NBSIR 86-3362

SCAT: A Vector Program to Solve A Transient MFIE

Egon Marx

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratories
Center for Manufacturing Engineering
Gaithersburg, MD 20899

April 1986

U.S. DEPARTMENT OF COMMERCE
BUREAU OF STANDARDS

NBSIR 86-3362

SCAT: A VECTOR PROGRAM TO SOLVE A TRANSIENT MFIE

Egon Marx

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratories
Center for Manufacturing Engineering
Gaithersburg, MD 20899

April 1986

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

Abstract

The FORTRAN program SCAT is used to solve the magnetic field integral equation (MFIE) to determine the fields scattered by a perfectly conducting sphere. The incident field is a plane-wave pulse, and a stepping-in-time procedure is used to determine the surface current density induced on the sphere. The program does not take advantage of the special symmetry of the scatterer because it is intended to serve as a verified starting point for more general programs. The output is compared to that of the program PERF, which computes the same fields via a Fourier transform of the monochromatic fields obtained from the Mie formulas. The contributions of the self-patch and neighboring patches to the singular integral are optionally computed by using their expansions in the linear size of the patches. The self-patch term is important for the solution of other integral equations that may be of the first kind. For the MFIE, these corrections are small but not negligible. The program takes advantage of the vector programming features of the CYBER 205.

Key words: computer programs; electromagnetic pulse scattering; magnetic field integral equation; Mie scattering; perfectly conducting sphere; stepping-in-time procedure; transient electromagnetic fields; vector programming.

Table of Contents

		Page
1.	Introduction	1
2.	Program PERF	5
3.	Program SCAT	8
	3.1 The Main Loop	9
	3.2 The Interpolation Formulas	9
	3.3 Subroutine RUNT	12
	3.4 Subroutine INFL	13
	3.5 Subroutine FARFL	14
4.	Self-Patch Corrections	15
5.	Neighboring-Patch Corrections	17
6.	Vector Programming Considerations	23
7.	Ancillary Programs	25
	7.1 Program PERFPLT	25
	7.2 Program CURCHK	26
	7.3 Program SCATP	26
	7.4 Subroutine VINUFT	27
8.	Program Inputs and Outputs	28
9.	Conclusions	38
References		39
Appendix		41

1. Introduction

There are many procedures related to the metrology mission of the National Bureau of Standards that require the analysis of waves scattered by an obstacle. Comparison of the scattered waves with those computed for known obstacles can serve to determine parameters related to the scatterer. Codes used in inverse scattering problems also may require the solution of the direct problem. If the scatterer is a homogeneous body, the scattered fields can be computed from tangential fields defined on the surface of the scatterer. In turn, these fields can be computed by solving a singular integral equation.

The electromagnetic fields scattered by a perfect conductor can be obtained form the surface current density $\vec{J}_s$ [1]*. This tangential field can be determined by solving the magnetic field integral equation (MFIE),

$$\vec{J}_s(\vec{x},t) = (2/\mu_0)\hat{n} \times \vec{B}^{in}(\vec{x},t)$$

$$- \frac{1}{2\pi} \hat{n} \times \oint_S dS' \vec{R} \times \left[\frac{1}{R^2 c} \frac{\partial \vec{J}_s(\vec{x}',\tau)}{\partial t'} + \frac{1}{R^3} \vec{J}_s(\vec{x}',\tau) \right], \qquad (1)$$

where $\vec{B}^{in}$ is the magnetic field of the incident pulse, μ_0 is the permeability of free space, $\hat{n}$ is the outside normal to the

*Numbers in square brackets indicate the literature references at the end of his report. There are further references in these publications.

scatterer, τ is the retarded time

$$\tau = t - R/c, \tag{2}$$

and

$$\vec{R} = \vec{x} - \vec{x}', \quad R = |\vec{R}|. \tag{3}$$

This is a singular integral equation because the integrand diverges where $R = 0$, and it is an integral equation of the second kind because the unknown field $\vec{J}_s$ also appears outside the integral, on the left side of the equation.

An example of a singular integral equation of the first kind is the electric field integral equation (EFIE),

$$0 = \hat{n} \times \vec{E}^{in}(\vec{x},t) + \frac{1}{4\pi}\hat{n} \times \oint_S dS' \left[\frac{\vec{R}}{\epsilon_0 R^2 c} \frac{\partial \rho_s(\vec{x}',\tau)}{\partial t'} \right.$$

$$\left. + \frac{\vec{R}}{R^3}\rho_s(\vec{x}',\tau) - \frac{\mu_0}{R} \frac{\partial \vec{J}_s(\vec{x}',\tau)}{\partial t'} \right], \tag{4}$$

where $\vec{E}^{in}$ is the electric field of the incident pulse, ϵ_0 is the permittivity of free space, and ρ_s is the surface charge density, related to $\vec{J}_s$ by conservation of charge.

We have shown [2, 3] how the scattered and transmitted fields of a pulse scattered off a homogeneous dielectric body can be determined from a single tangential vector field that obeys a singular integral equation of the first kind. We have further

generalized [4] this formulation to the scattering by a dispersive body, such as an imperfect conductor. In some papers we also derive equations for the scattering of other types of fields by homogeneous bodies, as well as the corresponding equations for the scattering of monochromatic waves.

The solution of the integral equations by means of the stepping-in-time procedure takes advantage of causality and the finite speed of propagation of the wave. The new values of the unknown field are expressed in terms of an integral over the retarded values of the field, which have been calculated in previous steps.

The integration can be performed numerically by dividing the surface of the scatterer into patches and adding the products of the integrand at the center of each patch times the area of the patch. This prescription does not apply to the contribution of the self-patch because this is the patch where the integrand becomes infinite as the field point $\vec{x}$ and the source point $\vec{x}'$ coincide. For an integral equation of the second kind, the contribution of the self-patch can be neglected in the computation of a new value of the field in comparison to the large number of contributions from the other patches. This is not so for an integral equation of the first kind, because the self-patch contribution to the integral is the only term that depends on the new value of the field.

We have expanded the contribution of the self-patch to first order in the linear size of the patch for an arbitrary surface [5], and in this form it may be included in the integral. We

have similarly expanded the contribution of a neighboring patch, because the integrand cannot be well approximated by its value at the center of such a patch when $1/R$ or $1/R^2$ varies significantly over the patch [6].

We have developed the program SCAT to implement the computation of the surface current density and the far fields scattered by a perfectly conducting sphere by solving the MFIE. In this program we provide the option of computing the correction to the surface fields due to the self-patch contribution and the more accurate computation of the neighboring-patch contributions.

We found that terms that contain the time derivatives of the fields, terms that are negligible in most computations, cannot be ignored when the fields themselves are small [7]. The expansions of the surface current density $\vec{J}_s$ and its derivatives that we use in SCAT are explained in [7]; they differ from those we used in [5] and [6] in that the dependence of the retarded time τ on the curvilinear coordinates u and v is taken into account when a partial derivative with respect to u or v is taken.

We intend SCAT to serve as a validated starting point for other programs that will determine the fields scattered by more general types of scatterers. For this reason we do not take advantage of special symmetry properties of the sphere. To validate the program, we compare the far fields to those computed by means of another program, PERF, which uses the Mie formulas [7] for monochromatic fields and a Fourier transform. This program is discussed in section 2.

The general structure of SCAT is presented in section 3. Section 4 is devoted to a discussion of the issues related to the

computation of the contributions of the patches neighboring the self-patch, and in section 5 we examine the contribution of the self-patch itself. In section 6 we point out where we use vector programming as appropriate for the CYBER 205. We describe a number of ancillary programs in section 7, and we show some examples of results in section 8. Listings of computer programs are given in the appendix.

We define the incident plane-wave pulse so that it reaches the sphere at t = 0. We choose a double-exponential smoothed at the beginning by a power factor for the profile of the plane wave. The incident magnetic field is

$$B_1^{in}(\zeta) = \beta\zeta(e^{\alpha\zeta} - a^{\beta\zeta})\mathcal{U}(-\zeta), \quad B_2^{in}(\zeta) = B_3^{in}(\zeta) = 0, \tag{5}$$

where $\mathcal{U}$ is the unit step function, α is the rise constant of the pulse, β the decay constant, and

$$\zeta = z + a - ct, \tag{6}$$

where a is the radius of the sphere.

2. Program PERF

The Mie formulas give the fields scattered by a homogeneous sphere from an incident plane monochromatic wave. For a perfect conductor and a wave incident along the z-axis, the scattered far field components reduce to

$$E_{\phi} = \frac{\exp(ikr)}{ikr} \sin\phi \sum_{n=1}^{\infty} \frac{2n + 1}{n(n + 1)} \left\{ a_n \frac{P_n^{(1)}(\cos\theta)}{\sin\theta} \right.$$

$$\left. + b_n \frac{dP_n^{(1)}(\cos\theta)}{d\theta} \right\} (-1)^{n+1}, \qquad (7)$$

$$E_{\theta} = - \frac{\exp(ikr)}{ikr} \cos\phi \sum_{n=1}^{\infty} \frac{2n + 1}{n(n + 1)} \left\{ a_n \frac{dP_n^{(1)}(\cos\theta)}{d\theta} \right.$$

$$\left. + b_n \frac{P_n^{(1)}(\cos\theta)}{\sin\theta} \right\} (-1)^{n+1}, \qquad (8)$$

where k is the wavenumber, r is the distance from the center of the sphere to point where the field is computed, θ is the polar or scattering angle, ϕ is the azimuthal or polarization angle, $P_n^{(1)}$ is the associated Legendre function, and a_n and b_n are coefficients given by

$$a_n(\rho) = - j_n(\rho) / h_n^{(1)}(\rho), \qquad (9)$$

$$b_n(\rho) = - [\rho j_n(\rho)]' / [\rho h_n^{(1)}(\rho)]', \qquad (10)$$

where j_n is a spherical Bessel function, $h_n^{(1)}$ is a spherical Hankel function of the first kind, and

$$\rho = ka = 2\pi a/\lambda. \qquad (11)$$

The primes in eq. (11) indicate derivatives with respect to ρ.

The Fourier transform of the incident wave (5) is

$$B_\omega^{in}(z) = \exp[ik(z + a)](\beta/c)[(\beta - ik)^{-2} - (\alpha - ik)^{-2}], \qquad (12)$$

where $k = \omega/c$. The coefficient of $\exp(ikz)$ multiplies the amplitude of the far fields given in eqs. (7) and (8). The steady state component of the incident pulse does not contribute to the scattered fields.

In the program PERF, we compute first the required Bessel and Hankel functions by means of the subroutines BESRJ and BESRY [8], then the scattering coefficients a_n and b_n in the subroutine SCACO, and the monochromatic fields in the subroutine FLD. The inverse Fourier transform is carried out in subroutine INUFT [9] or VINUFT (its vector version); we do not use a fast Fourier transform because we determined that it was better to cover low and high frequencies at different intervals than to use an equispaced grid. The frequencies are selected in subroutine OMEGA [9] and the intervals increase in a geometric progression, which gives a thorough coverage of low frequencies and a sparse but extended coverage of high frequencies. The input to the program requires the specification of α, β, and the radius of the sphere, the number of scattering angles and the scattering angles. In addition, we have to specify the lower and upper

frequency limits at which the fields are computed and the number of frequency and time points. The output consists of the intensity of the scattered fields for both polarizations of the scattered fields; they are saved in a file and they are plotted using the subroutines DRAW3 [10] or DRAW4 [11].

3. Program SCAT

In this program we compute the surface current density $\vec{J}_s$ induced on the sphere by solving the MFIE (1) in the time domain by a stepping-in-time procedure. The scattered fields are then obtained by integration, and we compute the intensities in the far zone for comparison with the output of PERF. The magnetic field in the radiation zone is

$$c\vec{B}^{rad}(\vec{x},t) = -\frac{\mu_0}{4\pi r}\hat{r} \times \oint_S dS' \frac{\partial \vec{J}_s(\vec{x}',\tau)}{\partial t'}. \qquad (13)$$

The surface current density is tangential to the surface and has only two independent components. To decrease the amount of memory required to run the program, we decided to store only the components along the meridians and parallels on the sphere. To sum the contributions from different patches, we first compute the components of the vectors in a Cartesian coordinate system. Once the two tangential components of the surface current density are determined, they are saved on disk for future use.

3.1 The Main Loop

Most of the execution time needed for the program is spent computing the surface current density by solving the MFIE. The components of this tangential vector are computed at the center of each patch starting at t = 0 and after each time increment Δt.

If we ignore the self-patch contribution to the integral in eq. (1), the contributions from the other patches can be computed because the fields at their centers have previously been calculated at the retarded times. We have to choose a time interval that is smaller than the distance between any pair of patches (divided by the speed of light c) to obtain a retardation that is sufficiently large. It is unlikely that the current density will have been calculated at precisely the retarded time, and an interpolation is needed to obtain the necessary values. The interpolation is discussed in the next subsection.

After the values of the surface current density are computed in this way, we can perform the corrections for the self-patch, for the neighboring patches, or for both; these corrections are described in detail in separate sections.

3.2 The Interpolation Formulas

The interpolation method [12] turned out to affect significantly the outcome of the calculations. The program SCAT allows for two alternatives. In the first one, the fields at the retarded times are computed from a polynomial of degree 4 that passes through five points chosen so that they fall on both sides

of the retarded time. The relation has the form

$$y(t) = a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{14}$$

and we rescale the independent variable so that it takes integer values from -2 to 2. In terms of the corresponding values of the dependent variable, the coefficients are

$$a_4 = \frac{1}{24}y(2) - \frac{1}{6}y(1) + \frac{1}{4}y(0) - \frac{1}{6}y(-1) + \frac{1}{24}y(-2), \tag{15}$$

$$a_3 = \frac{1}{12}y(2) - \frac{1}{6}y(1) + \frac{1}{6}y(-1) - \frac{1}{12}y(-2), \tag{16}$$

$$a_2 = -\frac{1}{24}y(2) + \frac{2}{3}y(1) - \frac{5}{4}y(0) + \frac{2}{3}y(-1) - \frac{1}{24}y(-2), \tag{17}$$

$$a_1 = -\frac{1}{12}y(2) + \frac{2}{3}y(1) - \frac{2}{3}y(-1) + \frac{1}{12}y(-2), \tag{18}$$

$$a_0 = y(0). \tag{19}$$

In the second method, a polynomial of degree 2 is fitted to the same five points by a least-squares approximation. The coefficients a_4 and a_3 are zero, and the other coefficients are

$$a_2 = \frac{1}{7}y(2) - \frac{1}{14}y(1) - \frac{1}{7}y(0) - \frac{1}{14}y(-1) + \frac{1}{7}y(-2), \tag{20}$$

$$a_1 = \frac{1}{5}y(2) + \frac{1}{10}y(1) - \frac{1}{10}y(-1) - \frac{1}{5}y(-2), \qquad (21)$$

$$a_0 = -\frac{3}{35}y(2) + \frac{12}{35}y(1) + \frac{17}{35}y(0) + \frac{12}{35}y(-1) - \frac{3}{35}y(-2). \qquad (22)$$

The polynomial interpolation gives more accurate results, but leads more easily to numerical instabilities [7, 13]. The time derivatives of the fields are calculated from

$$y'(t) = 4a_4t^3 + 3a_3t^2 + 2a_2t + a_1. \qquad (23)$$

The values of the fields and their derivatives are set to zero for times before the time the incident pulse reaches the center of the corresponding patch on the sphere. The interpolated values do not necessarily vanish at these times and the error increases when they are used.

The interpolation formulas are implemented in a form suitable for the execution of vector instructions in subroutines INTRP0 and INTRP1, and INTRP2 and INTRP3 are scalar versions needed only for the computation of the derivatives of the fields at the self-patch. The choice of the five points for the interpolation also has an effect on the accuracy of the results, which is improved if the retarded time is close to the center of the interval. We choose the points in this way except for patches that are so close to the self-patch that the retardation is too small for all the needed values to be available. The values of the components of the surface current density and of their time derivatives at the self-patch itself are needed to

carry out the self- and neighboring-patch corrections; we use the uncorrected values at the self-patch as a first approximation where necessary.

3.3 Subroutine RUNT

This subroutine is executed once at the beginning of the program to perform the division of the sphere into patches and to compute coefficients that will be needed during the execution of the program.

We do not take advantage of the symmetry of the sphere to simplify the integral equations, but we have to take into account the characteristics of the sphere to choose a good way to divide it into patches. We naturally use spherical coordinates, the polar angle θ and the azimuthal angle ϕ, which form a principal orthogonal curvilinear coordinate system [5].

We divide the surface of the sphere into patches by meridians and parallels. We first divide the surface into n_1 strips of equal increments of θ. Each strip is then divided into a number of patches that is roughly proportional to the length of the strip, that is, to $\sin\theta$. We leave two undivided caps at the poles, where spherical coordinates become undefined; we use Cartesian coordinates for these caps, which requires special formulas for the computations. We divide the strips next to the caps into a given number n_3 of patches, usually 5. Then the other strips are divided into approximately

$$n = (n_2 - n_3)\sin\theta + n_3 \qquad (24)$$

patches of equal size by lines of constant ϕ.

This more or less asymmetrical division of the surface into patches leads to computed nonzero values for some intermediate variables that should vanish by symmetry. Although this discrepancy does not seem to make much difference in the final results, we get intermediate values that are closer to zero by restricting the number of patches in each strip to a power of 2.

For strips near the poles, n is not large and the values of the increment in ϕ, $\Delta\phi = 2\pi/n$, may not be small compared to 1. The side of the patch, $a\sin\theta\Delta\phi$, is small compared to a.

To obtain patches that are approximately square in form, we would have to choose $n_2 = 2n_1$. A choice that works essentially just as well is $n_2 = n_1$. The consequent reduction in memory requirements and computing time is a made possible by the periodicity of the fields in ϕ, which greatly improves the accuracy of the integration over that variable [7].

We compute and store the components of the radius vector, of the tangential vectors, the surface elements, and the coefficients required for self- and neighboring-patch corrections. These computations involve the evaluation of trigonometric functions and logarithms, the performance of numerical integrations, and other time-consuming operations, but they are done only once in each run.

3.4 Subroutine INFL

The incident field $\vec{B}^{in}$ has to be computed at the center of

each patch at each time step; this is done in subroutine INFL. For the plane-wave pulse, eq. (5) allows us to calculate the magnetic field for arbitrary t. If the pulse is spatially limited, the incident fields have to be computed from their initial values by solving Maxwell's equations. This can be done by numerical integration over the information-collecting sphere [14].

3.5 Subroutine FARFL

The far field is calculated in subroutine FARFL by doing the integration in eq. (13) for any number of scattering directions given by the corresponding values of θ and ϕ for the direction of the scattered pulse. The intensities of the fields are plotted and they are also saved on disk for comparison with the results of other calculations.

The time scale is referred back to the center of the sphere, as is appropriate to compare the fields with the output from PERF. The time the scattered pulse starts is a function of the scattering angle,

$$t_0 = (a/c)[1 - 2\sin(\theta/2)], \qquad (25)$$

and the duration of the pulse is comparable to the length of the pulse and to the size of the sphere.

4. Self-Patch Corrections

The surface integral in eq. (9) is singular because the integrand diverges when $R = 0$, which happens on the self-patch. The magnitude of the first term in the integrand is proportional to R^{-1} and the integral of that term is well defined. The magnitude of the second term is proportional to R^{-2}, and the integral is defined in a limiting sense analogous to the principal value of a one-dimensional integral. In an expansion in the size of the patch [5], the leading term in the integral is not absolutely integrable, but it vanishes by symmetry when the field is computed at the center of the patch. The next term is linear in the size of the patch.

The self-patch contribution to the integral is usually neglected, although there is no more reason to neglect the contribution of this patch than that of any other patch. Actually, the self-patch contribution is linear in the size of the patch, while those of most other terms are quadratic. Also the self-patch contribution is of fundamental importance in an integral equation of the first kind, where the unknown field does not appear outside the integral. We thus include in the program the option to compute the self-patch correction.

The expansion of the integral of the first term is of order 2 in the size of the patch, but we cannot neglect this term when $\Delta\vec{J}_s$ is comparable to $\vec{J}_s$ [7]. In this case, because $c\Delta t$ is less than R in the denominators, the integral of the first term is larger than the integral of the second term. This happens, for instance, when the pulse first reaches a patch. To make sure

that we do not neglect an important contribution, we keep this term in all self-patch corrections. The importance of this term complicates the implementation of a similar solution for an integral equation of the first kind.

We include terms with spatial derivatives of the surface current density, which are often ignored or assumed to be negligible by requiring that the fields vary slowly over the scatterer [15]. We also have to take into account the spatial variation of the tangential unit vectors.

The resulting expression for the self patch correction is

$$\Delta\vec{J}_s \approx \begin{cases} \frac{a}{4\pi}\left[a\left(C_2 - \sin^2\theta C_3\right)\left(-J_\theta\hat{\theta} + J_\phi\hat{\phi}\right)\right. \\ \left. + \frac{1}{c}\left(C_7 - \sin^2\theta C_8\right)\left(-\frac{\partial J_\theta}{\partial t}\hat{\theta} + \frac{\partial J_\phi}{\partial t}\hat{\phi}\right)\right], \quad \theta \neq 0,\ \pi, \\ \\ \frac{\Delta\theta}{4}\left[\vec{J}_s \pm a\left(\frac{\partial J_{s3}}{\partial x}\hat{i} + \frac{\partial J_{s3}}{\partial y}\hat{j}\right)\right] \\ + \frac{a(\Delta\theta)^2}{16c}\left\{\frac{\partial\vec{J}_s}{\partial t} \pm a\left[\frac{\partial}{\partial x}\left(\frac{\partial J_{s3}}{\partial t}\right)\hat{i} + \frac{\partial}{\partial y}\left(\frac{\partial J_{s3}}{\partial t}\right)\hat{j}\right]\right\}, \quad \theta = 0,\ \pi, \end{cases} \tag{26}$$

where the coefficients are given in [5] or [7], $\hat{\theta}$ and $\hat{\phi}$ are the

tangential unit vectors along the coordinate curves, and $\Delta\theta$ is the angular width of a strip. To compute the corrections, we need to know the fields and their time derivatives at the self-patch, where the time retardation vanishes. We determine their values from the uncorrected results. If the integral of the term proportional to $\partial\vec{J}_s/\partial t$ is neglected, the correction is proportional to $\vec{J}_s$ and it can be incorporated in the coefficient of the term on the left side of the equation.

Derivatives with respect to ϕ are calculated by approximating them by a quotient of differences of the respective quantities for the patches on both sides. Derivatives with respect to θ are calculated in a similar manner. If the values of the functions at the same ϕ on the strips above and below are not available, they have to be obtained by interpolation. There is an additional problem when one of the neighboring strips is a polar cap because the θ- and ϕ-components of a tangential vector are not well defined there; we define the θ-direction tangent to the meridian that passes through the center of the self-patch, and the corresponding ϕ-direction from $\hat{\phi} = \hat{n} \times \hat{\theta}$. At the poles, the derivatives with respect to x and y are obtained from the values on both sides on the neighboring strip.

Since there is only one self-patch contribution to the integral, the effect is hardly noticeable.

5. Neighboring-Patch Corrections

The integral over patches other than the self-patch is first computed by multiplying the value of the integrand at the center

of the patch by the surface of the patch. This is a reasonable approach if the integrand varies slowly over the patch. This is not so for factors like $1/R$ or $1/R^2$ where R is small, that is, for patches near the self-patch. Neighboring patches are those that are closer than a small multiple (set in subroutine RUNT) of the patch size to the center of the self-patch. We expand the contributions from the neighboring patches in the parameters that are of the order of magnitude of the size of the patch, which here includes the displacements between the centers of the patches [6].

As we explained in the previous section, the contribution of the time-derivative term is not negligible when the current density at the previous time is zero or small, and we also have to expand this term [7].

We have to compute $\hat{n} \times \vec{I}$, where

$$\vec{I} = \int_{S_2} dS' \, \vec{R} \times \left[\frac{1}{R^2 c} \frac{\partial \vec{J}_s(\vec{x}', \tau)}{\partial t'} + \frac{1}{R^3} \vec{J}_s(\vec{x}', \tau) \right], \qquad (27)$$

and S_2 is a neighboring patch. Since $\hat{n}$ is the unit normal at the center of the self-patch S_1, it is not perpendicular to the tangential vectors $\hat{\theta}'$ and $\hat{\phi}'$. If we expand $\hat{n}$ in terms of the vectors related to the patch S_2, the expansion is greatly simplified but the resulting contribution to $\vec{J}_s$ is not tangential. Also the change in $\hat{n}$ between neighboring patches is not small when these patches are on a strip near a pole of the sphere. Consequently, we expand $\vec{I}$ itself instead of $\hat{n} \times \vec{I}$. We

obtain

$$\vec{I} \approx \left\{\left[(-1 + \cot\theta'_0\Delta\theta_0)C(1,0,3) - \frac{1}{a}\cot\theta'_0 C(2,0,3)\right.\right.$$

$$\left. - 3\cot\theta'_0\Delta\theta_0 C(1,2,5) + \frac{3}{2a}\cot\theta'_0 C(2,2,5)\right]\hat{\theta}'$$

$$+ \left[(-1 + 2\cot\theta'_0\Delta\theta_0)C(0,1,3) - \frac{2}{a}\cot\theta'_0 C(1,1,3) + \cos\theta'_0\Delta\phi_0 C(1,0,3)\right.$$

$$\left. - 3\cot\theta'_0\Delta\theta_0 C(0,3,5) + \frac{3}{2a}\cot\theta'_0 C(1,3,5)\right]\hat{\phi}' + \left[\frac{1}{2a}C(2,0,3)\right.$$

$$\left. - \Delta\theta_0 C(1,0,3)\right]\hat{n}'_0 + \left[\frac{1}{2a}\csc\theta'_0 C(0,2,3) - \Delta\phi_0 C(0,1,3)\right]\hat{\rho}'\Big\} \times \vec{J}_s$$

$$+ \left\{\left[\Delta\theta_0 C(1,0,3) - \frac{1}{a}C(2,0,3)\right]\hat{\theta}' + \left[\Delta\theta_0 C(0,1,3)\right.\right.$$

$$\left. - \frac{1}{a}C(1,1,3)\right]\hat{\phi}'\Big\} \times \frac{\partial \vec{J}_s}{\partial\theta'} + \left\{\left[\Delta\phi_0 C(1,0,3) - \frac{1}{a}\csc\theta'_0 C(1,1,3)\right]\hat{\theta}'\right.$$

$$\left. + \left[\sin\theta'_0\Delta\phi_0 C(0,1,3) - \frac{1}{a}C(0,2,3)\right]\hat{\phi}'\right\} \times \csc\theta'_0 \frac{\partial \vec{J}_s}{\partial\phi'}$$

$$+ \left\{\left[(-1 + \cot\theta'_0\Delta\theta_0)C(1,0,2) - \frac{1}{a}\cot\theta'_0 C(2,0,2)\right.\right.$$

$$\left. - 2\cot\theta'_0\Delta\theta_0 C(1,2,4) + \frac{1}{a}\cot\theta'_0 C(2,2,4)\right]\hat{\theta}'$$

$$+ \left[(-1 + 2\cot\theta'_0\Delta\theta_0)C(0,1,2) - \frac{2}{a}\cot\theta'_0 C(1,1,2) + \cos\theta'_0\Delta\phi_0 C(1,0,2)\right.$$

$$\left. - 2\cot\theta'_0\Delta\theta_0 C(0,3,4) + \frac{1}{a}\cot\theta'_0 C(1,3,4)\right]\hat{\phi}' + \left[\frac{1}{2a}C(2,0,2)\right.$$

$$\left. - \Delta\theta_0 C(1,0,2)\right]\hat{n}'_0 + \left[\frac{1}{2a}\csc\theta'_0 C(0,2,2) - \Delta\phi_0 C(0,1,2)\right]\hat{\rho}'\Big\} \times \frac{1}{c}\frac{\partial \vec{J}_s}{\partial t'}$$

$$+ \left\{\left[\Delta\theta_0 C(1,0,2) - \frac{1}{a}C(2,0,2)\right]\hat{\theta}' + \left[\Delta\theta_0 C(0,1,2)\right.\right.$$

$$\left. - \frac{1}{a}C(1,1,2)\right]\hat{\phi}'\Big\} \times \frac{1}{c}\frac{\partial}{\partial\theta'}\frac{\partial \vec{J}_s}{\partial t'} + \left\{\left[\Delta\phi_0 C(1,0,2) - \frac{1}{a}\csc\theta'_0 C(1,1,2)\right]\hat{\theta}'\right.$$

$$+ \left.\left[\sin\theta'_0\Delta\phi_0 C(0,1,2) - \frac{1}{a}C(0,2,2)\right]\hat{\phi}'\right\} \times \frac{\csc\theta'_0}{c}\frac{\partial}{\partial\phi'}\frac{\partial \vec{J}_s}{\partial t'}, \qquad (28)$$

where $\hat{\theta}'$ and $\hat{\phi}'$ are the tangential unit vectors at $\vec{x}'_0$, the center

of S_2, $\hat{\rho}'$ is the radial unit vector in the xy-plane, and the current density and its derivatives are evaluated at $\vec{x}'_0$ at the corresponding retarde time τ_0. The derivatives with respect to θ or ϕ take into account the dependence of the retarded time on θ and ϕ. The coefficients are given in [6] and [7].

Some of the coefficients become undefined when one of the sides of the neighboring patch has the same coordinate as the center of the sef-patch [7]. We then have to evaluate these coefficients from alternative expressions. Also the azimuthal angles of the first and the last patches on a strip differ by an amount close to 2π; we have to subtract 2π from such a difference to get the proper value of $\Delta\phi_0$ for the calculations.

