SCATTERING BY RESISTIVE STRIPS AND PLATES

by

Thomas B.A. Senior
Radiation Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109

Final Report on P.O. No. 46-4385

Prepared for

Northrop Corporation
Aircraft Division
Mail Zone 3020/92
3901 West Broadway
Hawthorne, CA 90250

July 1985

CHAPTER I. INTRODUCTION

This is the final report of the work carried out for the Northrop Corporation under purchase order 36-4385 during the period ending 31 May 1985. The purpose of the study was to examine the effect of non-zero resistivity on the backscattering cross sections of strips and plates with particular reference to angles of incidence which are close to grazing.

In the case of a strip with the incident electric vector parallel to the surface attention is confined to edge-on incidence, and the analysis is presented in Chapter II. The backscattering cross section is determined for a variety of uniform and quadratically-tapered resistivities, and one of the significant conclusions is that tapering is not always better. When the incident magnetic vector is parallel to the surface, the scattering for edge-on incidence is zero regardless of the resistivity, and the incidence which is most important is that corresponding to the traveling wave lobe. The nature of this lobe is examined in Chapter III, and it is shown that even a small resistivity is sufficient to eliminate the traveling wave as such. Unfortunately, this merely bares a far side lobe of the specular flash.

A strip is the two-dimensional analogue of a finite plate and in spite of the useful information that can be derived from a study of the simpler two-dimensional structure, there are many circumstances under which the model is irrelevant to the realistic problems of a finite plate. Substantial effort has been devoted to the development of an efficient and effective code to compute the scattering from a finite resistive plate of arbitrary shape and resistivity. The work that has been accomplished is described in Chapter IV, and though there is still much to be done, the program in its present form does compute the current induced in the plate. The development of the program was carried out by Mr. J. W. Burns and Professor D. A. Ksienski (a present and former student, respectively, of the author), and it is a pleasure to acknowledge their assistance.

CHAPTER II. E POLARIZATION

One of the more difficult problems in cross section reduction is to reduce the scattering from the edge of an electrically thin structure such as an aircraft wing or tail fin. This is particularly true at angles close to grazing incidence with the electric vector parallel to the surface, and the significant scattering that can occur in this case is dominated by the edges. We shall consider here the problem of a resistive strip or ribbon illuminated by an E-polarized plane wave at edge-on incidence ($\phi_0 = \pi$) and examine the magnitude of the backscattered field compared with that of the corresponding perfectly conducting strip.

2.1 Uniform Resistivity

For a uniform resistive strip with resistivity R ohms per square the backscattered far field amplitude $P(\pi,\pi)$ can be expressed as a sum of front and rear edge contributions as (Senior, 1979a):

$$P(\pi,\pi) = P^f + P^r \quad . \tag{2.1}$$

In terms of P the backscattering cross section per unit length is

$$\sigma = \frac{2\lambda}{\pi} |P(\pi,\pi)|^2 \quad . \tag{2.2}$$

If the strip width w is more than about $\lambda/2$ where $\lambda = 2\pi/k$ is the wavelength, the front edge return P^f is identical to that for a half plane of the same resistivity and

$$p^f = -\frac{i\eta}{16}\{ZJ(0,\eta)\}^2 = -\frac{i}{4}\{K(0,\eta)\}^2 \tag{2.3}$$

where $\eta = 2R/Z$, $J(x,\eta)$ is the current on a resistive half plane at a distance x from the edge for the same incident plane wave, and $K(0,\eta)$ is a "split" function which appears in the Wiener-Hopf solution for a half plane. Z is the intrinsic impedance of free space and a time variation $e^{-i\omega t}$ has been assumed.

$K(0,\eta)$ is real for real η and its values have been tabulated (Senior, 1979a). For complex η

$$K(0,\eta) \cong \begin{cases} 2^{1/2} \exp\{-\eta/\pi\,[1 - \ln(\eta/2)]\} & |\eta| \ll 1 \\ \eta^{-1/2} \exp\{-1/(\pi\eta)\} & |\eta| \gg 1 \end{cases} \tag{2.4}$$

and $K(0,\eta)$ can also be expressed in terms of the function $\psi_\pi(z)$ introduced by Maliuzhinets (1958) as

$$K(0,\eta) = \frac{4\sqrt{\eta}}{(\sqrt{\eta} + \sqrt{1+\eta})^2} \left\{\frac{\psi_\pi(\chi)}{\psi_\pi(\pi/2)}\right\}^4 \tag{2.5}$$

where $\cos\chi = 1/\eta$. In a recent article (Volakis and Senior, 1985), two simple expressions for $\psi_\pi(z)$ are derived which, when used in conjunction with known identities, approximate the function to a high degree of accuracy throughout the entire complex z plane.

For the rear edge scattering and with the same restriction on w,

$$p^r = i\gamma\{ZJ(w,\eta)\}^2 \tag{2.6}$$

where (Senior, 1979b)

$$\gamma = \{4K(0,\eta)\}^{-2} , \tag{2.7}$$

so that

$$p^r = \frac{1}{4\eta}\left\{\frac{ZJ(w,\eta)}{ZJ(0,\eta)}\right\}^2 . \tag{2.8}$$

The above formulas are valid for complex η as well as real, and a procedure for computing the exact analytical expression for $J(x,\eta)$ is described in Senior (1981). In the special case of perfect conductivity,

$$ZJ(x,0) = 2\sqrt{\frac{2}{\pi kx}}\, e^{i(kx+\pi/4)} . \tag{2.9}$$

If η is real and non-zero, $J(x,\eta)$ is asymptotic to $J(x,0)$ as $kx \to \infty$, but if $\eta >> 1$, kx has to be very large indeed before (2.9) constitutes a good approximation to $J(x,\eta)$. The current on a resistive half plane never exceeds its value at the corresponding point of a perfectly conducting half plane, and since its magnitude is a monotonically decreasing function of x and η, it follows from (2.8) that

$$|p^r| \leq \frac{1}{4\eta} \tag{2.10}$$

for all w and η.

For a perfectly conducting strip the front and rear edge contributions have particularly simple expressions. Since $K(0,0) = \sqrt{2}$, Eq. (2.3) gives

$$p^f = -\frac{i}{2} , \tag{2.11}$$

and from (2.6), (2.7) and (2.9),

$$p^r = -\frac{e^{2ikw}}{4\pi kw} . \tag{2.12}$$

This is the smallest rear edge contribution that can be achieved with any uniform resistive strip whose width w is such that $kw > \eta$, and the behavior of $|p^r|$ as a function of η is illustrated in Fig. 2.1.

To make the edge-on backscattering cross section of a strip as small as possible over a wide band of frequencies, it is necessary to minimize the front and rear edge contributions individually. The smallest value of $|P^f|$ is achieved by choosing η as large as possible, and if, for example, $\eta = 4$, the resulting front edge contribution is almost 20 dB below that for a perfectly conducting strip. For the rear edge the preceding results suggest that the minimum return is obtained with a perfectly conducting strip. This is certainly true for all except the very narrowest strips, and Fig. 2.2 shows $|P^r|$ as a function of w/λ for five different values of (real) η. Having $\eta \neq 0$ inevitably increases $|P^r|$ and though, for fixed w/λ, the return ultimately decreases as η increases, it remains above that for $\eta = 0$ for all reasonable values of kw and η. The obvious solution is to taper the resistivity from a maximum at the front to zero at the rear, and provided this is done smoothly, it is natural to expect a rear edge contribution comparable to that for a perfectly conducting strip with no new source of scattering created.

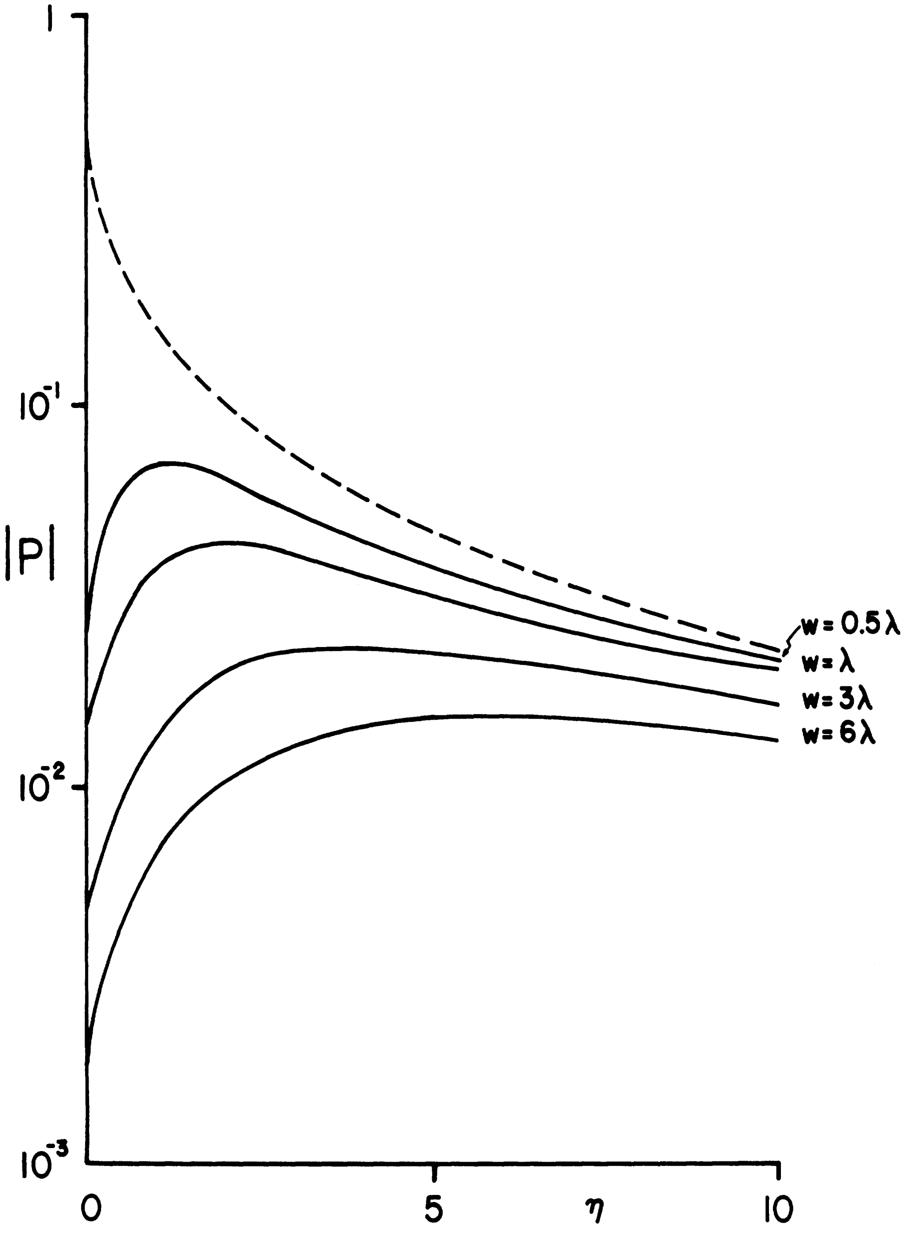

Fig. 2.1: The magnitudes of the front (---) and rear (——) edge contribution of uniform resistive strips as functions of η. (This is an expanded version of Fig. 2 of Senior and Liepa, 1984.)

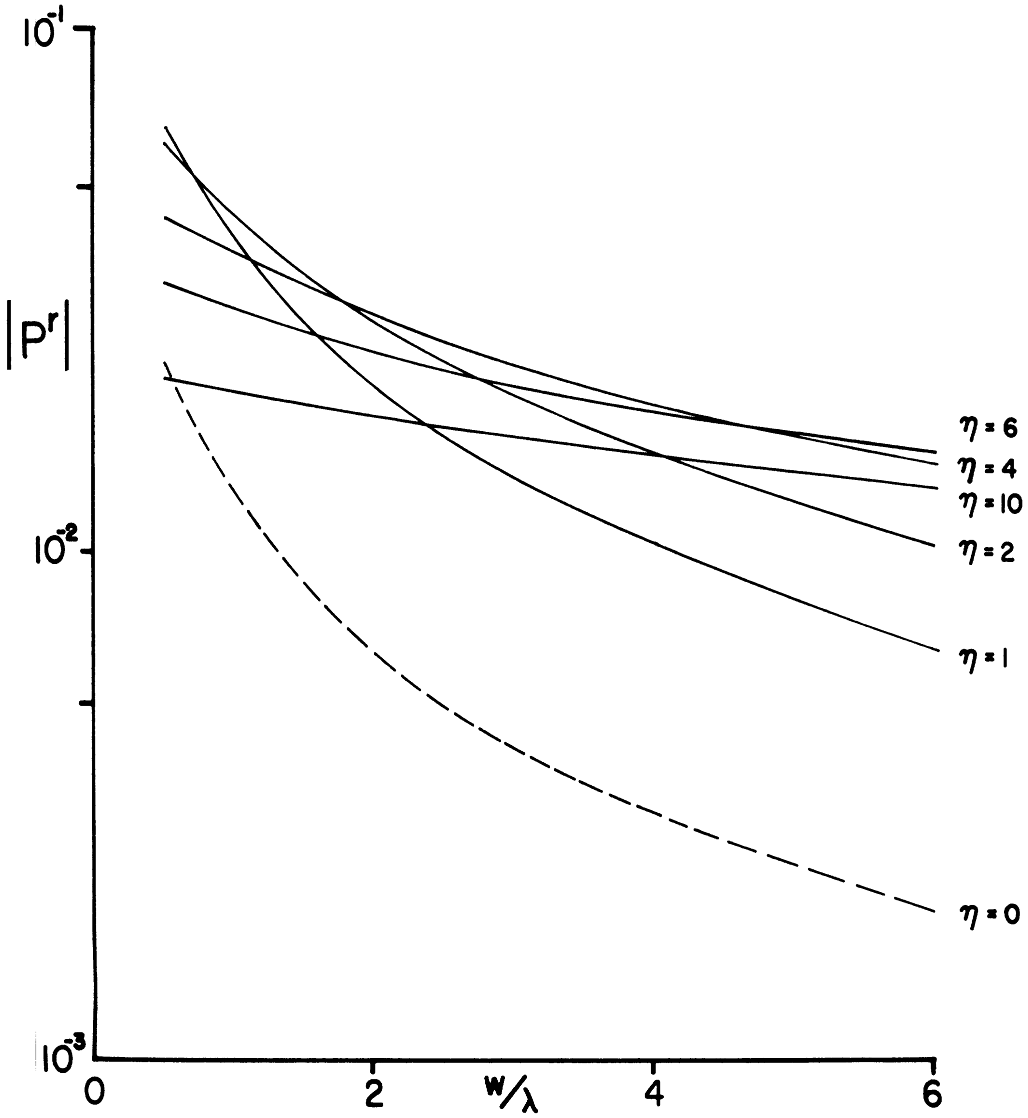

Fig. 2.2: The magnitudes of the rear edge contribution (——) for five uniform resistive strips, compared with the contribution for perfectly conducting strips (---) computed using (2.12).

2.2 Quadratically-Tapered Resistivity

Based on prior experience with resistive tapers, attention was confined to the quadratic form

$$R(x) = \frac{\eta}{2} Z \left(1 - \frac{x}{w}\right)^2 \qquad (2.13)$$

where x is measured from the front edge. Thus, η specifies the largest value of the resistivity, occurring at the front of the strip.

Using program REST-E which solves the integral equation for an E-polarized plane wave incident on a resistive strip, the total induced current and the backscattered far field were computed as a function of w/λ for a sequence of real η. From the computed data for $P(\pi,\pi)$, the real and imaginary parts of P^f and P^r were extracted (Ksienski, 1985a), and the resulting values of P^f were almost identical to those for the front edge contribution of a uniform resistive strip having the same η. The magnitudes of the rear edge contributions are plotted in Fig. 2.3, and we observe that they exceed the contribution for a perfectly conducting strip but show a similar dependence on w/λ. An empirical expression for P^r is

$$P^r = -\frac{e^{2ikw}}{4\pi kw}\left\{1 + \frac{5.8\eta}{(kw)^{2/3}}\right\}, \qquad (2.14)$$

and since the first term in parentheses in the rear edge contribution for a perfectly conducting strip, the second term is the additive effect of the resistive taper.

Under all circumstances, however, the rear edge return exceeds that for perfect conductivity, and for a narrow strip may even exceed

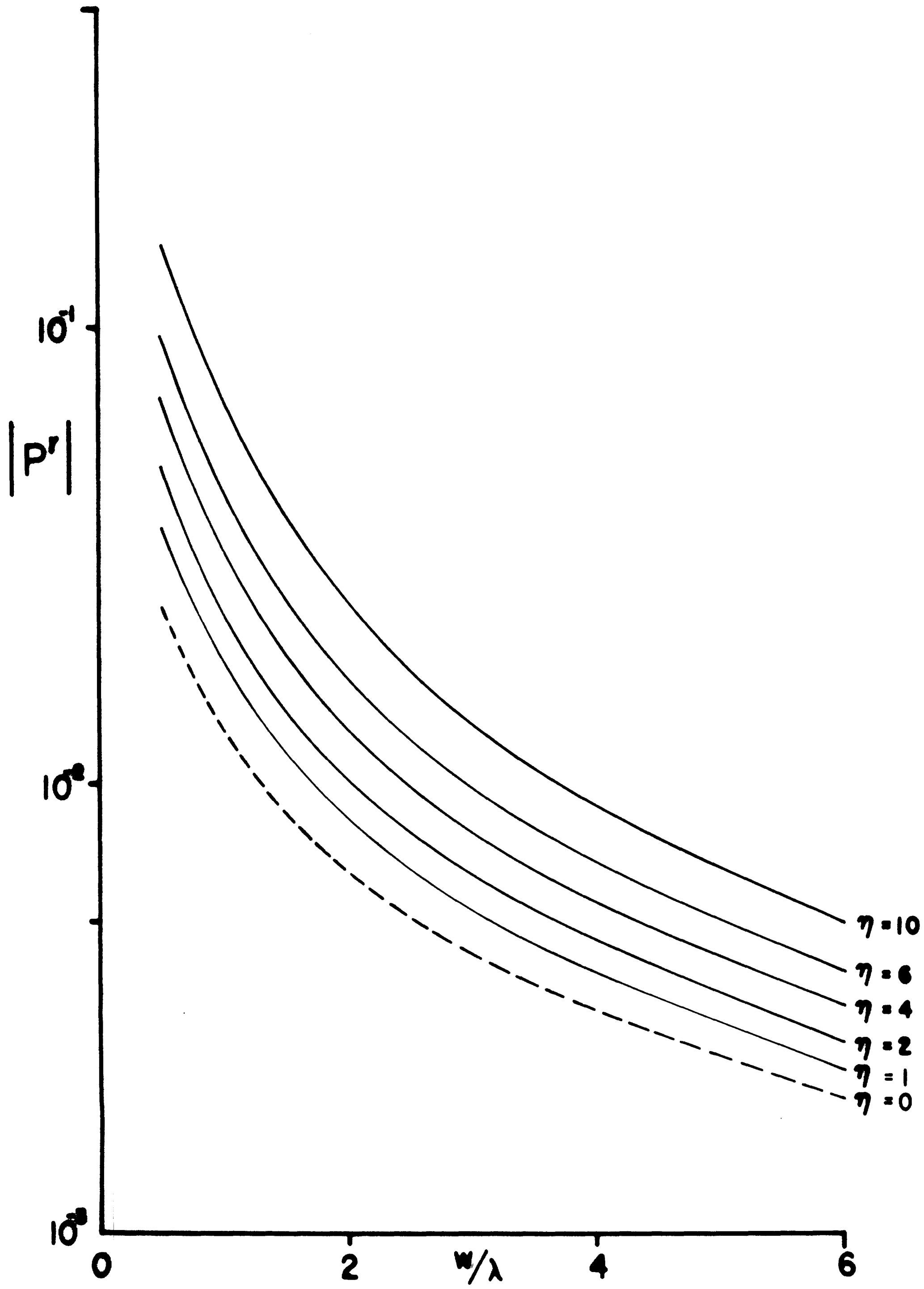

Fig. 2.3: The magnitudes of the rear edge contribution for tapered strips with five different values of η compared with the contribution (---) for perfectly conducting strips.

the return for the corresponding uniform resistive strip. Comparison of Figs. 2.1 and 2.3 shows that if $\eta = 10$ the tapering is only effective if $w > 2.4\ \lambda$, and the analogous conditions for $\eta = 6$ and $\eta = 4$ are $w > 1.4\ \lambda$ and $w > 0.9\ \lambda$ respectively. Thus, as the strip width and/or frequency is reduced, we ultimately reach a point at which tapering becomes counter productive.

