR=N*(I-1)
      DO 110 J=1,N
      JK=JQ+J
      HOLD=A(JK)
      JI=JR+J
      A(JK)=-A(JI)
  110 A(JI) =HOLD
  120 J=M(K)
      IF(J-K) 100,100,125
  125 KI=K-N
      DO 130 I=1,N
      KI=KI+N
      HOLD=A(KI)
      JI=KI-K+J
      A(KI)=-A(JI)
  130 A(JI) =HOLD
      GO TO 100
  150 RETURN
      END
```

```
      SUBROUTINE COND(M,ND,RE,IM)
      DIMENSION RE(ND,1),IM(ND,1)
      DOUBLE PRECISION RE,IM,SCALE
      MD = 1
      IF(M.GT.1) MD = 2*M-1
      MD1 = MD+1
      NBGR = ND
      NROW = NBGR
      DO 60 KR = MD1,NBGR
      SCALE = 1.0D0/IM(NROW,NROW)
      DO 8 LC = MD,NBGR
      RE(NROW,LC) = SCALE*RE(NROW,LC)
      IM(NROW,LC) = SCALE*IM(NROW,LC)
    8 CONTINUE
      MROW = NROW-1
      DO 20 MR = MD,MROW
      SCALE = IM(MR,NROW)
      DO 16 MC = MD,NBGR
      RE(MR,MC) = RE(MR,MC)-SCALE*RE(NROW,MC)
      IM(MR,MC) = IM(MR,MC)-SCALE*IM(NROW,MC)
   16 CONTINUE
   20 CONTINUE
      NROW = NROW-1
   60 CONTINUE
      NROW = NBGR-1
      DO 80 I = 1,NROW
      IB = I+1
      DO 72 J = IB,NBGR
      IM(I,J) = 0.0D0
   72 CONTINUE
   80 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ORTHO(M,ND,RE,IM,X,Y)
      DIMENSION RE(ND,1),IM(ND,1),X(1),Y(1)
      DOUBLE PRECISION RE,IM,X,Y,SUM1,SUM2
      MD = 1
      IF(M.GT.1) MD = 2*M-1
      NBGR = ND
      SUM1 = 0.0D0
      DO 20 K = MD,NBGR
      SUM1 = SUM1+RE(NBGR,K)**2+IM(NBGR,K)**2
   20 CONTINUE
      SUM1 = DSQRT(SUM1)
      DO 28 K = MD,NBGR
      RE(NBGR,K) = RE(NBGR,K)/SUM1
      IM(NBGR,K) = IM(NBGR,K)/SUM1
   28 CONTINUE
      NMI = NBGR-1
      NROW = NBGR
      DO 100 I = MD,NMI
      NROW = NROW-1
      MROW = NROW
      DO 36 K = MD,NBGR
      X(K) = RE(NROW,K)
      Y(K) = IM(NROW,K)
   36 CONTINUE
      DO 80 J = NROW,NMI
      SUM1 = 0.0D0
      SUM2 = 0.0D0
      MROW = MROW+1
      DO 40 K = MD,NBGR
      SUM1 = SUM1+RE(MROW,K)*RE(NROW,K)+IM(MROW,K)*IM(NROW,K)
      SUM2 = SUM2+RE(MROW,K)*IM(NROW,K)-IM(MROW,K)*RE(NROW,K)
   40 CONTINUE
      DO 48 K = MD,NBGR
      X(K) = X(K)-SUM1*RE(MROW,K)+SUM2*IM(MROW,K)
      Y(K) = Y(K)-SUM1*IM(MROW,K)-SUM2*RE(MROW,K)
   48 CONTINUE
   80 CONTINUE
      SUM1 = 0.0D0
      DO 84 K = MD,NBGR
      SUM1 = SUM1+X(K)**2+Y(K)**2
   84 CONTINUE
      SUM1 = DSQRT(SUM1)
      DO 88 K = MD,NBGR
      RE(NROW,K) = X(K)/SUM1
      IM(NROW,K) = Y(K)/SUM1
   88 CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PERFT(NP,M,NR,ND,AR,AI,BR,BI,T,RE,IM,X,Y)
      DIMENSION AR(NR,1),AI(NR,1),BR(NR,1),BI(NR,1),T(ND,1),RE(ND,1),
     1IM(ND,1),X(1),Y(1)
      DOUBLE PRECISION AR,AI,BR,BI,RE,IM,X,Y,FAC
      COMPLEX*16 T
      MD = 1
      IF(M.GT.1) MD = 2*M-1
      NR = ND/2
      FAC = 1.0D0
      IF(NP.GT.0) FAC = -1.0D0
      DO 10 I = 1,NR
      DO 10 J = 1,NR
      RE(2*I-1,2*J-1) = AR(I,J)
      RE(2*I-1,2*J) = FAC*BR(I,J)
      RE(2*I,2*J-1) = FAC*BR(I,J)
      RE(2*I,2*J) = -AR(I,J)
      IM(2*I-1,2*J-1) = AI(I,J)
      IM(2*I-1,2*J) = FAC*BI(I,J)
      IM(2*I,2*J-1) = FAC*BI(I,J)
      IM(2*I,2*J) = -AI(I,J)
      IF(I.EQ.J) IM(2*I,2*I) = 1.0D0-AI(I,I)
   10 CONTINUE
      CALL COND(M,ND,RE,IM)
      CALL ORTHO(M,ND,RE,IM,X,Y)
      DO 11 I = 1,ND
      DO 11 J = 1,ND
      T(I,J) = (0.0D0,0.0D0)
   11 CONTINUE
      DO 12 I = MD,ND
      DO 12 J = MD,ND
      DO 12 K = MD,ND
      T(I,J) = T(I,J)-DCMPLX(RE(K,I)*RE(K,J),-IM(K,I)*RE(K,J))
   12 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LINE(THETA,NIP,C,ALPHA,R,DR)
      DOUBLE PRECISION THETA,C,ALPHA,R,DR
      R = C*DSIN(ALPHA)/DSIN(THETA+DFLOAT(NIP)*ALPHA)
      DR = -R/DTAN(THETA+DFLOAT(NIP)*ALPHA)
      RETURN
      END
```