If the neighboring patch S_2 is the spherical cap at $\theta = 0, \pi$, we use Cartesian coordinates and obtain

$$\vec{I} \approx \left\{ \vec{\rho}_0 C'(0,0,3) - \hat{i}C'(1,0,3) - \hat{j}C'(0,1,3) \right.$$

$$\left. \mp \frac{1}{2a}\hat{k}\left[\rho_0^2 C'(0,0,3) - C'(2,0,3) - C'(0,2,3)\right]\right\} \times \vec{J}_s$$

$$+ \left[\vec{\rho}_0 C'(1,0,3) - \hat{i}C'(2,0,3) - \hat{j}C'(1,1,3)\right] \times \partial\vec{J}_s/\partial x'$$

$$+ \left[\vec{\rho}_0 C'(0,1,3) - \hat{i}C'(1,1,3) - \hat{j}C'(0,2,3)\right] \times \partial\vec{J}_s/\partial y'$$

$$+ \frac{1}{c}\left\{\vec{\rho}_0 C'(0,0,2) - \hat{i}C'(1,0,2) - \hat{j}C'(0,1,2)\right.$$

$$\left. \mp \frac{1}{2a}\hat{k}\left[\rho_0^2 C'(0,0,2) - C'(2,0,2) - C'(0,2,2)\right]\right\} \times \partial \vec{J}_s / \partial t'$$

$$+ \frac{1}{c}\left[\vec{\rho}_0 C'(1,0,2) - \hat{i}C'(2,0,2) - \hat{j}C'(1,1,2)\right] \times \frac{\partial}{\partial x'} \partial \vec{J}_s / \partial t'$$

$$+ \frac{1}{c}\left[\vec{\rho}_0 C'(0,1,2) - \hat{i}C'(1,1,2) - \hat{j}C'(0,2,2)\right] \times \frac{\partial}{\partial y'} \partial \vec{J}_s / \partial t', \quad (29)$$

where

$$\rho_0 = a\sin\theta \ , \ \vec{\rho}_0 = \rho_0(\hat{i}\cos\phi_0 + \hat{j}\sin\phi_0), \quad (30)$$

and the coefficients are defined in [7]. The computation of these coefficient involves a numerical integration [7], for which we use the subprogram Q1DB in CMLIB [16]. The integrand becomes undefined for certain values of the variable of integration, and an alternative expression has to be used where this happens.

The corrected neighboring-patch contributions replace those obtained in the simple way. We have compared the two computed values of the neighboring-patch contributions and we have found that sometimes they differ little while at other times the difference can be large. Since a patch can have about ten neighboring patches, the overall effect of this correction is more significant than that of the self-patch.

6. Vector Programming Considerations

The largest savings in execution time can be achieved by vectorizing the computations in the main loop.

The values of the components of $\vec{J}_s$ that may be needed for the integration are kept in memory in two-dimensional arrays, where one index represents the patch number and the other the time. After these fields are no longer needed, they are saved on a disk file.

An important observation that facilitates the vectorization of the interpolation in the time variable is that the location of the retarded fields in the two-dimensional array for a fixed field point is the same for the different components of the fields and is shifted by one column as the time is incremented for the interpolation. Thus the array IND is computed once and used in a Q8VGATHR function repeatedly. This function is used to collect the values of the components of $\vec{J}_s$ at the different retarded time in a single array. We originally computed the values stored in the array IND and the related array FT once for each field point and saved them, to be used at each time. This procedure requires square arrays for IND and FT, increasing the memory requirements. Although the CYBER 205 has virtual memory, paging generates much activity and slows significantly the execution of the program. The arrays that contain the values of the surface current density components are accessed in an irregular manner and they have to be in memory at the time of execution. The arrays IND and FT may be paged in gradually as the field point changes, but we have chosen to recompute their

values for each time point with only a small increase in the running time. We save square bit arrays BT3 and BT4 to control where the interpolation has to be shifted from the center of the interval.

The vectorization of this main loop produces a reduction of the execution time by a factor of 7 if the sphere is divided into 70 patches, and 20 for 550 patches. These factors can probably be reduced somewhat by improving the scalar version of the program, where we use the three Cartesian components of the fields instead of the two tangential components we use in the vector program. The vectorization has to be done explicitly to be able to take advantage of the Q8VGATHR function.

The subroutine RUNT is called only once at the beginning of the program, and we made no great effort to vectorize this part of the code. The subroutine FARFL can be called repeatedly to find the scattered fields in different directions, and its structure is not very different from the main loop since we are also doing surface integrals over retarded fields.

The computation of the self-patch correction takes only an additional one percent for execution. The additional execution time for the neighboring-patch corrections depends on the number of neighboring patches, which in turn depends on the criteria used to define them and the patch distribution on the sphere. It varies between ten and 40 percent of the basic execution time for the loop. It may be possible to decrease this time by vectorizing the computation of the neighboring-patch contributions.

7. Ancillary Programs

We have developed two programs that we use to examine the results of the main programs PERF and SCAT. They are PERFPLT, and CURCHK. These programs require very little execution time, but they run on the CYBER 205 because that is where the output files are stored.

We have written a program SCATP that runs on the front-end computer to determine the required dimensions of the different arrays in SCAT and the number of large pages needed to accomodate these arrays.

We have also written vector versions of subroutines NUFT and INUFT, which we use to compute the direct and inverse Fourier transform of functions that are not given at equispaced points.

7.1 Program PERFPLT

Program PERFPLT uses the output files PERFSV of PERF and SCFLxxx of SCAT, where xxx stands for the qualifier that identifies the run, and allows us to plot the far-field intensities on a single plot for each outgoing direction of propagation. We thus can compare the plotted output of PERF with up to 4 plots produced by SCAT. We can choose printer plots by using DRAW3 [10] or plots that can be produced on a Versatec printer/plotter by using DRAW4 [11] adapted to FORTRAN 77. Actually, the information for the plots is added to a file stored on the front-end computer. Then, when several plots have accumulated, this information is transfered to tape which is

taken to our local PE 3230 minicomputer for plotting.

We can select the initial and final times on the plots to show details of particular regions of the pulses.

7.2 Program CURCHK

Program CURCHK selects the values of the components of $\vec{J}_s$ saved in a file SCSVxxx for up to 10 patches for the whole time range or a given interval. These values can be printed or they can be plotted on the printer or on a plotter and they provide a useful check on the solution of the integral equation.

7.3 Program SCATP

We run this program on the CYBER 180/855, the front end, and then use the editor to pass the information to the parameter statements in SCAT and CURCHK. The class and priority of a job on the CYBER 205 depends on the memory and time requirements, and it is better to keep them as small as possible.

Program SCATP does the part of the calculations that will be made in SCAT to find the number of patches for the given n_1, n_2, and n_3, estimates the number of time steps that have to be kept in memory, and the number and distribution of neighboring patches if this correction will be made.

A discrepancy arose in these calculations due to the differences between the CYBER 170/855 and the CYBER 205. The former returned the result $\cos(\pi/6) = .4999...$ while the latter

returned $\cos(\pi/6) = .5$; rounding led to a different number of patches and an insufficient size for the arrays; this is why the size is incremented by 2 in SCATP.

7.4 Subroutine VINUFT

This subroutine is used by program PERF to compute the inverse Fourier transform of the scattered fields. It is related to subroutines NUFT and INUFT [9].

The subroutines NUFT can be used to compute the Fourier transform of a function $f(t)$, $a \leq t \leq b$ when the values of $f(t)$ are known at a set of values t_i that are not equispaced. The function is approximated by straight line segments joining adjacent points, the Fourier transform of this approximation is determined analytically, and the result is evaluated numerically at a given set of values ω_i.

Subroutine NUFT can be used twice to obtain the inverse Fourier transform in the analogous approximation, but we can do this more efficiently by using subroutine INUFT if we know that $f(t)$ is real, that is, when

$$\tilde{f}(-\omega) = \tilde{f}^*(\omega). \qquad (31)$$

Subroutines VNUFT and VINUFT are the vectorized versions of NUFT and INUFT, and they are listed in the appendix. The vector versions do not have the small angle approximation for the sine function used in the scalar versions.

8. Program Inputs and Outputs.

The inputs required to run PERF and SCAT are contained in the files PERFIN and SCINxx (filenames on the front end can have 7 characters only, and we add another digit to indicate the type of correction selected). The procedure used to submit a run to the computer causes a file SCINxxx to be saved, so that the input file SCINxx may be modified.

The information that is required for a given run relates both to the physical problem and to the numerical computation.

The constants α and β determine the shape of the pulse, and we also need the radius $\underline{a}$ of the sphere. Actually, the value of $\underline{a}$ simply fixes the scale of the problem, and we could set $\underline{a} = 1$ everywhere in the program. The outputs for $\underline{a} = 1$, α, β, and t are the same as those for $\underline{a}$, α/a, β/a, and $\underline{a}t$.

We have to specify the angles θ and ϕ of the direction of propagation of the scattered fields, and we may give times t_1 and t_2 to obtain plots of some part of the scattered fields. Several sets of directions and times may be specified, and all the calculations use the same previously determined surface currents. If t_1 or t_2 is not given, we make an estimate of the extreme values based on the time of arrival of the pulse and the time covered by the calculation of $\vec{J}_s$. Further sets of fields may be computed in separate runs without recomputing the surface current density, which is saved in SCSVxxx.

The parameters needed for the computation of the fields in SCAT are the time increment Δt and the patch distribution on the sphere. The distance light travels in the time Δt has to be

smaller than the separation between the centers of any two patches; the program SCAT makes this Δt equal to 0.8 times the smallest separation between the centers if none is given or if the given Δt is larger than this quantity. We can also give the number of time increments in the calculation, which determines the time interval for which the surface current density is determined; if this number is not given, the program selects a number that corresponds to propagation over a distance $6\underline{a}$. The division of the sphere into patches is controlled by the number of strips, n_1, the approximate number of patches on the equatorial strip, n_2, and the number of patches on the strip next to a polar cap, n_3, as discussed in section 3. To do the calculation is PERF we also have to give the first frequency increment, the highest frequency, and the number of frequency values where the monochromatic fields are to be calculated.

The agreement between the curves computed via the Fourier transform and by solving the MFIE is shown in figure 1. The output of SCAT in this figure corresponds to the largest run we made. We needed 28 large pages, which is essentially the size of the computer memory, the main loop executed in 10754 s with no self- or neighboring-patch corrections, and the total CPU-time for the job was 10812 s. The sphere was divided into 58 strips, and this was also the maximum specified for the number of patches in a strip (in powers of 2, the actual number goes from 8 near the poles to 64 at the equator), for a total of 2898 patches. We selected $\Delta t = 0.3 \times 10^{-10}$ s for the time interval and computed the current density for 900 time steps. There is some evidence

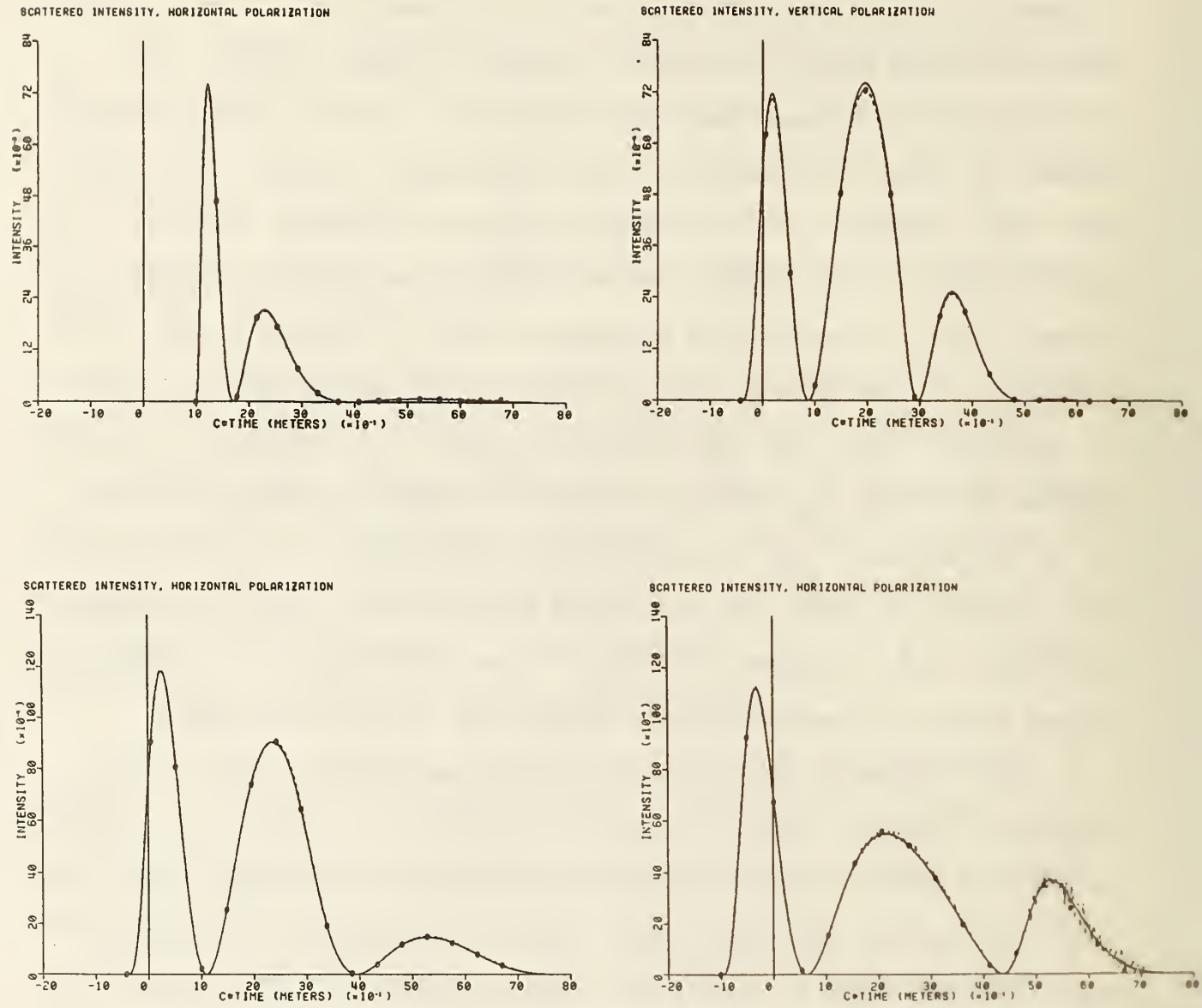


Figure 1. Intensity (in arbitrary units) of the scattered far fields as a function of ct in (a) the forward direction, (b) and (c) at 90 degrees with both polarizations, and (d) the backward direction for an incident pulse with $\alpha = 2\ m^{-1}$ and $\beta = 3\ m^{-1}$. The solid curves are computed via a Fourier transform from Mie scattering formulas and the dashed curves with the markers were computed via the MFIE. There is good agreement.

of oscillations at the later times, especially in backward scattering.

Many factors affect the accuracy of the computation of the fields from the integral equation, and these factors interact in complex ways. The two effects of the interpolation formula used for the time variable are illustrated in figure 2 for runs in which the sphere was divided into 19 strips with 8 to 32 patches in a strip (370 patches total) or 5 to 20 patches in a strip (258 patches total), and with $\Delta t = 2.75 \times 10^{-10}$ s, which is a relatively large interval. Near the first peak, the curves that correspond to the fourth-degree polynomial interpolation are much closer to the correct solution, while near the end of the graph these curves show the onset of oscillations. These oscillations tend to become larger as time progresses, limiting the usefulness of this interpolation scheme. We also see in figure 2 that the particular way in which the strips are divided into patches, whether we use powers of 2 or not, has little effect on the final results.

In figure 3 we show a details of the curves about the first peak to illustrate the effect of the self- and neighboring-patch corrections. The sphere was divided into 1986 patches, and the time increment was 0.5×10^{-10} s. We see that the effect of including the contribution of the self-patch is slight, while that of correcting the contributions of the neighboring patches is more significant.

The periodicity of the surface fields as functions of the azimuthal angle ϕ has the effect of reducing the number of patches that are required in a strip [7]. The trapezoidal rule

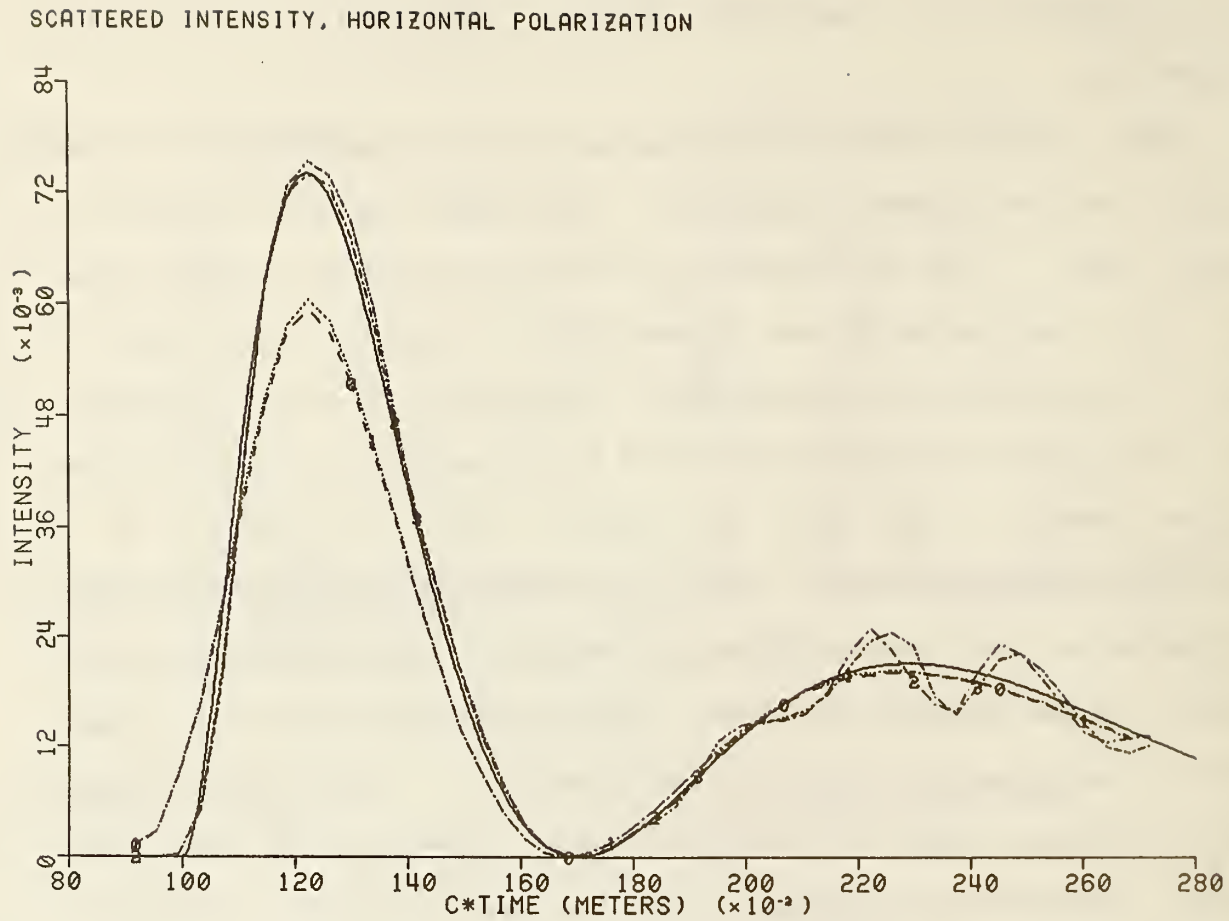


Figure 2. Illustration of the effects of the subdivision of the strips and the interpolation scheme. The curves marked with 0 and 2 correspond to patch numbers that are powers of 2 and curves 0 and 1 are computed with least-squares interpolation. The curves were cut off at ct = 2.8 m.

gives a much more accurate integral for periodic than for arbitrary functions when the integration is over one period. Thus, if the sphere is divided into n_1 strips, it is better to divide the equatorial strip into n_1 patches instead of the $2n_1$ required to make the patches more or less square. The smallness of the effect of this or even a greater reduction is shown in figure 4. The increase in the size of the patch has an adverse effect in the computation of the self- and neighboring-patch

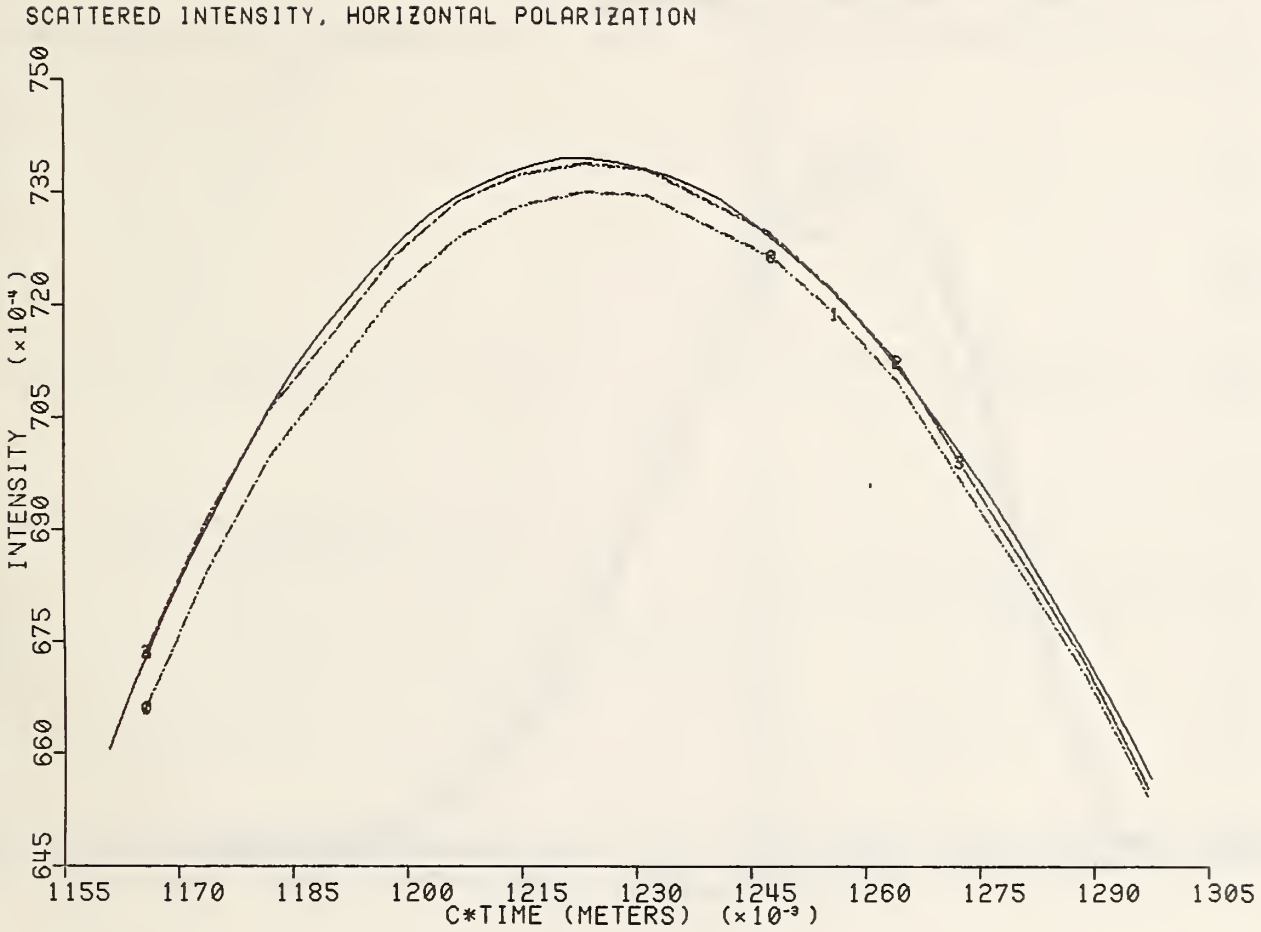


Figure 3. Effect of self-patch and neighboring-patch corrections. We show a detail of the curves near the first maximum. Curves 2 and 3, computed with neighboring-patch corrections, agree better with the solid curve than 0 and 1. The inclusion of self-patch corrections in the computation of 1 and 3 does not lead to a noticeable improvement over 0 and 2, respectively.

corrections, which is also illustrated in figure 4.

We show the components of the current density on the top cap in figure 5a; the x-component remains essentially equal to zero, as expected by symmetry. The computed surface current density remains exactly equal to zero until the incident wave reaches the patch; after that time, the current density is determined by the integral equation. On the other hand, the actual fields cannot

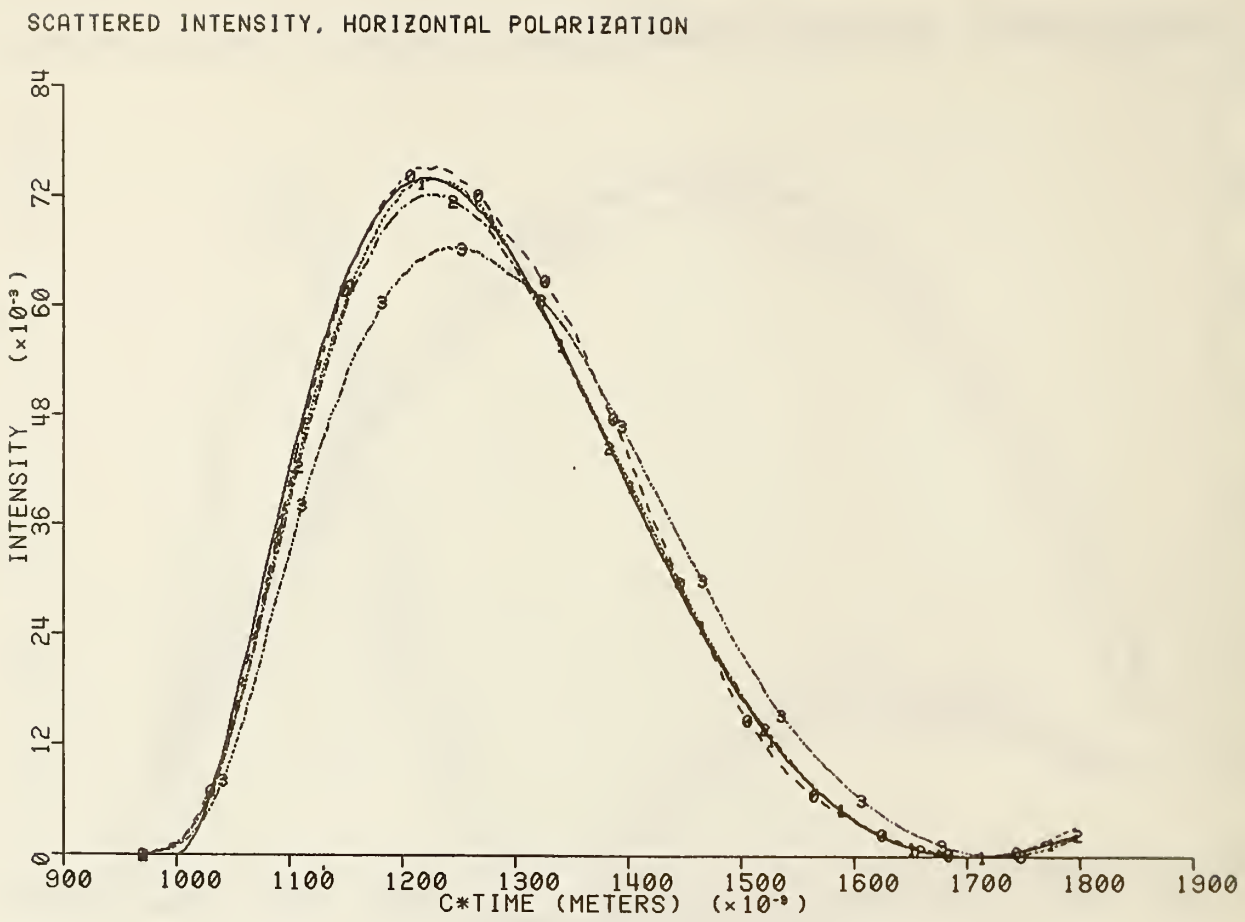


Figure 4. Illustration of the small effect of a variation in the number of patches per strip. The sphere is divided into 29 strips, and the number of patches in the equatorial strip is 15 for curve 0, 29 for curve 1, and 45 for curve 2. Curve 3 includes corrections with the same patch distribution used for curve 0, showing a deterioration of the results.

reach a patch until they go around the outside of the perfectly conducting sphere. This means that the solution of the integral equation at the top cap of the sphere has to remain zero between $ct = 2a$ and $ct = (1 + \pi)a$, an effect that is illustrated in figures 5b to 5f. The discrepancies shown for the different calculations are an indication of the accuracy that was attained in the solution of the integral equation.