Equations (2.14) and (2.3) are sufficient to provide the edge-on backscattered field of a resistive strip with a quadratic taper, and confirm that at high frequencies for which $w \gg \lambda$ the most effective way to reduce the scattering is to increase the resistivity at the front edge to as large a value as possible and to taper the resistivity smoothly to zero at the rear. Although we have considered only the particular taper (2.13), it seems probable that the results for other smooth monotonic tapers will be similar.

CHAPTER III: H POLARIZATION

Traveling waves are a major contributor to the backscattering cross section of a long thin body when the magnetic vector is perpendicular to the plane of incidence, and our work with H-polarization was concerned with studying the effect of traveling waves on the backscattering cross section of a strip or ribbon.

3.1 Traveling Wave Considerations*

Traveling waves are one of the most important contributors to the scattering from long slender bodies, and in the case of structures such as the fuselage of an aircraft it is possible to produce a reasonably complete description of the backscattering by taking into account the traveling waves and, where appropriate, the specular contributions and the side lobes thereof. When the incident magnetic vector is perpendicular to the plane formed by the direction of incidence and the axis of the body, the fan-shaped pattern characteristic of a traveling wave is a key feature of the overall scattering pattern, and the first lobe is often the major contributor to the scattering at angles close to end-on incidence.

The scattering pattern of a thin plate or strips also displays a similar lobe structure at angles close to edge-on incidence when the magetic vector is parallel to the surface, and it is customary to

*A shortened version of this section has been published by Senior and Yang (1984).

attribute at least the first lobe to a traveling wave. In effect, we are thinking of each narrow slice of the plate as a filament or wire supporting a traveling wave and the angle at which the lobe appears is consistent with this argument. Mathematically, however, there is no justification for this reasoning, and it is therefore of interest to compare the high frequency expressions for the backscattered fields of wires and strips.

For a wire of length ℓ viewed at an angle θ to end-on, an approximate form of the expression obtained by Peters (1958) for the backscattering cross section attributable to traveling waves is

$$\sigma = \frac{\lambda^2}{\pi} |S|^2$$

where

$$|S| = \frac{\Gamma}{C} \left\{ \frac{k\ell}{2} \sin\theta \, \mathrm{sinc} \left[\frac{k\ell}{2} (1 - \cos\theta) \right] \right\}^2 . \qquad (3.1)$$

$$\mathrm{sinc}\, X = \frac{\sin X}{X} ,$$

and Γ is the effective voltage reflection coefficient for the traveling wave. In deriving (3.1) we have assumed that the phase velocity is that of light and that $k\ell >> 1$, implying

$$C = \ln(2k\ell) - (1 - \gamma)$$

where $\gamma = 0.5772....$ is Euler's constant.

The angles $\theta = \theta_n$, $n = 1,2,3,...$, at which the maxima of the traveling wave lobes occur are given by the solutions of the transcendental equation

$$\tan \Phi_n = (1 + \cos \theta_n)\Phi_n \qquad (3.2)$$

with

$$\Phi_n = \frac{k\ell}{2} (1 - \cos \theta_n) \quad .$$

For large ℓ/λ

$$\theta_1 \cong 49.4\sqrt{\frac{\lambda}{\ell}} \quad \text{(degrees)} \quad , \qquad (3.3)$$

$$\theta_2 \cong 98.1\sqrt{\frac{\lambda}{\ell}} \quad \text{(degrees)} \quad , \qquad (3.4)$$

and comparison with numerical solutions of (3.2) shows that θ_1 is accurate to within one percent for $\ell/\lambda \geq 1.8$ and θ_2 is accurate to three percent for $\ell/\lambda \geq 3.6$. The results are also in good agreement with measured data. Chang and Liepa (1967) measured the backscattering patterns of a series of 81 wires having lengths varying from 0.3 to 5.42 λ and radius 6.27×10^{-3} λ, and the values of θ_1 and θ_2 obtained from these patterns are shown in Fig. 3.1, along with the curves corresponding to (3.3) and (3.4). We have also used the measured amplitude of the first lobe to determine the effective reflection coefficient Γ in (3.1). The amplitudes oscillate in a quite regular manner with a maximum to minimum ratio of about 6 dB and maxima occurring at $\ell/\lambda = 0.4 + 0.5n$, n = 0,1,2,... (approx.). It is clear that this is attributable to a resonance behavior of the wire, a feature which is not reproduced by (3.1), but if we incorporate this into the effective reflection coefficient, the resulting values of Γ appropriate to the

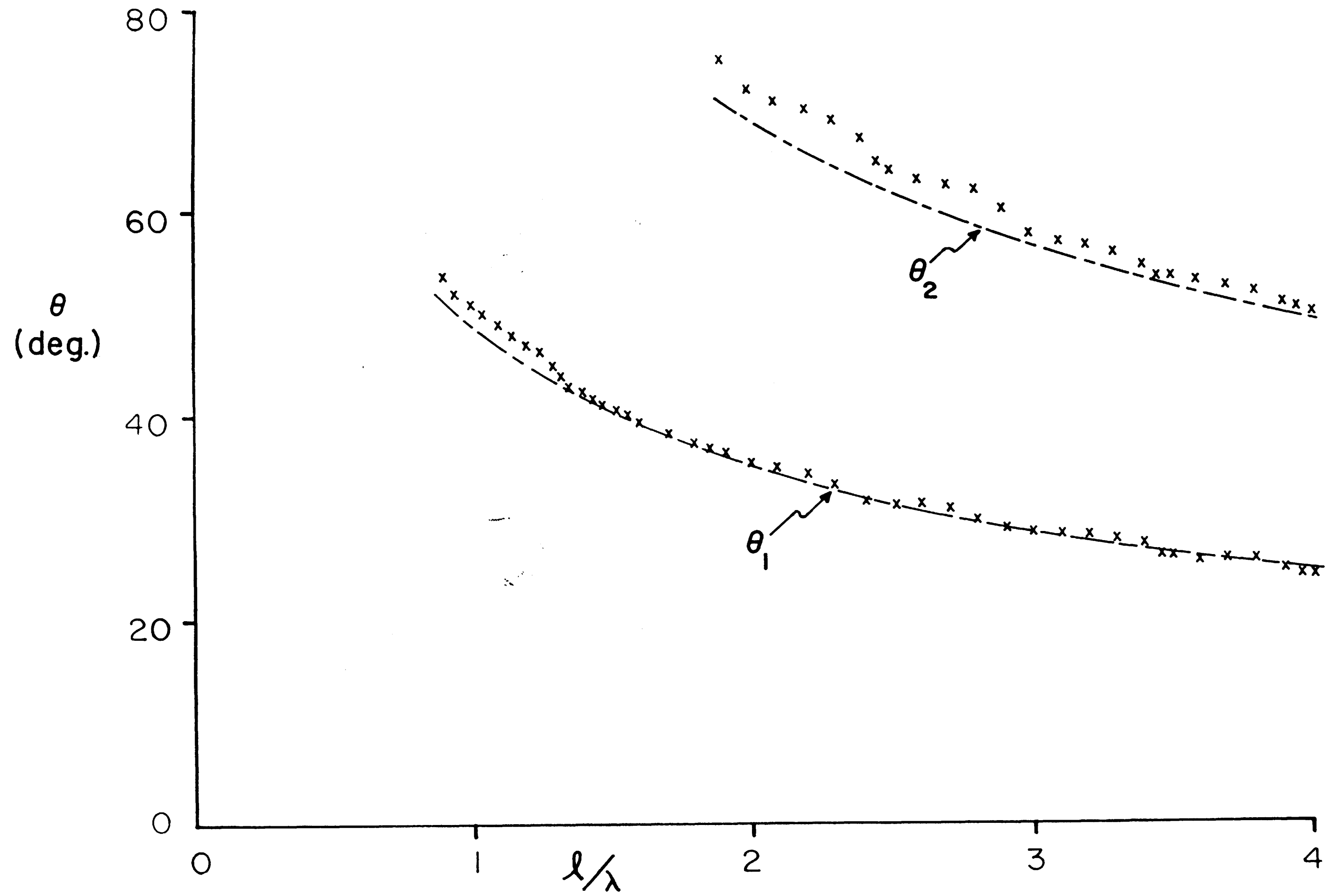

Fig. 3.1: Measured angles (xxx) of the first and second traveling wave lobes for a thin wire. The theoretical curves were computed using (3.3) and (3.4).

first lobe are shown in Fig. 3.2. The average is about 0.65 and varies little with ℓ/λ, and this is consistent with the value generally assumed (Ruck et al., 1970) for a pointed body.

We now consider an infinitesimally thin, perfectly conducting strip of width w occupying the region $0 \leq x \leq w$, $-\infty < \tau < \infty$ of the plane $y = 0$ of a Cartesian coordinate system x,y,z and illuminated by a plane wave having

$$\bar{H}^i = \hat{z}\, e^{-ik(x \cos \phi_0 + y \sin \phi_0)} \quad .$$

At large distances the scattered magnetic field can be written as

$$\bar{H}^s = \hat{z} \sqrt{\frac{2}{\pi k \rho}}\; e^{i(i\rho - \pi/4)}\; P(\phi,\phi_0)$$

where ρ,ϕ are cylindrical polar coordinates with $x = \rho \cos \phi$, $y = \rho \sin \phi$. In terms of P the scattering cross section per unit length in the z direction is

$$\sigma = \frac{2\lambda}{\pi} \left|P(\phi,\phi_0)\right|^2$$

and in the particular case of backscattering, $\phi_0 = \phi$.

For $kw >> 1$ a uniform second order GTD expression for the backscattered far field amplitude is (Senior, 1979b)

$$P(\phi,\phi) = -\frac{i}{4}\, \frac{1 + \cos \phi}{\cos \phi} \left\{1 - \frac{2}{\sqrt{\pi}}\, e^{-i\pi/4} \cos \frac{\phi}{2}\, F\left(\sqrt{2kw} \sin \frac{\phi}{2}\right)\right\}^2$$

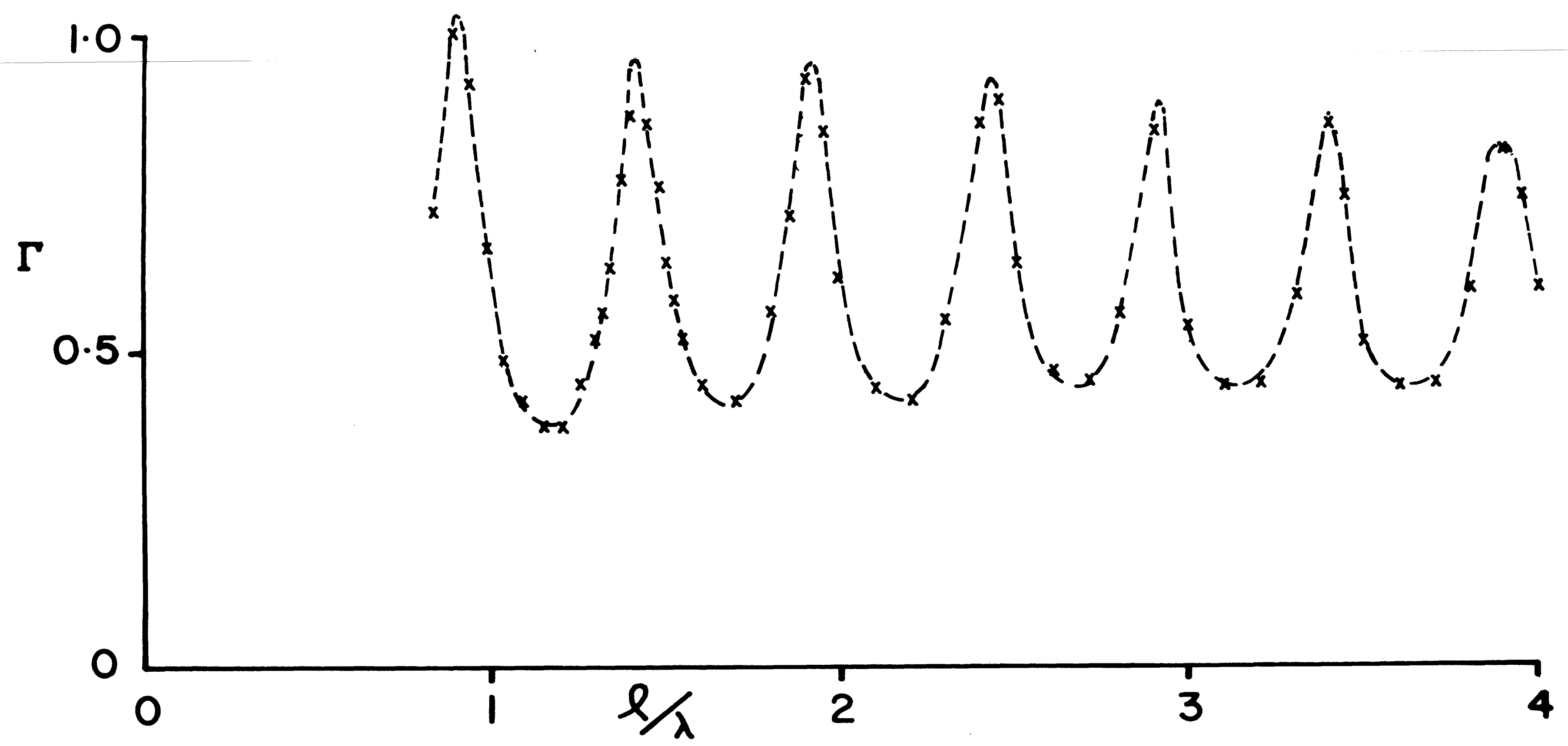

Fig. 3.2: Effective reflection coefficient Γ deduced from the thin wire data. The dashed curve is only to guide the eye.

$$\frac{i}{4}\,\frac{1-\cos\phi}{\cos\phi}\,e^{-2ikw\cos\phi}\left\{1-\frac{2}{\sqrt{\pi}}\,e^{-i\pi/4}\sin\frac{\phi}{2}\,F\left(\sqrt{2kw}\cos\frac{\phi}{2}\right)\right\}^2 \qquad (3.5)$$

$$+\;O([kw]^{-1}) \quad ,$$

valid for $0 \leq \phi \leq \pi$. $F(\tau)$ is the Fresnel integral

$$F(\tau) \;=\; \int_{\tau}^{\infty} e^{iu^2}\,du \quad ,$$

and (3.5) is identical to the result obtained by asymptotic expansion of the expression obtained by Khaskind and Vainshteyn (1964). A simple program to compute $P(\phi,\,\phi_0)$ has been developed for use on an IBM PC computer. It is designated P-RIB-H and is described in Appendix A.

When $\phi = 0$ or π, $|P| = 0$ as expected, and the backscattering pattern of a strip is illustrated by the plot of $|P|$ versus ϕ for $w/\lambda = 4$ in Fig. 3.3. Since the pattern is symmetrical about the broadside aspect, it is sufficient to consider only $0 \leq \phi \leq \pi/2$. The lobe centered on $\phi = 24$ degrees is the one usually attributed to a traveling wave, and the locations of this and the adjacent peaks are shown in Fig. 3.4. From the variation as a function of w/λ it is evident that all peaks other than the first are most logically associated with the side lobes of the specular return. The first peak is primarily determined by the first term in (3.5), and its magnitude is

$$\frac{1}{4}\left(\frac{2kw-u}{kw-u}\right)\left|1-\left(1-\frac{u}{2kw}\right)^{1/2}\left\{1-C(u)-S(u)+i[C(u)-S(u)]\right\}\right|^2$$

where

$$u \;=\; kw(1-\cos\phi)$$

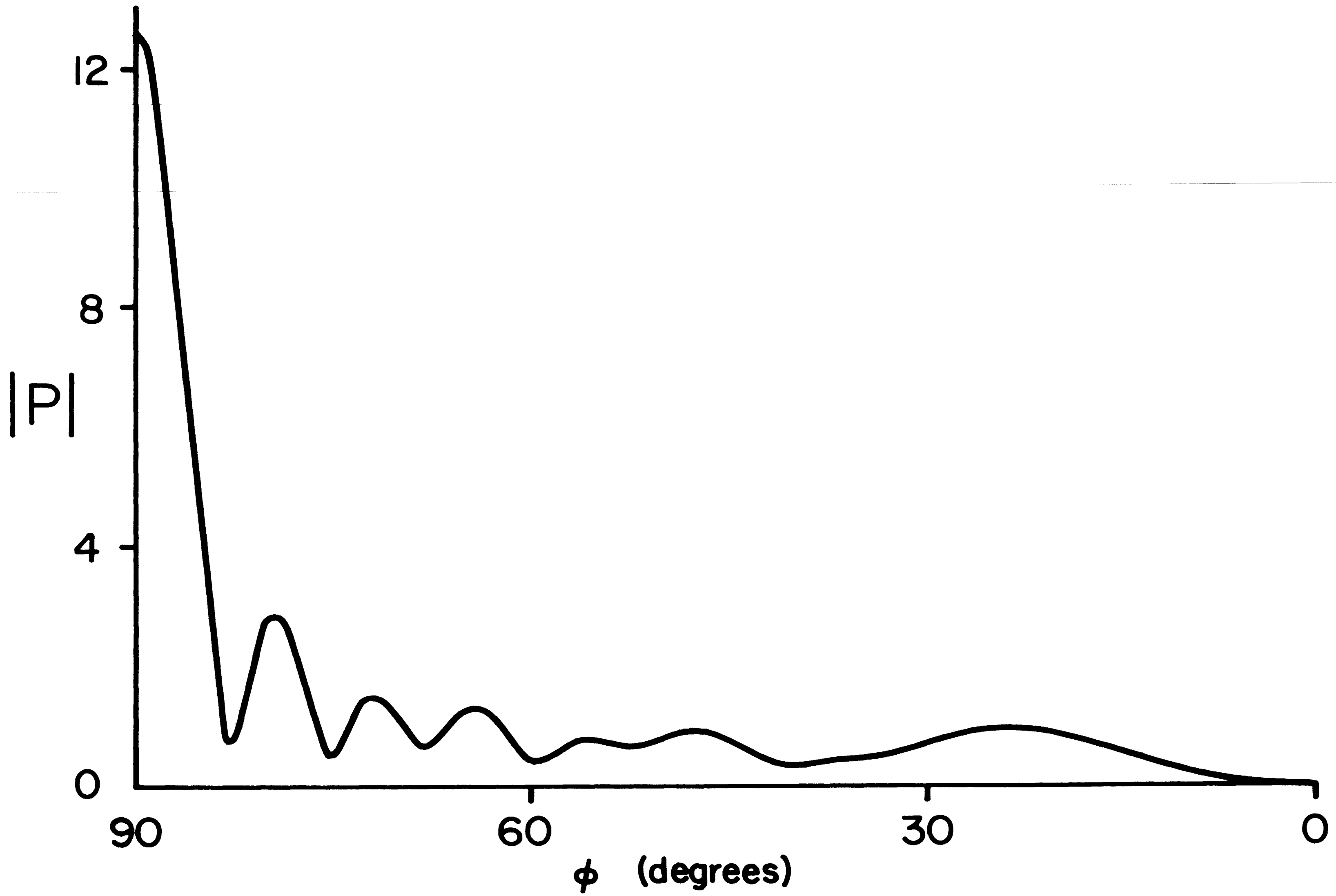

Fig. 3.3: Backscattering pattern of a strip of width w = 4λ computed using (3.5).