```
      SUBROUTINE TRCIR(STH,CTH,C,A,R,DR)
      DOUBLE PRECISION STH,CTH,C,A,R,DR,X,ST,CT
      ST = C*STH
      CT = C*CTH
      X = DSQRT(A**2-ST**2)
      R = CT+X
      DR = -ST-ST*CT/X
      RETURN
      END
```

```
      SUBROUTINE ELLIPS(STH,CTH,AZ,BRA,R,DR)
      DOUBLE PRECISION STH,CTH,AZ,BRA,R,DR
      R = AZ/DSQRT(CTH**2+(AZ*STH/BRA)**2)
      DR = R**3*STH*CTH*(1.0D0-(AZ/BRA)**2)/AZ**2
      RETURN
      END
```

```
      SUBROUTINE TRELLI(STH,CTH,AZ,BRA,C,R,DR)
      DOUBLE PRECISION STH,CTH,AZ,BRA,C,R,DR,NE,RO
      NE = (AZ*STH)**2+(BRA*CTH)**2
      RO = NE-(C*STH)**2
      RO = DSQRT(RO)
      R = (BRA**2*C*CTH+AZ*BRA*RO)/NE
      DR = -2.0D0*STH*CTH*(AZ**2-BRA**2)*R/NE+(-BRA**2*C*STH+STH*CTH*AZ*
     1BRA*(AZ**2-BRA**2-C**2)/RO)/NE
      RETURN
      END
```

Sphere-cone-sphere.

```
      A = 1.0D0
      ALPHA = PI*15.0D0/180.0D0
      NDLTH(1) = 64
      NDLTH(2) = 64
      NDLTH(3) = 64
      CALL SPCOSP(ALPHA,A,THETA1,THETA2)
      NSECT = 3
      NIP = -1
      B = 0.5D0*A*(1.0D0-DSIN(ALPHA))/(1.0D0+DSIN(ALPHA))
      C = 0.5D0*A*(DSIN(ALPHA)**2+DSIN(ALPHA)+2.0D0)/(DSIN(ALPHA)*(1.0D0
     1+DSIN(ALPHA)))
      D = -0.5D0*A
      E = A/(1.0D0+DSIN(ALPHA))
      THETA3 = PI
      CDH(1) = THETA1/DFLOAT(NDLTH(1))
      CDH(2) = (THETA2-THETA1)/DFLOAT(NDLTH(2))
      CDH(3) = (THETA3-THETA2)/DFLOAT(NDLTH(3))
```

```
      GO TO (1,2,3),ISECT
    1 CONTINUE
      CALL TRCIR(STH,CTH,B,A,R,DR)
      GO TO 4
    2 CONTINUE
      CALL LINE(THETA,NIP,C,ALPHA,R,DR)
      GO TO 4
    3 CONTINUE
      CALL TRCIR(STH,CTH,D,E,R,DR)
    4 CONTINUE
```

```
      SUBROUTINE SPCOSP(ALPHA,A,THETA1,THETA2)
      DOUBLE PRECISION ALPHA,SNA,CSA,A,BDA,Q,THETA1,THETA2
      SNA = DSIN(ALPHA)
      CSA = DCOS(ALPHA)
      BDA = 1.0D0/(1.0D0+SNA)
      Q = (1.0D0-BDA)*(1.0D0-SNA)/2.0D0
      THETA1 = DATAN(SNA*CSA/(Q-SNA**2))
      THETA2 = DATAN(BDA*SNA*CSA/(1.0D0-Q-BDA*CSA**2))
      THETA2 = 4.0D0*DATAN(1.0D0)-THETA2
      RETURN
      END
```

Translated circle

```
A = 1.0D0
C = -0.1D0
NDLTH(1) = 192
NSECT = 1
THETA1 = PI
CDH(1) = THETA1/DFLOAT(NDLTH(1))

CALL TRCIR(STH,CTH,C,A,R,DR)
```

Ellips

```
A = 1.0D0
B = 0.5D0
NDLTH(1) = 192
NSECT = 1
THETA1 = PI
CDH(1) = THETA1/DFLOAT(NDLTH(1))

CALL ELLIPS(STH,CTH,A,B,R,DR)
```

Translated ellips

```
A = 1.0D0
B = 0.5D0
C = -0.1D0
NDLTH(1) = 192
NSECT = 1
THETA1 = PI
CDH(1) = THETA1/DFLOAT(NDLTH(1))

CALL TRELLI(STH,CTH,A,B,C,R,DR)
```