The fields computed for the program-selected value of Δt are

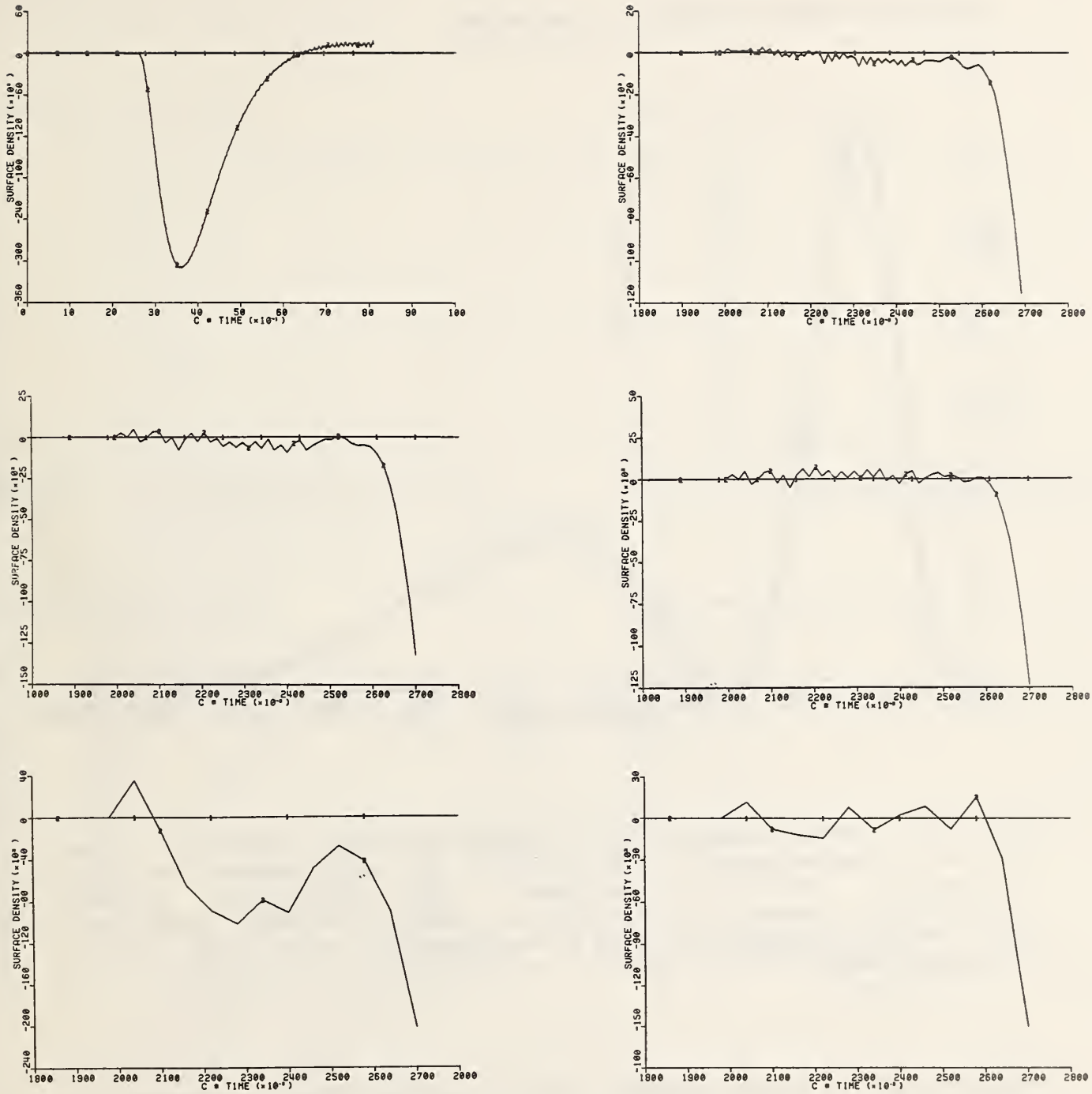


Figure 5. Components of the current density (in arbitrary units) at the top pole of the sphere; the x-component remains zero. We show the complete graph (a) in the run with 2898 patches with least-squares interpolation, and a detail of the region between ct = 2 m and ct = 1 + π/2 m for (b) the same run, (c) one with 1586 patches, (d) same with corrections, (e) one with 370 patches, and (f) 370 patches with polynomial interpolation.

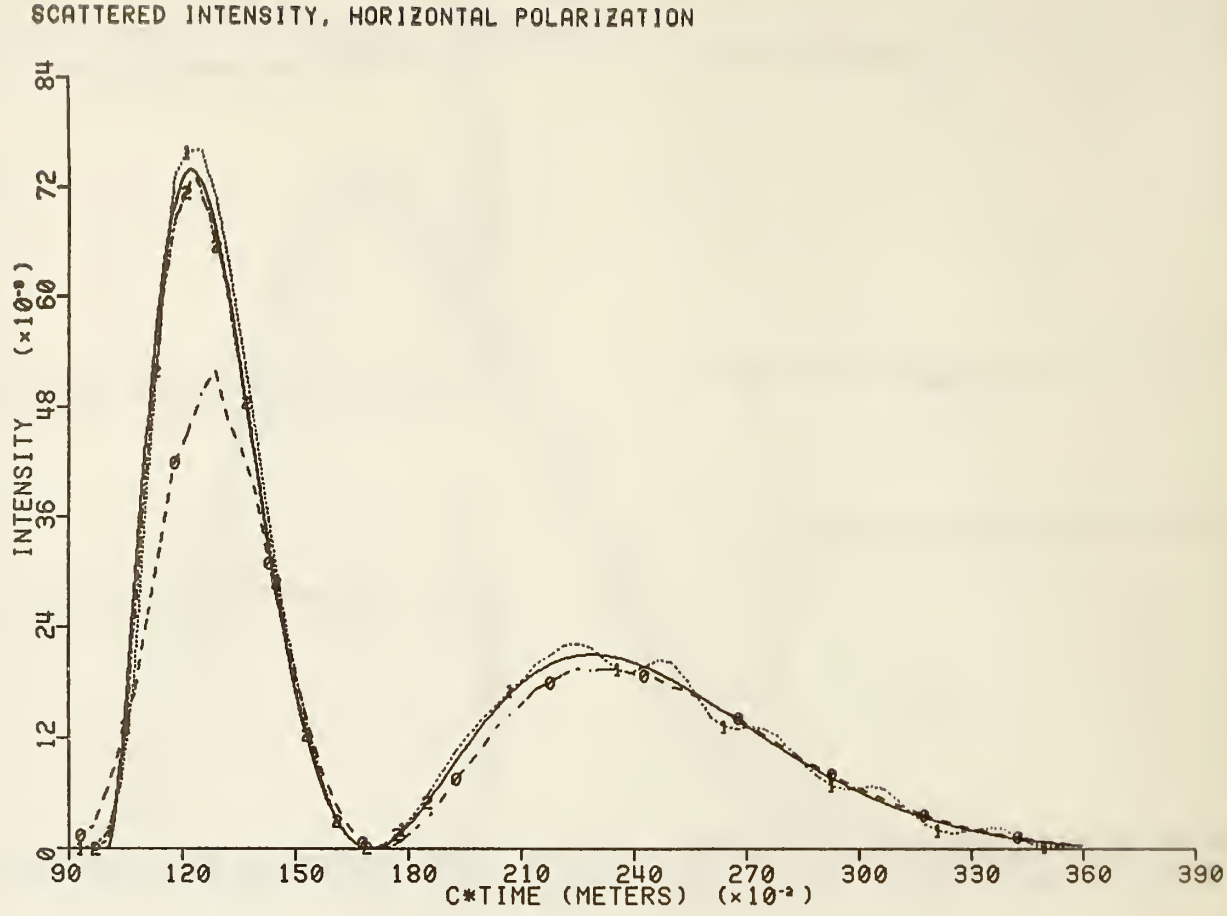
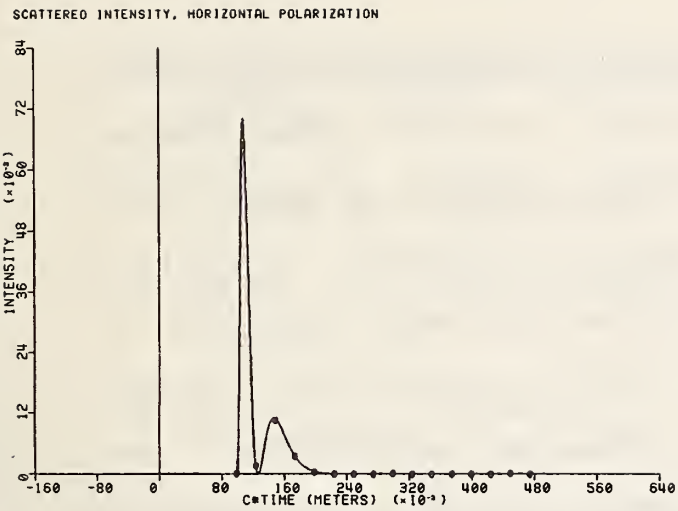
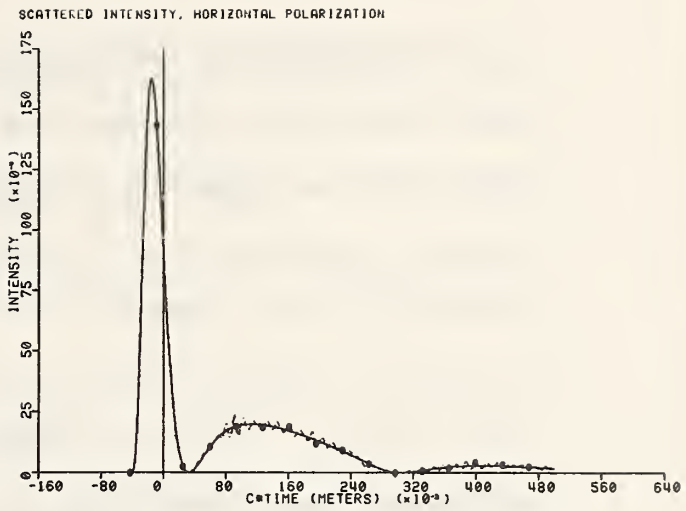


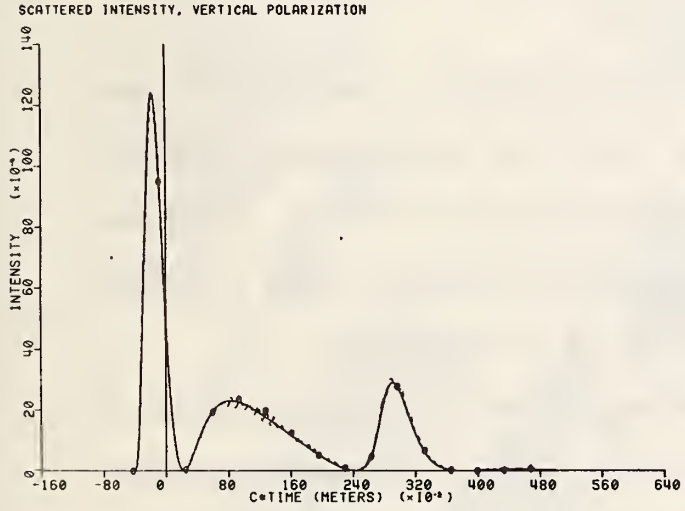
Figure 6. Fields computed with program-selected (minimum) values of Δt (approximately 3.5×10^{-10} s) and number of time increments with 370 patches. The computation without corrections shown in curve 0 is compared to the same with polynomial interpolation and corrections in curve 1 and with a time increment of $1. \times 10^{-10}$ s in curve 2.

compared with those computed with a smaller time increment in figure 6. The curves show that it is important to use the more accurate polynomial interpolation for a small number of patches.

The scattered fields for a sharper incident pulse, which has constants $\alpha = 6\ m^{-1}$, $\beta = 9\ m^{-1}$, are shown in figure 7. The fit at the peaks is not very good and there are oscillations at later times. Smaller patches are required for a sharper pulse.







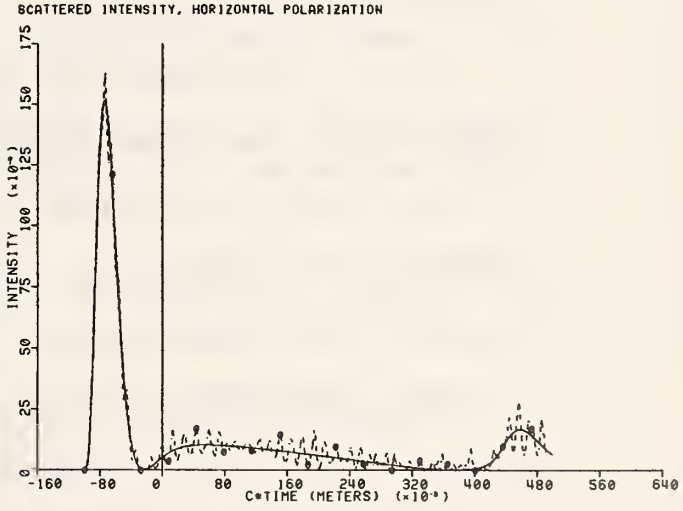


Figure 7. Intensity of the scattered far fields in (a) the forward direction, (b) and (c) at 90 degrees with both polarizations, and (d) in the backward direction for an incident pulse with $\alpha = 6\ m^{-1}$ and $\beta = 9\ m^{-1}$. The graphs, which correspond to runs with 1586 patches, show an increase in the oscillations after the initial peak.

9. Conclusions

The agreement between the results of programs PERF and SCAT shows that both programs are essentially correct. Thus, program SCAT can be used as the basis for more general programs that compute scattering of pulses by arbitrary perfectly conducting surfaces. The parts of the program that would have to be modified are mainly those related to the division of the surface into patches and the computation of spatial derivatives. The scattering of spatially localized pulses can also be included in the program by making the appropriate changes in the computation of the incident fields.

The importance of the time-derivative term in the computation of the contribution of the self-patch causes difficulties in the use of a similar procedure to solve integral equations of the first kind, such as the EFIE or the integral equation that describes the scattering by a dielectric.

The correction for the neighboring-patch contributions has a greater effect than inclusion of the self-patch, probably because there are a number of neighboring patches for each patch. On the other hand, the uncorrected integrals also gave satisfactory solutions with a sufficiently large number of patches.

The vectorization of the program reduces the running time by a factor that depends on the number of patches on the sphere. This factor is approximately equal to 20 for 550 patches. We also found that straight polynomial interpolation gave better accuracy than a least-squares polynomial interpolation at the cost of increased instability.

References

[1] Marx, E. Integral Equations for Transient Electromagnetic Fields, National Bureau of Standards Technical Note 1157, February 1982.

[2] Marx, E. Single Integral Equation for Wave Scattering, Journal of Mathematical Physics 23, 1057 (1982).

[3] Marx, E. Integral Equation for Scattering by a Dielectric, IEEE Transactions on Antennas and Propagation AP-32, 166 (1984).

[4] Marx, E. Transient Fields in Dispersive Media, Journal of Mathematical Physics 24, 2602 (1983).

[5] Marx, E. Self-Patch Integrals in Transient Electromagnetic Scattering, IEEE Transactions on Antennas and Propagation AP-32, 763 (1985).

[6] Marx, E. Neighboring-Patch Integrals in Transient Electromagnetic Scattering, IEEE Transactions on Antennas and Propagation AP-32, 768 (1985).

[7] Marx, E. Electromagnetic Pulse Scattered by a Sphere, IEEE Transactions on Antennas and Propagation, submitted for publication.

[8] Sookne, D. J. Bessel Functions of Real Argument and Integer Order, Journal of Research of the National Bureau of Standards, 77B, 125 (1973).

[9] Brandstein, A. G. and Marx, E. Numerical Fourier Transform, Harry Diamond Laboratories Technical Report 1748 September 1976.

[10] Marx, E. Printer Version of Plots Made by an Incremental Plotter, Harry Diamond Laboratories Technical Manual 75-33, December 1975.

[11] Noon, T. V. User's Manual for Modular Analysis-Package Libraries ANAPAC and TRANL, Harry Diamond Laboratories Technical Report 1782, November 1976.

[12] Hildebrand, F. B. Introduction to Numerical Analysis, p. 290 (McGraw-Hill, New York, 1956).

[13] Mieras, H. Time-Domain Scattering from Dielectric Solids, Sperry research Center, Sudbury, MA, June 1980.

[14] Marx, E. Free-Space Propagation of Light Pulses, National Bureau of Standards IR 84-2835, May 1984.

[15] Mieras, H. and Bennett, C. L. Space-Time Integral Approach to Dielectric Targets, IEEE Transactions on Antennas and Propagation AP-30, 2 (1982).

[16] Boisvert, R. F., Howe, S. E., and Kahaner, D. K. Guide to Available Mathematical Software, National Bureau of Standards IR 84-2824, January 1984.

Appendix

In this appendix we provide listings for the programs SCAT, SCATP, PERF, PERFPLT, and CURCHK together with most of the subprograms that they require. We also provide listings for the Fourier transform subroutines VNUFT and VINUFT.