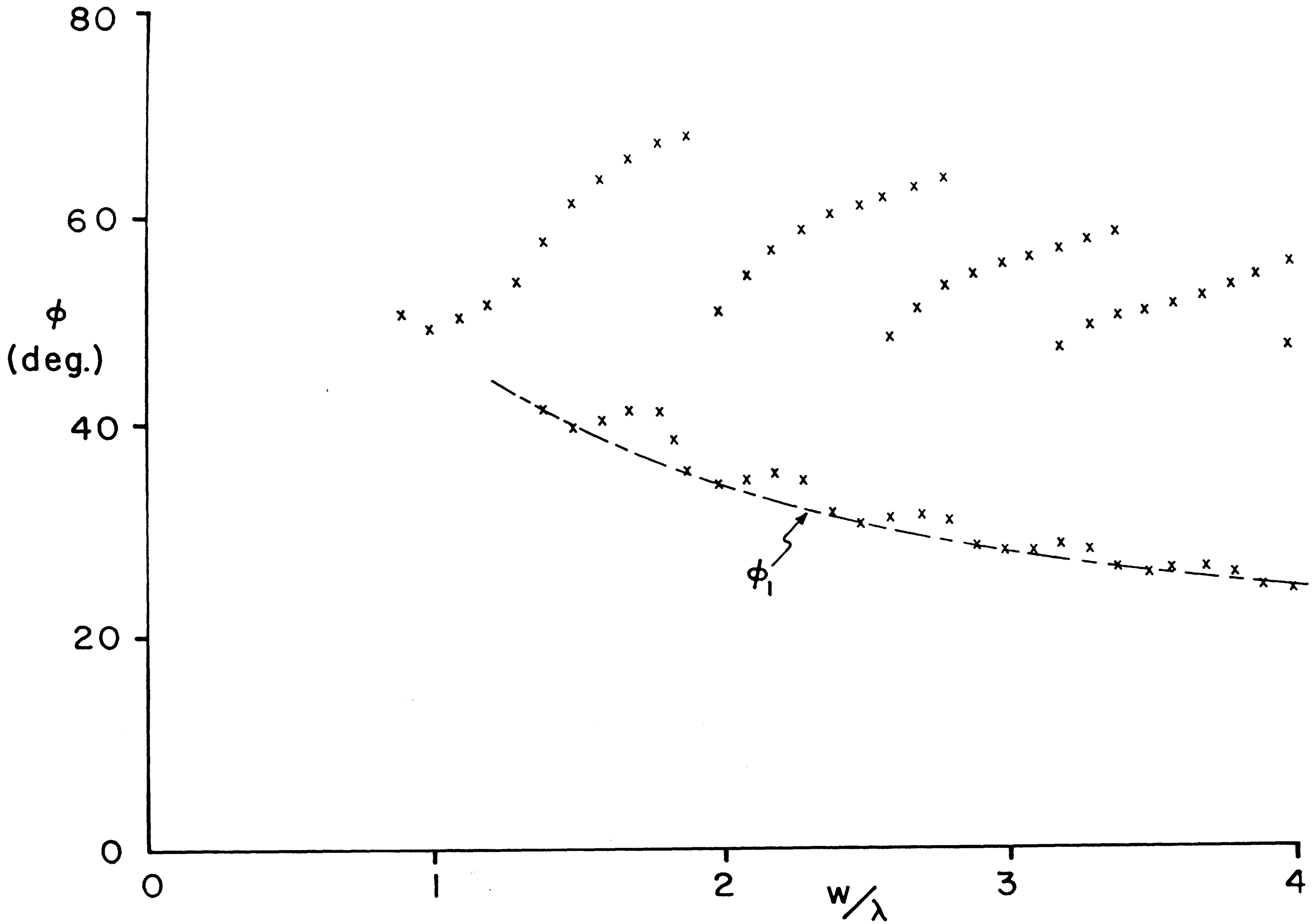

Fig. 3.4: Angles at which the maxima in the pattern occur. The theoretical curve was computed using (3.6).

and C(u) and S(u) are the real integrals defined and tabulated in Abramowitz and Stegun (1964). For kw >> 1 the magnitude is approximately $\{C(u)\}^2 + \{S(u)\}^2$, and its maximum is 0.901 occurring at $u = u_1 = 2.29$. The resulting value of ϕ is

$$\phi_1 = 48.9 \sqrt{\frac{\lambda}{w}} \quad \text{(degrees)} \quad , \tag{3.6}$$

and the corresponding curve is included in Fig. 3.4. To the next order in 1/(kw) the peak value is

$$\{C(u_1)\}^2 + \{S(u_1)\}^2 + \frac{u_1}{4kw}\{C(u_1) + S(u_1)\} = 0.901\left(1 + 0.136\,\frac{\lambda}{w}\right) \quad , \tag{3.7}$$

and this is plotted in Fig. 3.5 along with the peak values obtained from computations of $|P|$. The oscillations are attributable to the effect of the second term in (3.5) which was neglected in our analysis, and (3.7) is an excellent approximation to the mean.

In spite of the different mathematical formulas which describe the effects of traveling waves on wires and strips, the close agreement between the angles θ_1 and ϕ_1 at which the first (dominant) lobe occurs is remarkable. For all practical purposes, it is sufficient to use (3.3) to locate the lobe, thereby lending support to the idea of treating a planar structure as an assemblage of elementary wires, and to using the term "traveling wave" in the case of a strip.

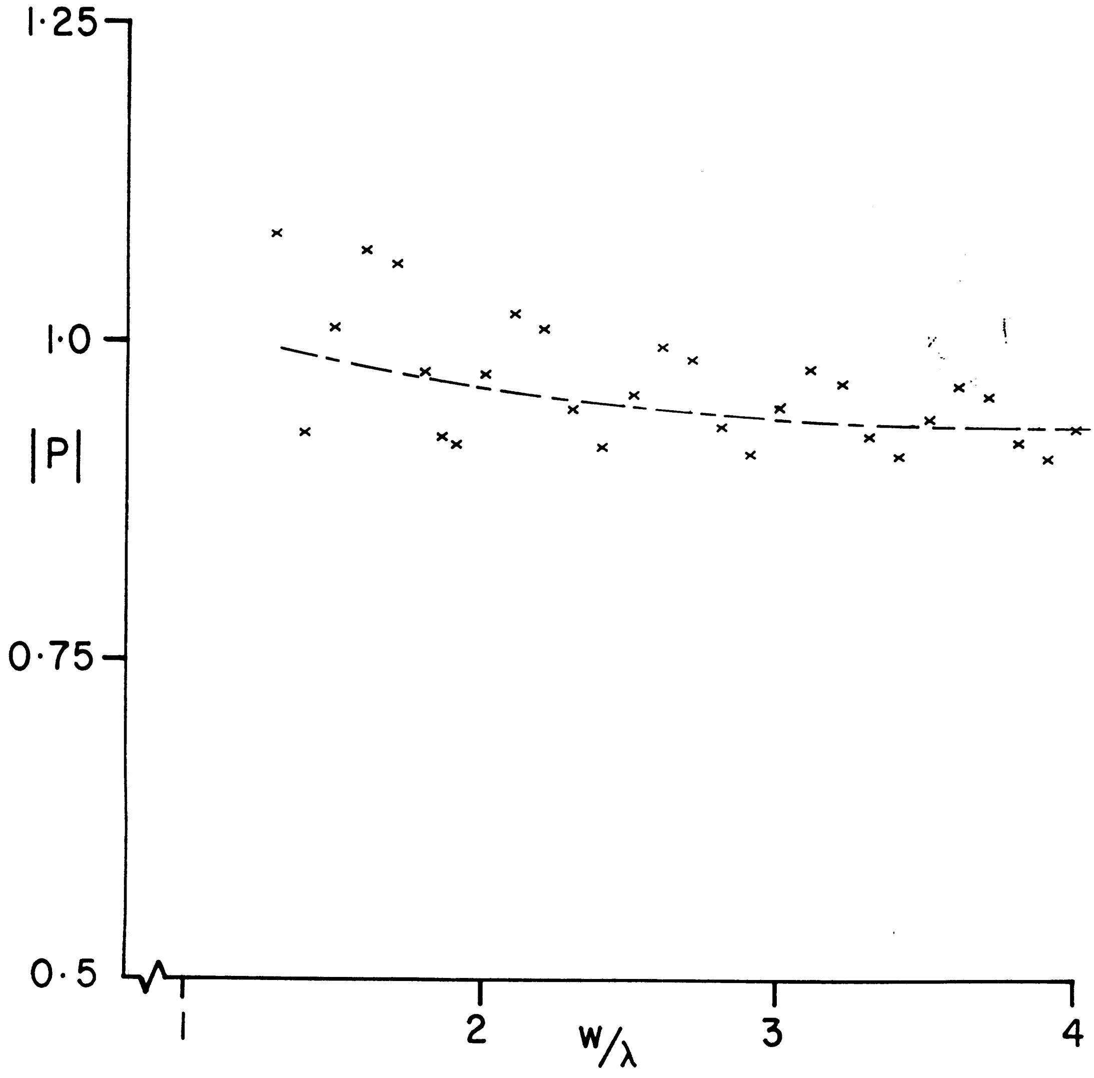

Fig. 3.5: Peak values (xxx) of the traveling wave lobe for a strip compared with the value (——) given by (3.7).

2.2 Effect of Non-Zero Resistivity

As we have seen, a traveling wave is a significant contributor to the backscattering cross section of a perfectly conducting strip at angles close to edge-on. It may be necessary to reduce the magnitude of the resulting peak, and the prevailing wisdom is that a relatively small amount of loss is adequate for this purpose. To determine the reduction as a function of the resistivity, we have used program REST-H or its specialized version STRIP-H (see Memorandum 2500-394-M) to compute the backscattered fields of uniform resistive strips of width $w = 1.0(0.1)4.0\ \lambda$ and have examined the magnitudes and locations of the peaks in the backscattered patterns.

For perfectly conducting strips the angle ϕ (measured in degrees from edge-on) at which the maxima are found to occur are shown in Fig. 3.6(a), and the results are almost identical to those obtained using the second order GTD expression (3.5) for the field. We observe that there are two distinct types of maxima; the ones associated with the main lobe of a traveling wave whose location is given approximately by the formula (3.6), and those which are more logically attributable to the side lobes of the specular flash. The latter correspond to the maxima of the pattern factor $(\sin X)/X$ with $X = kw \cos \phi$, and are given by

$$\phi = \text{arc cos} \left\{ (2n + 1) \frac{\lambda}{4w} \right\} \tag{3.8}$$

with $n = 1,2,3,\ldots$. As seen from Fig. 3.6(a), they lie on a sequence of trajectories which are distinct from the oscillatory curve describing the traveling wave peak. For large w/λ the traveling wave dominates the far side lobes, which then show up only as an oscillation in the

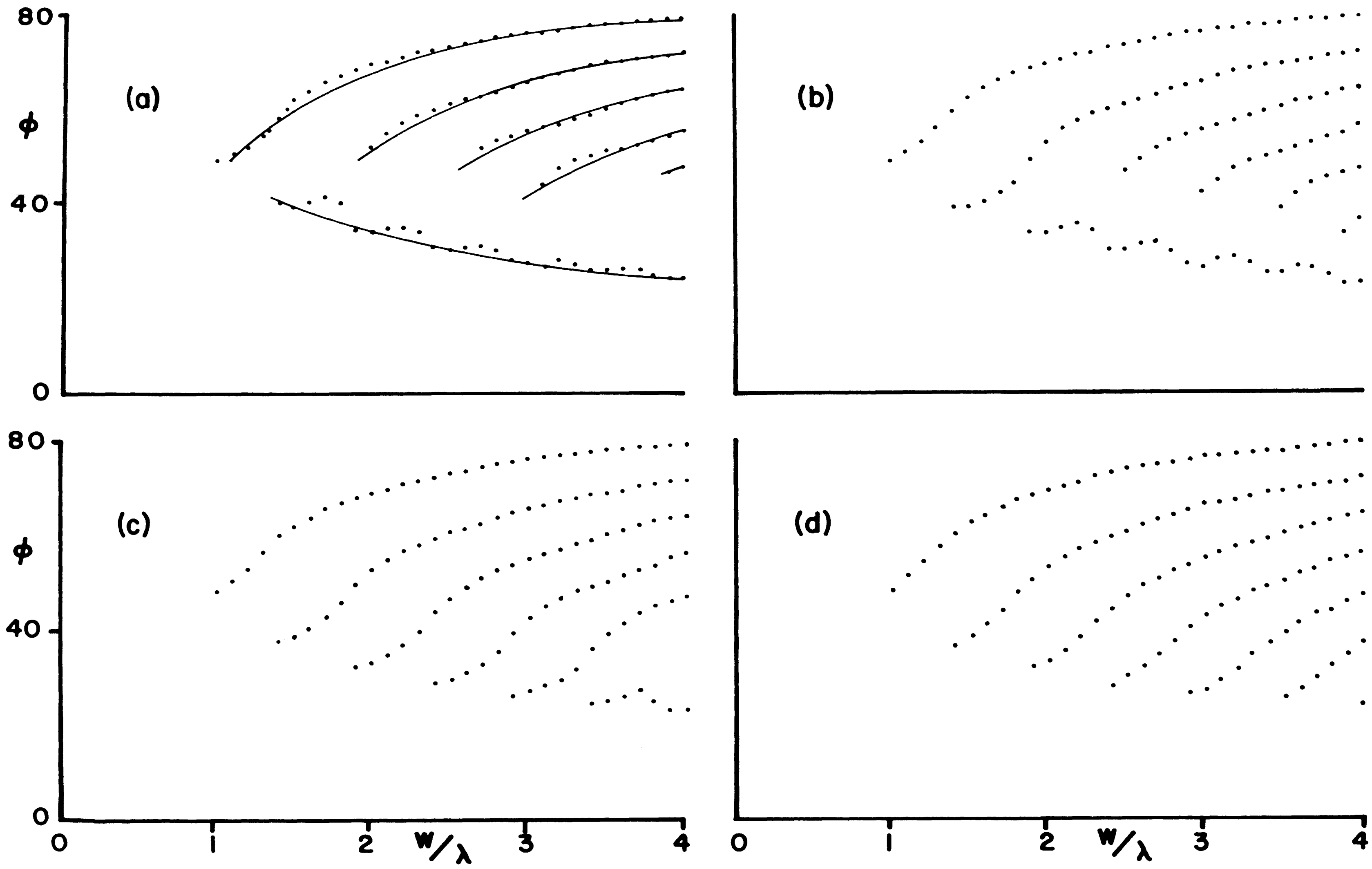

Fig. 3.6: Angles ϕ (in degrees) at which peaks in the backscattering pattern occur for (a) R = 0, (b) R/Z = 0.05, (c) R/Z = 0.10 and (d) R/Z = 0.20. The curves in (a) are plotted using (3.6) and (3.8).

location and the magnitude of the traveling wave lobe. The corresponding peak values of $|P|$ (in dB) are shown in Fig. 3.7(a).

The analogous results for resistive strips having $R/Z = 0.05$, 0.1 and 0.2 are shown in Figs. 3.6 and 3.7, (b) through (d). The point to be observed is that as R increases from zero the minimum strip width for which a traveling wave exists also increases. Thus, for $R/Z = 0.05$ (corresponding to 19 ohms), a traveling wave is present only for $w/\lambda \geq 1.9$, and for smaller w/λ the peak which occurs at almost the expected position (see Fig. 3.6(b)) is actually a side lobe of the specular flash. This is evident from the manner in which the peak location changes with increasing w/λ, and the logical explanation is that for $w/\lambda < 1.9$ the small amount of loss has reduced the traveling wave peak below the level of the side lobe. When $R/Z = 0.1$ there is no traveling wave lobe unless $w/\lambda \geq 3.4$. The way in which the peaks for $w/\lambda < 3.4$ are "taken over" by the side lobe of the specular flash is graphically illustrated in Fig. 3.7(c), and when $R/X = 0.2$ there is no evidence of a traveling wave for any $w/\lambda < 4$. It is apparent that even a small amount of loss is sufficient to eliminate a traveling wave for modest values of w/λ.

On the other hand, if the purpose of the non-zero resistivity is to reduce the near edge-on cross section, the presence of the peaks regardless of their origin could still be of concern. If we simply take the peak which occurs closest to edge-on, skipping from (side lobe) trajectory to trajectory as necessary, and plot its magnitude as a function of w/λ, the results shown in Fig. 3.8 are obtained. For $R/Z \lesssim 0.15$ the points form a reasonably smooth oscillatory curve for all $w/\lambda \geq 1.4$, but when $R/Z = 0.2$ the consequence of changing trajectories is

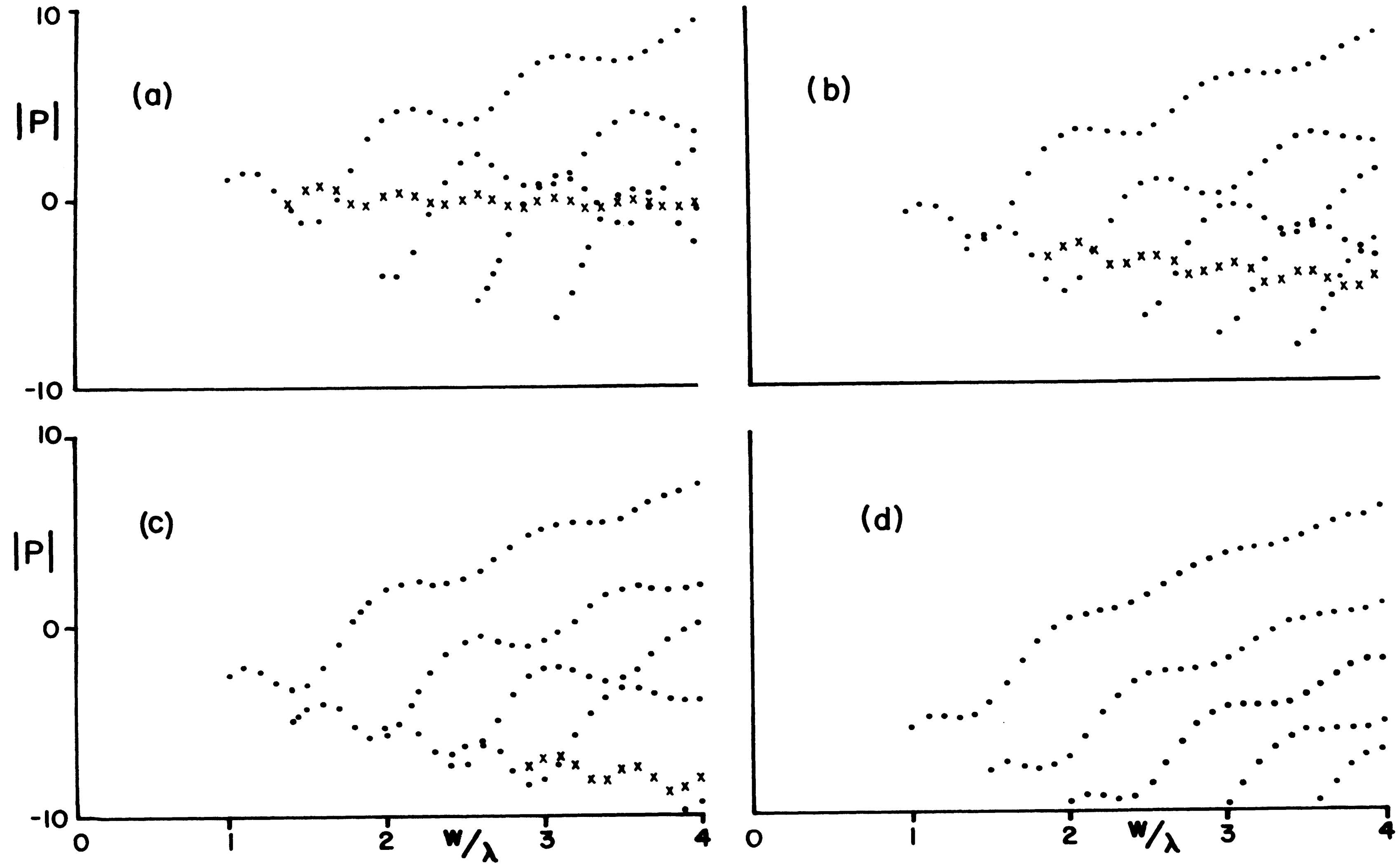

Fig. 3.7: Magnitudes |P| in dB of the peaks for (a) R = 0, (b) R/Z = 0.05, (c) R/Z = 0.10 and (D) R/Z = 0.2. The curves are merely to guide the eye and the crosses correspond to the traveling wave.