The plotting subroutines DRAW3 and DRAW4 are documented elsewhere, as are the integration subroutine Q1DB, the interval generating subroutine OMEGA, and the subroutines to compute Bessel functions BESRJ and BESRY.

```
      PROGRAM                    SCAT
C
C     VECTORIZED VERSION OF PROGRAM THAT COMPUTES SCATTERED FIELDS.
C     SELF-PATCH AND NEIGHBORING-PATCH CONTRIBUTIONS CAN BE INCLUDED.
C     NO CHARGE DENSITY IS COMPUTED
C
C     XR1, XR2, XR3: ARRAYS WITH THE COMPONENTS OF THE POSITION VECTOR
C                    ON THE SPHERE
C     SRF: (SOLID ANGLE SUBTENDED BY EACH PATCH)/(TWO PI)
C     TF: TIME OF FIRST ARRIVAL OF THE WAVE AT EACH PATCH
C     CO: COEFFICIENT FOR CONTRIBUTION TO SELF-PATCH
C     DO: SAME AS CO, FOR THE TIME-DERIVATIVE TERM
C     TH1, TH2, TH3: COMPONENTS OF THE UNIT VECTOR ALONG MERIDIANS
C     PH1, PH2: COMPONENTS OF UNIT VECTOR ALONG PARALLELS
C     FU1, FU2, FD1, FD2, FC1, FC2: INFORMATION FOR INTERPOLATION
C     THETA: POLAR ANGLE OF PATCH
C     PHII: AZIMUTHAL ANGLE OF PATCH
C     DPHII: INCREMENT OF AZIMUTHAL ANGLE
C     RB1, RB2, RB3, RB4: ARRAYS THAT CONTAIN BIT ARRAYS
C     COJ, COJT, COJU, COJTU, COJV, COJTV: ARRAYS WITH COEFFICIENTS
C           NEEDED TO COMPUTE NEIGHBORING-PATCH CORRECTIONS
C     SINT1, SINT2, SINT3: CARTESIAN COMPONENTS OF INTEGRALS
C     DR1, DR2, DR3: COMPONENTS OF R = X - X'
C     DR, CDRM2, DRM3: WORK ARRAYS
C     CJTHD, CJPHD: COMPONENTS OF THE RETARDED SURFACE CURRENT DENSITY
C           IN THE THETA- AND PHI-DIRECTIONS
C     DJTHD, DJPHD: SAME FOR TIME DERIVATIVE OF SURFACE CURRENT DENSITY
C     UTH, UPH, A0, A1, A2, A3, A4, YM2, YM1, Y0, Y1, Y2: WORK ARRAYS
C     CJTHA, CJPHA: COMPONENTS OF SURFACE CURRENT DENSITY AT TIME T
C     CJTH, CJPH: TWO-DIMENSIONAL ARRAYS WITH COMPONENTS OF ALREADY CALCULATED
C           CURRENT DENSITY
C     FT: INFORMATION FOR TIME INTERPOLATION
C     NL, NR, NU1, NU2, ND1, ND2: INFORMATION FOR SPATIAL INTERPOLATION
C     NEI, NPOINT: INFORMATION FOR NEIGHBORING-PATCH CORRECTION
C     IND: INDEX ARRAY FOR VECTOR PROCESSING
C     IJ: IJ(I) = I, FOR VECTOR PROCESSING
C     BT1: BIT ARRAY SELECTS ALL PATCHES EXCEPT THE SELF PATCH
C     BT2: BIT ARRAY TO INDICATE WHETHER THE PULSE HAS REACHED A PATCH
C     BT3, BT4: BIT ARRAYS TO INDICATE NECESSITY OF SHIFTING INTERPOLATION
C
      PARAMETER(LL=LLG,LC=LCG,LL12=LL12G,LL20=LL20G,
     1 LB=LL**2/64+1,LB1=LL/64+1)
      BIT BT1(LL),BT2(LL),BT3(LL,LL),BT4(LL,LL)
      CHARACTER TITLE*80,SUBT*80,XLAB*30,YLAB*30,QAL*4
      COMMON /RCOM/ XR1(LL),XR2(LL),XR3(LL),SRF(LL),TF(LL),
     1 CO(LL),DO(LL),TH1(LL),TH2(LL),TH3(LL),
     2 PH1(LL),PH2(LL),FU1(LL),FU2(LL),FD1(LL),FD2(LL),FC1(2),FC2(2),
     3 THETA(LL),PHII(LL),DPHII(LL),RB1(LB1),RB2(LB1),RB3(LB),RB4(LB),
     4 COJ(6,LL12),COJT(6,LL12),COJU(2,LL12),COJTU(2,LL12),
     5 COJV(2,LL12),COJTV(2,LL12),SINT1(LL),SINT2(LL),SINT3(LL),
     6 DR1(LL),DR2(LL),DR3(LL),DR(LL),CDRM2(LL),DRM3(LL),
     7 CJTHD(LL),CJPHD(LL),DJTHD(LL),DJPHD(LL),
     8 UTH(LL),UPH(LL),A0(LL),A1(LL),A2(LL),A3(LL),A4(LL),
     9 YM2(LL),YM1(LL),Y0(LL),Y1(LL),Y2(LL),
```

```
     A CJTHA(LL),CJPHA(LL),CJTH(LL,LC),CJPH(LL,LC),FT(LL),
     B NL(LL),NR(LL),NU1(LL),NU2(LL),ND1(LL),ND2(LL),
     C NEI(LL,LL20),NPOINT(LL,LL20),IND(LL),IJ(LL)
      COMMON /ICOM/ IX,IS,IS1,N1,N2,NI,ICORR,IPLOT,INTER,IDEB,
     1 IPATCH,N3,NU
      COMMON /LABEL/ TITLE,SUBT,XLAB,YLAB
      COMMON /PARAM/ ALPHA,BETA,DT,RAD,RADM1,CDTM1,THSC,SR1,SR2,SR3
      COMMON /CONST/ AMUO,TWOMUO,TWOEPO,C,CM1,PI,TOPI,PITO,OS,OTW,OTF
      EQUIVALENCE (RB1,BT1),(RB2,BT2),(RB3,BT3),(RB4,BT4)
      DATA IZO/O/,BT1/LL*B'1'/
      WRITE(XLAB,7)
      WRITE(YLAB,8)
      LL2=2*LL
      OS=1./7.
      OTW=1./12.
      OTF=1./24.
      C=3.E8
      CM1=1./C
      PITO=2.*ATAN(1.)
      PI=2.*PITO
      TOPI=4.*PITO
      CONV=PI/180.
      AMUO=PI*4.E-7
      TWOMUO=2./AMUO
      TWOEPO=2.*CM1**2/AMUO
      PRINT '(1H1)'
      READ *, ICORR,ISKIP,IPLOT,INTER,IDEB,IPATCH,QAL
      PRINT *, 'ICORR=',ICORR,',  ISKIP=',ISKIP,',  IPLOT=',IPLOT,
     1 ',  INTER=',INTER,',  IDEB=',IDEB,',  IPATCH=',IPATCH,',  QAL=',QAL
C
C     READ IN VALUES OF CONSTANTS FOR THE PULSE
C
      READ 4, DT,RAD,ALPHA,BETA
      READ 5, NI
      READ 5, N1,N2,N3
      IF(N1.GT.100) STOP 2
      IF(N3.EQ.0) N3=5
      RADM1=1./RAD
      IF(ISKIP.EQ.1) ICORR=0
C     COMPUTE COMPONENTS OF VECTORS ON SPHERE AND INFORMATION NEEDED FOR
C     SELF-PATCH AND NEIGHBORING-PATCH CORRECTIONS
      CALL RUNT
      IF(IDEB.EQ.1) THEN
        WRITE(7,*)'      I       XR1(I)       XR2(I)       XR3(I)       ',
     1  'SRF(I)       TF(I)         CO(I)        DO(I)       ',
     2  'TH1(I)       TH2(I)        TH3(I)'
        WRITE(7,*)'  PH1(I)       PH2(I)       FU1(I)',
     1  '       FU2(I)       FD1(I)       FD2(I)'
        WRITE(7,*)'  NL(I) NR(I) NU1(I) NU2(I) ND1(I) ND2(I)'
        DO 90 I=1,IS
          WRITE(7,2) I,XR1(I),XR2(I),XR3(I),SRF(I),TF(I),
     1    CO(I),DO(I),TH1(I),TH2(I),
     2    TH3(I),PH1(I),PH2(I),FU1(I),FU2(I),FD1(I),FD2(I)
          WRITE(7,1) NL(I),NR(I),NU1(I),NU2(I),ND1(I),ND2(I)
```

```
 90       CONTINUE
          IF(ICORR.GE.2) THEN
            DO 95 I=1,IS
              WRITE(7,1) I,(NEI(I,J),J=1,NEI(I,1))
 95         CONTINUE
          END IF
        END IF
C       NO CALCULATIONS IF NI < 0
        IF(NI.LT.0) STOP
        DO 100 I=1,IS
          IJ(I)=I
 100    CONTINUE
        DTM1=1./DT
        CDTM1=CM1*DTM1
C       COMPUTE THE NUMBER OF TIME STEPS TO A DIAMETER TO DIMENSION ARRAY
        NU=2.*RAD*CM1*DTM1+6.
C       COMPUTE THE NUMBER OF TIME STEPS IF NOT GIVEN
        IF(NI.EQ.0) NI=6.*RAD/(C*DT)
        PRINT *, 'NI=',NI,', N1=',N1,', N2=',N2,', N3=',N3,
       1 ', DT=',DT,', RAD=',RAD,', ALPHA=',ALPHA,', BETA=',BETA
        PRINT *, NU,' ROWS NEEDED OUT OF ',LC
        IF(NU.GT.LC) STOP 3
        NUM2=NU-2
        IF(ICORR.EQ.1) THEN
          PRINT *, 'SELF-PATCH CORRECTIONS ONLY'
        ELSE IF(ICORR.EQ.2) THEN
          PRINT *, 'NEIGHBORING-PATCH CORRECTIONS ONLY'
        ELSE IF(ICORR.EQ.3) THEN
          PRINT *, 'SELF- AND NEIGHBORING-PATCH CORRECTIONS'
        ELSE
          PRINT *, 'NO PATCH CORRECTIONS'
        END IF
C       ENCODE INFORMATION FOR TITLES OF GRAPHS
        IF(ICORR.EQ.1) THEN
          IF(INTER.EQ.0) THEN
            WRITE(SUBT,11) ALPHA,BETA,DT,NI,N1,N2,N3
          ELSE
            WRITE(SUBT,12) ALPHA,BETA,DT,NI,N1,N2,N3
          END IF
        ELSE IF(ICORR.EQ.2) THEN
          IF(INTER.EQ.0) THEN
            WRITE(SUBT,13) ALPHA,BETA,DT,NI,N1,N2,N3
          ELSE
            WRITE(SUBT,14) ALPHA,BETA,DT,NI,N1,N2,N3
          END IF
        ELSE IF(ICORR.EQ.3) THEN
          IF(INTER.EQ.0) THEN
            WRITE(SUBT,15) ALPHA,BETA,DT,NI,N1,N2,N3
          ELSE
            WRITE(SUBT,16) ALPHA,BETA,DT,NI,N1,N2,N3
          END IF
        ELSE
          IF(INTER.EQ.0) THEN
            WRITE(SUBT,9) ALPHA,BETA,DT,NI,N1,N2,N3
```

```
        ELSE
          WRITE(SUBT,10) ALPHA,BETA,DT,NI,N1,N2,N3
        END IF
      END IF
      CALL Q5RQUEST('LFN=','SCFL'//QAL,'MXR=',2400,'LEN=',50,
     1 'RT=','W')
      OPEN(9,FILE='SCFL'//QAL,FORM='UNFORMATTED')
      T=DT
      NF2=8*IS
      LENGTH=NI*IS*2/512+10
      IF(ISKIP.EQ.1) THEN
C     COMPUTE FAR FIELDS FROM SAVED FILE SCSV<QAL
        CALL Q5ATTACH('LFN=','SCSV'//QAL)
        OPEN(2,FILE='SCSV'//QAL,STATUS='OLD',ACCESS='DIRECT',
     1  FORM='UNFORMATTED',RECL=NF2)
        T=NI*DT
        GO TO 400
      END IF
      CALL Q5RQUEST('LFN=','SCSV'//QAL,'MXR=',NF2,'LEN=',LENGTH,
     1 'SFO=','D')
      OPEN(2,FILE='SCSV'//QAL,FORM='UNFORMATTED',
     1 RECL=NF2,ACCESS='DIRECT')
      TL=SECOND()
      NUM=1
      NREC=1
      IXH=0
C
C     INITIALIZE ARRAYS
C
      CJTHA(1,IS)=0.
      CJPHA(1,IS)=0.
      CJTHD(1,IS)=0.
      CJPHD(1,IS)=0.
      DJTHD(1,IS)=0.
      DJPHD(1,IS)=0.
      UTH(1,IS)=0.
      UPH(1,IS)=0.
      DO 120 J=1,LC
        CJTH(1,J,IS)=0.
        CJPH(1,J,IS)=0.
  120 CONTINUE
C
C     MAIN LOOP
C
      DO 300 IZ=2,NI
        NUM=NUM+1
        PRINT *, IZ,' OF ',NI
C     SHIFT THE NUMBERS IN THE ARRAYS
        DO 170 I=2,NU
          CJTH(1,I-1,IS)=CJTH(1,I,IS)
          CJPH(1,I-1,IS)=CJPH(1,I,IS)
  170   CONTINUE
C     COMPUTE DENSITIES AT EACH PATCH ON THE SPHERE
        DO 200 IX=1,IS
```

```
C      RETRIEVE THE RADIUS VECTOR (NORMAL)
           X=XR1(IX)
           Y=XR2(IX)
           Z=XR3(IX)
C      FIND THE INCIDENT FIELDS
           CALL INFL(SINT10,SINT20,SINT30,
     1     X,Y,Z,T,*210)
           DR1(1,IS)=X-XR1(1,IS)
           DR2(1,IS)=Y-XR2(1,IS)
           DR3(1,IS)=Z-XR3(1,IS)
           CDRM2(1,IS)=DR1(1,IS)**2+DR2(1,IS)**2+DR3(1,IS)**2
           DR(1,IS)=VSQRT(CDRM2(1,IS),DR(1,IS))
           DRM3(1,IS)=CDTM1*DR(1,IS)
           IND(1,IS)=VINT(DRM3(1,IS),IND(1,IS))
           FT(1,IS)=VAINT(DRM3(1,IS),IS)-DRM3(1,IS)
C      SHIFT THE FIVE POINTS FOR INTERPOLATION IF NECESSARY
           WHERE(BT3(1,IX,IS))
             IND(1,IS)=IND(1,IS)+1
             FT(1,IS)=FT(1,IS)+1.
           END WHERE
           WHERE(BT4(1,IX,IS))
             IND(1,IS)=IND(1,IS)+2
             FT(1,IS)=FT(1,IS)+2.
           END WHERE
           IND(1,IS)=IJ(1,IS)-IND(1,IS)*LL
           BT1(IX)=B'0'
C      DETERMINE WHETHER THE PULSE HAS REACHED THE PATCH
           BT2(1,IS)=T-CM1*DR(1,IS).LE.TF(1,IS)
           WHERE(BT1(1,IS)) DRM3(1,IS)=1./DR(1,IS)
           CDRM2(1,IS)=DRM3(1,IS)**2
           DRM3(1,IS)=CDRM2(1,IS)*DRM3(1,IS)
           CDRM2(1,IS)=CDRM2(1,IS)*CDTM1
C      DETERMINE RETARDED INTERPOLATED FIELDS
           IF(INTER.EQ.0) THEN
             CALL INTRPO(IND,FT,CJTH,CJTHD,DJTHD,A0,A1,A2,
     1       YM2,YM1,Y0,Y1,Y2)
             CALL INTRPO(IND,FT,CJPH,CJPHD,DJPHD,A0,A1,A2,
     1       YM2,YM1,Y0,Y1,Y2)
           ELSE
             CALL INTRP1(IND,FT,CJTH,CJTHD,DJTHD,A0,A1,A2,A3,A4,
     1       YM2,YM1,Y0,Y1,Y2)
             CALL INTRP1(IND,FT,CJPH,CJPHD,DJPHD,A0,A1,A2,A3,A4,
     1       YM2,YM1,Y0,Y1,Y2)
           END IF
C      SET FIELDS TO ZERO IF PULSE HAS NOT REACHED PATCH
C      (INTERPOLATION MAY MAKE FIELDS NONZERO BEFORE THEY ARE)
           WHERE(BT2(1,IS))
             CJTHD(1,IS)=0.
             CJPHD(1,IS)=0.
             DJTHD(1,IS)=0.
             DJPHD(1,IS)=0.
           END WHERE
C      DETERMINE THE CARTESIAN COMPONENTS OF THE CONTRIBUTION OF THE
C      PATCH TO THE INTEGRAL
```

```
      UTH(1,IS)=CDRM2(1,IS)*DJTHD(1,IS)+DRM3(1,IS)*CJTHD(1,IS)
      UPH(1,IS)=CDRM2(1,IS)*DJPHD(1,IS)+DRM3(1,IS)*CJPHD(1,IS)
      SINT1(1,IS)=-SRF(1,IS)*(DR2(1,IS)*UTH(1,IS)*TH3(1,IS)
     1  -DR3(1,IS)*(UTH(1,IS)*TH2(1,IS)+UPH(1,IS)*PH2(1,IS)))
      SINT2(1,IS)=-SRF(1,IS)*(DR3(1,IS)*(UTH(1,IS)*TH1(1,IS)
     1  +UPH(1,IS)*PH1(1,IS))-DR1(1,IS)*UTH(1,IS)*TH3(1,IS))
      SINT3(1,IS)=-SRF(1,IS)*(DR1(1,IS)*(UTH(1,IS)*TH2(1,IS)
     1  +UPH(1,IS)*PH2(1,IS))-DR2(1,IS)*(UTH(1,IS)*TH1(1,IS)
     2  +UPH(1,IS)*PH1(1,IS)))
      SNT1=Q8SSUM(SINT1(1,IS),BT1(1,IS))+SINT10
      SNT2=Q8SSUM(SINT2(1,IS),BT1(1,IS))+SINT20
      SNT3=Q8SSUM(SINT3(1,IS),BT1(1,IS))+SINT30
      BT1(IX)=B'1'
C
C     FILL IN APPROXIMATE VALUES FOR CURRENTS AND DERIVATIVES
C
      IF(IX.EQ.1) THEN
        CJTHD(1)=SNT2
        CJPHD(1)=-SNT1
        CJTH(1,NU)=SNT2
        CJPH(1,NU)=-SNT1
      ELSE
        CJTHD(IX)=-PH1(IX)*SNT1-PH2(IX)*SNT2
        CJPHD(IX)=TH1(IX)*SNT1+TH2(IX)*SNT2+TH3(IX)*SNT3
        CJTH(IX,NU)=CJTHD(IX)
        CJPH(IX,NU)=CJPHD(IX)
      END IF
      IF(INTER.EQ.0) THEN
        CALL INTRP2(2.,CJTH,IX,NU-2,DJTHDX)
        CALL INTRP2(2.,CJPH,IX,NU-2,DJPHDX)
      ELSE
        CALL INTRP3(2.,CJTH,IX,NU-2,DJTHDX)
        CALL INTRP3(2.,CJPH,IX,NU-2,DJPHDX)
      END IF
      DJTHD(IX)=DJTHDX
      DJPHD(IX)=DJPHDX
C
C     COMPUTE CORRECTIONS FOR NEIGHBORING PATCHES
C     IF ICORR = 0 OR 1, NEI(IX,1) = 1 AND LOOP IS SKIPPED
C
      DO 190 IXX=2,NEI(IX,1)
        IXP=NEI(IX,IXX)
        IF(T-CM1*DR(IXP).LE.TF(IXP)) GO TO 190
        NPNT=NPOINT(IX,IXX)
        COJ1=COJ(1,NPNT)
        COJ2=COJ(2,NPNT)
        COJ3=COJ(3,NPNT)
        COJ4=COJ(4,NPNT)
        COJ5=COJ(5,NPNT)
        COJ6=COJ(6,NPNT)
        COJT1=COJT(1,NPNT)
        COJT2=COJT(2,NPNT)
        COJT3=COJT(3,NPNT)
        COJT4=COJT(4,NPNT)
```

```
            COJT5=COJT(5,NPNT)
            COJT6=COJT(6,NPNT)
            COJU1=COJU(1,NPNT)
            COJU2=COJU(2,NPNT)
            COJTU1=COJTU(1,NPNT)
            COJTU2=COJTU(2,NPNT)
            COJV1=COJV(1,NPNT)
            COJV2=COJV(2,NPNT)
            COJTV1=COJTV(1,NPNT)
            COJTV2=COJTV(2,NPNT)
            IF(IXP.EQ.1.OR.IXP.EQ.IS) THEN
C     COMPUTE NEIGHBORING-PATCH CORRECTIONS FROM CAPS, IF NECESSARY
              IF(IXP.EQ.1) THEN
                ISP=2
              ELSE
                ISP=IS1
              END IF
C     DETERMINE THE INCREMENTS WITH RESPECT TO X
              CJ1D2=TH1(ISP)*CJTHD(ISP)+PH1(ISP)*CJPHD(ISP)
              CJ2D2=TH2(ISP)*CJTHD(ISP)+PH2(ISP)*CJPHD(ISP)
              CJ3D2=TH3(ISP)*CJTHD(ISP)
              N1=NL(IXP)
              N2=NR(IXP)
              CJ1DL=TH1(N1)*CJTHD(N1)+PH1(N1)*CJPHD(N1)
              CJ2DL=TH2(N1)*CJTHD(N1)+PH2(N1)*CJPHD(N1)
              CJ3DL=TH3(N1)*CJTHD(N1)
              CJ1DR=TH1(N2)*CJTHD(N2)+PH1(N2)*CJPHD(N2)
              CJ2DR=TH2(N2)*CJTHD(N2)+PH2(N2)*CJPHD(N2)
              CJ3DR=TH3(N2)*CJTHD(N2)
              IF(IXP.EQ.1) THEN
                DXCJ1=CJ1D2-(CJ1DL*FC1(1)+CJ1DR*FC2(1))
                DXCJ2=CJ2D2-(CJ2DL*FC1(1)+CJ2DR*FC2(1))
                DXCJ3=CJ3D2-(CJ3DL*FC1(1)+CJ3DR*FC2(1))
              ELSE
                DXCJ1=CJ1D2-(CJ1DL*FC1(2)+CJ1DR*FC2(2))
                DXCJ2=CJ2D2-(CJ2DL*FC1(2)+CJ2DR*FC2(2))
                DXCJ3=CJ3D2-(CJ3DL*FC1(2)+CJ3DR*FC2(2))
              END IF
C     DETERMINE THE INCREMENTS WITH RESPECT TO X
              N1=NU1(ISP)
              N2=NU2(ISP)
              CJ1PU=(TH1(N1)*CJTHD(N1)+PH1(N1)*CJPHD(N1))*FU1(ISP)
     1        +(TH1(N2)*CJTHD(N2)+PH1(N2)*CJPHD(N2))*FU2(ISP)
              CJ2PU=(TH2(N1)*CJTHD(N1)+PH2(N1)*CJPHD(N1))*FU1(ISP)
     1        +(TH2(N2)*CJTHD(N2)+PH2(N2)*CJPHD(N2))*FU2(ISP)
              CJ3PU=TH3(N1)*CJTHD(N1)*FU1(ISP)
     1        +TH3(N2)*CJTHD(N2)*FU2(ISP)
              N1=ND1(ISP)
              N2=ND2(ISP)
              CJ1PD=(TH1(N1)*CJTHD(N1)+PH1(N1)*CJPHD(N1))*FD1(ISP)
     1        +(TH1(N2)*CJTHD(N2)+PH1(N2)*CJPHD(N2))*FD2(ISP)
              CJ2PD=(TH2(N1)*CJTHD(N1)+PH2(N1)*CJPHD(N1))*FD1(ISP)
     1        +(TH2(N2)*CJTHD(N2)+PH2(N2)*CJPHD(N2))*FD2(ISP)
              CJ3PD=TH3(N1)*CJTHD(N1)*FD1(ISP)
```

```
     1          +TH3(N2)*CJTHD(N2)*FD2(ISP)
                DYCJ1=CJ1PU-CJ1PD
                DYCJ2=CJ2PU-CJ2PD
                DYCJ3=CJ3PU-CJ3PD
C       DETERMINE INCREMENTS OF TIME DERIVATIVES WITH RESPECT TO X
                DJ1D2=TH1(ISP)*DJTHD(ISP)+PH1(ISP)*DJPHD(ISP)
                DJ2D2=TH2(ISP)*DJTHD(ISP)+PH2(ISP)*DJPHD(ISP)
                DJ3D2=TH3(ISP)*DJTHD(ISP)
                N1=NL(IXP)
                N2=NR(IXP)
                DJ1DL=TH1(N1)*DJTHD(N1)+PH1(N1)*DJPHD(N1)
                DJ2DL=TH2(N1)*DJTHD(N1)+PH2(N1)*DJPHD(N1)
                DJ3DL=TH3(N1)*DJTHD(N1)
                DJ1DR=TH1(N2)*DJTHD(N2)+PH1(N2)*DJPHD(N2)
                DJ2DR=TH2(N2)*DJTHD(N2)+PH2(N2)*DJPHD(N2)
                DJ3DR=TH3(N2)*DJTHD(N2)
                IF(IXP.EQ.1) THEN
                  DXDJ1=DJ1D2-(DJ1DL*FC1(1)+DJ1DR*FC2(1))
                  DXDJ2=DJ2D2-(DJ2DL*FC1(1)+DJ2DR*FC2(1))
                  DXDJ3=DJ3D2-(DJ3DL*FC1(1)+DJ3DR*FC2(1))
                ELSE
                  DXDJ1=DJ1D2-(DJ1DL*FC1(2)+DJ1DR*FC2(2))
                  DXDJ2=DJ2D2-(DJ2DL*FC1(2)+DJ2DR*FC2(2))
                  DXDJ3=DJ3D2-(DJ3DL*FC1(2)+DJ3DR*FC2(2))
                END IF
C       DETERMINE INCREMENTS OF TIME DERIVATIVES WITH RESPECT TO Y
                N1=NU1(ISP)
                N2=NU2(ISP)
                DJ1PU=(TH1(N1)*DJTHD(N1)+PH1(N1)*DJPHD(N1))*FU1(ISP)
     1          +(TH1(N2)*DJTHD(N2)+PH1(N2)*DJPHD(N2))*FU2(ISP)
                DJ2PU=(TH2(N1)*DJTHD(N1)+PH2(N1)*DJPHD(N1))*FU1(ISP)
     1          +(TH2(N2)*DJTHD(N2)+PH2(N2)*DJPHD(N2))*FU2(ISP)
                DJ3PU=TH3(N1)*DJTHD(N1)*FU1(ISP)
     1          +TH3(N2)*DJTHD(N2)*FU2(ISP)
                N1=ND1(ISP)
                N2=ND2(ISP)
                DJ1PD=(TH1(N1)*DJTHD(N1)+PH1(N1)*DJPHD(N1))*FD1(ISP)
     1          +(TH1(N2)*DJTHD(N2)+PH1(N2)*DJPHD(N2))*FD2(ISP)
                DJ2PD=(TH2(N1)*DJTHD(N1)+PH2(N1)*DJPHD(N1))*FD1(ISP)
     1          +(TH2(N2)*DJTHD(N2)+PH2(N2)*DJPHD(N2))*FD2(ISP)
                DJ3PD=TH3(N1)*DJTHD(N1)*FD1(ISP)
     1          +TH3(N2)*DJTHD(N2)*FD2(ISP)
                DYDJ1=DJ1PU-DJ1PD
                DYDJ2=DJ2PU-DJ2PD
C       DETERMINE CORRECTION TO INTEGRAL FROM CAPS
                DYDJ3=DJ3PU-DJ3PD
                SNT1P=COJ3*CJPHD(IXP)-COJU2*DXCJ3-COJV2*DYCJ3
     1          +COJT3*DJPHD(IXP)-COJTU2*DXDJ3-COJTV2*DYDJ3
                SNT2P=-COJ3*CJTHD(IXP)+COJU1*DXCJ3+COJV1*DYCJ3
     1          -COJT3*DJTHD(IXP)+COJTU1*DXDJ3+COJTV1*DYDJ3
                SNT3P=-COJ1*CJPHD(IXP)+COJ2*CJTHD(IXP)-COJU1*DXCJPH
     1          +COJU2*DXCJTH-COJV1*DYCJPH+COJV2*DYCJTH
     2          -COJT1*DJPHD(IXP)+COJT2*DJTHD(IXP)-COJTU1*DXDJPH
     3          +COJTU2*DXDJTH-COJTV1*DYDJPH+COJTV2*DYDJTH
```

```
            SNT1=SNT1-SINT1(IXP)+SNT1P
            SNT2=SNT2-SINT2(IXP)+SNT2P
            SNT3=SNT3-SINT3(IXP)+SNT3P
          ELSE
C      COMPUTE NEIGHBORING-PATCH CORRECTIONS FROM REGULAR PATCHES
C      DETERMINE INCREMENT OF FIELDS AND TIME DERIVATIVES WITH RESPECT TO THETA
            CALL AVCUR(IXP,NU1,NU2,FU1,FU2,CJTHUO,CJPHUO,
     1      CJTHD,CJPHD)
            CALL AVCUR(IXP,ND1,ND2,FD1,FD2,CJTHDO,CJPHDO,
     1      CJTHD,CJPHD)
            CALL AVCUR(IXP,NU1,NU2,FU1,FU2,DJTHUO,DJPHUO,
     1      DJTHD,DJPHD)
            CALL AVCUR(IXP,ND1,ND2,FD1,FD2,DJTHDO,DJPHDO,
     1      DJTHD,DJPHD)
            DTHCJTH=CJTHDO-CJTHUO
            DTHCJPH=CJPHDO-CJPHUO
            DTHDJTH=DJTHDO-DJTHUO
            DTHDJPH=DJPHDO-DJPHUO
C      DETERMINE INCREMENT OF FIELDS AND TIME DERIVATIVES WITH RESPECT TO PHI
            IX1=NR(IXP)
            IX2=NL(IXP)
            CJTHX1=CJTHD(IX1)
            CJPHX1=CJPHD(IX1)
            CJTHX2=CJTHD(IX2)
            CJPHX2=CJPHD(IX2)
            DPHCJTH=CJTHX1-CJTHX2
            DPHCJPH=CJPHX1-CJPHX2
            DJTHX1=DJTHD(IX1)
            DJPHX1=DJPHD(IX1)
            DJTHX2=DJTHD(IX2)
            DJPHX2=DJPHD(IX2)
            DPHDJTH=DJTHX1-DJTHX2
            DPHDJPH=DJPHX1-DJPHX2
            CJTHDP=CJTHD(IXP)
            CJPHDP=CJPHD(IXP)
            DJTHDP=DJTHD(IXP)
            DJPHDP=DJPHD(IXP)
C      DETERMINE CONTRIBUTION TO INTEGRAL FROM NEIGHBORING PATCH
            CJITH=COJ1*CJTHDP+COJ4*CJPHDP
            DJITH=COJT1*DJTHDP+COJT4*DJPHDP
            CJIPH=COJ2*CJTHDP+COJ5*CJPHDP
            DJIPH=COJT2*DJTHDP+COJT5*DJPHDP
            CJINO=COJ3*CJTHDP+COJ6*CJPHDP+COJU1*DTHCJTH
     1      +COJU2*DTHCJPH+COJV1*DPHCJTH+COJV2*DPHCJPH
            DJINO=COJT3*DJTHDP+COJT6*DJPHDP+COJTU1*DTHDJTH
     1      +COJTU2*DTHDJPH+COJTV1*DPHDJTH+COJTV2*DPHDJPH
            CITH=CJITH+DJITH
            CIPH=CJIPH+DJIPH
            CINO=CJINO+DJINO
            TH1P=TH1(IXP)
            TH2P=TH2(IXP)
            TH3P=TH3(IXP)
            PH1P=PH1(IXP)
            PH2P=PH2(IXP)
```

```
              XP=RADM1*XR1(IXP)
              YP=RADM1*XR2(IXP)
              ZP=RADM1*XR3(IXP)
              SNT1P=-TH1P*CITH-PH1P*CIPH-XP*CINO
              SNT2P=-TH2P*CITH-PH2P*CIPH-YP*CINO
              SNT3P=-TH3P*CITH-ZP*CINO
              SNT1=SNT1-SINT1(IXP)+SNT1P
              SNT2=SNT2-SINT2(IXP)+SNT2P
              SNT3=SNT3-SINT3(IXP)+SNT3P
            END IF
 190      CONTINUE
C     STORE CORRECTED VALUES OF INTEGRAL
          IF(IX.EQ.1) THEN
            CJTHA(1)=SNT2
            CJPHA(1)=-SNT1
          ELSE
            CJTHA(IX)=-PH1(IX)*SNT1-PH2(IX)*SNT2
            CJPHA(IX)=TH1(IX)*SNT1+TH2(IX)*SNT2+TH3(IX)*SNT3
          END IF
C
C     CORRECT FOR SELF-PATCH CONTRIBUTIONS IF REQUIRED
C
          IF(ICORR.EQ.1.OR.ICORR.EQ.3) THEN
C
C     DO BOTTOM CAP (THETA = PI)
C
            IF(IX.EQ.1) THEN
              CJ3D2=TH3(2)*CJTHD(2)
              N1=NL(1)
              N2=NR(1)
              CJ3DL=TH3(N1)*CJTHD(N1)
              CJ3DR=TH3(N2)*CJTHD(N2)
              DXCJ3=CJ3D2-(CJ3DL*FC1(1)+CJ3DR*FC2(1))
              N1=NU1(1)
              N2=NU2(1)
              CJ3PU=TH3(N1)*CJTHD(N1)*FU1(1)+TH3(N2)*CJTHD(N2)*FU2(1)
              N1=ND1(1)
              N2=ND2(1)
              CJ3PD=TH3(N1)*CJTHD(N1)*FD1(1)+TH3(N2)*CJTHD(N2)*FD2(1)
              DYCJ3=CJ3PU-CJ3PD
              DJ3D2=TH3(2)*DJTHD(2)
              N1=NL(1)
              N2=NR(1)
              DJ3DL=TH3(N1)*DJTHD(N1)
              DJ3DR=TH3(N2)*DJTHD(N2)
              DXDJ3=DJ3D2-(DJ3DL*FC1(1)+DJ3DR*FC2(1))
              N1=NU1(1)
              N2=NU2(1)
              DJ3PU=TH3(N1)*DJTHD(N1)*FU1(1)+TH3(N2)*DJTHD(N2)*FU2(1)
              N1=ND1(1)
              N2=ND2(1)
              DJ3PD=TH3(N1)*DJTHD(N1)*FD1(1)+TH3(N2)*DJTHD(N2)*FD2(1)
              DYDJ3=DJ3PU-DJ3PD
              PC=CO(1)
```

```
                PD=DO(1)
                PDD=DO(IS)
                CJTHA(1)=CJTHA(1)*PC+DJTHD(1)*PD-.125*DXCJ3-PDD*DXDJ3
                CJPHA(1)=CJPHA(1)*PC+DJPHD(1)*PD-.125*DYCJ3-PDD*DYDJ3
              ELSE IF(IX.NE.IS) THEN
C
C       DO REGULAR PATCHES
C
                CJTHA(IX)=CJTHA(IX)*(1.-CO(IX))-DJTHD(IX)*DO(IX)
                CJPHA(IX)=CJPHA(IX)*(1.+CO(IX))+DJPHD(IX)*DO(IX)
              ELSE
C
C       DO TOP CAP IF REQUIRED
C
                CJ3D2=TH3(IS1)*CJTHD(IS1)
                N1=NL(IS)
                N2=NR(IS)
                CJ3DL=TH3(N1)*CJTHD(N1)
                CJ3DR=TH3(N2)*CJTHD(N2)
                DXCJ3=CJ3D2-(CJ3DL*FC1(2)+CJ3DR*FC2(2))
                N1=NU1(IS)
                N2=NU2(IS)
                CJ3PU=TH3(N1)*CJTHD(N1)*FU1(IS)+TH3(N2)*CJTHD(N2)*FU2(IS)
                N1=ND1(IS)
                N2=ND2(IS)
                CJ3PD=TH3(N1)*CJTHD(N1)*FD1(IS)+TH3(N2)*CJTHD(N2)*FD2(IS)
                DYCJ3=CJ3PU-CJ3PD
                DJ3D2=TH3(IS1)*DJTHD(IS1)
                N1=NL(IS)
                N2=NR(IS)
                DJ3DL=TH3(N1)*DJTHD(N1)
                DJ3DR=TH3(N2)*DJTHD(N2)
                DXDJ3=DJ3D2-(DJ3DL*FC1(2)+DJ3DR*FC2(2))
                N1=NU1(IS)
                N2=NU2(IS)
                DJ3PU=TH3(N1)*DJTHD(N1)*FU1(IS)+TH3(N2)*DJTHD(N2)*FU2(IS)
                N1=ND1(IS)
                N2=ND2(IS)
                DJ3PD=TH3(N1)*DJTHD(N1)*FD1(IS)+TH3(N2)*DJTHD(N2)*FD2(IS)
                DYDJ3=DJ3PU-DJ3PD
                CJTHA(IS)=CJTHA(IS)*PC+DJTHD(IS)*PD+.125*DXCJ3+PDD*DXDJ3
                CJPHA(IS)=CJPHA(IS)*PC+DJPHD(IS)*PD+.125*DYCJ3+PDD*DYDJ3
              END IF
            END IF
 200      CONTINUE
 210      IXH=IX-1
C       STORE NEW VALUES OF FIELDS
          CJTH(1,NU,IS)=CJTHA(1,IS)
          CJPH(1,NU,IS)=CJPHA(1,IS)
          IF(NUM.EQ.NU) THEN
C       WRITE RESULTS OUT TO DISK IF ARRAY IS ALL NEW
            DO 280 I=1,NU
              WRITE(2,REC=NREC) CJTH(1,I,IS)
              WRITE(2,REC=NREC+1) CJPH(1,I,IS)
```

```
            NREC=NREC+2
 280      CONTINUE
          NUM=0
        END IF
C     INCREMENT TIME
        T=T+DT
 300  CONTINUE
      TL=SECOND()-TL                          MAIN
      PRINT *,'EXECUTION TIME FOR THE LOOP IS ',TL,' SECONDS'
C     WRITE REMAINING RESULTS OUT TO DISK
      DO 310 I=NU-NUM+1,NU
        WRITE(2,REC=NREC) CJTH(1,I;IS)
        WRITE(2,REC=NREC+1) CJPH(1,I;IS)
        NREC=NREC+2
 310  CONTINUE
C     MAKE DISK FILE PERMANENT
      CLOSE(2)
      CALL Q5PURGE('LFN=','SCSV'//QAL,'STATUS=',ISTAT)
      IF(ISTAT.GT.0.AND.ISTAT.NE.1402) STOP 4
      CALL Q5DEFINE('LFN=','SCSV'//QAL)
      OPEN(2,FILE='SCSV'//QAL,STATUS='OLD',ACCESS='DIRECT',
     1 FORM='UNFORMATTED',RECL=NF2)
 400  READ(*,4,END=500) THSC,PHSC
      ITH=THSC
      IPHI=PHSC
      WRITE(TITLE,6) ITH,IPHI
      THSC=THSC*CONV
      PHSC=PHSC*CONV
C     COMPUTE COMPONENTS OF SCALED VECTOR TO FIELD POINT
      SR1=SIN(THSC)*COS(PHSC)*CDTM1
      SR2=SIN(THSC)*SIN(PHSC)*CDTM1
      SR3=COS(THSC)*CDTM1
C     COMPUTE FAR FIELDS
      CALL FARFL
      GO TO 400
 500  CLOSE(9)
C     SAVE FILE WITH COMPUTED FIELD INTENSITIES
      CALL Q5PURGE('LFN=','SCFL'//QAL,'STATUS=',ISTAT)
      IF(ISTAT.GT.0.AND.ISTAT.NE.1402) STOP 5
      CALL Q5DEFINE('LFN=','SCFL'//QAL)
      STOP
  1   FORMAT(20(1X,I5))
  2   FORMAT(1X,I4,3X,10(2X,1PE10.3))
  4   FORMAT(E12.5)
  5   FORMAT(3I4)
  6   FORMAT('PULSE SCATTERED IN THE DIRECTION THETA= ',I4,
     1 ',  PHI= ',I5)
  7   FORMAT('TIME*C IN METERS')
  8   FORMAT('INTENSITY    ')
  9   FORMAT('LST. SQ,  ALPHA=',F4.1,',  BETA=',F4.1,',  DT=',1PE7.1,
     1 ',  NT=',I3,',  N1=',I2,',  N2=',I2,',  N3=',I2,',  NO CORR')
 10   FORMAT('INTERP.,  ALPHA=',F4.1,',  BETA=',F4.1,',  DT=',1PE7.1,
     1 ',  NT=',I3,',  N1=',I2,',  N2=',I2,',  N3=',I2,',  NO CORR')
 11   FORMAT('LST. SQ,  ALPHA=',F4.1,',  BETA=',F4.1,',  DT=',1PE7.1,