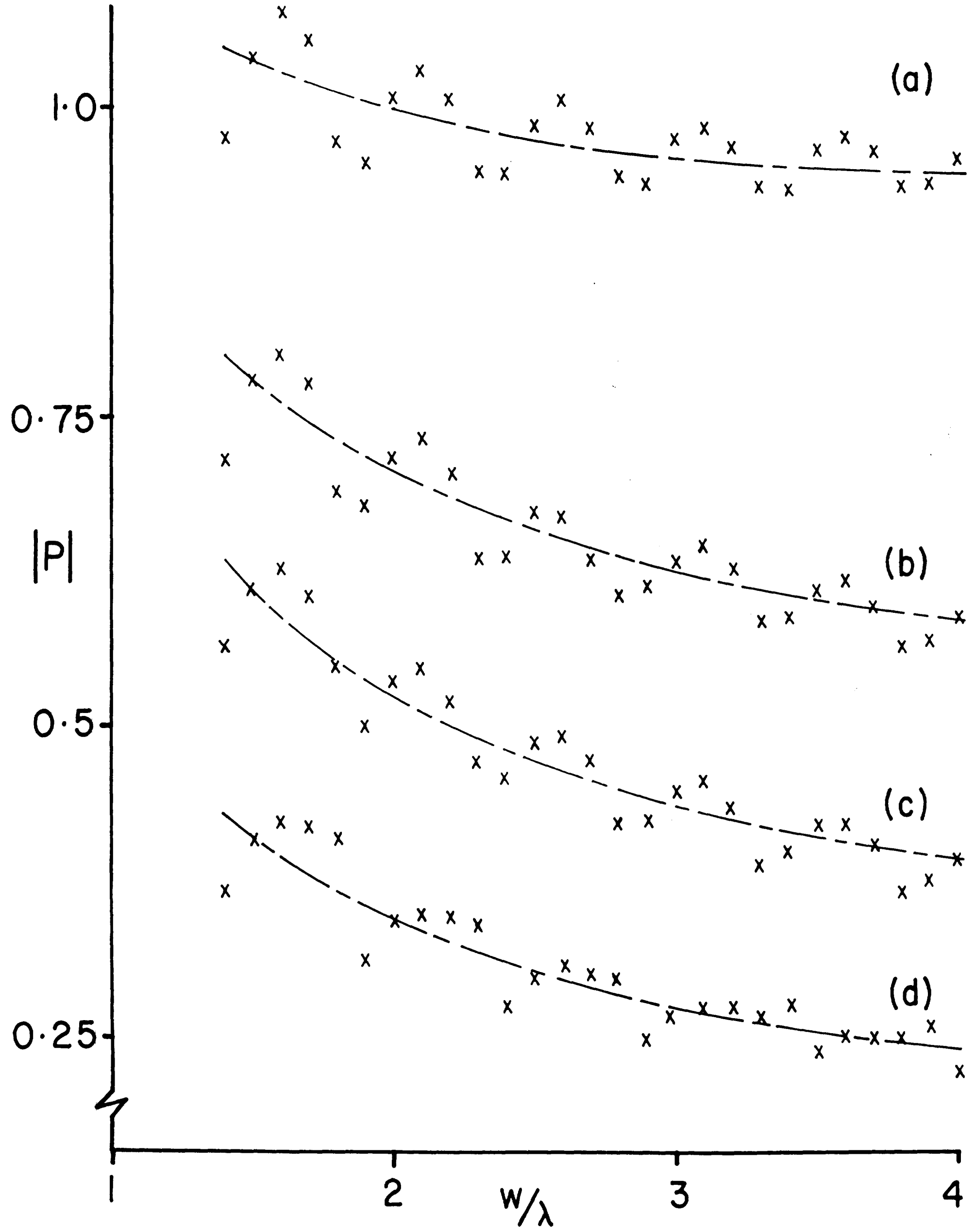

Fig. 3.8: Magnitudes of the actual or nearby traveling wave peak for (a) R = 0, (b) R/Z = 0.05, (c) R/Z = 0.10 and (d) R/Z = 0.20. The curves were obtained using (3.9).

to produce noticable discontinuities. As expected, a non-zero resistivity decreases the magnitude of the peak, and the reduction below the value for a perfectly conducting strip increases with increasing R/Z and w/λ. For R/Z = 0.2 and w/λ = 4 the reduction is approximately 12 dB.

We have not succeeded in developing a theoretical formula for the traveling wave peak showing the dependence on R/Z and w/λ. Though a uniform rigorous second order GTD expression for the bistatic scattered field of a uniform resistive strip is available (Senior 1979b), the result for H polarization is discontinuous in the limit $R \to 0$ and cannot be used to show the effect of resistivities as small as those of interest here. Nevertheless, from an examination of the data in Fig. 3.8 it appears that an expression of the form

$$|P| = A + B\frac{\lambda}{w} \qquad (3.9)$$

is adequate to predict the average return as a function of w/λ for a given R. By a lease squares fit it is found that

$$\begin{aligned} R/Z &= 0 \;, & A &= 0.88 \;, & B &= 0.23 \\ &= 0.05 \;, & &= 0.67 & &= 0.45 \\ &= 0.10 \;, & &= 0.26 & &= 0.53 \\ &= 0.20 \;, & &= 0.13 & &= 0.42 \;, \end{aligned}$$

and it is reasonable that these values be used in conjunction with (3.9) to estimate the effect of a small amount of resistivity.

CHAPTER IV. RESISTIVE PLATES

The strip or ribbon is a two-dimensional analogue of a finite plate, and there is a considerable amount of useful information about the scattering from a lossy plate that can be obtained by considering the simpler two-dimensional problem. Nevertheless, the information is limited in its applicability, and in cases such as near-grazing incidence when the side edges of a plate play a role it is necessary to consider the plate directly.

4.1 Background

A major activity has been the development of an effective and efficient code to compute the scattering from a finite, planar resistive plate of infinitesimal thickness when illuminated by an incident plane wave. We remark that the restriction to a resistive plate is not, in fact, a restriction at all. As pointed out in a recent publication (Senior, 1985: included here as Appendix B), a thin layer whose permittivity and permeability both differ from the free space values of the surrounding medium can be simulated using superposed (electrically) resistive and (magnetically) "conductive" sheets, and these sheets are uncoupled when they are planar. A conductive sheet is the electromagnetic dual of a resistive one, and thus by running the program twice with the polarization and the sheet parameters appropriately defined, the solution for the most general imperfect plate can be obtained by addition. A special case of a combination sheet is the opaque plate having an impedance boundary condition imposed at its surfaces.

Several years ago a program was developed (Naor and Senior, 1981) to solve the integro-differential equations for the components of the total electric current induced in a resistive plate when illuminated by a plane wave. The program employs rectangular subdomains and uses simple differencing to carry out the differentiations numerically, and because of this the accuracy and generality are less than what we would like. Based on our recent experience with a program to treat dielectric plates at low frequencies (Ksienski, 1985) where a static analysis is appropriate, it was felt that the numerical differentiation could be avoided, and that a more efficient, accurate and general program would result. The efficiency is important because of our desire to treat plates up to a square wavelength or two in area with a matrix of only modest (e.g. 128 x 128) size, and this was one factor that led to our use of the C language. It is also important that the program accommodate plates of general shape (and this motivated the choice of arbitrary triangular subdomains), having an arbitrarily specified non-uniform resistivity when illuminated by a plane wave incident in any direction with any polarization. It is believed that these objectives are met by the program we have developed.

4.2 Program Description

The plate is assumed to lie in the plane z = 0 of a Cartesian coordinate system x,y,z, and is simulated by a resistive sheet whose resistivity R (ohms per square) is

$$R = \frac{iZ}{kt(\varepsilon_r - 1)} \qquad (4.1)$$

where t and ε_r are the thickness and relative permittivity of the plate material, k and Z are the propagation constant and intrinsic impedance of free space, and a time factor $e^{-i\omega t}$ is assumed and suppressed. The boundary conditions at the plate are

$$\hat{z} \times \bar{E}\Big|_{-}^{+} = 0$$

and

$$\hat{z} \times \bar{E} = R\hat{z} \times \bar{J} \quad ,$$

where

$$R = \hat{z} \times \bar{H}\Big|_{-}^{+}$$

is the total electric current, and an electric field integral equation (EFIE) for $\bar{J}$ is then

$$R\,\hat{z} \times \bar{J} = \hat{z} \times \bar{E}^{inc} + i\,\frac{kZ}{4\pi}\,\hat{z} \times \left(1 + \frac{1}{k^2}\,\nabla\nabla\cdot\right) \int_S \bar{J}\,G\,dS' \qquad (4.2)$$

where G is the free space Green's function $G = (e^{ikr})/r$.

An equivalent version of (4.2) can be derived using vector and scalar potentials and is

$$R\,\hat{z} \times \bar{J} = \hat{z} \times \bar{E}^{inc} + i\,\frac{kZ}{4\pi}\,\hat{z} \times \int_S \bar{J}\,G\,dS' - \frac{1}{4\pi}\,\hat{z} \times \nabla \int_S \rho/\varepsilon\,G\,dS' \quad (4.3)$$

with

$$\frac{Z}{ik}\,\nabla \cdot \bar{J} = \rho/\varepsilon \qquad (4.4)$$

and this is actually the form used by Naor and Senior (1981). In both instances the differentiation was carried out numerically. To avoid this, we first split (4.2) into Cartesian components as follows:

$$RJ_x = E_x^{inc} + i\,\frac{kZ}{4\pi}\int_S J_x\,\frac{e^{ikr}}{r}\,dS' + \frac{iZ}{4\pi k}\int_S\left(J_x\,\frac{\partial^2}{\partial x^2} + J_y\,\frac{\partial^2}{\partial x\partial y}\right)\frac{e^{ikr}}{r}\,dS' \tag{4.5}$$

$$RJ_y = E_y^{inc} + i\,\frac{kZ}{4\pi}\int_S J_y\,\frac{e^{ikr}}{r}\,dS' + \frac{iZ}{4\pi k}\int_S\left(J_x\,\frac{\partial^2}{\partial x\partial y} + J_y\,\frac{\partial^2}{\partial y^2}\right)\frac{e^{ikr}}{r}\,dS' \;.$$

When discretized and put in matrix form the equations become

$$\begin{aligned}[R](J_x) &= (E_x^{inc}) + [A](J_x) + [B](J_x)\\ [R](J_x) &= (E_y^{inc}) + [C](J_x) + [D](J_y)\end{aligned} \tag{4.6}$$

where $(E_{x(y)}^{inc})$ is an N-element column vector containing the x(y) component of incident field on each subdomain,

$(J_{x(y)})$ is an N-element column vector containing the x(y) component of the induced current at the sampling points,

[R] is an N x N matrix containing the resistivities at the sampling points, and

[A],[B],[C],[D] are N x N matrices containing the mutual impedances between the subdomains.

The equations (4.6) can be written as a single matrix equation as

$$\begin{pmatrix} -E_x^{inc} \\ \hline -E_y^{inc} \end{pmatrix} = \left[\begin{array}{c|c} [A]-[R] & [B] \\ \hline [C] & (D]-[R] \end{array}\right]\begin{pmatrix} J_x \\ \hline J_y \end{pmatrix}$$

↑ 2N-element column matrix; ↑ 2N x 2N matrix; ↑ 2N-element column matrix so that

$$\left(\begin{array}{c} J_x \\ \hline J_y \end{array} \right) = \left[\begin{array}{c|c} [A]-[R] & [B] \\ \hline [C] & [D]-[R] \end{array} \right]^{-} \left(\begin{array}{c} -E_x^{inc} \\ \hline -E_y^{inc} \end{array} \right) . \tag{4.8}$$

The matrix inversion is carried out using standard IBM FORTRAN library routines employing LU decomposition contained in the MTS NAAS package.

The discretization is based on triangular subdomains, with "tent" subsectional basis functions. The latter produce a current which is linearly varying over each subdomain and piecewise continuous over the plate. The concepts are illustrated in Figs. 4.1 and 4.2. If the currents at the vertices V1, V2 and V3 of a triangular subdomain are

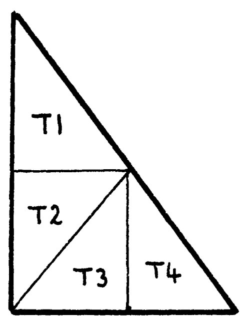

Fig. 4.1: Simple example of triangular subdomains for a triangular plate.

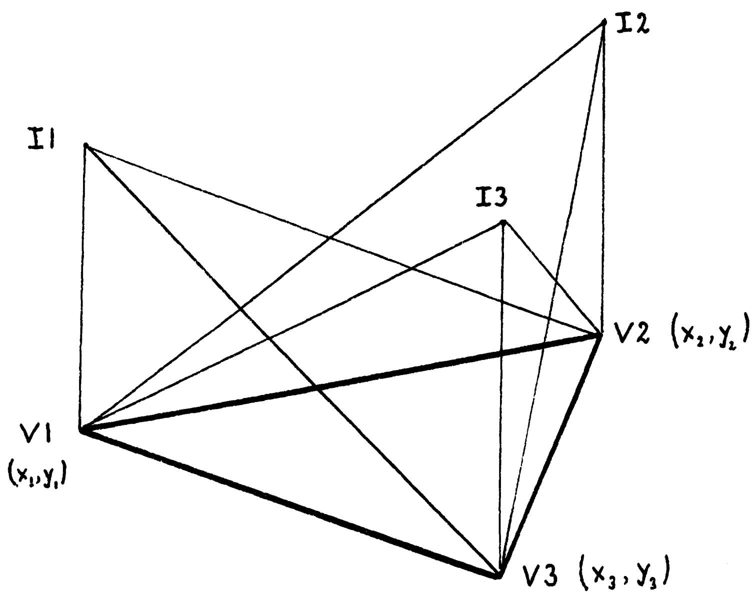

Fig. 4.2: "Tent" current basis functions.

I1, I2 and I3 respectively, then

$$
\begin{aligned}
I1 &= \alpha_1 x_1 + \beta_1 y_1 + \gamma_1 \\
I2 &= \alpha_2 x_2 + \beta_2 y_2 + \gamma_2 \\
I3 &= \alpha_3 x_3 + \beta_3 y_3 + \gamma_3
\end{aligned}
$$

and the current over the entire subdomain is

$$I(x,y) = (\alpha_1 + \alpha_2 + \alpha_3)x + (\beta_1 + \beta_2 + \beta_3)y + \gamma_1 + \gamma_2 + \gamma_3 \quad .$$

The resistivity is specified at the vertices of the subdomains, and assumed to be linearly varying over the subdomain and piecewise continuous over the plate.

Special care is necessary when computing the matrix elements in (4.8) and the novel features of the program are concerned with this. In the progenitor (static) program (Ksienski, 1985b) the vertices of the subdomains were used as the sampling points, and because the singularity of the kernel of the integral equation is integrable, there is no difficulty in computing the field of the current over each subdomain at the vertices. For the dynamic problem, the higher order singularities make this approach impossible, and after examining a variety of generalized function theory methods (e.g., Fikioris, 1965; Lee et al, 1980; Asvestas, 1983; Miron, 1983), an alternative procedure was developed. For each subdomain including the self cell, the kernel was rationalized in the manner shown below and the sampling points were chosen at the centroids of the triangles. The program still computes the current at the vertices, but does so by linearly interpolating the values of the current at the centroids of the adjacent subdomains. Thus, in Fig. 4.3, the current at V1 is found by interpolating the known currents at C1, C2, C3 and C4 found from the integral equations.

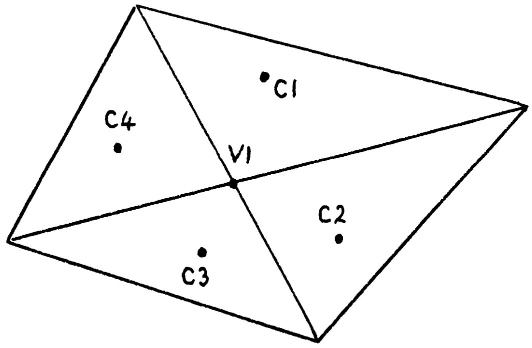

Fig. 4.3: Current computation based on the centroids.

As much as possible of the integration and differentiation was done analytically to maximize the speed and accuracy of the computation. To this end, the singular parts of the dynamic kernel which are most troublesome numerically were treated analytically, and the integrals over the subdomains were evaluated by conversion to line integrals using the method of Wilton et al (1984). Thus, the kernel was written as

$$\frac{e^{ikr}}{r} = \frac{1}{r} + ik - \frac{k^2 r}{2} + \left[\frac{e^{ikr}}{r} - \frac{1}{r} - ik + \frac{k^2 r}{2}\right]$$

and the numerical integration and differentiation was limited to the bracketted terms. The resulting equation for the x component of the current at the jth centroid is as follows:

$$RJ_{xj} = E_{xj}^{inc} + \frac{ikZ}{4\pi} \sum_i \int_{S_i} J_{xi} \left(\frac{1}{r} + ik - \frac{k^2 r}{2}\right) dS' \quad \text{(evaluated analytically)}$$

$$+ \frac{ikZ}{4\pi} \sum_i \int_{S_i} J_{xi} \left(\frac{e^{ikr}}{r} - \frac{1}{r} - ik + \frac{k^2 r}{2}\right) dS' \text{(evaluated numerically)}$$

(cont.)

$$+ \frac{iZ}{4\pi k} \sum_i \frac{\partial^2}{\partial x^2} \int_{S_i} J_{xi} \left(\frac{1}{r} + ik - \frac{k^2 r}{2}\right) dS' \text{ (evaluated analytically)}$$

$$+ \frac{iZ}{4\pi k} \sum_i \int_{S_i} J_{xi} \frac{\partial^2}{\partial x^2} \left(\frac{e^{ikr}}{r} - \frac{1}{r} - ik + \frac{k^2 r}{2}\right) dS' \text{ (evaluated numerically)}$$

$$+ \frac{iZ}{4\pi k} \sum_i \frac{\partial^2}{\partial x \partial y} \int_{S_i} Jy_i \left(\frac{1}{r} + ik - \frac{k^2 r}{2}\right) dS' \text{ (evaluated analytically)}$$

$$+ \frac{iZ}{4\pi k} \sum_i \int_{S_i} J_{yi} \frac{\partial^2}{\partial x \partial y} \left(\frac{e^{ikr}}{r} - \frac{1}{r} - ik + \frac{k^2 r}{2}\right) dS' \text{ (evaluated numerically)}$$

The equation for J_{yj} is similar. Note that the derivatives are taken inside the integrals only when the integrands are non-singular. The numerical integration was carried out using the method of Hammer et al (1956), which is exact for a fifth order polynomial.

Because of time limitations the program is not yet complete. The final part necessary to compute the scattered field of the plate has not been finished, nor does the program make use of any symmetries that the plate may possess. In its present form the program is limited to the computation of the current induced in a plate of arbitrary size and resistivity. The source code consists of approximately 2200 lines of C language code, plus approximately 1000 lines of IBM FORTRAN NAAS library routines. The program compiles and, to judge from the few runs that have been made on a VAX, appears to produce accurate results. Nevertheless, there is further work that must be done and apart from the additions necessary to compute the scattered field we are also aware of some changes that should be made to improve the accuracy.

A source listing is included as an attachment to this report.

APPENDIX A: IBM PC PROGRAM P-RIB-H

A program has been developed for use on an IBM PC computer to determine the high frequency bistatic scattered field of a perfectly conducting strip or ribbon for H polarization.

A perfectly conducting strip of width w occupies the region $0 \le x \le w$, $-\infty < z < \infty$ of the plane $y = 0$ of a Cartesian coordinate system x,y,z, and is illuminated by an H-polarized plane wave having

$$\bar{H}^i = \hat{z}\, e^{-ik(x \cos \phi_0 + y \sin \phi_0)}$$

where a time factor $e^{-i\omega t}$ is assumed and suppressed. At large distances the scattered magnetic field can be written as

$$\bar{H}^s \sim \hat{z} \sqrt{\frac{2}{\pi k\rho}}\, e^{i(k\rho-\pi/4)}\, P(\phi,\phi_0)$$

where ρ,ϕ are cylindrical polar coordinates with $x = \rho \cos \phi$, $y = \rho \sin \phi$. In terms of the two-dimensional far field amplitude P the scattering cross section per unit length in the z direction is

$$\sigma(\phi,\phi_0) = \frac{2\lambda}{\pi} |P(\phi,\phi_0)|^2 \quad , \qquad (1)$$

and in the particular case of backscattering, $\phi_0 = \phi$. From symmetry it is sufficient to consider only $\pi/2 \le \phi \le \pi$ where $\phi = \pi$ corresponds to grazing incidence.