```

```
     1  ',  NT=',I3,',  N1=',I2,',  N2=',I2,',  N3=',I2,',  SLF PCH')
  12   FORMAT('INTERP.,  ALPHA=',F4.1,',  BETA=',F4.1,',  DT=',1PE7.1,
     1  ',  NT=',I3,',  N1=',I2,',  N2=',I2,',  N3=',I2,',  SLF PCH')
  13   FORMAT('LST.  SQ,  ALPHA=',F4.1,',  BETA=',F4.1,',  DT=',1PE7.1,
     1  ',  NT=',I3,',  N1=',I2,',  N2=',I2,',  N3=',I2,',  NEI PCH')
  14   FORMAT('INTERP.,  ALPHA=',F4.1,',  BETA=',F4.1,',  DT=',1PE7.1,
     1  ',  NT=',I3,',  N1=',I2,',  N2=',I2,',  N3=',I2,',  NEI PCH')
  15   FORMAT('LST.  SQ,  ALPHA=',F4.1,',  BETA=',F4.1,',  DT=',1PE7.1,
     1  ',  NT=',I3,',  N1=',I2,',  N2=',I2,',  N3=',I2,',  SLF & N')
  16   FORMAT('INTERP.,  ALPHA=',F4.1,',  BETA=',F4.1,',  DT=',1PE7.1,
     1  ',  NT=',I3,',  N1=',I2,',  N2=',I2,',  N3=',I2,',  SLF & N')
       END
C
       SUBROUTINE AVCUR(IN,NX1,NX2,FX1,FX2,CJTHP,CJPHP,
     1 CJTHX,CJPHX)
C
C      TO COMPUTE CURRENTS AT DESIRED PHI BY AVERAGING
       PARAMETER(LL=LLG,LC=LCG)
       DIMENSION NX1(LL),NX2(LL),FX1(LL),FX2(LL),
     1 CJTHX(LL),CJPHX(LL)
       N1=NX1(IN)
       N2=NX2(IN)
       F1=FX1(IN)
       F2=FX2(IN)
       IF(N1.EQ.N2) THEN
         IF(N1.EQ.1) THEN
C        FOR BOTTOM CAP
           CJTHP=-F1*CJTHX(N1)-F2*CJPHX(N1)
         ELSE
C        FOR TOP CAP
           CJTHP=F1*CJTHX(N1)+F2*CJPHX(N1)
         END IF
         CJPHP=-F2*CJTHX(N1)+F1*CJPHX(N1)
       ELSE
C        FOR OTHER PATCHES
         CJTHP=F1*CJTHX(N1)+F2*CJTHX(N2)
         CJPHP=F1*CJPHX(N1)+F2*CJPHX(N2)
       END IF
       RETURN
       END
C
       SUBROUTINE FARFL
C
C      THIS SUBROUTINE COMPUTES THE RADIATION FIELDS AT TIMES IN T1
C      IN THE DIRECTION GIVEN BY SR1, SR2, SR3.
C      TMP:  TIME ARRAY
C      AMP:  AMPLITUDE ARRAY
C
       PARAMETER(LL=LLG,LC=LCG,LL12=LL12G,LL20=LL20G,
     1 LB=LL**2/64+1,LB1=LL/64+1)
       DIMENSION TMP(300),AMP(300)
       COMMON /RCOM/ XR1(LL),XR2(LL),XR3(LL),SRF(LL),TF(LL),
     1 CO(LL),DO(LL),TH1(LL),TH2(LL),TH3(LL),
     2 PH1(LL),PH2(LL),FU1(LL),FU2(LL),FD1(LL),FD2(LL),FC1(2),FC2(2),
```

```
     3 THETA(LL),PHII(LL),DPHII(LL),RB1(LB1),RB2(LB1),RB3(LB),RB4(LB),
     4 COJ(6,LL12),COJT(6,LL12),COJU(2,LL12),COJTU(2,LL12),
     5 COJV(2,LL12),COJTV(2,LL12),SINT1(LL),SINT2(LL),SINT3(LL),
     6 DR1(LL),DR2(LL),DR3(LL),DR(LL),CDRM2(LL),DRM3(LL),
     7 CJTHD(LL),CJPHD(LL),DJTHD(LL),DJPHD(LL),
     8 UTH(LL),UPH(LL),A0(LL),A1(LL),A2(LL),A3(LL),A4(LL),
     9 YM2(LL),YM1(LL),Y0(LL),Y1(LL),Y2(LL),
     A CJTHA(LL),CJPHA(LL),CJTH(LL,LC),CJPH(LL,LC),FT(LL),
     B NL(LL),NR(LL),NU1(LL),NU2(LL),ND1(LL),ND2(LL),
     C NEI(LL,LL20),NPOINT(LL,LL20),IND(LL),IJ(LL)
      COMMON /ICOM/ IX,IS,IS1,N1,N2,NI,ICORR,IPLOT,INTER,IDEB,
     1 IPATCH,N3,NU
      CHARACTER TITLE*80,SUBT*80,XLAB*30,YLAB*30
      COMMON /LABEL/ TITLE,SUBT,XLAB,YLAB
      COMMON /PARAM/ ALPHA,BETA,DT,RAD,RADM1,CDTM1,THSC,SR1,SR2,SR3
      COMMON /CONST/ AMUO,TWOMUO,TWOEPO,C,CM1,PI,TOPI,PITO,OS,OTW,OTF
      PRINT *, 'START FAR-FIELD COMPUTATION'
      CDTM2=CDTM1**2
      RCDTM1=RAD*CDTM1
      FAC=(1.E-7*TOPI*RADM1)**2
      NU=1
      NI2=NI*2
      READ(*,1) T1,T2
C     COMPUTE FIRST TIME OF ARRIVAL IF T1 = 0
      IF(T1.EQ.0.) T1=RAD*CM1*(1.-2.*SIN(.5*THSC))-DT
C     COMPUTE LAST TIME IF T2 = 0
      IF(T2.EQ.0.) T2=(NI-5)*DT-RAD*CM1
      T1DT=T1/DT
      READ(*,2) N
      IF(N.GT.300) STOP 6
      AMP(1:300)=-1.
      DT1=(T2-T1)/((N-1)*DT)
      NMAX=T1DT+RCDTM1+4.
      NMIN=T1DT-RCDTM1-3.
      NTOT=NMAX-NMIN+1
      IF(NTOT.GT.LC) THEN
        PRINT *, 'NTOT=',NTOT,' GREATER THAN LC=',LC
        STOP 7
      END IF
      IF(NMIN.LE.0) THEN
        DO 130 J1=1,-NMIN+1
          CJTH(1,J1,IS)=0.
          CJPH(1,J1,IS)=0.
130     CONTINUE
        NREC=0
      ELSE
        NREC=(NMIN-1)*2
        J1=1
      END IF
      DO 150 J=J1,NTOT
        NREC=NREC+2
        IF(NREC.GT.NI2) THEN
          PRINT *,'TIME LIMIT ON SPHERE EXCEEDED DURING INTEGRATION'
          PRINT *,'T2= ',T2,',  T= ',T1DT*DT,',  K= ',K,' OUT OF ',N
```

```
        STOP 8
      END IF
      READ(2,REC=NREC-1) CJTH(1,J,IS)
      READ(2,REC=NREC) CJPH(1,J,IS)
150   CONTINUE
      DO 200 K=1,N
        NX=T1DT+RCDTM1+4.
        NN=T1DT-RCDTM1-3.
        IF(NX-NN+1.GT.LC) THEN
          PRINT *, 'NEW RANGE ',NX-NN+1,' GREATER THAN LC=',LC
          PRINT *, 'NN=',NN, ', NX=',NX, ', K=',K
          STOP 9
        END IF
        IF(NX-NMIN+1.GT.LC) THEN
          NA=NN-NMIN
          DO 160 L=1,NMAX-NN+1
            NA=NA+1
            CJTH(1,L,IS)=CJTH(1,NA,IS)
            CJPH(1,L,IS)=CJPH(1,NA,IS)
160       CONTINUE
          NMIN=NN
        ELSE
          L=NMAX-NMIN+2
        END IF
        IF(NX.GT.NMAX) THEN
        DO 170 J=L,LC
          NREC=NREC+2
          IF(NREC.GT.NI2) THEN
            PRINT *,'TIME LIMIT ON SPHERE EXCEEDED DURING INTEGRATION'
            PRINT *,'T2= ',T2, ', T= ',T1DT*DT, ', K= ',K,' OUT OF ',N
            STOP 10
          END IF
          READ(2,REC=NREC-1) CJTH(1,J,IS)
          READ(2,REC=NREC) CJPH(1,J,IS)
170     CONTINUE
          NMAX=NMIN+LC-1
        END IF
        NMIN1=NMIN-1
        TMP(K)=T1DT*DT*C
        DR(1,IS)=T1DT+XR1(1,IS)*SR1+XR2(1,IS)*SR2+XR3(1,IS)*SR3
        IND(1,IS)=VINT(DR(1,IS),IND(1,IS))
        FT(1,IS)=DR(1,IS)-VAINT(DR(1,IS),IS)
        IND(1,IS)=IJ(1,IS)+(IND(1,IS)-NMIN1)*LL
        IF(INTER.EQ.0) THEN
          CALL INTRPO(IND,FT,CJTH,CJTHD,DJTHD,A0,A1,A2,
     1    YM2,YM1,Y0,Y1,Y2)
          CALL INTRPO(IND,FT,CJPH,CJPHD,DJPHD,A0,A1,A2,
     1    YM2,YM1,Y0,Y1,Y2)
        ELSE
          CALL INTRP1(IND,FT,CJTH,CJTHD,DJTHD,A0,A1,A2,A3,A4,
     1    YM2,YM1,Y0,Y1,Y2)
          CALL INTRP1(IND,FT,CJPH,CJPHD,DJPHD,A0,A1,A2,A3,A4,
     1    YM2,YM1,Y0,Y1,Y2)
        END IF
```

```
         C1=Q8SSUM((DJTHD(1,IS)*TH1(1,IS)+DJPHD(1,IS)*PH1(1,IS))
     1   *SRF(1,IS))
         C2=Q8SSUM((DJTHD(1,IS)*TH2(1,IS)+DJPHD(1,IS)*PH2(1,IS))
     1   *SRF(1,IS))
         C3=Q8SSUM(DJTHD(1,IS)*TH3(1,IS)*SRF(1,IS))
         AMP(K)=((C1**2+C2**2+C3**2)*CDTM2-(C1*SR1+C2*SR2
     1   +C3*SR3)**2)*FAC
         T1DT=T1DT+DT1
 200   CONTINUE
       WRITE(9) TMP
       WRITE(9) AMP
       IF(IPLOT.EQ.1.OR.IPLOT.EQ.2)
     1   CALL DRAW1(1,16,12,56,80,N,0.,0.,XLAB,YLAB,TITLE,SUBT,TMP,AMP)
       IF(IPLOT.EQ.0.OR.IPLOT.EQ.2)
     1   CALL DRAW2(1,16,12,56,80,N,0.,0.,XLAB,YLAB,TITLE,SUBT,TMP,AMP)
       RETURN
 1     FORMAT(E12.5)
 2     FORMAT(2I4)
       END
C
       SUBROUTINE INFL(SINT1,SINT2,SINT3,X,Y,Z,T,*)
C
C      THIS SUBROUTINE PROVIDES THE INCIDENT FIELDS
C
       COMMON /PARAM/ ALPHA,BETA,DT,RAD,RADM1,CDTM1,THSC,SR1,SR2,SR3
       COMMON /CONST/ AMUO,TWOMUO,TWOEPO,C,CM1,PI,TOPI,PITO,OS,OTW,OTF
       ZCT=Z-C*T+RAD
       IF(ZCT.GE.0.) RETURN 1
       AZ=ALPHA*ZCT
       BZ=BETA*ZCT
       IF(BZ.GE.-25.) THEN
         B1=BZ*(EXP(AZ)-EXP(BZ))
         E2=-B1*C
       ELSE
         B1=0.
         E2=0.
       END IF
       SINT1=TWOMUO*B1
       SINT2=0.
       SINT3=0.
       RETURN
       END
C
       SUBROUTINE INTRPO(IND,FT,ARR,CA,DA,A0,A1,A2,
     1 YM2,YM1,Y0,Y1,Y2)
C
C      LEAST SQUARES INTERPOLATION
C
       PARAMETER(LL=LLG,LC=LCG)
       COMMON /ICOM/ IX,IS,IS1,N1,N2,NI,ICORR,IPLOT,INTER,IDEB,
     1 IPATCH,N3,NU
       COMMON /CONST/ AMUO,TWOMUO,TWOEPO,C,CM1,PI,TOPI,PITO,OS,OTW,OTF
       DIMENSION ARR(LL,LC),CA(LL),DA(LL),YM2(LL),YM1(LL),Y0(LL),
     1 Y1(LL),Y2(LL),A0(LL),A1(LL),A2(LL),IND(LL),FT(LL)
```

```
      Y2(1,IS)=Q8VGATHR(ARR(1,NU+2,1),IND(1,IS),Y2(1,IS))
      Y1(1,IS)=Q8VGATHR(ARR(1,NU+1,1),IND(1,IS),Y1(1,IS))
      YO(1,IS)=Q8VGATHR(ARR(1,NU,1),IND(1,IS),YO(1,IS))
      YM1(1,IS)=Q8VGATHR(ARR(1,NU-1,1),IND(1,IS),YM1(1,IS))
      YM2(1,IS)=Q8VGATHR(ARR(1,NU-2,1),IND(1,IS),YM2(1,IS))
      AO(1,IS)=(YM2(1,IS)+YM1(1,IS)+YO(1,IS)+Y1(1,IS)+Y2(1,IS))*.2
      A1(1,IS)=(Y2(1,IS)+.5*(Y1(1,IS)-YM1(1,IS))-YM2(1,IS))*.2
      A2(1,IS)=(Y2(1,IS)-.5*(Y1(1,IS)+YM1(1,IS))-YO(1,IS)+YM2(1,IS))*OS
      CA(1,IS)=AO(1,IS)+FT(1,IS)*A1(1,IS)
     1 +(FT(1,IS)**2-2.)*A2(1,IS)
      DA(1,IS)=A1(1,IS)+FT(1,IS)*2.*A2(1,IS)
      RETURN
      END
C
      SUBROUTINE INTRP1(IND,FT,ARR,CA,DA,AO,A1,A2,A3,A4,
     1 YM2,YM1,YO,Y1,Y2)
C
C     POLYNOMIAL INTERPOLATION
C
      PARAMETER(LL=LLG,LC=LCG)
      COMMON /ICOM/ IX,IS,IS1,N1,N2,NI,ICORR,IPLOT,INTER,IDEB,
     1 IPATCH,N3,NU
      COMMON /CONST/ AMUO,TWOMUO,TWOEPO,C,CM1,PI,TOPI,PITO,OS,OTW,OTF
      DIMENSION ARR(LL,LC),CA(LL),DA(LL),YM2(LL),YM1(LL),YO(LL),
     1 Y1(LL),Y2(LL),AO(LL),A1(LL),A2(LL),A3(LL),A4(LL),
     2 IND(LL),FT(LL)
      Y2(1,IS)=Q8VGATHR(ARR(1,NU+2,1),IND(1,IS),Y2(1,IS))
      Y1(1,IS)=Q8VGATHR(ARR(1,NU+1,1),IND(1,IS),Y1(1,IS))
      YO(1,IS)=Q8VGATHR(ARR(1,NU,1),IND(1,IS),YO(1,IS))
      YM1(1,IS)=Q8VGATHR(ARR(1,NU-1,1),IND(1,IS),YM1(1,IS))
      YM2(1,IS)=Q8VGATHR(ARR(1,NU-2,1),IND(1,IS),YM2(1,IS))
C     AO(1,IS)=YO(1,IS)
      A1(1,IS)=OTW*(YM2(1,IS)-Y2(1,IS)-8.*(YM1(1,IS)-Y1(1,IS)))
      A2(1,IS)=OTF*(-YM2(1,IS)-Y2(1,IS)+16.*(YM1(1,IS)+Y1(1,IS))
     1 -30.*YO(1,IS))
      A3(1,IS)=OTW*(-YM2(1,IS)+Y2(1,IS)+2.*(YM1(1,IS)-Y1(1,IS)))
      A4(1,IS)=OTF*(YM2(1,IS)+Y2(1,IS)-4.*(YM1(1,IS)+Y1(1,IS))
     1 +6.*YO(1,IS))
      CA(1,IS)=YO(1,IS)+FT(1,IS)*(A1(1,IS)+FT(1,IS)
     1 *(A2(1,IS)+FT(1,IS)*(A3(1,IS)+FT(1,IS)*A4(1,IS))))
      DA(1,IS)=A1(1,IS)+FT(1,IS)*(2.*A2(1,IS)+FT(1,IS)
     1 *(3.*A3(1,IS)+FT(1,IS)*4.*A4(1,IS)))
      RETURN
      END
      SUBROUTINE INTRP2(TT,ARR,I,NTP,DA)
C
C     LEAST-SQUARES INTERPOLATION, SCALAR VERSION, DERIVATIVE ONLY
C
      PARAMETER(LL=LLG,LC=LCG)
      DIMENSION ARR(LL,LC)
      COMMON /CONST/ AMUO,TWOMUO,TWOEPO,C,CM1,PI,TOPI,PITO,OS,OTW,OTF
      YM2=ARR(I,NTP-2)
      YM1=ARR(I,NTP-1)
      YO=ARR(I,NTP)
```

```
      Y1=ARR(I,NTP+1)
      Y2=ARR(I,NTP+2)
C     A0=(YM2+YM1+Y0+Y1+Y2)*.2
      A1=(Y2+.5*(Y1-YM1)-YM2)*.2
      A2=(Y2-.5*(Y1+YM1)-Y0+YM2)*OS
C     CA=A0+TT*A1+(TT**2-2.)*A2
      DA=A1+TT*2.*A2
      RETURN
      END
C
      SUBROUTINE INTRP3(TT,ARR,I,NTP,DA)
C
C     POLYNOMIAL INTERPOLATION, SCALAR VERSION, DERIVATIVE ONLY
C
      PARAMETER(LL=LLG,LC=LCG)
      DIMENSION ARR(LL,LC)
      COMMON /CONST/ AMU0,TWOMU0,TWOEPO,C,CM1,PI,TOPI,PITO,OS,OTW,OTF
      YM2=ARR(I,NTP-2)
      YM1=ARR(I,NTP-1)
      Y0=ARR(I,NTP)
      Y1=ARR(I,NTP+1)
      Y2=ARR(I,NTP+2)
      A1=OTW*(YM2-Y2-8.*(YM1-Y1))
      A2=OTF*(-YM2-Y2+16.*(YM1+Y1)-30.*Y0)
      A3=OTW*(-YM2+Y2+2.*(YM1-Y1))
      A4=OTF*(YM2+Y2-4.*(YM1+Y1)+6.*Y0)
C     CA=Y0+FT*(A1+TT*(A2+TT*(A3+TT*A4)))
      DA=A1+TT*(2.*A2+TT*(3.*A3+TT*4.*A4))
      RETURN
      END
      SUBROUTINE RUNT
C
C     THIS SUBROUTINE COMPUTES COMPONENTS OF THE RADIUS VECTOR
C     AND FACTORS NEEDED FOR CORRECTIONS
C
C     N1=NUMBER OF STRIPS ON THE SPHERE (VALUES OF THETA)
C     N2=MAXIMUM NUMBER OF PATCHES IN A STRIP (VALUES OF PHI ON EQUATOR)
C     N3=MINIMUM NUMBER OF PATCHES IN A STRIP (NEXT TO A POLAR CAP)
C     NPHI IS THE ARRAY WHERE THE NUMBER OF PATCHES IN A STRIP IS STORED
C     IS=TOTAL NUMBER OF ANGLE PAIRS
C
      PARAMETER(LL=LLG,LC=LCG,LL12=LL12G,LL20=LL20G,
     1 LB=LL**2/64+1,LB1=LL/64+1)
      DIMENSION NPHI(100)
      BIT BT3(LL,LL),BT4(LL,LL)
      EXTERNAL F003,F103,F013,F203,F113,F023
      EXTERNAL F002,F102,F012,F202,F112,F022
      COMMON /RCOM/ XR1(LL),XR2(LL),XR3(LL),SRF(LL),TF(LL),
     1 C0(LL),D0(LL),TH1(LL),TH2(LL),TH3(LL),
     2 PH1(LL),PH2(LL),FU1(LL),FU2(LL),FD1(LL),FD2(LL),FC1(2),FC2(2),
     3 THETA(LL),PHII(LL),DPHII(LL),RB1(LB1),RB2(LB1),RB3(LB),RB4(LB),
     4 COJ(6,LL12),COJT(6,LL12),COJU(2,LL12),COJTU(2,LL12),
     5 COJV(2,LL12),COJTV(2,LL12),SINT1(LL),SINT2(LL),SINT3(LL),
     6 DR1(LL),DR2(LL),DR3(LL),DR(LL),CDRM2(LL),DRM3(LL),
```

```
     7 CJTHD(LL),CJPHD(LL),DJTHD(LL),DJPHD(LL),
     8 UTH(LL),UPH(LL),A0(LL),A1(LL),A2(LL),A3(LL),A4(LL),
     9 YM2(LL),YM1(LL),Y0(LL),Y1(LL),Y2(LL),
     A CJTHA(LL),CJPHA(LL),CJTH(LL,LC),CJPH(LL,LC),FT(LL),
     B NL(LL),NR(LL),NU1(LL),NU2(LL),ND1(LL),ND2(LL),
     C NEI(LL,LL20),NPOINT(LL,LL20),IND(LL),IJ(LL)
      COMMON /ICOM/ IX,IS,IS1,N1,N2,NI,ICORR,IPLOT,INTER,IDEB,
     1 IPATCH,N3,NU
      COMMON /PARAM/ ALPHA,BETA,DT,RAD,RADM1,CDTM1,THSC,SR1,SR2,SR3
      COMMON /CONST/ AMUO,TWOMUO,TWOEPO,C,CM1,PI,TOPI,PITO,OS,OTW,OTF
      COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
      EQUIVALENCE (RB3,BT3),(RB4,BT4)
      DATA PC,PC2,PC3,ZRO/4*0./,EPS/1.E-10/
C     COMPUTE DELTA-THETA
      O3=1./3.
      PIM1=1./PI
      TOPIM1=1./TOPI
      FOPIM1=.5*TOPIM1
      RCDTM1=RAD*CDTM1
      TPCTM1=RCDTM1*TOPIM1
      FPCTM1=RCDTM1*FOPIM1
      DTH=PI/(N1-1)
      SDTHM1=1./SIN(DTH)
      DTH2=DTH*.5
      DTHM1=1./DTH
      DTHM1H=FOPIM1/DTH
      DTHM1C=DTHM1H*RCDTM1
      SDTH2=SIN(DTH2)
      SDT2M1=1./SDTH2
C     COMPUTE NUMBER OF PATCHES IN STRIP
      TH=PI
      IS=2
      NPHI(1)=1
      NPHI(N1)=1
      N1H=(N1+1)/2
      DPH=1.
      IF(IPATCH.EQ.0) THEN
C     NUMBER OF PATCHES IN A STRIP HAS TO BE A POWER OF 2
        NPHX=1
 90     NPHX=NPHX*2
        IF(NPHX.LT.N3) GO TO 90
        DO 100 I=2,N1H
          TH=TH-DTH
          STH=SIN(TH)
          NPH=(N2-N3)*STH+N3+.5
          IF(NPH.GT.NPHX) NPHX=NPHX*2
          NPHI(I)=NPHX
          IS=IS+NPHX
          DPH=AMIN1(DPH,STH*SIN(PI/NPHX))
 100    CONTINUE
      ELSE
        DO 105 I=2,N1H
          TH=TH-DTH
          STH=SIN(TH)
```

```
            NPH=(N2-N3)*STH+N3+.5
            NPHI(I)=NPH
            IS=IS+NPH
            DPH=AMIN1(DPH,STH*SIN(PI/NPH))
  105     CONTINUE
        END IF
C       TO ASSURE SYMMETRY
        IF(MOD(N1,2).EQ.1) N1H=N1H-1
        DO 110 J=I,N1-1
          NPH=NPHI(N1H)
          NPHI(J)=NPH
          IS=IS+NPH
          N1H=N1H-1
  110   CONTINUE
C       CHOOSE AN ADEQUATE (MINIMUM) TIME INTERVAL
        DT1=1.6*RAD*CM1*AMIN1(SDTH2,DPH)
        IF(DT.EQ.0.) THEN
C       USE DT1 IF NO TIME INCREMENT IS GIVEN
          DT=DT1
        ELSE
          IF(DT1.LT.DT) THEN
C       USE DT1 IF GIVEN TIME INCREMENT IS TOO LARGE
            PRINT 3, DT,DT1
            NI=NI*DT/DT1
            DT=DT1
          END IF
        END IF
C       PRINT THE COMPUTED VALUES
C       FILE 'OUTPT' IS USED BECAUSE THE OUTPUT FILE TENDS TO FILL UP
        IF(IDEB.EQ.1) THEN
          LENGTH=IS*133/1024+10
          CALL Q5RQUEST('LFN=','OUTPT','MXR=',133,'LEN=',LENGTH)
          OPEN(7,FILE='OUTPT')
          WRITE(7,'(1H1)')
          WRITE(7,*) 'NUMBER OF PATCHES IN STRIPS'
          DO 120 I=1,N1,10
            WRITE(7,1) NPHI(I),NPHI(I+1),NPHI(I+2),NPHI(I+3),NPHI(I+4),
       1    NPHI(I+5),NPHI(I+6),NPHI(I+7),NPHI(I+8),NPHI(I+9)
  120     CONTINUE
          WRITE(7,'(//)')
        END IF
        PRINT *, 'NUMBER OF PATCHES IS ',IS,', NUMBER OF ROWS IS ',LL
        IF(IS.GT.LL) STOP 11
        IS1=IS-NPH
        TH=PI
        S1=2.*(RAD*SIN(DTH*.25))**2
        IN=1
C       COMPUTE COMPONENT OF RADIUS VECTOR, SURFACE ELEMENT, TANGENTIAL UNIT
C       VECTORS, ETC.
C       BOTTOM CAP, THETA = PI
        XR1(1)=0.
        XR2(1)=0.
        XR3(1)=-RAD
        SRF(1)=S1
```

```
      TF(1)=0.
      THETA(1)=PI
      PHII(1)=0.
      DPHII(1)=TOPI
      TH1(1)=1.
      TH2(1)=0.
      TH3(1)=0.
      PH1(1)=0.
      PH2(1)=1.
      NP2=1
      NP3=NPHI(2)
      INB2=1
      INB3=2
      DPHI2=TOPI
      DPHI3=TOPI/NP3
      IF(ICORR.NE.0) THEN
C     COMPUTE COEFFICIENTS FOR SELF-PATCH CORRECTIONS FOR CAPS
        CO(1)=1.+.25*DTH
        DO(1)=.0625*DTH**2*RCDTM1
        DO(IS)=.03125*DTH*RCDTM1
        CALL UD(PI,DPHI3,NP3,1,2,NL,NR,FC1,FC2)
        CALL UD(PITO,DPHI3,NP3,1,2,NU1,NU2,FU1,FU2)
        CALL UD(PITO*3.,DPHI3,NP3,1,2,ND1,ND2,FD1,FD2)
      END IF
C     REGULAR PATCHES
      DO 200 I=2,N1-1
        TH=TH-DTH
        STH=SIN(TH)
        CTH=COS(TH)
        RSTH=RAD*STH
        RCTH=RAD*CTH
        TFI=CM1*(RAD+RCTH)
        NP1=NP2
        NP2=NP3
        NP3=NPHI(I+1)
        INB1=INB2
        INB2=INB3
        INB3=INB2+NP2
        DPHI1=DPHI2
        DPHI2=DPHI3
        DPHI3=TOPI/NP3
        PHI=0.
C
C     SOLID ANGLE FOR THE PATCH
C
        SOLG=PIM1*DPHI2*STH*SDTH2*RAD**2
C
C     PATCH COEFFICIENT
C
        IF(ICORR.NE.0) THEN
C     COMPUTE COEFFICIENTS FOR SELF-PATCH CORRECTIONS
          STHDPH=STH*DPHI2
          STHDPH1=1./STHDPH
          DGAMA2=DTH**2+STHDPH**2
```

```
          DGAMMA=SQRT(DGAMA2)
          AL1=ALOG((DGAMMA+DTH)/(DGAMMA-DTH))
          AL2=ALOG((DGAMMA+STHDPH)/(DGAMMA-STHDPH))
          PC2=FOPIM1*STHDPH*AL1
          PC3=FOPIM1*DTH*AL2
          PC=PC2-PC3
          ATDG=.5*DGAMA2*ATAN(STHDPH*DTHM1)
          TAB=.5*DTH*STHDPH
          SNPH2=.5*PITO*STHDPH**2
          PC7=FPCTM1*(TAB+ATDG-SNPH2)
          PC8=FPCTM1*(TAB-ATDG+SNPH2)
          PD=PC7-PC8
        END IF
        DO 190 J=1,NP2
          IN=IN+1
          CPH=COS(PHI)
          SPH=SIN(PHI)
          XR1(IN)=RSTH*CPH
          XR2(IN)=RSTH*SPH
          XR3(IN)=RCTH
          SRF(IN)=SOLQ
          THETA(IN)=TH
          PHII(IN)=PHI
          DPHII(IN)=DPHI2
          TH1(IN)=CTH*CPH
          TH2(IN)=CTH*SPH
          TH3(IN)=-STH
          PH1(IN)=-SPH
          PH2(IN)=CPH
          TF(IN)=TFI
          IF(ICORR.NE.0) THEN
            CO(IN)=PC
            DO(IN)=PD
            IF(J.EQ.1) THEN
              NL(IN)=IN+NP2-1
            ELSE
              NL(IN)=IN-1
            END IF
            IF(J.EQ.NP2) THEN
              NR(IN)=IN-NP2+1
            ELSE
              NR(IN)=IN+1
            END IF
            CALL UD(PHI,DPHI1,NP1,IN,INB1,ND1,ND2,FD1,FD2)
            CALL UD(PHI,DPHI3,NP3,IN,INB3,NU1,NU2,FU1,FU2)
          END IF
          PHI=PHI+DPHI2
  190   CONTINUE
  200 CONTINUE
C     TOP CAP, THETA = 0
      XR1(IS)=0.
      XR2(IS)=0.
      XR3(IS)=RAD
      SRF(IS)=S1
```

```
      THETA(IS)=0.
      PHII(IS)=0.
      DPHII(IS)=TOPI
      TH1(IS)=1.
      TH2(IS)=0.
      TH3(IS)=0.
      IF(ICORR.NE.0) THEN
        CALL UD(PI,DPHI2,NP2,IS,INB2,NL,NR,PH1,PH2)
        FC1(2)=PH1(IS)
        FC2(2)=PH2(IS)
        CALL UD(PITO,DPHI2,NP2,IS,INB2,NU1,NU2,FU1,FU2)
        CALL UD(PITO*3,DPHI2,NP2,IS,INB2,ND1,ND2,FD1,FD2)
      END IF
      PH1(IS)=0.
      PH2(IS)=1.
      TF(IS)=2.*CM1*RAD
C     COMPUTE QUANTITIES RELATED TO TWO PATCHES
      DL1=DTH*RAD
      NMAX=0
      IN=1
      NPNT=0
      NP=0
      DO 240 I=1,IS
        X=XR1(I)
        Y=XR2(I)
        Z=XR3(I)
        TH=THETA(I)
        PHI=PHII(I)
        STH=SIN(TH)
        RSTH=RAD*STH
        NM=1
        IF(NP.EQ.0) THEN
          NPH=NPHI(IN)
          DPH=TOPI/NPH
          DL2=DPH*RSTH
          RMAX=SQRT(2.)*AMAX1(DL1,DL2)
          IF(I.EQ.1.OR.I.EQ.IS) RMAX=1.5*DTH*RAD
          NP=NPH
          IN=IN+1
        END IF
        NP=NP-1
        DR1(1,IS)=X-XR1(1,IS)
        DR2(1,IS)=Y-XR2(1,IS)
        DR3(1,IS)=Z-XR3(1,IS)
        DR(1,IS)=VSQRT(DR1(1,IS)**2+DR2(1,IS)**2+DR3(1,IS)**2,
     1  DR(1,IS))
        DRM3(1,IS)=CDTM1*DR(1,IS)
        IND(1,IS)=VINT(DRM3(1,IS),IND(1,IS))
C     DETERMINE WHERE SHIFTS IN ITERPOLATION ARE NEEDED
        BT3(1,I,IS)=IND(1,IS).EQ.2
        BT4(1,I,IS)=IND(1,IS).EQ.1
        IF(ICORR.LT.2) GO TO 230
C     COMPUTE COEFFICIENTS FOR NEIGHBORING-PATCH INTEGRALS
        DO 220 J=1,IS
```

```
      IF(J.EQ.I.OR.DR(J).GT.RMAX) GO TO 220
      NM=NM+1
      IF(NM.GT.LL20 OR. NPNT.GT.LL12) THEN
        PRINT *, 'NM=',NM, ', NPNT=',NPNT, ', I=',I, ', J=',J
        STOP 12
      END IF
      NEI(I,NM)=J
      NPNT=NPNT+1
      NPOINT(I,NM)=NPNT
      IF(J.EQ.1.OR.J.EQ.IS) THEN
C     NEIGHBORING PATCH IS A SPHERICAL CAP
        RHOQ=STH*SDT2M1
        TRHOQ=2.*RHOQ
        RHOQH=STH*FOPIM1*SDTHM1
        RHOQCT=RHOQH*RCDTM1
        RHOQI=1./RHOQ
        TRHOQI=2.*RHOQI
        RHOSQ=1.+RHOQ**2
        RHOSQI=1.+RHOQI**2
        RHCTM1=STH*RCDTM1*TOPIM1
        RHCTM1H=.5*RHCTM1
C
C     USE Q1DB FROM CMLIB TO DO NUMERICAL INTEGRATIONS
C
C     CP003 INCLUDES AN ADDITIONAL FACTOR RHOO
        CALL Q1DB(F003,ZRO,TOPI,EPS,CP003,ER,KF,IFLAG)
        IF(IFLAG.GT.3) THEN
          PRINT *, 'PROBLEM IN CP003=',CP003,'; ERROR=',ER,
     1     ', NO. OF COMPUTATIONS=',KF,', FLAG=',IFLAG
          STOP
        END IF
        CALL Q1DB(F103,ZRO,TOPI,EPS,CP103,ER,KF,IFLAG)
        IF(IFLAG.GT.3) THEN
          PRINT *, 'PROBLEM IN CP103=',CP103,'; ERROR=',ER,
     1     ', NO. OF COMPUTATIONS=',KF,', FLAG=',IFLAG
          STOP
        END IF
        CALL Q1DB(F013,ZRO,TOPI,EPS,CP013,ER,KF,IFLAG)
        IF(IFLAG.GT.3) THEN
          PRINT *, 'PROBLEM IN CP013=',CP013,'; ERROR=',ER,
     1     ', NO. OF COMPUTATIONS=',KF,', FLAG=',IFLAG
          STOP
        END IF
C     CP203 INCLUDES AN ADDITIONAL FACTOR 1/RHOO
        CALL Q1DB(F203,ZRO,TOPI,EPS,CP203,ER,KF,IFLAG)
        IF(IFLAG.GT.3) THEN
          PRINT *, 'PROBLEM IN CP203=',CP203,'; ERROR=',ER,
     1     ', NO. OF COMPUTATIONS=',KF,', FLAG=',IFLAG
          STOP
        END IF
C     CP113 INCLUDES AN ADDITIONAL FACTOR 1/RHOO
        CALL Q1DB(F113,ZRO,TOPI,EPS,CP113,ER,KF,IFLAG)
        IF(IFLAG.GT.3) THEN
          PRINT *, 'PROBLEM IN CP113=',CP113,'; ERROR=',ER,
```

```
     1         ', NO. OF COMPUTATIONS=', KF, ', FLAG=', IFLAG
               STOP
             END IF
C      CP023 INCLUDES AN ADDITIONAL FACTOR 1/RH00
             CALL Q1DB(F023, ZRO, TOPI, EPS, CP023, ER, KF, IFLAG)
             IF(IFLAG.GT.3) THEN
               PRINT *, 'PROBLEM IN CP023=', CP023, '; ERROR=', ER,
     1         ', NO. OF COMPUTATIONS=', KF, ', FLAG=', IFLAG
               STOP
             END IF
             CALL Q1DB(F002, ZRO, TOPI, EPS, CP002, ER, KF, IFLAG)
             IF(IFLAG.GT.3) THEN
               PRINT *, 'PROBLEM IN CP002=', CP002, '; ERROR=', ER,
     1         ', NO. OF COMPUTATIONS=', KF, ', FLAG=', IFLAG
               STOP
             END IF
C      CP102 INCLUDES AN ADDITIONAL FACTOR 1/RH00
             CALL Q1DB(F102, ZRO, TOPI, EPS, CP102, ER, KF, IFLAG)
             IF(IFLAG.GT.3) THEN
               PRINT *, 'PROBLEM IN CP102=', CP102, '; ERROR=', ER,
     1         ', NO. OF COMPUTATIONS=', KF, ', FLAG=', IFLAG
               STOP
             END IF
C      CP012 INCLUDES AN ADDITIONAL FACTOR 1/RH00
             CALL Q1DB(F012, ZRO, TOPI, EPS, CP012, ER, KF, IFLAG)
             IF(IFLAG.GT.3) THEN
               PRINT *, 'PROBLEM IN CP012=', CP012, '; ERROR=', ER,
     1         ', NO. OF COMPUTATIONS=', KF, ', FLAG=', IFLAG
               STOP
             END IF
C      CP202 INCLUDES AN ADDITIONAL FACTOR 1/RH00**2
             CALL Q1DB(F202, ZRO, TOPI, EPS, CP202, ER, KF, IFLAG)
             IF(IFLAG.GT.3) THEN
               PRINT *, 'PROBLEM IN CP202=', CP202, '; ERROR=', ER,
     1         ', NO. OF COMPUTATIONS=', KF, ', FLAG=', IFLAG
               STOP
             END IF
C      CP112 INCLUDES AN ADDITIONAL FACTOR 1/RH00**2
             CALL Q1DB(F112, ZRO, TOPI, EPS, CP112, ER, KF, IFLAG)
             IF(IFLAG.GT.3) THEN
               PRINT *, 'PROBLEM IN CP112=', CP112, '; ERROR=', ER,
     1         ', NO. OF COMPUTATIONS=', KF, ', FLAG=', IFLAG
               STOP
             END IF
C      CP022 INCLUDES AN ADDITIONAL FACTOR 1/RH00**2
             CALL Q1DB(F022, ZRO, TOPI, EPS, CP022, ER, KF, IFLAG)
             IF(IFLAG.GT.3) THEN
               PRINT *, 'PROBLEM IN CP022=', CP022, '; ERROR=', ER,
     1         ', NO. OF COMPUTATIONS=', KF, ', FLAG=', IFLAG
               STOP
             END IF
             CPH=COS(PHI)
             SPH=SIN(PHI)
             COJ(1,NPNT)=TOPIM1*(CPH*CP003-CP103)
```

```
            COJ(2,NPNT)=TOPIM1*(SPH*CP003-CP013)
            COJT(1,NPNT)=RHCTM1*(CPH*CP002-CP102)
            COJT(2,NPNT)=RHCTM1*(SPH*CP002-CP012)
            IF(J.EQ.1) THEN
              COJ(3,NPNT)=FOPIM1*STH*(CP003-CP203-CP023)
              COJT(3,NPNT)=RHCTM1H*STH*(CP002-CP202-CP022)
            ELSE
              COJ(3,NPNT)=-FOPIM1*STH*(CP003-CP203-CP023)
              COJT(3,NPNT)=-RHCTM1H*STH*(CP002-CP202-CP022)
            END IF
            COJU(1,NPNT)=RHOQH*(CPH*CP103-CP203)
            COJU(2,NPNT)=RHOQH*(SPH*CP103-CP113)
            COJV(1,NPNT)=RHOQH*(CPH*CP013-CP113)
            COJV(2,NPNT)=RHOQH*(SPH*CP013-CP023)
            COJTU(1,NPNT)=RHOQCT*(CPH*CP102-CP202)
            COJTU(2,NPNT)=RHOQCT*(SPH*CP102-CP112)
            COJTV(1,NPNT)=RHOQCT*(CPH*CP012-CP112)
            COJTV(2,NPNT)=RHOQCT*(SPH*CP012-CP022)
          ELSE
C       OTHER NEIGHBORING PATCHES
            THP=THETA(J)
            PHIP=PHII(J)
            STHP=SIN(THP)
            CTHP=COS(THP)
            CSCTHP=1./STHP
            COTTHP=CTHP*CSCTHP
            SPHP=SIN(PHIP)
            CPHP=COS(PHIP)
            STCP=STHP*CPHP
            STSP=STHP*SPHP
            CTCP=CTHP*CPHP
            CTSP=CTHP*SPHP
            DELTH=THP-TH
            DELPHI=PHIP-PHI
            IF(I.EQ.1.OR.I.EQ.IS) THEN
              DELPHI=0.
            ELSE
              IF(DELPHI.GT.PI) DELPHI=DELPHI-TOPI
              IF(DELPHI.LT.-PI) DELPHI=DELPHI+TOPI
            END IF
            STDELP=STHP*DELPHI
            THTP=DELTH+.5*DTH
            THTP2=THTP**2
            THTP3=THTP2*THTP
            THTP4=THTP3*THTP
            THTM=DELTH-.5*DTH
            THTM2=THTM**2