Using a GTD approach, Senior (1979b) developed an expression for the bistatic scattered field of a uniform resistive strip through second order terms, and when specialized to the case of a perfectly conducting strip with $\phi_0 = \phi$, the result is

$$P(\phi,\phi) = -\frac{i}{4}\,\frac{1+\cos\phi}{\cos\phi} + \frac{i}{4}\,\frac{1-\cos\phi}{\cos\phi}\,e^{-2ikw\cos\phi}$$

$$-\left(\frac{2}{\pi kw}\right)^{1/2} e^{-i\pi/4}\,\frac{e^{ikw(1-\cos\phi)}}{\sin\phi} + O([kw]^{-1}) \quad , \quad (2)$$

where the origin of phase is at the left hand edge of the strip. At broadside ($\phi = \pi/2$) the infinities of the first two terms cancel and (2) reduces to

$$P\left(\frac{\pi}{2},\frac{\pi}{2}\right) = \frac{1}{2}\,kw - \frac{i}{2} - \left(\frac{2}{\pi kw}\right)^{1/2} e^{i(kw-\pi/4)} + O([kw]^{-1}) \quad . \qquad (3)$$

For $\phi > \pi/2$ the first and second terms on the right hand side of (2) are the contributions of the front and rear edges respectively, with the former vanishing for grazing incidence. The third term is the second order contribution and this clearly fails when $\phi = \pi$, but by expressing (2) in terms of the half plane current and then using a uniform asymptotic representation of the current, a uniform expression for $P(\phi,\phi)$ valid for $\pi/2 < \phi \leq \pi$ is found to be

$$P(\phi,\phi) = -\frac{i}{4}\,\frac{1+\cos\phi}{\cos\phi}\left[1-\left(\frac{2}{\pi kw}\right)^{1/2} e^{i\pi/4+ikw(1-\cos\phi)}\cot\frac{\phi}{2}\right.$$

$$\left.+\frac{i}{4}\,\frac{1-\cos\phi}{\cos\phi}\,e^{-2ikw\cos\phi}\left[1-\frac{2}{\sqrt{\pi}}\,e^{-i\pi/4}\sin\frac{\phi}{2}\,F\left(\sqrt{2kw}\cos\frac{\phi}{2}\right)\right]^{2}\right.$$

$$+ O([kw]^{-1}) \qquad (4)$$

where $F(\tau)$ is the Fresnel integral

$$F(\tau) = \int_{\tau}^{\infty} e^{iu^2} \, du \quad . \qquad (5)$$

For large values of τ,

$$F(\tau) \sim \frac{i}{2\tau} e^{i\tau^2}$$

and when this is substituted into (4) we recover the asymptotic expression (2). On the other hand, for small τ

$$F(\tau) = \frac{1}{2}\sqrt{\pi}\, e^{i\pi/4} + O(\tau)$$

showing that $P(\pi,\pi) = 0$, as expected. In general

$$F(\tau) = \sqrt{\frac{\pi}{2}} \left\{ \frac{1}{2} - C(\tau^2) + i \left[\frac{1}{2} - S(\tau^2) \right] \right\} \quad , \qquad (6)$$

where C and S are cosine and sine integrals whose computation is described by Boersma (1960). Thus

$$\frac{2}{\sqrt{\pi}} \, e^{-i\pi/4} \sin\frac{\phi}{2} F(\sqrt{2kw}\cos\phi) = \sin\frac{\phi}{2} \{1 - C(u) - S(u) + i[C(u) - S(u)]\}$$

where

$$u = kw(1 + \cos\phi) \quad .$$

The introduction of the Fresnel integral to provide a uniform behavior in the vicinity of $\phi = \pi$ produces a difficulty near broadside since the infinities of the first two terms on the right hand side of (4) no longer cancel precisely when $\phi = \pi/2$. For this reason it is desirable to introduce a second Fresnel integral whose main effect is to produce a uniform behavior near $\phi = 0$ in spite of the fact that we are only concerned with $\phi \geq \pi/2$. The result is

$$P(\phi,\phi) = -\frac{i}{4}\,\frac{1+\cos\phi}{\cos\phi}\left\{1 - \frac{2}{\sqrt{\pi}}\,e^{-i\pi/4}\,\cos\frac{\phi}{2}\,F\left(\sqrt{2kw}\,\sin\frac{\phi}{2}\right)\right\}^2$$

$$+\frac{i}{4}\,\frac{1-\cos\phi}{\cos\phi}\,e^{-2ikw\cos\phi}\left\{1 - \frac{2}{\sqrt{\pi}}\,e^{-i\pi/4}\,\sin\frac{\phi}{2}\,F\left(\sqrt{2kw}\,\cos\frac{\phi}{2}\right)\right\}^2$$

$$+ \, O([kw]^{-1}) \qquad (7)$$

valid for $0 \leq \phi \leq \pi$. The infinities at $\phi = \pi/2$ now cancel precisely, and we remark that (7) is identical to the result obtained by asymptotic expansion of the uniform expression of Khaskind and Vainshteyn (1964).

A program designated P-RIB-H (perfectly conducting-ribbon-H polarization) has been written to compute the function $P(\phi,\phi)$ using (7). It is coded in BASIC on our IBM personal computer and computes σ/λ (in dB), $|P|$ and arg P (in degrees) as functions of the angle ϕ for $91 \leq \phi \leq 179$ degrees. The broadside angle $\phi = 90$ degrees is omitted to avoid numerical problems, but in practice, a knowledge of the far field at 91 degrees is sufficient to determine the broadside return. When $\phi = 180$ degrees, $|P| = 0$, and this angle is also omitted to avoid problems in computing arg P.

Some results obtained with P-RIB-H are compared with those of an integral equation solution by the moment method in Figs. 1 through 6.

The agreement is good for all $w/\lambda \gtrsim 0.5$, though we do observe a somewhat larger phase discrepancy than expected for $\phi \geq 130$ degrees with $w/\lambda = 2$. The traveling wave lobe is faithfully reproduced and this enables us to use (7) or, indeed, (4) for studying traveling wave effects. From the GTD viewpoint the Fresnel integrals were introduced to match the second order expansion for $0 < \phi < \pi$ into the zero values at grazing incidence, but one interesting consequence is that (7), as opposed to (2), is reasonably accurate even for small values of w/λ.

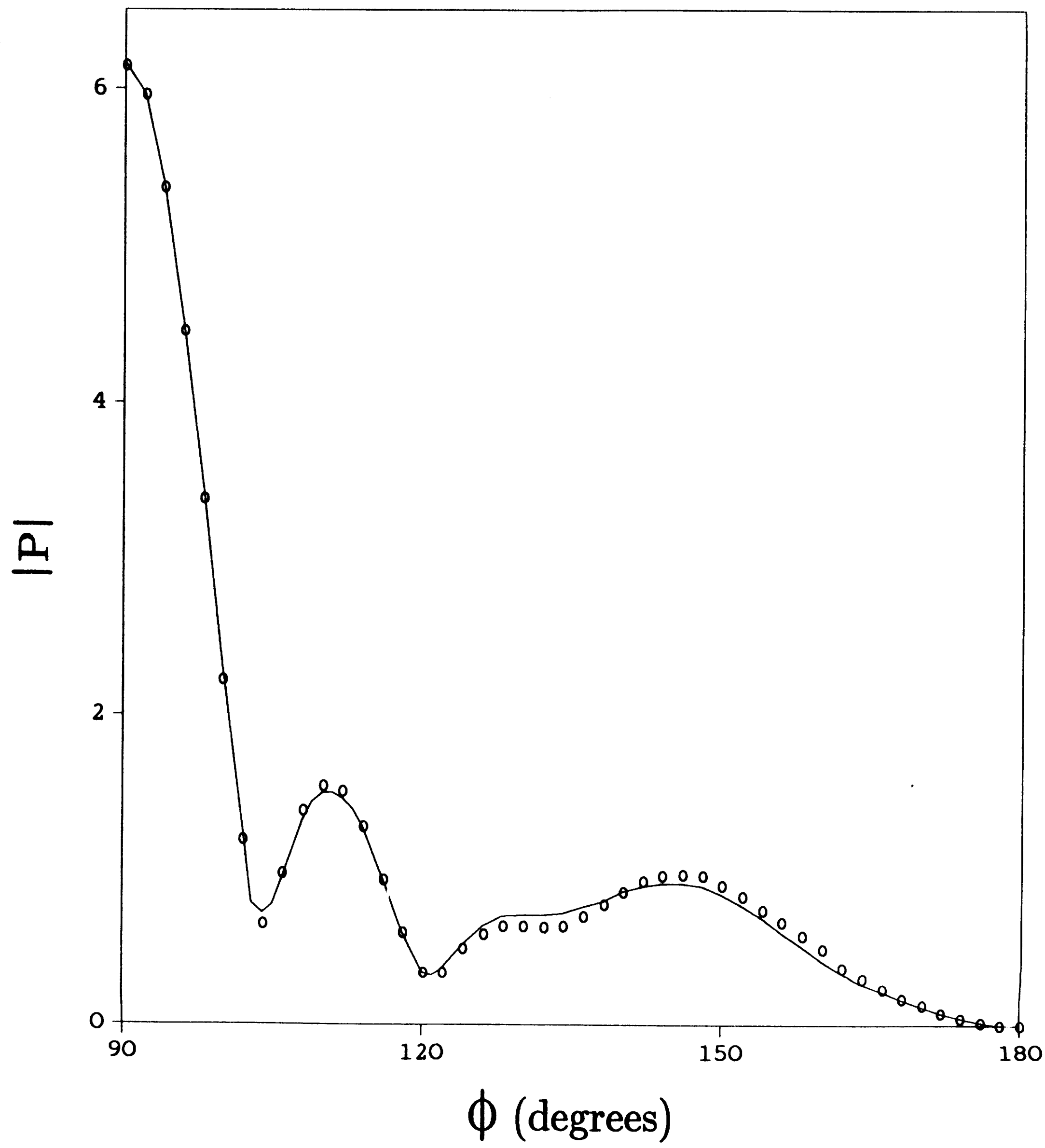

Fig. 1: |P| for w/λ = 2 computed using P-RIB-H (ooo) compared with the results of an integral equation method (——).

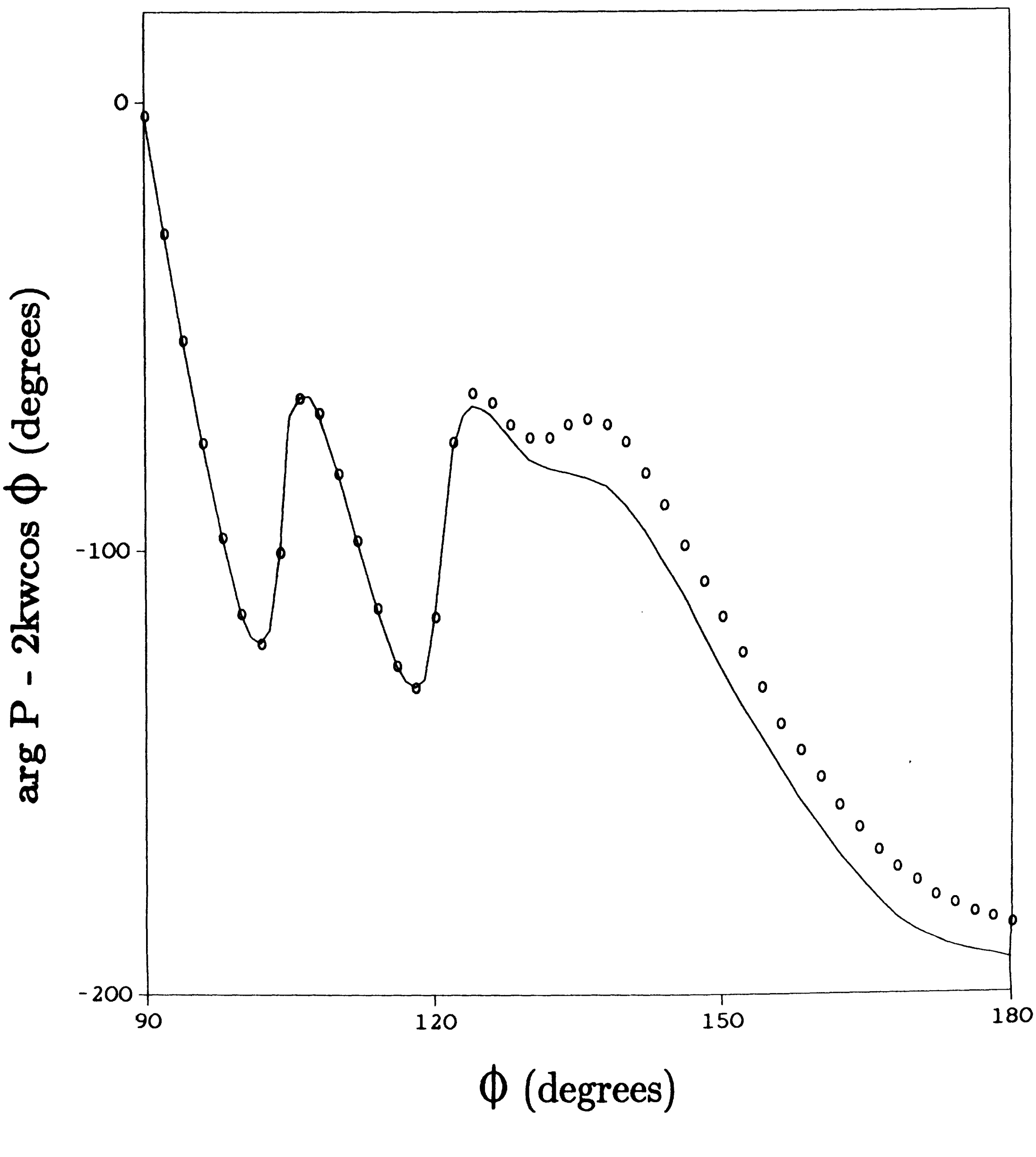

Fig. 2: Arg P for w/λ = 2 computed using P-RIB-H (ooo) compared with the results of an integral equation method (——).

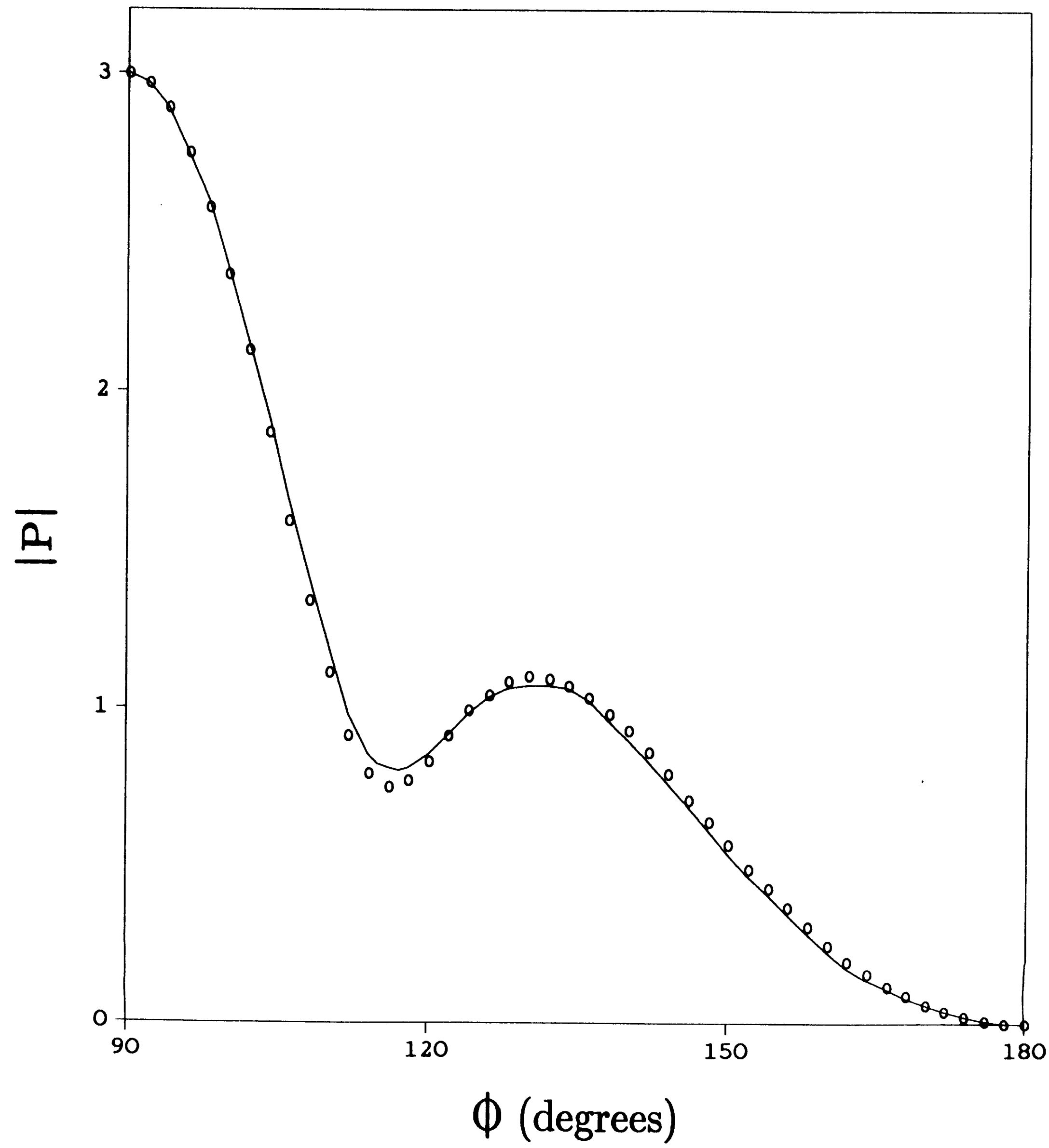

Fig. 3: Arg P for w/λ = 1 computed using P-RIB-H (ooo) compared with the results of an integral equation method (——).

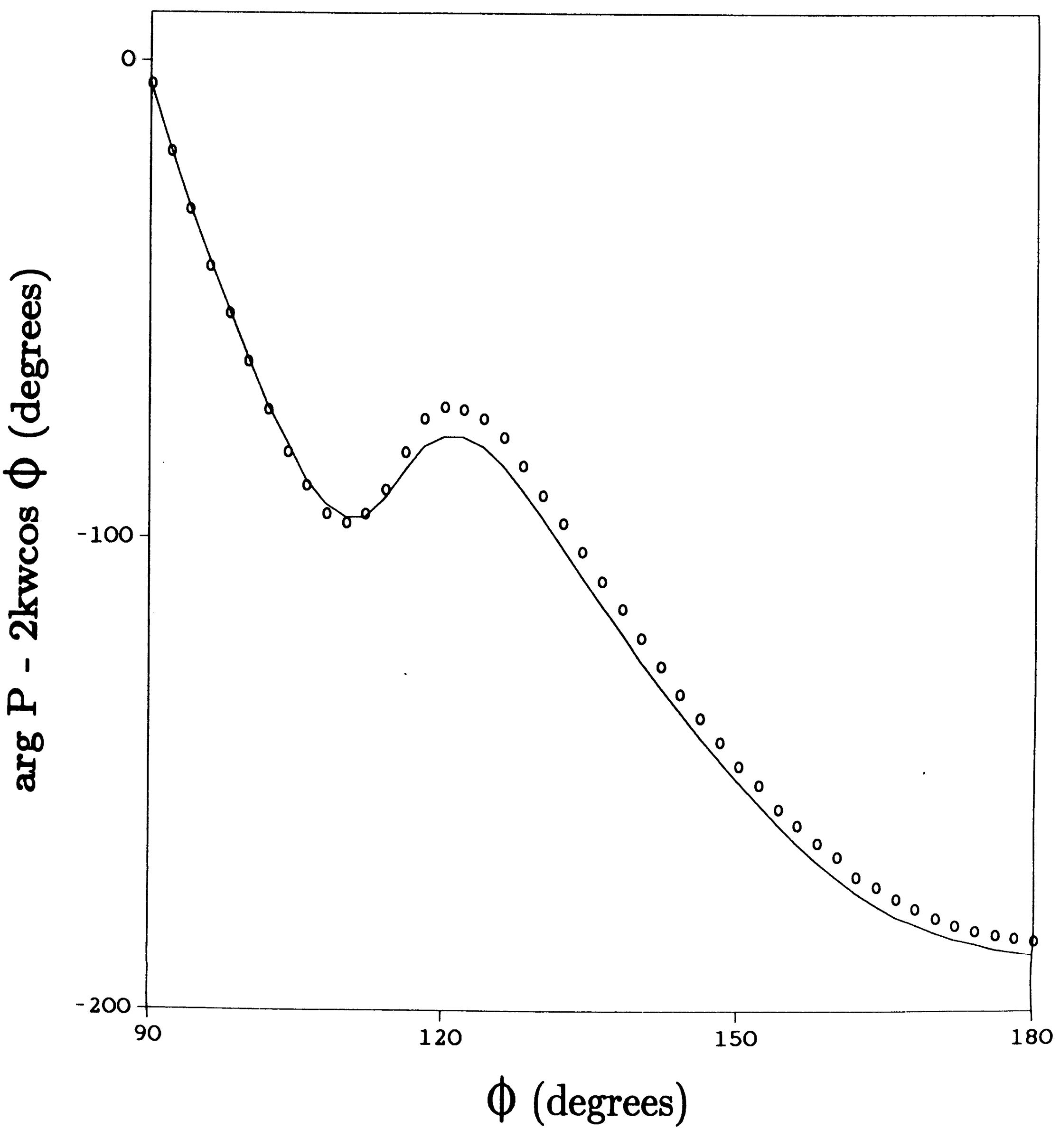

Fig. 4: Arg P for $w/\lambda = 1$ computed using P-RIB-H (ooo) compared with the results of an integral equation method (——).

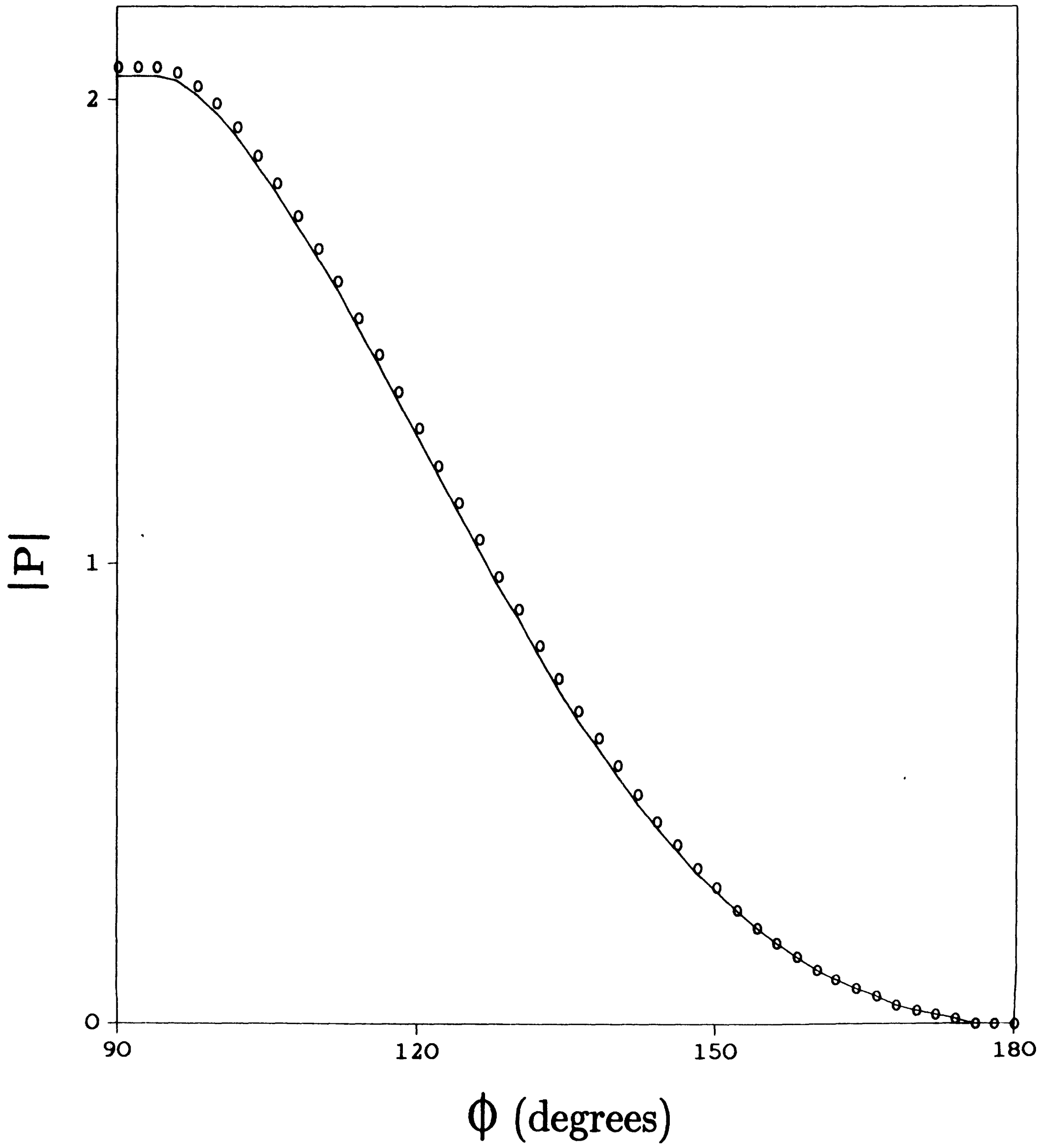

Fig. 5: Arg P for w/λ = 0.5 computed using P-RIB-H (ooo) compared with the results of an integral equation method (——).

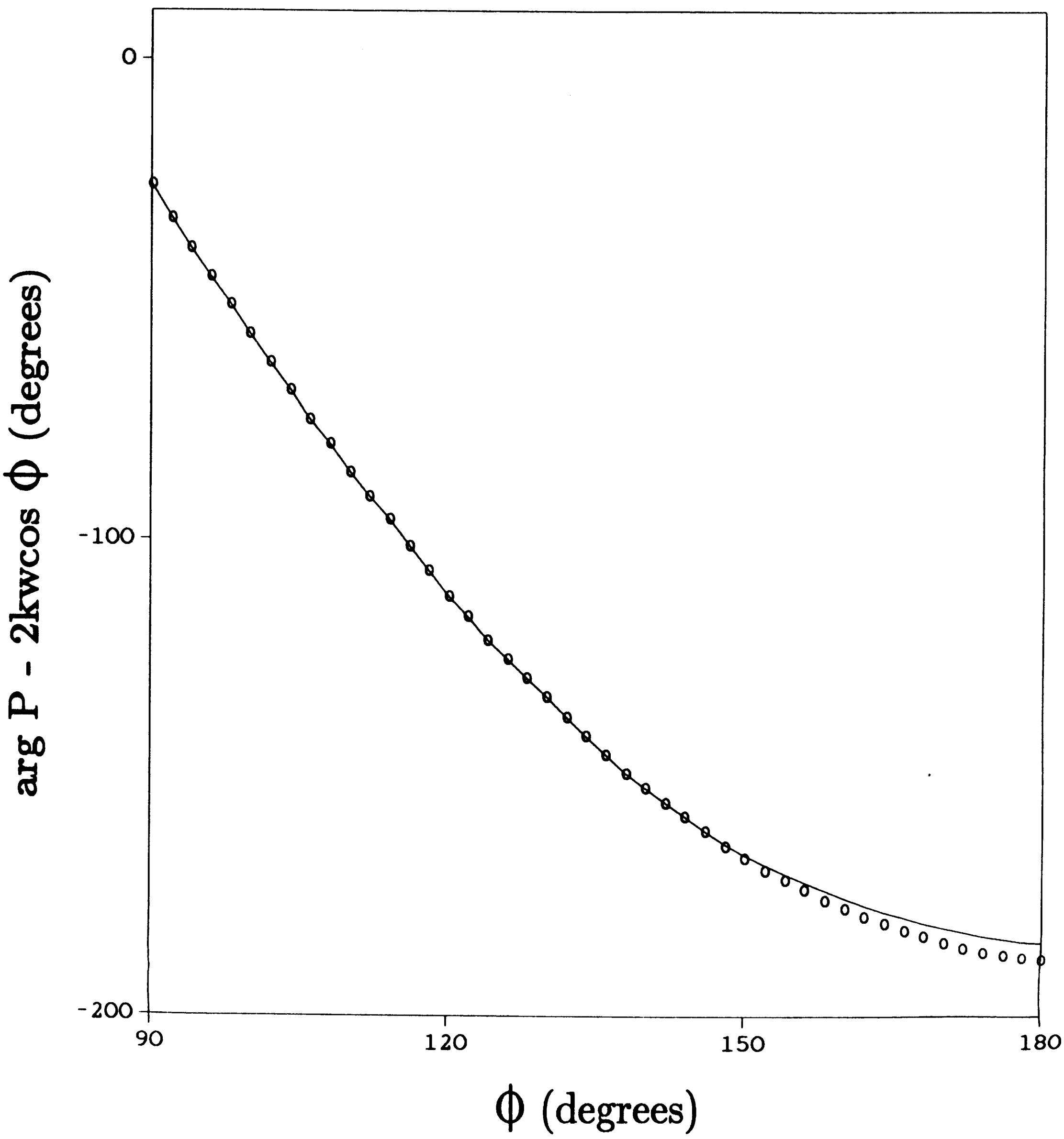

Fig. 6: Arg P for $w/\lambda = 0.5$ computed using P-RIB-H (ooo) compared with the results of an integral equation method (——).

```
5 '                    PROGRAM  P-RIB-H
7 '
10 'THIS PROGRAM CALCULATES THE BACKSCATTERED FIELD OF A PERFECTLY CONDUCTING
20 'STRIP (OR RIBBON) FOR H POLARIZATION  USING A SECOND ORDER UNIFORM GTD
30 'APPROXIMATION. THE QUANTITIES COMPUTED ARE BACKSCATTERING CROSS SECTION PER
   UNIT LENGTH SIGMA/LAMDA [DB],MAG P AND PHASE P [DEGREES],WHERE P IS THE FAR
40 'FIELD COEFFICIENT, AS FUNCTIONS OF THE INCIDENT ANGLE PHI, 91 <= PHI <= 179
45 '[DEGREES], FOR A SPECIFIED VALUE OF KW WHERE W IS THE STRIP WIDTH.
50 '
60 'THE USER ENTERS:
70 '   1) DX -  THE ANGULAR STEP INCREMENT USED IN PLOTTING. .5 TO 2 DEGREES
80 '            IS GOOD
90 '   2) W/LAMDA -  WHERE LAMDA =WAVELENGTH, W=STRIP WIDTH
100 '   3) MAXP -  THE MAXIMUM HEIGHT FOR THE PLOT.
110 INPUT "ENTER ANGLE INCREMENT IN DEGREES    ",DX
120 INPUT "ENTER W/LAMDA   ",W
130 INPUT "ENTER MAXIMUM Y COORDINATE   ",MAXP
132 LPRINT "W/LAMDA = "; W
140 PI=3.141593
145 KW=2*PI*W
150 R2=SQR(2)
160 RPI=SQR(PI)
161 D=.1
162 D1=.025
170 '
180 'SET UP THE SCREEN
190 CLS
195 KEY OFF
200 SCREEN 2
210 WINDOW (1,-2)-(3.5,MAXP+1)
220 LINE (PI/2,0)-(PI,0)
230 LINE (PI/2,0)-(PI/2,MAXP)
232 FOR I=1 TO MAXP STEP 1
233 LINE (PI/2-D1,I)-(PI/2+D1,I)
234 NEXT
235 FOR J= 1 TO 9 STEP 1
236 LINE (PI/2+J*PI/18,D)-(PI/2+J*PI/18,-D)
237 NEXT
240 LOCATE 5,40:  PRINT "W/LAMDA = ";W
241 LOCATE 3,15:  PRINT USING "##.#";MAXP
242 LOCATE 21,15: PRINT "0.0"
243 LOCATE 22,18: PRINT "90"
244 LOCATE 22,70: PRINT "180"
245 LOCATE 22,35: PRINT "ANGLE [DEGREE]"
246 LOCATE 10,13: PRINT "MAG P"
250 'ALL THE FOLLOWING FUNCTIONS ARE DEFINED BECAUSE THERE IS NO
260 'COMPLEX ARITHEMATIC.  THE FINAL FUNCTION TO BE PLOTTED IS CALLED
270 'MAGP.
280 DEF FNA(X)=(1+COS(X))/(4*COS(X))
300 DEF FNC(X)=(1-COS(X))/(4*COS(X))
320 DEF FNANG3(X)=-2*KW*COS(X)
370 DEF FNREP(X)= FA*FR1I - FC*(FR2R*SIN(ANG3) + FR2I*COS(ANG3))
380 DEF FNIMPP(X)= -FA*FR1R +FC*(FR2R*COS(ANG3) -FR2I*SIN(ANG3))
390 DEF FNMAGP(X)=SQR(FNREP(X)^2+FNIMPP(X)^2)
400 '
```

```
...
410 'THE FIRST POINT TO BE PLOTTED IS FOR A VALUE OF 91 DEGREES.  THE FOLLOWING
420 'PRESETS THIS FIRST POINT SO WE CAN MOVE INTO THE LOOP.
430 'WE DO NOT USE 90 DEGREES TO AVOID DIVISION BY ZERO.
631 LPRINT "ANGLE   SIGMA/LAM [DB]        MAG P    PHASE P
640 '
650 'NOW WE ARE READY TO LOOP THROUGH AND PLOT THE FUNCTION FROM 91 DEGREES
660 'TO 179 DEGREES.
662 I=1
670 DX=DX*(PI/180)
680 FOR X=PI/2+PI/180 TO PI-PI/180  STEP DX
690 Z=KW*(1-COS(X))
700 GOSUB 900
710 CS1=CS
720 SN1=SN
730 Z= KW*(1+ COS(X))
740 GOSUB 900
750 CS2=CS
760 SN2=SN
810 ANG3=FNANG3(X)
820 FA=FNA(X)
830 FC=FNC(X)
840 FR1= 1-COS(X/2)*(1-CS1-SN1):  FR2=COS(X/2)*(SN1-CS1)
850 FS1= 1-SIN(X/2)*(1-CS2-SN2):  FS2=SIN(X/2)*(SN2-CS2)
855 FR1R= FR1^2 - FR2^2 :  FR1I= 2* FR1 * FR2
857 FR2R= FS1^2 - FS2^2 :  FR2I= 2* FS1 * FS2
860 YY=FNMAGP(X)
865 IF I=1, THEN PRESET (PI/2,YY)    ELSE  LINE  -(X,YY)
870 I=I+1
871 GOSUB 1200
880 NEXT
890 END
900 '
910 'THIS SUBROUTINE COMPUTES THE FRESNEL INTEGRAL.  IT USES TWO DIFFERENT
920 'APPROXIMATIONS FOR ARGUMENTS GREATER OR LESS THAN FOUR.
930 'REFERENCES:  "COMPUTATION OF FRESNEL INTEGRAL" BY BOERSMA, MATHEMATICAL
940 '              TABLES AND OTHER AIDS TO COMPUTATION, VOL. 14, 1960
950 '              NO. 72, PAGE 380
960 'C(X)=REAL PART OF INTEGRAL EXP(it)/SQRT(2*pai*t) FROM 0 TO X
970 'S(X)=IMAGINARY PART OF THE ABOVE INTEGRAL
990 IF Z>4 GOTO 1080
1000 ZY=Z/4
1010 AR=1.59576914# - 1.702E-06*Z  -6.808568854#*ZY^2 -5.76361E-04*ZY^3  + 6.920
691902#*ZY^4 -.016898657#*ZY^5 -3.05048566#*ZY^6  - .075752419#*ZY^7 +.850663781
#*ZY^8 - .025639041#*ZY^9 -.15023096#*ZY^10 + .034404779#*ZY^11
1030 AI=-(-3.3E-08 + 4.255387524#*ZY-9.281E-05*ZY^2 -7.7800204#*ZY^3 - 9.520895E
-03*ZY^4  + 5.075161298#*ZY^5 - .138341947#*ZY^6 - 1.363729124#*ZY^7 -.403349276
#*ZY^8 + .702222016#*ZY^9 -.216195929#*ZY^10 + .019547031#*ZY^11)
1040 CS=SQR(ZY)*(AR*COS(Z)-AI*SIN(Z))
1050 SN=SQR(ZY)*(AR*SIN(Z)+AI*COS(Z))
1070 RETURN
```

```
1080 ZZ=4/Z
1110 BR=-.024933975#*ZZ + 3.936E-06*ZZ^2 + 5.770956E-03*ZZ^3 +6.89892E-04*ZZ^4
-9.497136E-03*ZZ^5 + .011948809#*ZZ^6 -6.748873E-03*ZZ^7 +2.4642E-04*ZZ^8
+2.102967E-03*ZZ^9-1.21793E-03*ZZ^10 +2.33939E-04*ZZ^11
1120 BI=-(.19947114# +2.3E-08*ZZ -9.351341E-03*ZZ^2 +2.3006E-05*ZZ^3 +4.851466E-
03*ZZ^4 +1.903218E-03*ZZ^5 -.017122914#*ZZ^6 +2.906467E-02*ZZ^7 -.027928955#*ZZ^
8 +.016497308#*ZZ^9 -5.598515E-03*ZZ^10 +8.38386E-04*ZZ^11)
1130 CS=.5 + SQR(ZZ)*(BR*COS(Z) - BI*SIN(Z))
1140 SN=.5 + SQR(ZZ)*(BR*SIN(Z) + BI*COS(Z))
1180 RETURN
1200 ANGLE= 57.29578*X
1210 IF FNMAGP(X)=0,THEN SCS=-999.99 ELSE SCS=20*LOG(FNMAGP(X))/LOG(10)-1.9612
1215 FFF=FNIMPP(X)/FNREP(X)
1220 IF FNREP(X)=0, THEN PHASY=-999.99 ELSE PHASY=57.29578*ATN(FFF)
1222 IF FNREP(X) >= 0,THEN PHASE= PHASY   ELSE PHASE= PHASY +SGN(FNIMPP(X))*180
1230 LPRINT USING "#####.##"; ANGLE;SCS;
1240 LPRINT USING "##########.#####"; FNMAGP(X);
1250 LPRINT USING "#####.###"; PHASE
1260 RETURN
```

IEEE Transactions on Antennas and
Propagation, Vol. AP-33, No. 5, May 1985

Combined Resistive and Conductive Sheets

THOMAS B. A. SENIOR, FELLOW, IEEE

Abstract—To simulate a thin layer of material whose permittivity and permeability both differ from the values for the surrounding medium, a combination resistive and conductive sheet is defined and its properties described.

I. INTRODUCTION

Thin layers of lossy material are of obvious interest for cross section reduction purposes. A mathematical model of such a layer is a resistive sheet, and during the last few years the scattering from this type of sheet has been extensively explored [1]-[3]. The electromagnetic dual of an electrically resistive sheet is a "magnetically conductive" one [4], and to simulate a thin layer whose permittivity and permeability both differ from the surrounding medium, it may be necessary to include this sheet as well.

The properties of a combined sheet consisting of coincident resistive and conductive ones are examined. Although the two sheets are, in general, coupled, with each affecting the scattering from the other, decoupling occurs when the sheets lie in a plane. This is true regardless of the (planar) configuration, and has implications for the development of analytical and numerical methods to predict the scattering from lossy plates.

II. BOUNDARY CONDITIONS

An electrically resistive sheet is simply an electric current sheet whose strength is proportional to the tangential electric field at its surface, and in recent years the concept of such a sheet has found many useful applications. As noted by Levi-Civita (see [4, p. 19]), its electromagnetic properties are completely specified by its resistivity R in ohms per square, and the boundary conditions at its surface are

$$\hat{n}\times\bar{E}(+)-\hat{n}\times\bar{E}(-)=0, \tag{1}$$

$$\hat{n}\times\bar{H}(+)-\hat{n}\times\bar{H}(-)=\bar{J} \tag{2}$$

with

$$\hat{n}\times\{\hat{n}\times\bar{E}(\pm)\}=-R\bar{J} \tag{3}$$

where $\hat{n}$ is a unit vector normal drawn outward to the positive (plus) side of the sheet and $\bar{J}$ is the total electric current supported. Equation (1) implies that there is no magnetic current and, hence, that the permeability of the corresponding layer is the same as that of the surrounding (free space) medium. Because of this, (3) can be written as

$$\hat{n}\times\{\hat{n}\times[\bar{E}(+)+\bar{E}(-)]\}=-2R\bar{J}, \tag{4}$$

and in the special case $R=0$ the sheet is perfectly conducting, whereas if $R=\infty$ the sheet is no longer present. For a material of large conductivity σ,

$$R=(\sigma\tau)^{-1}$$

where τ is the thickness of the layer, and an alternative expression is [5]

$$R=\frac{iZ}{(\epsilon_r-1)k\tau} \tag{5}$$

valid if $|\epsilon_r-1|\gg 1$. In (5), k and $Z\,(=1/Y)$ are the propagation constant and impedance, respectively, of free space, ϵ_r is the relative permittivity of the layer material, and a time factor $e^{-i\omega t}$ has been assumed.