            THTM3=THTM2*THTM
            THTM4=THTM3*THTM
            DPHI=DPHII(J)
            PHTP=(DELPHI+.5*DPHI)*STHP
            PHTP2=PHTP**2
            PHTP3=PHTP2*PHTP
            PHTP4=PHTP3*PHTP
```

```
      PHTM=(DELPHI-.5*DPHI)*STHP
      PHTM2=PHTM**2
      PHTM3=PHTM2*PHTM
      PHTM4=PHTM3*PHTM
      GAM1T=SQRT(THTM2+PHTM2)
      GAM1TI=1./GAM1T
      GAM1TI2=GAM1TI**2
      GAM1TI3=GAM1TI2*GAM1TI
      GAM2T=SQRT(THTP2+PHTM2)
      GAM2TI=1./GAM2T
      GAM2TI2=GAM2TI**2
      GAM2TI3=GAM2TI2*GAM2TI
      GAM3T=SQRT(THTP2+PHTP2)
      GAM3TI=1./GAM3T
      GAM3TI2=GAM3TI**2
      GAM3TI3=GAM3TI2*GAM3TI
      GAM4T=SQRT(THTM2+PHTP2)
      GAM4TI=1./GAM4T
      GAM4TI2=GAM4TI**2
      GAM4TI3=GAM4TI2*GAM4TI
      THMAX2=AMAX1(THTM2,THTP2)*1.E-12
      PHMAX2=AMAX1(PHTM2,PHTP2)*1.E-12
      IF(THTM2.LT.PHMAX2.AND.PHTP.LT.0.) THEN
        C103=ALOG(PHTM*(GAM2T+PHTM)/((GAM3T+PHTP)*PHTP))
      ELSE IF(THTP2.LT.PHMAX2.AND.PHTP.LT.0.) THEN
        C103=ALOG((GAM4T+PHTP)*PHTP/(PHTM*(GAM1T+PHTM)))
      ELSE
        C103=ALOG((GAM4T+PHTP)*(GAM2T+PHTM)/((GAM3T+PHTP)*
     1    (GAM1T+PHTM)))
      END IF
      IF(PHTM2.LT.THMAX2.AND.THTP.LT.0.) THEN
        C013=ALOG(THTM*(GAM4T+THTM)/((GAM3T+THTP)*THTP))
      ELSE IF(PHTP2.LT.THMAX2.AND.THTP.LT.0.) THEN
        C013=ALOG((GAM2T+THTP)*THTP/(THTM*(GAM1T+THTM)))
      ELSE
        C013=ALOG((GAM2T+THTP)*(GAM4T+THTM)/((GAM3T+THTP)*
     1    (GAM1T+THTM)))
      END IF
      IF(PHTM2.LT.THMAX2) THEN
        C203=-PHTM*ABS(ALOG(THTP/THTM))
     2    +.5*PHTP*ALOG((GAM3T+THTP)*(GAM4T-THTM)/
     3    ((GAM3T-THTP)*(GAM4T+THTM)))
      ELSE IF(PHTP2.LT.THMAX2) THEN
        C203=-.5*PHTM*ALOG((GAM2T+THTP)*(GAM1T-THTM)/
     1    ((GAM2T-THTP)*(GAM1T+THTM)))
     2    +PHTP*ABS(ALOG(THTP/THTM))
      ELSE
        C203=-.5*PHTM*ALOG((GAM2T+THTP)*(GAM1T-THTM)/
     1    ((GAM2T-THTP)*(GAM1T+THTM)))
     2    +.5*PHTP*ALOG((GAM3T+THTP)*(GAM4T-THTM)/
     3    ((GAM3T-THTP)*(GAM4T+THTM)))
      END IF
      C113=-GAM1T+GAM2T-GAM3T+GAM4T
      IF(THTM2.LT.PHMAX2) THEN
```

```
          C023=-THTM*ABS(ALOG(PHTP/PHTM))
     2      +.5*THTP*ALOG((GAM3T+PHTP)*(GAM2T-PHTM)/
     3      ((GAM3T-PHTP)*(GAM2T+PHTM)))
        ELSE IF(THTP2.LT.PHMAX2) THEN
          C023=-.5*THTM*ALOG((GAM4T+PHTP)*(GAM1T-PHTM)/
     1      ((GAM4T-PHTP)*(GAM1T+PHTM)))
     2      +THTP*ABS(ALOG(PHTP/PHTM))
        ELSE
          C023=-.5*THTM*ALOG((GAM4T+PHTP)*(GAM1T-PHTM)/
     1      ((GAM4T-PHTP)*(GAM1T+PHTM)))
     2      +.5*THTP*ALOG((GAM3T+PHTP)*(GAM2T-PHTM)/
     3      ((GAM3T-PHTP)*(GAM2T+PHTM)))
        END IF
        C125=O3*((PHTP*(GAM3TI-GAM4TI)-PHTM*(GAM2TI-GAM1TI))
     1      +C103)
        C035=O3*((THTM*(GAM4TI-GAM1TI)-THTP*(GAM3TI-GAM2TI))
     1      +2.*C013)
        C225=O3*(THTP*(PHTP*GAM3TI-PHTM*GAM2TI)
     1      -(THTM*(PHTP*GAM4TI-PHTM*GAM1TI)))
        C135=THTM2*(O3*THTM2*(GAM1TI3-GAM4TI3)-GAM1TI
     1      +GAM4TI)+PHTP4*O3*(GAM4TI3-GAM3TI3)
     2      +THTP2*(O3*THTP2*(GAM3TI3-GAM2TI3)-GAM3TI
     3      +GAM2TI)+PHTM4*O3*(GAM2TI3-GAM1TI3)
        DTHT=THTP-THTM
        DPHT=PHTP-PHTM
        PHT1=ATAN2(PHTM,THTM)
        PHT2=ATAN2(PHTM,THTP)
        PHT3=ATAN2(PHTP,THTP)
        PHT4=ATAN2(PHTP,THTM)
        PHT12=PHT1-PHT2
        PHT23=PHT2-PHT3
        PHT34=PHT3-PHT4
        PHT41=PHT4-PHT1
        IF(PHTM.LT.0..AND.PHTP.GT.0..AND.THTP.LT.0.) THEN
          PHT23=PHT23+TOPI
          PHT41=PHT41-TOPI
        END IF
        ALG12=ALOG(GAM1T*GAM2TI)
        ALG23=ALOG(GAM2T*GAM3TI)
        ALG34=ALOG(GAM3T*GAM4TI)
        ALG41=ALOG(GAM4T*GAM1TI)
        C102=-THTP*PHT23+PHTP*ALG34-THTM*PHT41+PHTM*ALG12
        C012=-PHTP*PHT34-THTP*ALG23-PHTM*PHT12-THTM*ALG41
        C202=.5*(DTHT*DPHT+PHTM2*PHT12-THTP2*PHT23
     1      +PHTP2*PHT34-THTM2*PHT41)
        C112=.5*(PHTM2*ALG12-THTP2*ALG23
     1      +PHTP2*ALG34-THTM2*ALG41)
        C022=.5*(DTHT*DPHT-PHTM2*PHT12+THTP2*PHT23
     1      -PHTP2*PHT34+THTM2*PHT41)
        C124=.5*(PHTM3*(GAM2TI2-GAM1TI2)-THTP*PHT23
     1      +THTP2*(PHTM*GAM2TI2-PHTP*GAM3TI2)
     2      +PHTP3*(GAM4TI2-GAM3TI2)-THTM*PHT41
     3      +THTM2*(PHTP*GAM4TI2-PHTM*GAM1TI2))
        C034=.5*(-PHTM*PHT12-PHTM2*(THTP*GAM2TI2-THTM*GAM1TI2)
```

```
     1       +THTP*(PHTM2*GAM2TI2-PHTP2*GAM3TI2-2. *ALG23)
     2       -PHTP*PHT34-PHTP2*(THTM*GAM4TI2-THTP*GAM3TI2)
     3       +THTM*(PHTP2*GAM4TI2-PHTM2*GAM1TI2-2. *ALG41))
             C224=. 25*(-PHTM2*PHT12+PHTM3*(THTP*GAM2TI2-THTM*GAM1TI2)
     1       -THTP2*PHT23+THTP3*(PHTM*GAM2TI2-PHTP*GAM3TI2)
     2       -PHTP2*PHT34+PHTP3*(THTM*GAM4TI2-THTP*GAM3TI2)
     3       -THTM2*PHT41+THTM3*(PHTP*GAM4TI2-PHTM*GAM1TI2))
             C134=. 25*(PHTM4*(GAM2TI2-GAM1TI2)
     1       +THTP2*(PHTM2*GAM2TI2-PHTP2*GAM3TI2-2. *ALG23)
     2       +PHTP4*(GAM4TI2-GAM3TI2)
     3       +THTM2*(PHTP2*GAM4TI2-PHTM2*GAM1TI2-2. *ALG41))
             DTHCOT=DELTH*COTTHP
             COETH=C103*(-1. +DTHCOT)+(-C203+1. 5*C225)*COTTHP
     1       -3. *C125*DTHCOT
             COEPH=C013*(-1. +2. *DTHCOT)+CTHP*DELPHI*C103
     1       +COTTHP*(-2. *C113+1. 5*C135)-3. *DTHCOT*C035
             COENO=. 5*C203-DELTH*C103
             COER=. 5*C023*CSCTHP-DELPHI*C013
             CTHTH=DELTH*C103-C203
             CTHPH=DELTH*C013-C113
             CPHTH=STDELP*C103-C113
             CPHPH=STDELP*C013-C023
C        COJ: 1: JTHTH, 2: JTHPH 3 JTHN, 4: JPHTH, 5 JPHPH 6: JPHN
             COJ(1,NPNT)=-TOPIM1*CTHPH
             COJ(2,NPNT)=TOPIM1*(COENO+COER*STHP+CTHTH)
             COJ(3,NPNT)=TOPIM1*(-COEPH+CPHTH*COTTHP)
             COJ(4,NPNT)=TOPIM1*(-COENO-COER*STHP-CPHPH)
             COJ(5,NPNT)=TOPIM1*CPHTH
             COJ(6,NPNT)=TOPIM1*(COETH+COER*CTHP+CPHPH*COTTHP)
             COJU(1,NPNT)=-DTHM1H*CTHPH
             COJU(2,NPNT)=DTHM1H*CTHTH
             STDM1H=TOPIM1/(STHP*DPHI)
             COJV(1,NPNT)=-STDM1H*CPHPH
             COJV(2,NPNT)=STDM1H*CPHTH
             COETH=C102*(-1. +DTHCOT)+(-C202+C224)*COTTHP
     1       -2. *C124*DTHCOT
             COEPH=C012*(-1 +2. *DTHCOT)+CTHP*DELPHI*C102
     1       +COTTHP*(-2. *C112+C134)-2. *DTHCOT*C034
             COENO=. 5*C202-DELTH*C102
             COER=. 5*C022*CSCTHP-DELPHI*C012
             CTHTH=DELTH*C102-C202
             CTHPH=DELTH*C012-C112
             CPHTH=STDELP*C102-C112
             CPHPH=STDELP*C012-C022
             COJT(1,NPNT)=-TPCTM1*CTHPH
             COJT(2,NPNT)=TPCTM1*(COENO+COER*STHP+CTHTH)
             COJT(3,NPNT)=TPCTM1*(-COEPH+CPHTH*COTTHP)
             COJT(4,NPNT)=TPCTM1*(-COENO-COER*STHP-CPHPH)
             COJT(5,NPNT)=TPCTM1*CPHTH
             COJT(6,NPNT)=TPCTM1*(COETH+COER*CTHP+CPHPH*COJTHP)
             COJTU(1,NPNT)=-DTHM1C*CTHPH
             COJTU(2,NPNT)=DTHM1C*CTHTH
             STDM1C=STDM1H*CRDTM1
             COJTV(1,NPNT)=-STDM1C*CPHPH
```

```
            COJTV(2,NPNT)=STDM1C*CPHTH
          END IF
220     CONTINUE
230     NEI(I,1)=NM
        IF(NM.GT.NMAX) NMAX=NM
240   CONTINUE
      PRINT *, 'NUMBER OF COLUMNS NEEDED IS ',NMAX-1,' OUT OF 19,',
     1 ' TOTAL NUMBER OF NEIGHBORING PATCHES IS ',NPNT,
     2 ', WITH A MAXIMUM OF ',LL12
      RETURN
1     FORMAT(10(2X,I10))
3     FORMAT(' GIVEN DT=',E9.2,' GREATER THAN "MINIMUM" DT=',E9.2)
      END
C
      SUBROUTINE UD(PHI,DPHI,NP,IN,INB,NX1,NX2,FX1,FX2)
C
C     THIS SUBROUTINE IS USED TO COMPUTE QUANTITIES NEEDED FOR PHI-INTERPOLATION
C
      PARAMETER(LL=LLG,LC=LCG)
      DIMENSION NX1(LL),NX2(LL),FX1(LL),FX2(LL)
      F1=PHI/DPHI
      N1=F1
      F1=F1-REAL(N1)
      NX1(IN)=INB+N1
      IF(N1.EQ.NP-1) THEN
        NX2(IN)=INB
      ELSE
        NX2(IN)=INB+N1+1
      END IF
      IF(N1.EQ.N2) THEN
C     FOR POLAR CAPS
        FX1(IN)=COS(PHI)
        FX2(IN)=SIN(PHI)
      ELSE
        FX1(IN)=1.-F1
        FX2(IN)=F1
      END IF
      RETURN
      END
C
      FUNCTION FOO3(X)
      COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
      COSX=COS(X)
      EKS=SQRT(RHOSQ-TRHOQ*COSX)
      FOO3=1./(EKS*(EKS+RHOQ-COSX))
      RETURN
      END
C
      FUNCTION F103(X)
      COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
      COSX=COS(X)
      RCOSX=RHOQ*COSX
      TCOSX=2.*RCOSX
      EKS=SQRT(RHOSQ-TCOSX)
```

```
      ECOSX=EKS*COSX
      IF(COSX.GT..5) THEN
        F103=COS(X+PHI)*(ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     1  2.*COSX/(EKS*(EKS+RHOQ-COSX))-
     2  (1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
      ELSE IF(COSX.GT.-.5) THEN
        F103=COS(X+PHI)*(ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     1  2.*COSX/(EKS*(EKS+RHOQ-COSX))-
     2  (1.+ECOSX-RCOSX)/(EKS*(1.-COSX**2)))
      ELSE
        F103=COS(X+PHI)*(ALOG((EKS-RCOSX+1.)/(RHOQ-RCOSX))+
     1  2.*COSX/(EKS*(EKS+RHOQ-COSX))-
     2  (1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
      END IF
      RETURN
      END
C
      FUNCTION F013(X)
      COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,D3
      COSX=COS(X)
      RCOSX=RHOQ*COSX
      TCOSX=2.*RCOSX
      EKS=SQRT(RHOSQ-TCOSX)
      ECOSX=EKS*COSX
      IF(COSX.GT..5) THEN
        F013=SIN(X+PHI)*(ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     1  2.*COSX/(EKS*(EKS+RHOQ-COSX))-
     2  (1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
      ELSE IF(COSX.GT.-.5) THEN
        F013=SIN(X+PHI)*(ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     1  2.*COSX/(EKS*(EKS+RHOQ-COSX))-
     2  (1.+ECOSX-RCOSX)/(EKS*(1.-COSX**2)))
      ELSE
        F013=SIN(X+PHI)*(ALOG((EKS-RCOSX+1.)/(RHOQ-RCOSX))+
     1  2.*COSX/(EKS*(EKS+RHOQ-COSX))-
     2  (1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
      END IF
      RETURN
      END
C
      FUNCTION F203(X)
      COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,D3
      COSX=COS(X)
      RCOSX=RHOQ*COSX
      TCOSX=2.*RCOSX
      EKS=SQRT(RHOSQ-TCOSX)
      ECOSX=EKS*COSX
      IF(COSX.GT..5) THEN
        F203=COS(X+PHI)**2*(RHOQI*EKS-1.+
     1  3.*COSX*ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     2  (4.*COSX**2-1.)/(EKS*(EKS+RHOQ-COSX))-
     3  2.*COSX*(1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
      ELSE IF(COSX.GT.-.5) THEN
        F203=COS(X+PHI)**2*(RHOQI*EKS-1.+
```

```
     1   3.*COSX*ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     2   (4.*COSX**2-1.)/(EKS*(EKS+RHOQ-COSX))-
     3   2.*COSX*(1.+ECOSX-RCOSX)/(EKS*(1.-COSX**2)))
         ELSE
           F203=COS(X+PHI)**2*(RHOQI*EKS-1.+
     1   3.*COSX*ALOG((EKS-RCOSX+1.)/(RHOQ-RCOSX))+
     2   (4.*COSX**2-1.)/(EKS*(EKS+RHOQ-COSX))-
     3   2.*COSX*(1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
         END IF
         RETURN
         END
C
         FUNCTION F113(X)
         COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
         COSX=COS(X)
         RCOSX=RHOQ*COSX
         TCOSX=2.*RCOSX
         EKS=SQRT(RHOSQ-TCOSX)
         ECOSX=EKS*COSX
         IF(COSX.GT..5) THEN
           F113=COS(X+PHI)*SIN(X+PHI)*(RHOQI*EKS-1.+
     1   3.*COSX*ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     2   (4.*COSX**2-1.)/(EKS*(EKS+RHOQ-COSX))-
     3   2.*COSX*(1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
         ELSE IF(COSX.GT.-.5) THEN
           F113=COS(X+PHI)*SIN(X+PHI)*(RHOQI*EKS-1.+
     1   3.*COSX*ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     2   (4.*COSX**2-1.)/(EKS*(EKS+RHOQ-COSX))-
     3   2.*COSX*(1.+ECOSX-RCOSX)/(EKS*(1.-COSX**2)))
         ELSE
           F113=COS(X+PHI)*SIN(X+PHI)*(RHOQI*EKS-1.+
     1   3.*COSX*ALOG((EKS-RCOSX+1.)/(RHOQ-RCOSX))+
     2   (4.*COSX**2-1)/(EKS*(EKS+RHOQ-COSX))-
     3   2.*COSX*(1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
         END IF
         RETURN
         END
C
         FUNCTION FO23(X)
         COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
         COSX=COS(X)
         RCOSX=RHOQ*COSX
         TCOSX=2.*RCOSX
         EKS=SQRT(RHOSQ-TCOSX)
         ECOSX=EKS*COSX
         IF(COSX.GT..5) THEN
           FO23=SIN(X+PHI)**2*(RHOQI*EKS-1.+
     1   3.*COSX*ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     2   (4.*COSX**2-1.)/(EKS*(EKS+RHOQ-COSX))-
     3   2.*COSX*(1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
         ELSE IF(COSX.GT.-.5) THEN
           FO23=SIN(X+PHI)**2*(RHOQI*EKS-1.+
     1   3.*COSX*ALOG((RHOQ+RCOSX)/(EKS+RCOSX-1.))+
     2   (4.*COSX**2-1.)/(EKS*(EKS+RHOQ-COSX))-
```

```
     3   2.*COSX*(1.+ECOSX-RCOSX)/(EKS*(1.-COSX**2)))
         ELSE
           FO23=SIN(X+PHI)**2*(RHOQI*EKS-1.+
     1   3.*COSX*ALOG((EKS-RCOSX+1.)/(RHOQ-RCOSX))+
     2   (4.*COSX**2-1.)/(EKS*(EKS+RHOQ-COSX))-
     3   2.*COSX*(1.-TCOSX)/(EKS*(1.-ECOSX-RCOSX)))
         END IF
         RETURN
         END
C
         FUNCTION FOO2(X)
         COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
         SINX=SIN(X)
         COSX=COS(X)
         EKS2=RHOSQI-TRHOQI*COSX
         IF(ABS(SINX).GE.1.E-4) THEN
           FO2=ATAN(SINX/(RHOQ-COSX))/SINX
         ELSE
           DEN=1./(RHOQ-COSX)
           FO2=DEN-O3*SINX**2*DEN**3
         END IF
         FOO2=.5*ALOG(EKS2)+COSX*FO2
         RETURN
         END
C
         FUNCTION F102(X)
         COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
         SINX=SIN(X)
         COSX=COS(X)
         EKS2=RHOSQI-TRHOQI*COSX
         IF(ABS(SINX).GE.1.E-4) THEN
           FO2=ATAN(SINX/(RHOQ-COSX))/SINX
         ELSE
           DEN=1./(RHOQ-COSX)
           FO2=DEN-O3*SINX**2*DEN**3
         END IF
         F102=COS(X+PHI)*(RHOQI+COSX*ALOG(EKS2)
     1  -(1.-2.*COSX**2)*FO2)
         RETURN
         END
C
         FUNCTION FO12(X)
         COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
         SINX=SIN(X)
         COSX=COS(X)
         COSX2=COSX**2
         EKS2=RHOSQI-TRHOQI*COSX
         IF(ABS(SINX).GE.1.E-4) THEN
           FO2=ATAN(SINX/(RHOQ-COSX))/SINX
         ELSE
           DEN=1./(RHOQ-COSX)
           FO2=DEN-O3*SINX**2*DEN**3
         END IF
         FO12=SIN(X+PHI)*(RHOQI+COSX*ALOG(EKS2)
```

```
     1  -(1.-2.*COSX**2)*FO2)
        RETURN
        END
C
        FUNCTION F202(X)
        COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
        SINX=SIN(X)
        COSX=COS(X)
        COSX2=COSX**2
        EKS2=RHOSQI-TRHOQI*COSX
        IF(ABS(SINX).GE.1.E-4) THEN
          FO2=ATAN(SINX/(RHOQ-COSX))/SINX
        ELSE
          DEN=1./(RHOQ-COSX)
          FO2=DEN-O3*SINX**2*DEN**3
        END IF
        F202=COS(X+PHI)**2*(RHOQI*(.5*RHOQI+2.*COSX)
     1  +(2.*COSX2-.5)*ALOG(EKS2)-COSX*(3.-4.*COSX2)*FO2)
        RETURN
        END
C
        FUNCTION F112(X)
        COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
        SINX=SIN(X)
        COSX=COS(X)
        COSX2=COSX**2
        EKS2=RHOSQI-TRHOQI*COSX
        IF(ABS(SINX).GE.1.E-4) THEN
          FO2=ATAN(SINX/(RHOQ-COSX))/SINX
        ELSE
          DEN=1./(RHOQ-COSX)
          FO2=DEN-O3*SINX**2*DEN**3
        END IF
        F112=COS(X+PHI)*SIN(X+PHI)*(RHOQI*(.5*RHOQI+2.*COSX)
     1  +(2.*COSX2-.5)*ALOG(EKS2)-COSX*(3.-4.*COSX2)*FO2)
        RETURN
        END
C
        FUNCTION FO22(X)
        COMMON /PARFUN/ PHI,RHOQ,RHOQI,TRHOQ,TRHOQI,RHOSQ,RHOSQI,O3
        SINX=SIN(X)
        COSX=COS(X)
        COSX2=COSX**2
        EKS2=RHOSQI-TRHOQI*COSX
        IF(ABS(SINX) GE.1.E-4) THEN
          FO2=ATAN(SINX/(RHOQ-COSX))/SINX
        ELSE
          DEN=1./(RHOQ-COSX)
          FO2=DEN-O3*SINX**2*DEN**3
        END IF
        FO22=SIN(X+PHI)**2*(RHOQI*(.5*RHOQI+2.*COSX)
     1  +(2.*COSX2-.5)*ALOG(EKS2)-COSX*(3.-4.*COSX2)*FO2)
        RETURN
        END
```

```
      PROGRAM                    SCATP
C
C     TO COMPUTE THE SIZES OF ARRAYS FOR PROGRAM SCAT AND RELATED PROGRAMS
C
      CHARACTER QAL*4, SUF*3, JCAT*2, PRI*2
      COMMON /PARAM/ RAD, ALPHA, BETA, DT, DTM1, THDIR
      COMMON /ICOM/ IX, IS, IS1, NTOT, N1, N2, N3, NI, ISKIP, IPATCH, ICORR
      COMMON /CONST/ C, CM1, PI, TOPI
      DATA C/3.E8/
      CM1=1./C
      PI=4.*ATAN(1.)
      TOPI=2.*PI
      CONV=PI/180.
      OPEN(1,FILE='SCTQAL',STATUS='OLD')
      READ(1,*) QAL, ISKIP, NT, IPATCH, PRI, ICORR, NTL, NLP1
      PRINT *,'INPUT FILE: SCIN'//QAL,' SKIP=',ISKIP,',  PATCH=',IPATCH,
     1 ',  CORR=',ICORR
      OPEN(2,FILE='SCIN'//QAL,STATUS='OLD')
      READ(2,4) DT,RAD,ALPHA,BETA
      READ(2,5) NI
      READ(2,5) N1,N2,N3
      IF(N1.GT.100) STOP 1
      IF(N3.EQ.0) N3=5
C     FIND NUMBER OF PATCHES AND NUMBER OF NEIGHBORING PATCHES
      CALL RUNT1(LL12,LL20)
      NU=2.*RAD*CM1*DTM1+6.
      IF(NI.EQ.0) NI=6.*RAD/(C*DT)
      IF(NI.LT.0) NU=1
      PRINT *,  'DT=',DT,',  RAD=',RAD,',  ALPHA=',ALPHA,',  BETA=',BETA,
     1 ',  NI=',NI,',  N1=',N1,',  N2=',N2,',  N3=',N3
      PRINT *, NU,' ROWS NEEDED'
C     FIND DIMENSION OF ARRAY NEEDED FOR FAR-FIELD COMPUTATIONS
  400 READ(2,4,END=500) THDIR,PHDIR
      ITH=THDIR
      IPHI=PHDIR
      THDIR=THDIR*CONV
      PHDIR=PHDIR*CONV
      CALL FARFL
      IF(NTOT.GT.NTM) NTM=NTOT
      GO TO 400
  500 LL=IS+2
C     ADD 2 BECAUSE COMPUTATIONS ON 855 AND 205 SOMETIMES DISAGREE
      LC=MAXO(NU,NTM)+2
C     PRELIMINARY CALCULATIONS ONLY
      IF(NI.LT.0) LC=1
C     FILE WITH DIRECTIVES FOR THE FULL SCREEN EDITOR
      OPEN(5,FILE='SCTFSE',STATUS='NEW')
      LB=LL**2/64+1
      LB1=LL/64+1
C     COMPUTE NUMBER OF LARGE PAGES
      NLP=(LL*(2*LC+2*LL20+55)+20*LL12+2*(LB+LB1)+65539)/65536
      PRINT *,  'LL=',LL,',  LC=',LC,',  LL12=',LL12,',  LL20=',LL20
      PRINT *,  'COMPUTED NLP=',NLP
      IF(LL.GT.99999.OR.LC.GT.9999.OR.LL12.GT.999999.OR.LL20.GT.999)
```

```
      1 STOP 10
       WRITE(5,6) LL,LC,LL12,LL20
C      IF NUMBER OF LARGE PAGES IS GIVEN
       IF(NLP1.NE.0) NLP=NLP1
       IF(NLP.GT.28.OR.NTL.GT.599940) STOP
C      ASSIGN TIME LIMITS (IF NOT GIVEN) AND PRIORITIES
       IF(NLP.LE.5) THEN
         IF(NTL.EQ.0) NTL=500
         IF(NTL.LE.900) THEN
           JCAT='P2'
         ELSE IF(NTL.LE.3600) THEN
           JCAT='P3'
         ELSE
           JCAT='P5'
         END IF
       ELSE IF(NLP.LE.10) THEN
         IF(NTL.EQ.0) NTL=900
         IF(NTL.LE.3600) THEN
           JCAT='P3'
         ELSE
           JCAT='P5'
         END IF
       ELSE IF(NLP.LE.20) THEN
         IF(NTL.EQ.0) NTL=900
         IF(NTL.LE.900) THEN
           JCAT='P4'
         ELSE
           JCAT='P5'
         END IF
       ELSE
         IF(NTL.EQ.0) NTL=5000
         JCAT='P5'
       END IF
       IF(PRI.NE.'XX') THEN
         IF(PRI.LT.JCAT) THEN
           PRINT *, 'ASSIGNED PRIORITY TOO LOW'
           PRINT *, 'PRI=',PRI,', JCAT=',JCAT
           STOP
         ELSE
           JCAT=PRI
         END IF
       END IF
       OPEN(4,FILE='SCTRES',STATUS='NEW')
       WRITE(4,*) 'RESOURCE,TL=',NTL,',LP=',NLP,',JCAT=',JCAT,
      1 ',NT=',NT,','
       PRINT *,'PRIORITY ',JCAT,', TIME LIMIT ',NTL,', ',NT,' TAPE'
       STOP
  4    FORMAT(E12.5)
  5    FORMAT(3I4)
  6    FORMAT('REPLACE/LLG/',I5,'/ALL QUIET',/,'REPLACE/LCG/',I4,
      1 '/ALL QUIET',/,'REPLACE/LL12G/',I6,'/ALL QUIET',/,
      2 'REPLACE/LL20G/',I3,'/ALL QUIET',/,'QUIT')
       END
C
```

```
      SUBROUTINE RUNT1(LL12,LL20)
C
C     THIS SUBROUTINE DETERMINES THE NUMBER OF PATCHES AND THE DISTRIBUTION
C     NEIGHBORING PATCHES IF ICORR = 2 OR 3
C
      DIMENSION NPHI(100)
      DIMENSION XR1(5000),XR2(5000),XR3(5000),THETA(5000),PHII(5000)
      COMMON /PARAM/ RAD,ALPHA,BETA,DT,DTM1,THDIR
      COMMON /ICOM/ IX,IS,IS1,NTOT,N1,N2,N3,NI,ISKIP,IPATCH,ICORR
      COMMON /CONST/ C,CM1,PI,TOPI
C
      DTH=PI/(N1-1)
      SDTH2=SIN(DTH*.5)
      TH=PI
      IS=2
      NPHI(1)=1
      NPHI(N1)=1
      N1H=(N1+1)/2
      DPH=1.
      IF(IPATCH.EQ.0) THEN
        NPHX=1
 90     NPHX=NPHX*2
        IF(NPHX.LT.N3) GO TO 90
        DO 100 I=2,N1H
          TH=TH-DTH
          STH=SIN(TH)
          NPH=(N2-N3)*STH+N3+.5
          IF(NPH.GT.NPHX) NPHX=NPHX*2
          NPHI(I)=NPHX
          IS=IS+NPHX
          DPH=AMIN1(DPH,STH*SIN(PI/NPHX))
 100    CONTINUE
      ELSE
        DO 105 I=2,N1H
          TH=TH-DTH
          STH=SIN(TH)
          NPH=(N2-N3)*STH+N3+.5
          NPHI(I)=NPH
          IS=IS+NPH
          DPH=AMIN1(DPH,STH*SIN(PI/NPH))
 105    CONTINUE
      END IF
      IF(MOD(N1,2).EQ.1) N1H=N1H-1
      DO 110 J=1,N1-1
        NPH=NPHI(N1H)
        NPHI(J)=NPH
        IS=IS+NPH
        N1H=N1H-1
 110  CONTINUE
      DT1=1.6*RAD*CM1*AMIN1(SDTH2,DPH)
      IF(DT.EQ.0.) THEN
        DT=DT1
      ELSE
        IF(DT1.LT.DT) THEN
```

```
          PRINT 3, DT,DT1
          NI=NI*DT/DT1
          DT=DT1
        END IF
      END IF
      DTM1=1./DT
      PRINT *, 'NUMBER OF PATCHES IS ',IS
C     RETURN IF NO NEIGHBORING-PATCH CORRECTIONS ARE GOING TO BE MADE
      IF(ICORR.LT.2.OR.ISKIP.EQ.1) THEN
        LL12=1
        LL20=1
        RETURN
      END IF
C     CHECK IF ARRAYS ARE BIG ENOUGH TO CALCULATE THE NUMBER OF NEIGHBORING
C     PATCHES
      IF(IS.GT.5000) STOP 5
      IS1=IS-NPH
      TH=PI
      IN=1
      XR1(1)=0.
      XR2(1)=0.
      XR3(1)=-RAD
      THETA(1)=PI
      PHII(1)=0.
      NP2=1
      NP3=NPHI(2)
      INB2=1
      INB3=2
      DPHI2=TOPI
      DPHI3=TOPI/NP3
      DO 200 I=2,NI-1
        TH=TH-DTH
        STH=SIN(TH)
        CTH=COS(TH)
        RSTH=RAD*STH
        RCTH=RAD*CTH
        NP1=NP2
        NP2=NP3
        NP3=NPHI(I+1)
        INB1=INB2
        INB2=INB3
        INB3=INB2+NP2
        DPHI1=DPHI2
        DPHI2=DPHI3
        DPHI3=TOPI/NP3
        PHI=0
        DO 190 J=1,NP2
          IN=IN+1
          CPH=COS(PHI)
          SPH=SIN(PHI)
          XR1(IN)=RSTH*CPH
          XR2(IN)=RSTH*SPH
          XR3(IN)=RCTH
          THETA(IN)=TH
```

```
          PHII(IN)=PHI
          PHI=PHI+DPHI2
  190   CONTINUE
  200 CONTINUE
      XR1(IS)=0.
      XR2(IS)=0.
      XR3(IS)=RAD
      THETA(IS)=0.
      PHII(IS)=0.
      DL1=DTH*RAD
      NMAX=0
      IN=1
      NPNT=0
      NP=0
      DO 240 I=1,IS
        X=XR1(I)
        Y=XR2(I)
        Z=XR3(I)
        TH=THETA(I)
        PHI=PHII(I)
        NM=1
        IF(NP.EQ.0) THEN
          NPH=NPHI(IN)
          DPH=TOPI/NPH
          DPHPR=DPH*1.1
          DL2=DPH*SIN(TH)*RAD
          RMAX=SQRT(2.)*AMAX1(DL1,DL2)
          IF(I.EQ.1.OR.I.EQ.IS) RMAX=1.5*DTH*RAD
          NP=NPH
          IN=IN+1
        END IF
        NP=NP-1
        DO 220 J=1,IS
          DR=SQRT((X-XR1(J))**2+(Y-XR2(J))**2+(Z-XR3(J))**2)
          IF(J.EQ.I.OR.DR.GT.RMAX) GO TO 220
          NM=NM+1
          NPNT=NPNT+1
  220   CONTINUE
        IF(NM.GT.NMAX) NMAX=NM
  240 CONTINUE
      PRINT *, 'NUMBER OF COLUMNS NEEDED IS ',NMAX-1,
     1 ' TOTAL NUMBER OF NEIGHBORING PATCHES IS ',NPNT
      LL12=NPNT
      LL20=NMAX
      RETURN
    3 FORMAT(' GIVEN DT=',E9.2,' GREATER THAN "MINIMUM" DT=',E9.2)
      END
C
      SUBROUTINE FARFL
C
      COMMON /PARAM/ RAD,ALPHA,BETA,DT,DTMJ,THDIR
      COMMON /ICOM/ IX,IS,IS1,NTOT,N1,N2,N3,NI,ISKIP,IPATCH,ICORR
      COMMON /CONST/ C,CM1,PI,TOPI
      READ(2,1) T1,T2
```

```
C       COMPUTE FIRST TIME OF ARRIVAL IF T1 = 0
        IF(T1.EQ.0.) T1=RAD*CM1*(1.-2.*SIN(.5*THDIR))-DT
C       COMPUTE LAST TIME IF T2 = 0
        IF(T2.EQ.0.) T2=(NI-5)*DT-RAD*CM1
        READ(2,2) N
        NMAX=(T1+RAD*CM1)/DT+4.
        NMIN=(T1-RAD*CM1)/DT-3.
        NTOT=NMAX-NMIN+1
        PRINT *, 'NTOT=',NTOT
        RETURN
  1     FORMAT(E12.5)
  2     FORMAT(2I4)
        END
```

```
      PROGRAM PERF
C     COMPUTES MIE SCATTERING FOR A PERFECT CONDUCTOR
      PARAMETER(NW=8192,NTM=3000,NT4=1200)
      CHARACTER XLAB*30,YLAB*30,PTLAB1*80,PTLAB2*80,SBLAB*80
      REAL PTI(NTM),TI(NTM),AMP1(NTM),AMP2(NTM),WW(NW),
     1 S1(NW),S2(NW),S3(NW),S4(NW)
      DIMENSION PI(5,NT4),P(NT4),TAU(5,NT4),X(NT4)
      COMPLEX FT1(NW),FT2(NW),FAC,EYE
      COMMON/PERAR/ANGLE(5),AG1R(5,NW),AG1I(5,NW),
     1 AG2R(5,NW),AG2I(5,NW),
     2 BES(NT4),ENEU(NT4),RA(NT4),EIA(NT4),RB(NT4),EIB(NT4),
     3 PTI,TI,AMP1,AMP2,WW,FT1,FT2,S1,S2,S3,S4,PI,P,TAU,X
      COMMON/VAR/ NANGL,RHO,EM
      DATA XLAB/'C*TIME (METERS) '/,YLAB/'INTENSITY    '/
      DATA PTLAB1/'SCATTERED INTENSITY, HORIZONTAL POLARIZATION'/
      DATA PTLAB2/'SCATTERED INTENSITY, VERTICAL POLARIZATION  '/
      DATA C/3.E8/,EYE/(0.,1.)/
      CM1=1./C
      READ *, IPLOT
      OPEN(7,FILE='PERFSV',FORM='UNFORMATTED')
      READ(*,1) CTMAX,CTMIN,R1,ALPHA,BETA
      READ(*,2) NT,NSTAR,NANGL
      CDT=(CTMAX-CTMIN)/(NT-1)
      IF(NANGL.GT.5) STOP 1
      IF(NSTAR.GT.NW) STOP 8
      IF(NT.GT.NTM) STOP 3
      READ *, (ANGLE(I),I=1,NANGL)
      READ(*,1) DW,WMAX
      NREC=1
      CTIM=CTMIN
      DO 90 I=1,NT
        PTI(I)=CTIM
        TI(I)=PTI(I)*CM1
        CTIM=CTIM+CDT
90    CONTINUE
      CALL OMEGA(WW,WMAX,NSTAR,DW)
      B2A2C2=(BETA**2-ALPHA**2)*C**2
      BAC=-(BETA-ALPHA)*C*2.
      ABC=-(ALPHA+BETA)*C
      BC2=BETA*C**2
      ABC2=ALPHA*BC2
      MMAX=WMAX*R1/C+7.
      IF(MMAX.GT.NT4-2) THEN
        PRINT *, 'MMAX ',MMAX
        STOP 20
      END IF
      IX=0
      TL=SECOND()
      DO 100 IW=2,NSTAR
        RHO=R1*WW(IW)/C
        EM=RHO+7.
        M=EM+.5
        IF(M.GT.NT4-2) STOP 21
        IX=IX+1
```

```
        CALL BESRJ(RHO,.5,M+2,1,BES,NCALC)
        IF(NCALC.NE.M+2) THEN