The electromagnetic dual is a magnetically conductive sheet [6] supporting only a magnetic current $\bar{J}^*$, and by analogy with (1), (2), and (4) the boundary conditions at its surface are

$$\hat{n}\times\bar{H}(+)-\hat{n}\times\bar{H}(-)=0, \tag{6}$$

$$\hat{n}\times\bar{E}(+)-\hat{n}\times\bar{E}(-)=-\bar{J}^* \tag{7}$$

with

$$\hat{n}\times\{\hat{n}\times[\bar{H}(+)+\bar{H}(-)]\}=-2R^*\bar{J}^* \tag{8}$$

where R^* is the conductivity. By virtue of (6), a layer modeled by such a sheet must have its relative permittivity unity, and from (5) an expression for R^* is

$$R^*=\frac{iY}{(\mu_r-1)k\tau} \tag{9}$$

valid for $|\mu_r-1|\gg 1$ where μ_r is the relative permeability of the layer material.

III. COMBINATION SHEET

To model a layer whose permittivity and permeability both differ from their free space values a logical approach is to consider a combination sheet consisting of coincident resistive and conductive sheets at which the boundary conditions are (2)-(4) and (6)-(8). If $\hat{s}$ and $\hat{t}$ are unit vectors in the plane of the sheet such that at every point $\hat{s}\cdot\hat{t}=0$ and $\hat{s}\times\hat{t}=\hat{n}$, implying that $\hat{s}$, $\hat{t}$, and $\hat{n}$ form a right-handed system, (3), (4), (7), and (8) can be written as

$$E_s(+)+E_s(-)=-2R\{H_t(+)-H_t(-)\},$$

$$E_s(+)-E_s(-)=-(2R^*)^{-1}\{H_t(+)+H_t(-)\};$$

$$E_t(+)+E_t(-)=2R\{H_s(+)-H_s(-)\},$$

$$E_t(+)-E_t(-)=(2R^*)^{-1}\{H_s(+)+H_s(-)\}.$$

Addition and subtraction of the equations in pairs gives

$$\begin{aligned}E_s(\pm)&=-R\left(1\pm\frac{1}{4RR^*}\right)H_t(+)+R\left(1\mp\frac{1}{4RR^*}\right)H_t(-)\\E_t(\pm)&=R\left(1\pm\frac{1}{4RR^*}\right)H_s(+)-R\left(1\mp\frac{1}{4RR^*}\right)H_s(-)\end{aligned} \tag{10}$$

Manuscript received October 4, 1984.

The author is with the Radiation Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

showing that, in general, the sheet is partially transparent, but if

$$4RR^* = 1 \tag{11}$$

the conditions (10) reduce to Leontovich boundary conditions [7] on the two sides of the sheet. The combined sheet is then opaque and is simply an impedance (boundary condition) sheet with the same surface impedance $\eta = 2R$ on each side. In view of (11) an equivalent formula for the surface impedance is $\eta = (R/R^*)^{1/2}$, and when the expressions for R and R^* are inserted, we find

$$\eta = Z\left(\frac{\mu_r - 1}{\epsilon_r - 1}\right)^{1/2} \cong Z\left(\frac{\mu_r}{\epsilon_r}\right)^{1/2} \tag{12}$$

as expected.

Under most circumstances the resistive and conductive sheets comprising a combination sheet are coupled inasmuch as the strength (as measured by the current supported) and the scattering of each sheet are effected by the presence of the other. To illustrate this fact, consider a closed cylindrical sheet illuminated by an E-polarized wave. In terms of the cylindrical polar coordinates ρ, ϕ, z, the sheet is defined as the surface $\rho = a$ and the incident plane wave is assumed to have

$$\bar{E}^i = \hat{z}e^{-ik\rho\cos\phi}.$$

By expanding the scattered and interior fields in cylindrical wave functions, the solution for each type of sheet can be obtained by mode matching. In particular, the interior field is

$$\bar{E}^{int} = \hat{z}\sum_{n=0}^{\infty} \epsilon_n(-i)^n A_n J_n(k\rho)\cos n\phi \tag{13}$$

where $\epsilon_0 = 1$ and $\epsilon_n = 2, n > 0$, and for a resistive sheet

$$A_n = \left\{1 + \frac{\pi ka}{2}\frac{Z}{R}J_n(ka)H_n^{(1)}(ka)\right\}^{-1}. \tag{14}$$

For a conductive sheet

$$A_n = \left\{1 + \frac{\pi ka}{2}\frac{Y}{R^*}J_n'(ka)H_n^{(1)\prime}(ka)\right\}^{-1} \tag{15}$$

where the prime denotes differentiation with respect to ka, but for a combination sheet

$$A_n = \left(1 - \frac{1}{4RR^*}\right)\left\{1 + \frac{1}{4RR^*} + \frac{\pi ka}{2}\frac{Z}{R}J_n(ka)H_n^{(1)}(ka)\right.$$

$$\left. + \frac{\pi ka}{2}\frac{Y}{R^*}J_n'(ka)H_n^{(1)\prime}(ka)\right\}^{-1}. \tag{16}$$

If (11) is satisfied the field interior to the combination sheet vanishes, showing that the sheet is then opaque, and we also observe that (14) and (15) can be obtained from (16) by taking the limits $R^* \to \infty$ and $R \to \infty$, respectively. More importantly, however, the coefficient A_n for the combination sheet is not the sum of the coefficients for the individual sheets, which demonstrates the coupling.

IV. PLANAR SHEET

Although the sheets comprising a combination sheet are generally coupled, an exception occurs in the special case of a planar sheet. To prove this, let

$$\bar{E} = \bar{E}^i + \bar{E}^{(1)} + \bar{E}^{(2)}$$

where $\bar{E}^{(1)}$ and $\bar{E}^{(2)}$ are the electric fields radiated by the induced electric and magnetic currents, respectively. If $\bar{H}^{(1)}$ and $\bar{H}^{(2)}$ are the associated magnetic fields, the symmetry about a planar magnetic sheet implies

$$\hat{n} \times \{\bar{H}^{(2)}(+) - H^{(2)}(-)\} = 0$$

and

$$\hat{n} \times \bar{E}^{(2)}(\pm) = \pm\tfrac{1}{2}\bar{J}^*.$$

Hence

$$\hat{n} \times \{\bar{E}^{(2)}(+) + \bar{E}^{(2)}(-)\} = 0,$$

and when these expressions are substituted into the boundary condition (4) for a combination sheet, we find

$$\hat{n} \times \{\hat{n} \times [\bar{E}^i(+) + \bar{E}^{(1)}(+) + \bar{E}^i(-) + \bar{E}^{(1)}(-)]\}$$

$$= -R\hat{n} \times \{\bar{H}^{(1)}(+) - \bar{H}^{(1)}(-)\}$$

which is simply the condition for the corresponding resistive sheet in isolation. Similarly, (8) reduces to the condition for a conductive sheet by itself, showing that for a planar combination sheet the resistive and conductive parts scatter independently of one another.

This is true for a plate of any (planar) configuration and it is therefore sufficient to restrict the development of analytical and/or numerical procedures to the simple case of a resistive plate. By application of the duality principle the solution for the corresponding conductive plate can be deduced, and the solution for the combination plate then follows by addition of the component solutions. Thus, for a planar plate, we can achieve the added generality of a combination sheet without any increase in complexity. The resulting plate is partially transparent unless (11) is satisfied, which represents the special case when the impedance boundary condition is applied.

REFERENCES

[1] T. B. A. Senior, "Scattering by resistive strips," *Radio Sci.*, vol. 14, pp. 911-924, Sept./Oct. 1979.

[2] ——, "Backscattering from resistive strips," *IEEE Trans. Antennas Propagat.*, vol. AP-27, pp. 808-813, Nov. 1979.

[3] T. B. A. Senior and V. V. Liepa, "Backscattering from tapered resistive strips," *IEEE Trans. Antennas Propagat.* vol. AP-32, pp. 747-751, July 1984.

[4] H. Bateman, *Electrical and Optical Wave Motion*, Cambridge, England: Cambridge Univ. Press, 1915.

[5] R. F. Harrington and J. R. Mautz, "An impedance sheet approximation for thin dielectric shells," *IEEE Trans. Antennas Propagat.*, vol. AP-23, pp. 531-534, July 1975.

[6] T. B. A. Senior, "Some extensions of Babinet's principle in electromagnetic theory," *IEEE Trans. Antennas Propagat.*, vol. AP-25, pp. 417-420, May 1977.

[7] ——, "Impedance boundary conditions for imperfectly conducting surfaces," *Appl. Sci. Res.*, vol. 8, sec. B, pp. 418-436, 1960.

References

Abramowitz, M. and I. E. Stegun (1964), Handbook of Mathematical Functions, National Bureau of Standards Appl. Math Series 55, U.S. Government Printing Office, Washington, D.C,; p. 321.

Asvestas, J. S. (1983), "Comments on 'Singularity in Green's function and its numerical evaluation'", IEEE Trans. Antennas Propagat. AP-31, 174-177.

Boersma, J. (1960), "Computation of Fresnel integrals", Math Tables and Other Aids to Comp. 14, 380.

Chang, S. and V. V. Liepa (1967), "Measured backscattering cross section of thin wires", University of Michigan Radiation Laboratory Report No. 8077-4-T.

Fikioris, J. G. (1965), "Electromagnetic field inside a current carrying region", J. Math. Phys. 6, 1617-1620.

Hammer, P. C., O. J. Marlowe and A. H. Stroud (1956), "Numerical integration over simplexes and cones", Math Tables and Other Aids to Comp. 10, 130-137.

Khaskind, M. D. and L. A. Vainshteyn (1964), "Diffraction of plane waves by a slit and a tape", Radio Eng. Electron. 10, 1492-1502.

Ksienski, D. A. (1985a), "A method of resolving data into two maximally smooth components", Proc. IEEE 73, 166-168.

Ksienski, D. A. (1985b), "Scattering by distributions of small thin particles", Ph.D. dissertation, University of Michigan.

Lee, S. W., J. Boersma, C. L. Law and G. A. Deschamps (1980), "Singularity in Green's function and its numerical evaluation", IEEE Trans. Antennas Propagat. AP-28, 311-317.

Maliuzhinets, G. D. (1958), "Excitation, reflection and emission of surface waves from a wedge with given face impedances", Sov. Phys. Dokl. 3, 752-755.

Miron, D. B. (1983), "The singular integral problem in surfaces", IEEE Trans. Antennas Propagat. AP-31, 507-509.

Naor, M. and T.B.A. Senior (1981), "Scattering by resistive plates", University of Michigan Radiation Laboratory Report No. 018803-1-T.

Peters, L., Jr. (1958), "End-fire echo area of long, thin bodies", IRE Trans. Antennas Propagat. 6, 133-139.

Ruck, G. T., D. E. Barrick, W. D. Stuart and C. K. Krichbaum (1970), Radar Cross Section Handbook (Plenum Press, New York); p. 132.

Senior, T.B.A. (1979a), "Backscattering from resistive strips", IEEE Trans. Antennas Propagat. AP-27, 808-813.

Senior, T.B.A. (1979b), "Scattering by resistive strips", Radio Sci. 14, 911-924.

Senior, T.B.A. (1981), "The current induced in a resistive half plane", Radio Sci. 16, 1249-1254.

*Senior, T.B.A. (1985), "Combined resistive and conductive sheets", IEEE Trans. Antennas Propagat. AP-33, 577-579.

Senior, T.B.A. and V. V. Liepa (1984), "Backscattering from tapered resistive strips", IEEE Trans. Antennas Propagat. AP-32, 747-751.

*Senior, T.B.A. and S. J. Yang (1984), "Traveling waves on thin bodies", Electronics Lett. 20, 1050-1051.

*Volakis, J. L. and T.B.A. Senior (1985), "Simple expressions for a function occurring in diffraction theory", IEEE Trans. Antennas Propagat. AP-33, 678-680.

Wilton, D. R., S. M. Rao, A. W. Glisson, D. H. Schaubert, O. M. Al-Bundak and C. M. Butler (1984), "Potential integrals for uniform and linear source distributions on polygonal and polyhedral domains", IEEE Trans. Antennas Propagat. AP-32, 276-281.