          PRINT *, 'ERROR IN COMPUTATION OF BESSEL FUNCTIONS ',NCALC
          STOP 10
        END IF
        CALL BESRY(RHO,.5,M+2,1,ENEU,NCALC)
        IF(NCALC.NE.M+2) THEN
          PRINT *,'ERROR IN COMPUTATION OF NEUMANN FUNCTIONS ',NCALC
          STOP 11
        END IF
C     COMPUTATION OF SCATTERING COEFFICIENTS
        CALL SCACO
C     COMPUTATION OF MONOCHROMATIC FIELDS
        CALL FLD(IW,M)
100   CONTINUE
      TL=SECOND()-TL
      PRINT *,'TIME TO COMPUTE MONOCHROMATIC FIELDS IS ',TL,' SECONDS'
        DO 210 IANG=1,NANGL
          FT1(1)=0.
          FT2(1)=0.
          DO 200 I=2,NSTAR
          W=WW(I)
          WRC=W*R1/C
          FAC=CMPLX(B2A2C2,BAC*W)/CMPLX(ABC2-W**2,ABC*W)**2
          FAC=FAC*CMPLX(COS(WRC),SIN(WRC))
          FAC=BC2*FAC/W
          FT1(I)=EYE*CMPLX(AG1R(IANG,I),AG1I(IANG,I))*FAC
          FT2(I)=EYE*CMPLX(AG2R(IANG,I),AG2I(IANG,I))*FAC
200       CONTINUE
          TL=SECOND()
          CALL VINUFT(TI,AMP1,NT,NSTAR,WW,1,FT1,S1,S2,S3,S4,-1)
          TL=SECOND()-TL
          PRINT *,'TIME TO COMPUTE A FOURIER TRANSFORM IS ',TL,'SECONDS'
          WRITE(7) AMP1
C       COMPUTE OTHER POLARIZATION UNLESS THETA = 0, 180
          IF(ANGLE(IANG).NE.0..AND.ANGLE(IANG).NE.180.) THEN
            TL=SECOND()
            CALL VINUFT(TI,AMP2,NT,NSTAR,WW,1,FT2,S1,S2,S3,S4,-1)
            TL=SECOND()-TL
            PRINT *,'TIME TO COMPUTE A FOURIER TRANSFORM IS ',TL,'SECONDS'
            WRITE(7) AMP2
          END IF
210     CONTINUE
        REWIND 7
        DO 300 IANG=1,NANGL
          WRITE(SBLAB,3) ALPHA,BETA,R1,ANGLE(IANG)
          IF(ANGLE(IANG).NE.0..AND.ANGLE(IANG).NE.180.) THEN
            READ(7) AMP2
            READ(7) AMP1
          ELSE
            READ(7) AMP1
          END IF
          DO 220 I=1,NT
            AMP1(I)=AMP1(I)**2
```

```
220       CONTINUE
          IF(IPLOT.EQ.1.OR.IPLOT.EQ.2) THEN
            CALL DRAW4(1,2,16,12,44,72,XLAB,YLAB,PTLAB1,SBLAB)
            CALL DRAW4A(1,1,NT,'   ',0,0,PTI,AMP1,0.,0.)
            CALL DRAW4B(1,0,0,0,NT,PTI,AMP1,2.,2.)
          END IF
          IF(IPLOT.EQ.0.OR.IPLOT.EQ.2) THEN
            CALL DRAW3(1,2,16,12,44,72,XLAB,YLAB,PTLAB1,SBLAB)
            CALL DRAW3A(1,1,NT,'   ',0,0,PTI,AMP1,0.,0.)
            CALL DRAW3B(1,0,0,0,NT,PTI,AMP1,2.,2.)
          END IF
          IF(ANGLE(IANG).NE.0..AND.ANGLE(IANG).NE.180.) THEN
            DO 260 I=1,NT
              AMP2(I)=AMP2(I)**2
260         CONTINUE
            IF(IPLOT.EQ.1.OR.IPLOT.EQ.2) THEN
              CALL DRAW4(1,2,16,12,44,72,XLAB,YLAB,PTLAB2,SBLAB)
              CALL DRAW4A(1,1,NT,'   ',0,0,PTI,AMP2,0.,0.)
              CALL DRAW4B(1,0,0,0,NT,PTI,AMP2,2.,2.)
            END IF
            IF(IPLOT.EQ.0.OR.IPLOT.EQ.2) THEN
              CALL DRAW3(1,2,16,12,44,72,XLAB,YLAB,PTLAB2,SBLAB)
              CALL DRAW3A(1,1,NT,'   ',0,0,PTI,AMP2,0.,0.)
              CALL DRAW3B(1,0,0,0,NT,PTI,AMP2,2.,2.)
            END IF
          END IF
300     CONTINUE
        CLOSE(7)
        CALL Q5PURGE('LFN=','PERFSV','STATUS=',ISTAT)
        IF(ISTAT.GT.0.AND.ISTAT.NE.1402) STOP 4
        CALL Q5DEFINE('LFN=','PERFSV')
        STOP
1       FORMAT(E12.5)
2       FORMAT(I4)
3       FORMAT('ALPHA = ',F5.2,'          BETA = ',F5.2,
       1 '       RADIUS = ',F4.2,'        ANGLE = ',F4.0)
        END
C
        SUBROUTINE SCACO
        PARAMETER(NW=912,NTM=3000,NT4=1200)
        REAL PTI(NTM),TI(NTM),AMP1(NTM),AMP2(NTM),WW(NW),
       1 S1(NW),S2(NW),S3(NW),S4(NW)
        DIMENSION PI(5,NT4),P(NT4),TAU(5,NT4),X(NT4)
        COMPLEX FT1(NW),FT2(NW),FAC,EYE
        COMMON/PERAR/ANGLE(5),AG1R(5,NW),AG1I(5,NW),
       1 AG2R(5,NW),AG2I(5,NW),
       2 BES(NT4),ENEU(NT4),RA(NT4),EIA(NT4),RB(NT4),EIB(NT4),
       3 PTI,TI,AMP1,AMP2,WW,FT1,FT2,S1,S2,S3,S4,PI,P,TAU,X
        COMMON/VAR/ NANGL,RHO,EM
        IERR=0
        M=EM
C          CALCULATION FOR PERFECTLY CONDUCTING SPHERES
        DO 82 I=1,M
          AY=I
```

```
      ATER=(AY+1. )*BES(I+1)-RHO*BES(I+2)
      BTER=(AY+1. )*ENEU(I+1)-RHO*ENEU(I+2)
      A2=ATER**2
      FRAC=1. /(A2+BTER**2)
      RB(I)=-A2*FRAC
      EIB(I)=ATER*BTER*FRAC
      ATER=BES(I+1)
      BTER=ENEU(I+1)
      A2=ATER**2
      FRAC=1. /(A2+BTER**2)
      RA(I)=-A2*FRAC
      EIA(I)=ATER*BTER*FRAC
82    CONTINUE
      RETURN
      END
C
      SUBROUTINE FLD(IW,LMAX)
      PARAMETER(NW=8192,NTM=3000,NT4=1200)
      REAL PTI(NTM),TI(NTM),AMP1(NTM),AMP2(NTM),WW(NW),
     1 S1(NW),S2(NW),S3(NW),S4(NW)
      DIMENSION PI(5,NT4),P(NT4),TAU(5,NT4),X(NT4)
      COMPLEX FT1(NW),FT2(NW),FAC,EYE
      COMMON/PERAR/ANGLE(5),AG1R(5,NW),AG1I(5,NW),
     1 AG2R(5,NW),AG2I(5,NW),
     2 BES(NT4),ENEU(NT4),RA(NT4),EIA(NT4),RB(NT4),EIB(NT4),
     3 PTI,TI,AMP1,AMP2,WW,FT1,FT2,S1,S2,S3,S4,PI,P,TAU,X
      COMMON/VAR/ NANGL,RHO,EM
      DO 3 J=1,NANGL
      THETA=ANGLE(J)*    1.745329251994E-2
      P(1)=COS (THETA)
      PI(J,1)=1.
      TAU(J,1)=P(1)
      P(2)=1.5*P(1)**2-.5
      PI(J,2)=3. *P(1)
      TAU(J,2)=4. *P(2)-1.
      DO 2 L=3,LMAX
      EL=L
      P(L)=(2. *EL-1. )/EL*P(1)*P(L-1)-(EL-1. )/EL*P(L-2)
      PI(J,L)=P(1)*PI(J,L-1)+EL*P(L-1)
    2 TAU(J,L)=EL*(EL+1. )*P(L)-P(1)*PI(J,L)
    3 CONTINUE
      DO 4 L=1,LMAX
      EL=L
    4 X(L)=(2. *EL+1. )/(EL*(EL+1. ))
      M=EM+.0001
      DO 6 J= 1,NANGL
      AG1R(J,IW)=0.
      AG1I(J,IW)=0.
      AG2R(J,IW)=0.
      AG2I(J,IW)=0.
      DO 6 L=1,M
      AG1R(J,IW)=AG1R(J,IW)+X(L)*(RA(L)*PI(J,L)+RB(L)*TAU(J,L))
      AG1I(J,IW)=AG1I(J,IW)+X(L)*(EIA(L)*PI(J,L)+EIB(L)*TAU(J,L))
      AG2R(J,IW)=AG2R(J,IW)+X(L)*(RB(L)*PI(J,L)+RA(L)*TAU(J,L))
```

```
      AG2I(J, IW)=AG2I(J, IW)+X(L)*(EIB(L)*PI(J, L)+EIA(L)*TAU(J, L))
6     CONTINUE
      RETURN
      END
```

```
      PROGRAM PERFPLT(INPUT,OUTPUT)
C     PLOTS PREVIOUSLY CALCULATED SCATTERED FIELDS
      CHARACTER XLAB*30,YLAB*30,PTLAB1*80,PTLAB2*80,SBLAB*80
      DIMENSION PTI(3000),AMP01(3000),AMP02(3000),
     1 T1(300),T2(300),T3(300),T4(300),AMP1(300),AMP2(300),
     2 AMP3(300),AMP4(300),ANGLE(5)
      LOGICAL TWO,NEWT1,NEWT2
      CHARACTER*4 QL1,QL2,QL3,QL4,SL(4)
      CHARACTER*40 L(6)
      DATA XLAB/'C*TIME (METERS) '/,YLAB/'INTENSITY    '/
      DATA PTLAB1/'SCATTERED INTENSITY, HORIZONTAL POLARIZATION'/
      DATA PTLAB2/'SCATTERED INTENSITY, VERTICAL POLARIZATION  '/
      DATA C/3.E8/,NEWT1,NEWT2/2*.FALSE./
      READ *, QL1,QL2,QL3,QL4,CT1,CT2,IPLOT,INVER,(L(I),I=1,6)
      PRINT *,'COMPARISON OF PLOTS PRODUCED BY PERF WITH THOSE',
     1 ' PRODUCED BY SCAT, IF ANY'
      PRINT *,'QUALIFIERS ',QL1,QL2,QL3,QL4
      DO 100 I=1,6
        PRINT 4
        PRINT *, L(I)
100   CONTINUE
      PRINT 4
      IF(CT1.NE.0..AND.CT2.NE.0..AND.CT1.GT.CT2) THEN
        PRINT *,'INITIAL TIME ',CT1,' GREATER THAN FINAL TIME ',CT2
        STOP 10
      END IF
      IF(CT1.NE.0.) NEWT1=.TRUE.
      IF(CT2.NE.0.) NEWT2=.TRUE.
      CALL Q5ATTACH('LFN=','PERFSV')
      OPEN(7,FILE='PERFSV',FORM='UNFORMATTED',STATUS='OLD')
      READ 1, CTMAX,CTMIN,R1,ALPHA,BETA
      READ 2, NT,NSTAR,NANGL
      CDT=(CTMAX-CTMIN)/(NT-1)
      IF(NANGL.GT.5) STOP 1
      IF(NT.GT.3000) STOP 3
      READ *, (ANGLE(I),I=1,NANGL)
      READ 1, DW,WMAX
      NREC=1
      NQ=0
      IF(QL1.NE.'NO') THEN
        NQ=NQ+1
        SL(NQ)=QL1
        CALL Q5ATTACH('LFN=','SCFL'//QL1)
        OPEN(8,FILE='SCFL'//QL1,STATUS='OLD',
     1  ACCESS='DIRECT',FORM='UNFORMATTED',RECL=2400)
      END IF
      IF(QL2.NE.'NO') THEN
        NQ=NQ+1
        SL(NQ)=QL2
        CALL Q5ATTACH('LFN=','SCFL'//QL2)
        OPEN(9,FILE='SCFL'//QL2,STATUS='OLD',
     1  ACCESS='DIRECT',FORM='UNFORMATTED',RECL=2400)
      END IF
      IF(QL3.NE.'NO') THEN
```

```
      NQ=NQ+1
      SL(NQ)=QL3
      CALL Q5ATTACH('LFN=','SCFL'//QL3)
      OPEN(10,FILE='SCFL'//QL3,STATUS='OLD',
     1  ACCESS='DIRECT',FORM='UNFORMATTED',RECL=2400)
     END IF
     IF(QL4.NE.'NO') THEN
      NQ=NQ+1
      SL(NQ)=QL4
      CALL Q5ATTACH('LFN=','SCFL'//QL4)
      OPEN(11,FILE='SCFL'//QL4,STATUS='OLD',
     1  ACCESS='DIRECT',FORM='UNFORMATTED',RECL=2400)
     END IF
     DO 500 IANG=1,NANGL
     WRITE(SBLAB,3) ALPHA,BETA,R1,ANGLE(IANG),(SL(I),I=1,NQ)
     IF(ANGLE(IANG).NE.0..AND.ANGLE(IANG).NE.180.) THEN
      TWO=.TRUE.
      IF(INVER.EQ.1) THEN
       READ(7,END=600) AMP02
       READ(7) AMP01
      ELSE
       READ(7,END=600) AMP01
       READ(7) AMP02
      END IF
     ELSE
      TWO=.FALSE.
      READ(7,END=600) AMP01
     ENDIF
     CTIM=CTMIN
     DO 210 I=1,NT
      AMP01(I)=AMP01(I)**2
      PTI(I)=CTIM
      CTIM=CTIM+CDT
210  CONTINUE
     IF(NEWT1) THEN
      DO 222 IO1=1,NT
       IF(PTI(IO1).GE.CT1) GO TO 223
222   CONTINUE
      PRINT *,'INITIAL TIME ',CT1,' TOO LARGE'
      STOP 10
     ELSE
      IO1=1
     END IF
223  IF(NEWT2) THEN
      DO 224 IO2=NT,1,-1
       IF(PTI(IO2).LE.CT2) GO TO 225
224   CONTINUE
      PRINT *,'FINAL TIME ',CT2,' TOO SMALL'
      STOP 10
     ELSE
      IO2=NT
     END IF
225  IF(IO1.GT.1) THEN
      DO 226 I3=IO1,IO2
```

```
          AMP01(I3-I01+1)=AMP01(I3)
          PTI(I3-I01+1)=PTI(I3)
226     CONTINUE
      END IF
      NTP=I02-I01+1
      IF(QL1.NE.'NO') THEN
        READ(8,END=600) T1
        READ(8) AMP1
        DO 240 NT1=1,300
          IF(AMP1(NT1).LT.0.) GO TO 245
240     CONTINUE
245     NT1=NT1-1
        IF(NEWT1) THEN
          DO 252 I1=1,NT1
            IF(T1(I1).GE.CT1) GO TO 253
252       CONTINUE
        ELSE
          I1=1
        END IF
253     IF(NEWT2) THEN
          DO 254 I2=NT1,1,-1
            IF(T1(I2).LE.CT2) GO TO 255
254       CONTINUE
        ELSE
          I2=NT1
        END IF
255     IF(I1.NE.1) THEN
          DO 256 I3=I1,I2
            AMP1(I3-I1+1)=AMP1(I3)
            T1(I3-I1+1)=T1(I3)
256       CONTINUE
        END IF
        NT1=I2-I1+1
      END IF
      IF(QL2.NE.'NO') THEN
        READ(9,END=600) T2
        READ(9) AMP2
        DO 260 NT2=1,300
          IF(AMP2(NT2).LT.0.) GO TO 265
260     CONTINUE
265     NT2=NT2-1
        IF(NEWT1) THEN
          DO 272 I1=1,NT2
            IF(T2(I1).GE.CT1) GO TO 273
272       CONTINUE
        ELSE
          I1=1
        END IF
273     IF(NEWT2) THEN
          DO 274 I2=NT2,1,-1
            IF(T2(I2).LE.CT2) GO TO 275
274       CONTINUE
        ELSE
          I2=NT2
```

```
        END IF
275     IF(I1.NE.1) THEN
          DO 276 I3=I1,I2
            AMP2(I3-I1+1)=AMP2(I3)
            T2(I3-I1+1)=T2(I3)
276       CONTINUE
        END IF
        NT2=I2-I1+1
      END IF
      IF(QL3.NE.'NO') THEN
        READ(10,END=600) T3
        READ(10) AMP3
        DO 280 NT3=1,300
          IF(AMP3(NT3).LT.0.) GO TO 285
280     CONTINUE
285     NT3=NT3-1
        IF(NEWT1) THEN
          DO 292 I1=1,NT3
            IF(T3(I1).GE.CT1) GO TO 293
292       CONTINUE
        ELSE
          I1=1
        END IF
293     IF(NEWT2) THEN
          DO 294 I2=NT3,1,-1
            IF(T3(I2).LE.CT2) GO TO 295
294       CONTINUE
        ELSE
          I2=NT3
        END IF
295     IF(I1.NE.1) THEN
          DO 296 I3=I1,I2
            AMP3(I3-I1+1)=AMP3(I3)
            T3(I3-I1+1)=T3(I3)
296       CONTINUE
        END IF
        NT3=I2-I1+1
      END IF
      IF(QL4.NE.'NO') THEN
        READ(11,END=600) T4
        READ(11) AMP4
        DO 300 NT4=1,300
          IF(AMP4(NT4).LT.0.) GO TO 305
300     CONTINUE
305     NT4=NT4-1
        IF(NEWT1) THEN
          DO 312 I1=1,NT4
            IF(T4(I1).GE.CT1) GO TO 313
312       CONTINUE
        ELSE
          I1=1
        END IF
313     IF(NEWT2) THEN
          DO 314 I2=NT4,1,-1
```

```
            IF(T4(I2).LE.CT2) GO TO 315
314         CONTINUE
          ELSE
            I2=NT4
          END IF
315       IF(I1.NE.1) THEN
            DO 316 I3=I1,I2
              AMP4(I3-I1+1)=AMP4(I3)
              T4(I3-I1+1)=T4(I3)
316         CONTINUE
          END IF
          NT4=I2-I1+1
        END IF
        NREC=NREC+2
        NM1=NT1/20+4
        NM2=NM1+1
        NM3=NM2+1
        NM4=NM3+1
        IF(IPLOT.EQ.1.OR.IPLOT.EQ.2) THEN
          CALL DRAW4(1,1,16,12,44,80,XLAB,YLAB,PTLAB1,SBLAB)
          CALL DRAW4A(1,1,NTP,'   ',0,0,PTI,AMP01,0.,0.)
          IF(QL1.NE.'NO') CALL DRAW4A(1,1,NT1,' 0',NM1,1,T1,AMP1,0.,0.)
          IF(QL2.NE.'NO') CALL DRAW4A(1,1,NT2,' 1',NM2,2,T2,AMP2,0.,0.)
          IF(QL3.NE.'NO') CALL DRAW4A(1,1,NT3,' 2',NM3,3,T3,AMP3,0.,0.)
          IF(QL4.NE.'NO') CALL DRAW4A(1,1,NT4,' 3',NM4,4,T4,AMP4,0.,0.)
          CALL DRAW4B(1,0,0,0,3000,PTI,AMP01,2.,2.)
        END IF
        IF(IPLOT.EQ.0.OR.IPLOT.EQ.2) THEN
          CALL DRAW3(1,1,16,12,44,80,XLAB,YLAB,PTLAB1,SBLAB)
          CALL DRAW3A(1,1,NTP,'   ',0,0,PTI,AMP01,0.,0.)
          IF(QL1.NE.'NO') CALL DRAW3A(1,1,NT1,' 0',NM1,1,T1,AMP1,0.,0.)
          IF(QL2.NE.'NO') CALL DRAW3A(1,1,NT2,' 1',NM2,2,T2,AMP2,0.,0.)
          IF(QL3.NE.'NO') CALL DRAW3A(1,1,NT3,' 2',NM3,3,T3,AMP3,0.,0.)
          IF(QL4.NE.'NO') CALL DRAW3A(1,1,NT4,' 3',NM4,4,T4,AMP4,0.,0.)
          CALL DRAW3B(1,0,0,0,3000,PTI,AMP01,2.,2.)
        END IF
        IF(TWO) THEN
          DO 320 I=1,NT
            AMP02(I)=AMP02(I)**2
320       CONTINUE
          IF(I01.NE.1) THEN
            DO 326 I3=I01,I02
              AMP02(I3-I01+1)=AMP02(I3)
326         CONTINUE
          END IF
          IF(QL1.NE.'NO') THEN
            READ(8,END=600) T1
            READ(8) AMP1
            DO 340 NT1=1,300
              IF(AMP1(NT1).LT.0.) GO TO 345
340         CONTINUE
345         NT1=NT1-1
            IF(NEWT1) THEN
              DO 352 I1=1,NT1
```

```
              IF(T1(I1).GE.CT1) GO TO 353
352           CONTINUE
            ELSE
              I1=1
            END IF
353         IF(NEWT2) THEN
              DO 354 I2=NT1,1,-1
                IF(T1(I2).LE.CT2) GO TO 355
354           CONTINUE
            ELSE
              I2=NT1
            END IF
355         IF(I1.NE.1) THEN
              DO 356 I3=I1,I2
                AMP1(I3-I1+1)=AMP1(I3)
                T1(I3-I1+1)=T1(I3)
356           CONTINUE
            END IF
            NT1=I2-I1+1
          END IF
          IF(QL2.NE.'NO') THEN
            READ(9,END=600) T2
            READ(9) AMP2
            DO 360 NT2=1,300
              IF(AMP2(NT2).LT.0.) GO TO 365
360         CONTINUE
365         NT2=NT2-1
            IF(NEWT1) THEN
              DO 372 I1=1,NT2
                IF(T2(I1).GE.CT1) GO TO 373
372           CONTINUE
            ELSE
              I1=1
            END IF
373         IF(NEWT2) THEN
              DO 374 I2=NT2,1,-1
                IF(T2(I2).LE.CT2) GO TO 375
374           CONTINUE
            ELSE
              I2=NT2
            END IF
375         IF(I1.NE.1) THEN
              DO 376 I3=I1,I2
                AMP2(I3-I1+1)=AMP2(I3)
                T2(I3-I1+1)=T2(I3)
376           CONTINUE
            END IF
            NT2=I2-I1+1
          END IF
          IF(QL3.NE.'NO') THEN
            READ(10,END=600) T3
            READ(10) AMP3
            DO 380 NT3=1,300
              IF(AMP3(NT3).LT.0.) GO TO 385
```

```
380         CONTINUE
385         NT3=NT3-1
            IF(NEWT1) THEN
              DO 392 I1=1,NT3
                IF(T3(I1).GE.CT1) GO TO 393
392           CONTINUE
            ELSE
              I1=1
            END IF
393         IF(NEWT2) THEN
              DO 394 I2=NT3,1,-1
                IF(T3(I2).LE.CT2) GO TO 395
394           CONTINUE
            ELSE
              I2=NT3
            END IF
395         IF(I1.NE.1) THEN
              DO 396 I3=I1,I2
                AMP3(I3-I1+1)=AMP3(I3)
                T3(I3-I1+1)=T3(I3)
396           CONTINUE
            END IF
            NT3=I2-I1+1
          END IF
          IF(QL4.NE.'NO') THEN
            READ(11,END=600) T4
            READ(11) AMP4
            DO 400 NT4=1,300
              IF(AMP4(NT4).LT.0.) GO TO 405
400         CONTINUE
405         NT4=NT4-1
            IF(NEWT1) THEN
              DO 412 I1=1,NT4
                IF(T4(I1).GE.CT1) GO TO 413
412           CONTINUE
            ELSE
              I1=1
            END IF
413         IF(NEWT2) THEN
              DO 414 I2=NT4,1,-1
                IF(T4(I2).LE.CT2) GO TO 415
414           CONTINUE
            ELSE
              I2=NT4
            END IF
415         IF(I1.NE.1) THEN
              DO 416 I3=I1,I2
                AMP4(I3-I1+1)=AMP4(I3)
                T4(I3-I1+1)=T4(I3)
416           CONTINUE
            END IF
            NT4=I2-I1+1
          END IF
          NREC=NREC+2
```

```
      NM1=NT1/20+4
      NM2=NM1+1
      NM3=NM2+1
      NM4=NM3+1
      IF(IPLOT.EQ.1.OR.IPLOT.EQ.2) THEN
        CALL DRAW4(1,1,16,12,44,80,XLAB,YLAB,PTLAB2,SBLAB)
        CALL DRAW4A(1,1,NTP,'  ',0,0,PTI,AMP02,0.,0.)
        IF(QL1.NE.'NO') CALL DRAW4A(1,1,NT1,' 0',NM1,1,T1,AMP1,0.,0.)
        IF(QL2.NE.'NO') CALL DRAW4A(1,1,NT2,' 1',NM2,2,T2,AMP2,0.,0.)
        IF(QL3.NE.'NO') CALL DRAW4A(1,1,NT3,' 2',NM3,3,T3,AMP3,0.,0.)
        IF(QL4.NE.'NO') CALL DRAW4A(1,1,NT4,' 3',NM4,4,T4,AMP4,0.,0.)
        CALL DRAW4B(1,0,0,0,3000,PTI,AMP02,2.,2.)
      END IF
      IF(IPLOT.EQ.0.OR.IPLOT.EQ.2) THEN
        CALL DRAW3(1,1,16,12,44,80,XLAB,YLAB,PTLAB2,SBLAB)
        CALL DRAW3A(1,1,NTP,'  ',0,0,PTI,AMP02,0.,0.)
        IF(QL1.NE.'NO') CALL DRAW3A(1,1,NT1,' 0',NM1,1,T1,AMP1,0.,0.)
        IF(QL2.NE.'NO') CALL DRAW3A(1,1,NT2,' 1',NM2,2,T2,AMP2,0.,0.)
        IF(QL3.NE.'NO') CALL DRAW3A(1,1,NT3,' 2',NM3,3,T3,AMP3,0.,0.)
        IF(QL4.NE.'NO') CALL DRAW3A(1,1,NT4,' 3',NM4,4,T4,AMP4,0.,0.)
        CALL DRAW3B(1,0,0,0,3000,PTI,AMP02,2.,2.)
      END IF
    END IF
500 CONTINUE
600 STOP
1   FORMAT(E12.5)
2   FORMAT(I4)
3   FORMAT('ALPHA=',F5.2,'  BETA=',F5.2,
   1 '  RADIUS=',F4.2,'  ANGLE=',F4.0:
   2 '   QUAL.:  ',A4:'  ',A4:'  ',A4:'  ',A4)
4   FORMAT(//)
    END
```

```
      PROGRAM                                CURCHK
      PARAMETER(LL=LLG)
C     LCG, LL12G, AND LL20G NOT NEEDED
      DIMENSION CJTH(900,10),CJPH(900,10),T(900)
      DIMENSION NPCH(10),CJA(LL)
      CHARACTER QAL*4,XLAB*30,YLAB*30,TITLE*80,SUBT*80
      COMMON /PASS/ N1,N2,N3,IS,IPATCH,RAD,C,DT1
      DATA XLAB/'C * TIME'/,YLAB/'SURFACE DENSITY'/,
     1 TITLE/'SURFACE CURRENT DENSITY'/
      DATA C/3.E8/
C
C     READ INFORMATION FOR CURRENT CHECK
C
      READ *, QAL,IPLOT,IPRINT,NN
      IF(NN.GT.10) STOP 1
      READ *, (NPCH(I),I=1,NN)
      READ *, CT1,CT2,IPATCH
C
C     READ IN VALUES OF CONSTANTS FOR THE PULSE
C
      READ 4, DT,RAD,ALPHA,BETA
      READ 5, NI
      READ 5, N1,N2,N3
      CALL RUNT2
      IF(DT.EQ.0.) DT=DT1
      IF(NI.EQ.0) NI=6.*RAD/(C*DT)
      CDT=C*DT
      NI1=CT1/CDT+1.
      NI2=CT2/CDT+1.
      CALL Q5ATTACH('LFN=','SCSV'//QAL)
      OPEN(2,FILE='SCSV'//QAL,STATUS='OLD',ACCESS='DIRECT',
     1 FORM='UNFORMATTED',RECL=IS*8)
      IF(NI2.EQ.1) NI2=NI
      IF(NI1.GE.NI2.OR.NI2.GT.900) THEN
        PRINT *,'NI1=',NI1,', NI2=',NI2,', CT1=',CT1,', CT2=',CT2
        STOP 2
      END IF
      NII=NI2-NI1+1
      NK=NII/12+2
      NK1=NK+1
      TT=(NI1-1)*CDT
      DO 200 I=NI1,NI2
        READ(2,REC=I*2-1) CJA(1;IS)
        DO 110 N=1,NN
          CJTH(I,N)=CJA(NPCH(N))
110     CONTINUE
        READ(2,REC=I*2) CJA(1;IS)
        DO 120 N=1,NN
          CJPH(I,N)=CJA(NPCH(N))
120     CONTINUE
        T(I)=TT
        TT=TT+CDT
200   CONTINUE
      IF(IPRINT.NE.0) THEN
```

```
      PRINT *, 'INPUT FILE SCIN',QAL
      PRINT *, 'DT=',DT,', RAD=',RAD,', ALPHA=',ALPHA,', BETA=',BETA,
     1  ', NI=',NI,', N1=',N1,', N2=',N2,', N3=',N3
      DO 220 N=1,NN
        NU=NPCH(N)
        DO 210 I=NI1,NI2
          PRINT 2, NU,I,CJTH(I,N),CJPH(I,N)
210     CONTINUE
        PRINT 3
220   CONTINUE
    END IF
    IF(IPRINT.NE.1) THEN
      DO 250 N=1,NN
        WRITE(SUBT,1) QAL,NPCH(N)
        IF(IPLOT.NE.1) THEN
          CALL DRAW3(1,7,8,15,23,33,XLAB,YLAB,TITLE,SUBT)
          CALL DRAW3A(1,1,NII,' 1',NK,0,T(NI1),CJTH(NI1,N),0.,0.)
          CALL DRAW3A(1,1,NII,' 2',NK1,0,T(NI1),CJPH(NI1,N),0.,0.)
          CALL DRAW3B(1,0,0,0,NI,CJA,T,2.,2.)
        END IF
        IF(IPLOT.NE.0) THEN
          CALL DRAW4(1,7,8,15,23,33,XLAB,YLAB,TITLE,SUBT)
          CALL DRAW4A(1,1,NII,' 1',NK,0,T(NI1),CJTH(NI1,N),0.,0.)
          CALL DRAW4A(1,1,NII,' 2',NK1,0,T(NI1),CJPH(NI1,N),0.,0.)
          CALL DRAW4B(1,0,0,0,NI,CJA,T,2.,2.)
        END IF
250   CONTINUE
    END IF
    STOP
1   FORMAT('QUALIFIER ',A4,', PATCH NUMBER ',I4)
2   FORMAT(1X,2I8,2(3X,1PE16.9))
3   FORMAT(///)
4   FORMAT(E12.5)
5   FORMAT(3I4)
    END
    SUBROUTINE RUNT2
C
C   THIS SUBROUTINE DETERMINES THE NUMBER OF PATCHES AND
C   DT FROM THE PATCH DISTRIBUTION ON THE SPHERE
C
    COMMON /PASS/ N1,N2,N3,IS,IPATCH,RAD,C,DT1
C
    PI=4.*ATAN(1.)
    DTH=PI/(N1-1)
    SDTH2=SIN(DTH*.5)
    TH=PI
    N1H=(N1+1)/2
    DPH=1.
    IS=2
    IF(IPATCH.EQ.0) THEN
      NPHX=1
 90   NPHX=NPHX*2
      IF(NPHX.LT.N3) GO TO 90
      DO 100 I=2,N1H
```

```
            TH=TH-DTH
            STH=SIN(TH)
            NPH=(N2-N3)*STH+N3+.5
            IF(NPH.GT.NPHX) NPHX=NPHX*2
            IS=IS+NPHX*2
            DPH=AMIN1(DPH,STH*SIN(PI/NPHX))
 100      CONTINUE
          IF(MOD(N1,2).EQ.1) IS=IS-NPHX
        ELSE
          DO 105 I=2,N1H
            TH=TH-DTH
            STH=SIN(TH)
            NPH=(N2-N3)*STH+N3+.5
            IS=IS+NPH*2
            DPH=AMIN1(DPH,STH*SIN(PI/NPH))
 105      CONTINUE
9         IF(MOD(N1,2).EQ.1) IS=IS-NPH
        END IF
        DT1=1.6*RAD/C*AMIN1(SDTH2,DPH)
        RETURN
        END
```

```
      SUBROUTINE VNUFT(X,Y,N,NU,OM,IOM,FT,DX,CX,SX,JFLAG)
C THIS SUBROUTINE CALCULATES THE FOURIER TRANSFORM OF THE
C FUNCTION GIVEN BY STRAIGHT LINES JOINING THE POINTS GIVEN BY THE
C ARRAYS X,Y.  THE NUMBER OF IMPUT POINTS IS N, THE FREQUENCY ARRAY OM IS
C PROVIDED AND HAS DIMENSION NU.  DX, CX, SX ARE REAL ARRAYS OF DIMENSION N
C THAT ARE USED FOR SCRATCH.  THE FOURIER TRANSFORM IS GIVEN BY THE
C COMPLEX ARRAY FT.
C IF IOM=1, INPUT AND OUTPUT OM ARE CIRCULAR FREQUENCIES
C IF IOM=2, INPUT ARRAY IS CIRCULAR FREQUENCY, OUTPUT IS FREQUENCY
C IF IOM=3, INPUT ARRAY IS FREQUENCY, OUTPUT IS CIRCULAR FREQUENCY
C IF IOM=4, INPUT AND OUTPUT OM ARE FREQUENCIES
C JFLAG=+1 IF THE FOURIER TRANSFORM HAS A FACTOR EXP(+IWT)
C JFLAG=-1 IF THE FOURIER TRANSFORM HAS A FACTOR EXP(-IWT)
      DIMENSION X(N),Y(N),DX(N),CX(N),SX(N),OM(NU)
      COMPLEX FT(NU)
      TOPI=8.*ATAN(1.)
      TPIN=1./TOPI
      IF(IOM.LE.2) GO TO 25
      OM(1;N)=OM(1;N)*TOPI
 25   NP=N-1
      DX(1;NP)=Q8VDELT(X(1;N);DX(1;NP))
      NO=1
      IF(OM(1).EQ.0.) THEN
        NO=2
        S=Q8SSUM(Q8VADJM(Y(1;N);NP)*DX(1;NP))
        FT(1)=CMPLX(S,0.)
      END IF
      DX(1;NP)=Q8VDELT(Y(1;N);NP)/DX(1;NP)
      DO 70 I=NO,NU
        W=OM(I)
        WIN=1./W
        CX(1;N)=VCOS(W*X(1;N);CX(1;N))
        SX(1;N)=VSIN(W*X(1;N);SX(1;N))
        S2R=Q8SSUM(Q8VDELT(CX(1;N);NP)*DX(1;NP))
        S2I=Q8SSUM(Q8VDELT(SX(1;N);NP)*DX(1;NP))
        S1R=Y(1)*CX(1)-Y(N)*CX(N)
        S1I=Y(1)*SX(1)-Y(N)*SX(N)
        FT(I)=CMPLX((-S1I+S2R*WIN)*WIN,(S1R+S2I*WIN)*WIN)
 70   CONTINUE
      IF(IOM.EQ.1.OR.IOM.EQ.3) GO TO 85
      OM(1;N)=OM(1;N)*TPIN
 85   IF(JFLAG.EQ.1) RETURN
      FT(1;NU)=VCONJG(FT(1;NU);FT(1;NU))
      RETURN
      END
C
      SUBROUTINE VINUFT(X,Y,N,NU,OM,IOM,FT,SCRR,SCRI,CX,SX,JFLAG)
C THIS SUBROUTINE CALCULATES THE INVERSE FOURIER TRANSFORM OF THE FUNCTION
C GIVEN BY STRAIGHT LINES JOINING THE POINTS GIVEN BY THE COMPLEX ARRAY FT
C AS A FUNCTION OF THE REAL ARRAY OM.  THE NUMBER OF INPUT POINTS IS NU.
C THE INVERSE F.T.  IS ASSUMED TO BE A REAL FUNCTION AND GOES IN THE
C ARRAY Y, AND THE REAL ARRAY X IS GIVEN AND HAS DIMENSION N.
C SCRR, SCRI ARE REAL ARRAYS OF DIMENSION NU AND IS USED FOR SCRATCH.
C IF IOM=1, INPUT AND OUTPUT OM ARE CIRCULAR FREQUENCIES
```

```
C IF IOM=2, INPUT ARRAY IS CIRCULAR FREQUENCY, OUTPUT IS FREQUENCY
C IF IOM=3, INPUT ARRAY IS FREQUENCY, OUTPUT IS CIRCULAR FREQUENCY
C IF IOM=4, INPUT AND OUTPUT OM ARE FREQUENCIES
C JFLAG=+1 IF THE INVERSE FOURIER TRANSFORM HAS A FACTOR EXP(+IWT)
C JFLAG=-1 IF THE INVERSE FOURIER TRANSFORM HAS A FACTOR EXP(-IWT)
      DIMENSION X(N),Y(N),OM(NU),SCRR(NU),SCRI(NU),CX(NU),SX(NU)
      COMPLEX FT(NU)
      TOPI=8.*ATAN(1.)
      TPIN=1./TOPI
      PIN=2.*TPIN
      IF(JFLAG.EQ.1) GO TO 15
      FT(1;NU)=VCONJG(FT(1;NU);FT(1;NU))
 15   IF(IOM.LE.2) GO TO 25
      OM(1;NU)=OM(1;NU)*TOPI
 25   NP=NU-1
      SCRI(1;NP)=1./Q8VDELT(OM(1;NU);NP)
      SCRR(1;NP)=Q8VDELT(VREAL(FT(1;NU);NU);NP)*SCRI(1;NP)
      SCRI(1;NP)=Q8VDELT(VAIMAG(FT(1;NU);NU);NP)*SCRI(1;NP)
      DO 70 I=1,N
        W=X(I)
        IF(ABS(W).LT.1.E-19) THEN
          Y(I)=PIN*Q8SSUM(Q8VADJM(VREAL(FT(1;NU);NU);NP)*
     1    Q8VDELT(OM(1;NU);NP))
        ELSE
          WIN=1./W
          CX(1;NU)=VCOS(W*OM(1;NU);CX(1;NU))
          SX(1;NU)=VSIN(W*OM(1;NU);SX(1;NU))
          S2R=Q8SSUM(Q8VDELT(CX(1;NU);NP)*SCRR(1;NP)-
     1    Q8VDELT(SX(1;NU);NP)*SCRI(1;NP))
          S1I=REAL(FT(1))*SX(1)+AIMAG(FT(1))*CX(1)
     1    -REAL(FT(NU))*SX(NU)-AIMAG(FT(NU))*CX(NU)
          Y(I)=PIN*(-S1I+S2R*WIN)*WIN
        END IF
 70   CONTINUE
      IF(IOM.EQ.1.OR.IOM.EQ.3) GO TO 85
      OM(1;NU)=OM(1;NU)*TPIN
 85   IF(JFLAG.EQ.1) RETURN
      FT(1;NU)=VCONJG(FT(1;NU);FT(1;NU))
      RETURN
      END
```

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See Instructions)	1. PUBLICATION OR REPORT NO. NBSIR 86-3362	2. Performing Organ. Report No.	3. Publication Date APRIL 1986

4. TITLE AND SUBTITLE

SCAT: A Vector Program to Solve a Transient MFIE

5. AUTHOR(S)

Egon Marx

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The FORTRAN program SCAT is used to solve the magnetic field integral equation (MFIE) to determine the fields scattered by a perfectly conducting sphere. The incident field is a plane-wave pulse, and a stepping-in-time procedure is used to determine the surface current density induced on the sphere. The program does not take advantage of the special symmetry of the scatterer because it is intended to serve as a verified starting point for more general programs. The output is compared to that of the program PERF, which computes the same fields via a Fourier transform of the monochromatic fields obtained from the Mie formulas. The contributions of the self-patch and neighboring patches to the singular integral are optionally computed by using their expansions in the linear size of the patches. The self-patch term is important for the solution of other integral equations that may be of the first kind. For the MFIE, these corrections are small but not negligible. The program takes advantage of the vector programming features of the CYBER 205.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

computer programs; electromagnetic pulse scattering; magnetic field integral equation; Mie scattering; perfectly conducting sphere; stepping-in-time procedure; transient electromagnetic fields; vector programming

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

104

15. Price

$16.95

USCOMM-DC 6043-P80