---

*Supported by present contract.

```
/* function definitions */
#include <math.h>
double darea(),area(),sumang();
/* macro definitions */
#define Darray(Ara,I1,I2,L1,L2) Ara[(I1)+(I2)*(L1)]  /* Macro to generate other\
                                                        macros to mimic fortran\
                                                        arrays    */
#define Mnumpoi 100 /* maximum number of points */
#define Mnumtri 100 /* maximum number of triangular patches */
#define Mnumatri 10 /* maximum number of triangles which may share a \
        common point, 6 is average for internal points, 8 accounts for\
        most schemes, thus 10 is a reasonable upper bound */
#define True 1 /* value of logical true */
#define False 0 /* value of logical false */
#define Tri(J1,J2) Darray(tri,J1,J2,Mnumatri,Mnumpoi)  /* set up tri as two\
                          dimensional array with limits (Mnumatri, Mnumpoi) */
#define Poi1(J1,J2) Darray(poi1,J1,J2,Mnumatri,Mnumpoi)  /* set up poi1 as two\
                          dimensional array with limits (Mnumatri, Mnumpoi) */
#define Poi2(J1,J2) Darray(poi2,J1,J2,Mnumatri,Mnumpoi)  /* set up poi2 as two\
                          dimensional array with limits (Mnumatri, Mnumpoi) */
#define Poi(J1,J2) Darray(poi,J1,J2,Mnumtri,3) /* set up poi as two dimensional\
                          array with limits (Mnumtri,3)  */
#define Matrix(J1,J2) Darray(matrix,J1,J2,Mnumpoi,Mnumpoi)  /* set up matrix as\
                          two dimensional array with limits (Mnumpoi,Mnumpoi) */
#define Cmatrix(J1,J2) Darray(cmatrix,J1,J2,Mnumpoi,Mnumpoi)  /* complex
                                version of Matrix, used with the complex
                                structure */
#define Cvmat(J1,J2) Darray(cvmat,J1,J2,2*Mnumtri,2*Mnumpoi) /* define Cvmat,
                                used to store centroid-vertex contributions */
#define Vvmat(J1,J2) Darray(vvmat,J1,J2,2*Mnumpoi,2*Mnumpoi) /* define Vvmat,
                                used to store vertex-vertex contributions */
struct complex {
        double real;
        double imag;
        };
/*                                                                          */
/*                                                                          */
/*  End of Definitions / Beginning of external variable declarations       */
/*                                                                          */
/*                                                                          */
int flagsym; /* flagsym may assume values of 0,1,2, or 3.  Flagsym is used
                by low level routines to incorporate symmetry in a manner
                invisible to the rest of the program.
                0 - indicates no symmetry
                1 - indicates object possesses mirror symmetry about x=0
                2 - indicates object possesses mirror symmetry about y=0
                3 - indicates object possesses mirror symmetry about x=0 and y=0
                Flagsym is set at the beginning of the main program and after
                that is never changed                                       */
int flagsyms; /* flagsyms is used in conjunction with flagsym to generate
                 mirror images of the source patches while calculating
                 contributions to the field point.  The initial object
                 description is assumed to lie in the first quadrant, however
                 if flagsym is 0, this is not necessary.  Flagsyms is always
                 less than or equal to flagsym, and may assume the values of
                 0,1,2, or 3.  These values are given the following meanings:
```

```
                  0 - source patch is original patch (presumably first quadrant)
                  1 - source patch is in second quadrant
                  2 - source patch is in fourth quadrant
                  3 - source patch is in third quadrant   */
int eof; /* end of file variable (=0 means end of file) */
int potflag; /* flag marks whether or not potentials have been computed
                 0 - potential has not been computed
                 1 - potential has been computed assuming x excitation
                 2 - potential has been computed assuming y excitation
                 3 - potential has been computed assuming z excitation  */
struct {
        double x[Mnumpoi];
        double y[Mnumpoi];
        struct complex res[Mnumpoi];
        struct complex j[2*Mnumpoi];
        struct complex jtotx[Mnumpoi];
        struct complex jtoty[Mnumpoi];
        }point ;    /* point is a structure which contains the locations of all
                 of the points used in defining the triangular patches.  A point
                 number 0 is permitted as C defines arrays beginning with 0.
                 X and y are the coordinates of the points, j is the computed
                 current resulting from a single excitation with x components
                 listed first and y components displaced by Mnumpoi.  jtotx and
                 jtoty contain the x and y components of the total current.  */

struct {
        int numatri[Mnumpoi];
        int Tri(Mnumatri,Mnumpoi-1);
        int Poi1(Mnumatri,Mnumpoi-1);
        int Poi2(Mnumatri,Mnumpoi-1);
        } apoint;  /* apoint is a structure which contains lists of triangles
                 associated with each point.  numatri contains the number of
                 triangles associated with each point; tri contains the list of
                 triangle numbers associated with each point; and poi1 and poi2
                 contain the other two vertices used in defining the triangle.
                 poi1 and poi2 are indices to point.  tri is an index to
                 triang.      */
struct {
        struct {
                double x[Mnumtri];
                double y[Mnumtri];
                } centroid;
        int Poi(Mnumtri,3-1);
        double poten[Mnumtri];
        struct complex cpoten[Mnumtri]; /* complex version of poten */
        } triang;   /*  triang is used to solve the scattering problem with z
                   excitation.  centroid .x and .y contain the coordinates
                   of the centroid of each triangle.  poi contains the list of
                   points (vertices) associated with each triangle, and is an
                   index to point.  poten is the potential of each triangle as
                   determined from solution of the matrix problem.    */
double Matrix(Mnumpoi,Mnumpoi-1);  /* This is "THE"  matrix.  Trivial points
                                    (ie, those which have zero potential from
                                    symmetry considerations) are skipped when
                                    reading or filling the matrix, which is
                                    always done sequentially.    */
```

```
struct complex Cmatrix(Mnumpoi,Mnumpoi-1);  /* complex version of Matrix */
struct complex Cvmat(2*Mnumtri,2*Mnumpoi-1);
struct complex Vvmat(2*Mnumpoi,2*Mnumpoi-1);
struct complex ecvec[2*Mnumtri];
struct complex evvec[2*Mnumpoi];
double fvect[Mnumpoi];    /* fvect is the forcing vector for the matrix problem
                             and after solution of the matrix problem contains
                             the solution vector.   */
struct complex cfvect[Mnumpoi];  /* complex version of fvect */
int mnumpoi;  /* This is the total number of points.  Points are assumed to be
                 numbered sequentially from 0.  mnumpoi <= Mnumpoi. */
int mnumtri;  /* This is the total number of triangles.  Triangles are assumed
                 to be numbered sequentially from 0.  mnumtri <= Mnumtri. */
union {
        char str[2];
        char let;
        } com;
              /* This is the first character of each input line, and is
                 interpreted as a command.  Valid commands are:
                        d - display linkup of points and triangles
                                and display potentials if defined
                        e - specify direction of exciting field,
                                and solve the resultant matrix problem.
                        g - graph potentials of points/triangles
                        h - enter heading, ie, a descriptive one line title.
                        m - enter material parameters of plate: t and tau.
                        p - enter coordinates of next point
                        r - regenerate matrix problem using finer mesh
                        s - define symmetry to be assumed in solving problem
                        t - enter definition of next triangle       */
char hedstr[82];  /*  contains one line heading, description of data */
double dYk;
double dYZ = 376.7; /* impedance of free space */
double t; /* Thickness of the plate */
double tau; /* permittivity of the plate */
double taui;  /* imaginary part of the permittivity.  This variable is also
                used as a flag: if taui is zero, computations are performed
                assuming "tau" is purely real; if taui is non-zero, the routines
                appropriate for a complex tau are invoked  */
double dipmom;  /* contains the real part of the computed dipole moment */
double dipmomi;  /* contains the imaginary part of the computed dipole moment */
double Epsilon = 1e-9; /* A very small number, used for approximate equality */
double Pi = 3.1415926535; /* Pi */
double dYxxr,dYxxi,dYxyr,dYxyi,dYyxr,dYyxi,dYyyr,dYyyi; /* used in interface */
double theta, phi, alpha;  /* specifies the direction of propagation (theta
                                and phi), and the polarization of the electric
                                vector with alpha.  alpha is specified in the
                                same manner as phi.  */
double rtheta, rphi, ralpha; /* contains angles in radians */
/***************************************************************************/
/*                                                                         */
/*  M A I N     P R O G R A M                                              */
/*                                                                         */
/*  This is a program which calculates scattering by an arbitrarily shaped, */
/*  thin, flat, resistive plate.  Excitation may be specified in the x, y,  */
/*  or z directions.  The plate is made up of an arbitrary number (nominally */
```

```
/*  less than 50) of triangular patches, upon which a method of moments       */
/*  solution is obtained.  Contributions from each patch is calculated via    */
/*  surface integrals which are evaluated analytically, thus contributions    */
/*  from each patch is obtained exactly (at least to 10 plus digits).  The    */
/*  only approximations which are made are that the potential varies linearly */
/*  with z inside the plate, and the division of the plate into triangular    */
/*  patches, inside of which the field varies linearly with x and y.  The     */
/*  shape and size of the triangular patches are completely arbitrary, and it */
/*  is noted that the patches need not be contiguous.                         */
/*                                                                            */
/******************************************************************************/
main() {
hedstr[0] = '\0';
for(eof=scanf("%1s",com.str);eof == 1;) {
        switch (com.let) {
                case 'P':
                case 'p':
                case 'T':
                case 't':
                        rdata();  /* read in the data */
                        continue;
                case 'R':
                case 'r':
                        rres();   /* read in the resistivity of each point */
                        continue;
                case 'H':
                case 'h':
                        gethed();
                        break;
                case 'E':
                case 'e':
                        suplup(); /* read in direction of excitation
                                     and solve the resulting matrix problem */
                default:
                        printf("%c is an invalid command\n",com.let);
                        while(scanf("%c",com.str),com.let != '\n');
                                  /* flush out bad command */
                        break;
                }
        eof=scanf("%1s",com.str);
        }
}
/******************************************************************************/
/*                                                                            */
/*  ROUTINE:   darea(i)                                                       */
/*                                                                            */
/*  darea is called by suplup to calculate the area of a triangle.  darea     */
/*  accomplishes this with a call to area.  The argument i specifies the      */
/*  triangle number.                                                          */
/*                                                                            */
/******************************************************************************/
double darea(i)
int i;  /* pointer to triangle */
{
return(area(point.x[triang.Poi(i,0)],point.y[triang.Poi(i,0)],
        point.x[triang.Poi(i,1)],point.y[triang.Poi(i,1)],
```

```
        point.x[triang.Poi(i,2)],point.y[triang.Poi(i,2)]));
}
/**************************************************************************/
/*                                                                        */
/*  ROUTINE:   suplup                                                     */
/*                                                                        */
/*  suplup is called by the main program to read in excitation, decompose the */
/*  problem into several symmetric problems if appropriate and possible, and  */
/*  compute the current induced on the resistive plate.                   */
/*                                                                        */
/**************************************************************************/
suplup() {
int i, j, i2;   /* loop variables */
scanf ("%le %le %le %le",&dYk,&theta,&phi,&alpha);
rtheta=theta*Pi/180.;
rphi=phi*Pi/180.;
ralpha=alpha*Pi/180.;
        for (i=0;i<mnumtri;i++) {
                ecvec[i].real = cos(dYk*(triang.centroid.x[i]*sin(rtheta)*
                        cos(rphi)+triang.centroid.y[i]*sin(rtheta)*sin(rphi)))*
                     sqrt(sin(rtheta)*sin(rtheta)*sin(rphi)*sin(rphi)+cos(rtheta)*
                        cos(rtheta))*
                     cos(ralpha);
                ecvec[i].imag = sin(dYk*(triang.centroid.x[i]*sin(rtheta)*
                        cos(rphi)+triang.centroid.y[i]*sin(rtheta)*sin(rphi)))*
                     sqrt(sin(rtheta)*sin(rtheta)*sin(rphi)*sin(rphi)+cos(rtheta)*
                        cos(rtheta))*
                     cos(ralpha);
                ecvec[i+mnumtri].real =cos(dYk*(triang.centroid.x[i]*sin(rtheta)*
                        cos(rphi)+triang.centroid.y[i]*sin(rtheta)*sin(rphi)))*
                     sqrt(sin(rtheta)*sin(rtheta)*cos(rphi)*cos(rphi)+cos(rtheta)*
                        cos(rtheta))*
                     sin(ralpha);
                ecvec[i+mnumtri].imag =sin(dYk*(triang.centroid.x[i]*sin(rtheta)*
                        cos(rphi)+triang.centroid.y[i]*sin(rtheta)*sin(rphi)))*
                     sqrt(sin(rtheta)*sin(rtheta)*cos(rphi)*cos(rphi)+cos(rtheta)*
                        cos(rtheta))*
                     sin(ralpha);
                for (i2=0;i2<mnumpoi;i2++) {    /* fill Vcmat */
                     cpcon(i,i2);
                     }
                }
        cv2vv();
        solvem(2*mnumpoi);  /* using Vvmat, evvec, and point.j */
        for (i=0;i<mnumpoi;i++) {
                point.jtotx[i].real=point.j[i].real;
                point.jtotx[i].imag=point.j[i].imag;
                point.jtoty[i].real=point.j[i+mnumpoi].real;
                point.jtoty[i].imag=point.j[i+mnumpoi].imag;
                }
        printy();
        }
/**************************************************************************/
/*                                                                        */
/*  ROUTINE:   cv2vv                                                      */
/*                                                                        */
```

```
/*  cv2vv is called by suplup to convert from the centroid vertex matrix to   */
/*  the vertex vertex matrix.
/*
/*****************************************************************************/
cv2vv(){
int i,j,j2;  /*loop variables*/
double sarea,tarea; /* stores area of triangles*/
for (j=0;j<mnumpoi;j++) {   /* convert to point-point */
      for (j2=0,sarea=0,evvec[j].real=evvec[j].imag=
            evvec[j+mnumpoi].real= evvec[j+mnumpoi].imag=0;
            j2<apoint.numatri[j];j2++) {
            tarea=darea(apoint.Tri(j2,j));
            sarea += tarea;
            evvec[j].real += tarea*ecvec[apoint.Tri(j2,j)].real;
            evvec[j].imag += tarea*ecvec[apoint.Tri(j2,j)].imag;
            evvec[j+mnumpoi].real += tarea*
            ecvec[apoint.Tri(j2,j)+mnumtri].real;
            evvec[j+mnumpoi].imag += tarea*
            ecvec[apoint.Tri(j2,j)+mnumtri].imag;
            }
      evvec[j].real /= sarea;
      evvec[j].imag /= sarea;
      evvec[j+mnumpoi].real /= sarea;
      evvec[j+mnumpoi].imag /= sarea;
      for (i=0;i<mnumpoi;i++) {
            for (j2=0,sarea=0,Vvmat(i,j).real=Vvmat(i,j).imag=
                        Vvmat(i,j+mnumpoi).real=Vvmat(i,j+mnumpoi).imag=
                        Vvmat(i+mnumpoi,j).real=Vvmat(i+mnumpoi,j).imag=
                        Vvmat(i+mnumpoi,j+mnumpoi).real=
                        Vvmat(i+mnumpoi,j+mnumpoi).imag=0;
                        j2<apoint.numatri[i];j2++) {
                  tarea=darea(apoint.Tri(j2,i));
                  sarea += tarea;
                  Vvmat(i,j).real += tarea*Cvmat(apoint.Tri(j2,i),j).real;
                  Vvmat(i,j).imag += tarea*Cvmat(apoint.Tri(j2,i),j).imag;
                  Vvmat(i,j+mnumpoi).real += tarea*
                        Cvmat(apoint.Tri(j2,i),j+mnumpoi).real;
                  Vvmat(j+mnumpoi,i).imag += tarea*
                        Cvmat(apoint.Tri(j2,i),j+mnumpoi).imag;
                  Vvmat(i+mnumpoi,j).real += tarea*
                        Cvmat(apoint.Tri(j2,i)+mnumtri,j).real;
                  Vvmat(i+mnumpoi,j).imag += tarea*
                        Cvmat(apoint.Tri(j2,i)+mnumtri,j).imag;
                  Vvmat(i+mnumpoi,j+mnumpoi).real += tarea*
                        Cvmat(apoint.Tri(j2,i)+mnumtri,j+mnumpoi).real;
                  Vvmat(i+mnumpoi,j+mnumpoi).imag += tarea*
                        Cvmat(apoint.Tri(j2,i)+mnumtri,j+mnumpoi).imag;
                  }
            Vvmat(i,j).real /= sarea;
            Vvmat(i,j).imag /= sarea;
            Vvmat(i,j+mnumpoi).real /= sarea;
            Vvmat(i,j+mnumpoi).imag /= sarea;
            Vvmat(i+mnumpoi,j).real /= sarea;
            Vvmat(i+mnumpoi,j).imag /= sarea;
            Vvmat(i+mnumpoi,j+mnumpoi).real /= sarea;
            Vvmat(i+mnumpoi,j+mnumpoi).imag /= sarea;
```

```
            }
        }
}
/*****************************************************************************/
/*                                                                          */
/*  ROUTINE:   cpcon(i,i2)                                                  */
/*                                                                          */
/*  cpcon calculates the elements of the matrix in terms of the contribution */
/*  of a source point to a centroid.  The source point contribution is broken */
/*  up into triangle contributions, which are analyzed by calling contrib.   */
/*  cpcon is the only routine which calls contrib, and thus is the          */
/*  interface between the control program, written by Dave Ksienski, and the */
/*  matrix element numerical evaluation routines, written by Joe Burns.  cpcon*/
/*  also calculates the contribution to the self cell from the RJ term.     */
/*                                                                          */
/*****************************************************************************/
cpcon(i,i2)
int i,i2;        /* field point i, source point i2 */
{
struct complex rsum;    /* holds values of resistance at centroid */
int iloop;       /* loop variable */
for (Cvmat(i,i2).real=Cvmat(i,i2).imag=Cvmat(i,i2+mnumpoi).real=
            Cvmat(i,i2+mnumpoi).imag=Cvmat(i+mnumtri,i2).real=
            Cvmat(i+mnumtri,i2).imag=Cvmat(i+mnumtri,i2+mnumpoi).real=
            Cvmat(i+mnumtri,i2+mnumpoi).imag=0,
            iloop=0;iloop<apoint.numatri[i2];iloop++) {
        contrib(triang.centroid.x[i],triang.centroid.y[i],point.x[i2],
                point.y[i2],point.x[apoint.Poi1(iloop,i2)],
                point.y[apoint.Poi1(iloop,i2)],point.x[apoint.Poi2(iloop,i2)],
                point.y[apoint.Poi2(iloop,i2)]);
/* >>                                                                   << */
/* >>  contrib returns values through the external variables dYabc, where << */
/* >>  a=x or y and represents the direction of the field               << */
/* >>  b=x or y and represents the direction of the current             << */
/* >>  c=r or i and represents either the real or imaginary part        << */
/* >>                                                                   << */
        Cvmat(i,i2).real+=dYxxr;
        Cvmat(i,i2).imag+=dYxxi;
        Cvmat(i,i2+mnumpoi).real+=dYxyr;
        Cvmat(i,i2+mnumpoi).imag+=dYxyi;
        Cvmat(i+mnumtri,i2).real+=dYyxr;
        Cvmat(i+mnumtri,i2).imag+=dYyxi;
        Cvmat(i+mnumtri,i2+mnumpoi).real+=dYyyr;
        Cvmat(i+mnumtri,i2+mnumpoi).imag+=dYyyi;
        if(apoint.Tri(iloop,i2) == i) { /* self cell contribution */
                rsum.real=(point.res[i2].real+
                   point.res[apoint.Poi1(iloop,i2)].real+
                   point.res[apoint.Poi2(iloop,i2)].real)/3;
                rsum.imag=(point.res[i2].imag+
                   point.res[apoint.Poi1(iloop,i2)].imag+
                   point.res[apoint.Poi2(iloop,i2)].imag)/3;
                Cvmat(i,i2).real+=rsum.real;
                Cvmat(i,i2).imag+=rsum.imag;
                Cvmat(i+mnumtri,i2+mnumpoi).real+=rsum.real;
                Cvmat(i+mnumtri,i2+mnumpoi).imag+=rsum.imag;
                }
```

```
        }
}
/****************************************************************************/
/*                                                                          */
/*  ROUTINE:   rres                                                         */
/*                                                                          */
/*  rres is called by the main program to read in the resistivity associated */
/*  with each point.  The resistivity is assumed to be complex and linearly  */
/*  varying.  Upon encountering a non-"r" command rres retruns to the main   */
/*  program.                                                                */
/*                                                                          */
/****************************************************************************/
rres() {
int i,j;                /* loop variables */
int inumb;      /* next point */
do {
        switch (com.let) {
                case 'R':
                case 'r':
                        scanf("%d",&inumb);
                        scanf("%le %le",&point.res[inumb].real,
                           &point.res[inumb].imag);
                        continue;
                default:
                        return;
                }
        }
        while (scanf("%1s",com.str) == 1);
}
/****************************************************************************/
/*                                                                          */
/*  ROUTINE:   ddata                                                        */
/*                                                                          */
/*  ddata is called by the main program to display data.  ddata display     */
/*  locations of points, connections of points, and potentials of points if */
/*  they have been computed.  ddata also displays status of various flags   */
/*  which indicate symmetries of plate and direction of excitation.         */
/*                                                                          */
/****************************************************************************/
printy() {
int i,j; /* loop variables */
printf("%s \n",hedstr);
printf("k=%lf, theta=%lf, phi=%lf, alpha=%lf\n",dYk,theta,phi,alpha);
for (i=0; i<mnumpoi; i++) {
        printf("point # %d is located at x = %lf, y = %lf\n",i,point.x[i],
          point.y[i]);
                printf("      and has current Jx=%lf +i%lf, Jy=%lf +i%lf\n",
                        point.jtotx[i].real,point.jtotx[i].imag,point.jtoty[i].real,
                        point.jtoty[i].imag);
                printf("      and has resistivity R=%lf +i%lf\n",
                        point.res[i].real,point.res[i].imag);
        printf("point %d is associated with points and triangles (tri,P1,P2)\n"
          ,i);
        for (j=0; j<apoint.numatri[i];j++) {
                printf("(%d,%d,%d),  ",apoint.Tri(j,i), apoint.Poi1(j,i),
                  apoint.Poi2(j,i));
```

```
                if (j%6 == 5 || j == apoint.numatri[i]-1) printf("\n");
                }
        }
}
/*************************************************************************/
/*                                                                       */
/*  ROUTINE:   rdata                                                     */
/*                                                                       */
/*  rdata is called by the main program to read in point and triangle   */
/*  definitions.  Upon encountering a non- "p" or "t" command, rdata calls */
/*  ldata to link the data, and then returns to the main program.        */
/*                                                                       */
/*************************************************************************/
rdata() {
int i; /* loop variable */
int inumb; /* number of next point or triangle */
do {
        switch (com.let) {
                case 'P':
                case 'p':
                        scanf("%d",&inumb);
                        scanf("%le %le",point.x+inumb,point.y+inumb);
                        mnumpoi=inumb+1;
                        continue;
                case 'T':
                case 't':
                        scanf("%d",&inumb);
                        for (i=0;i<3;i++) scanf("%d",&triang.Poi(inumb,i));
                        mnumtri=inumb+1;
                        continue;
                default:
                        ldata();
                        return;
                }
        }
        while (scanf("%1s",com.str) == 1);
}
/*************************************************************************/
/*                                                                       */
/*  ROUTINE:   ldata                                                     */
/*                                                                       */
/*  ldata links all information concerning points and triangles.  This  */
/*  information is needed to precisely define each patch on the plate.   */
/*                                                                       */
/*************************************************************************/
ldata() {
int i,j; /* loop variables */
int isi; /* scratch variable */
for (i=0;i<mnumpoi;i++) apoint.numatri[i]=0;
for (i=0;i<mnumtri;i++) {
        for (triang.centroid.x[i]=triang.centroid.y[i]=0,j=0;j<3;j++) {
                isi=triang.Poi(i,j);
                triang.centroid.x[i] += point.x[isi]/3;
                triang.centroid.y[i] += point.y[isi]/3;
                apoint.Tri(apoint.numatri[isi],isi) = i;
                apoint.Poii(apoint.numatri[isi],isi) = triang.Poi(i,(j+1)%3);
```

```
                apoint.Poi2(apoint.numatri[is1]++,is1) = triang.Poi(i,(j+2)%3);
                }
        }
return;
}
/*****************************************************************************/
/*                                                                           */
/*  ROUTINE:   solvem(N)                                                     */
/*                                                                           */
/*  solvem is used to solve the matrix problem.  The matrix and forcing vector*/
/*  are created in xsolv or ysolv.  N is the dimension of the vector.  solvem */
/*  solves the matrix problem by calling the two fortran routines dgeco and   */
/*  dgesl which preform a decomposition and back substitution.  solvem is the */
/*  only c routine which calls a fortran routine.                             */
/*                                                                           */
/*****************************************************************************/
solvem(N)
int N; /* N is the order of the matrix stored in the array matrix.  N is less
            than or equal to isize */
{
int i,j;
int isize,job;  /* arguments for fortran routines, isize is the size of the
                array matrix, job (=0) indicates type of matrix problem */
int ipvt[2*Mnumpoi];  /* an array used by the fortran routines to store the
                        pivoting vector   */
double rcond; /* the condition number of the matrix. */
double z[2*Mnumpoi*2]; /* scratch vector, lengthened for complex case */
isize=2*Mnumpoi;
job=0;
for(i=0;i<13;i++){
        for(j=0;j<13;j++){
                printf("%lf %lf\n",Vvmat(i,j));
                }
        }
for(i=0;i<N+1;i++) {
        point.j[i].real=evvec[i].real;
        point.j[i].imag=evvec[i].imag;
        }
cgeco_(vvmat,&isize,&N,ipvt,&rcond,z);
printf("condition number is %e\n",1/rcond);
cgesl_(vvmat,&isize,&N,ipvt,point.j,&job);
}
/*****************************************************************************/
/*                                                                           */
/*  ROUTINE:   gethed                                                        */
/*                                                                           */
/*  gethed is called by the main program to get the heading of the data.     */
/*  gethed fills hedstr with the remainder of the line (the entire line      */
/*  except for the letter H which must be in column 1.                       */
/*                                                                           */
/*****************************************************************************/
gethed() {
int i;  /* loop variable */
for (i=0; (hedstr[i]=getchar()) != '\n'; i++);
hedstr[i] = '\0';
return;
```

```
}
/***************************************************************************/
/*                                                                         */
/*  ROUTINE: area                                                          */
/*                                                                         */
/*  area is called by graphp to calculate the area of a triangle.  This is */
/*  needed in the computation of the area coordinates.                     */
/*                                                                         */
/***************************************************************************/
double area(x1,y1,x2,y2,x3,y3)
double x1,y1,x2,y2,x3,y3;       /*  These are the coordinates of the three
                                    points which delimit the triangle  */
{
return(fabs(x2*y3-y2*x3-x1*y3+y1*x3+x1*y2-y1*x2)/2);
}
#include <math.h>
double dYk,dYZ;
double Rk_re,Rk_im,Rkx_re,Rkx_im,Rky_re,Rky_im;
double Dx2rk_re,Dx2rk_im,Dx2rkx_re,Dx2rkx_im,Dx2rky_re,Dx2rky_im;
double Dy2rk_re,Dy2rk_im,Dy2rkx_re,Dy2rkx_im,Dy2rky_re,Dy2rky_im;
double Dxyrk_re,Dxyrk_im,Dxyrkx_re,Dxyrkx_im,Dxyrky_re,Dxyrky_im;
double IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9;
double DX2IT1,DY2IT1,DXYIT1;
double DX2IT2,DY2IT2,DXYIT2;
double DX2IT3,DY2IT3,DXYIT3;
double DX2IT7,DY2IT7,DXYIT7;
double DX2IT8,DY2IT8,DXYIT8;
double DX2IT9,DY2IT9,DXYIT9;
extern double dYxxr,dYxxi,dYxyr,dYxyi,dYyxr,dYyxi,dYyyr,dYyyi;


contrib(xo,yo,xs1,ys1,xs2,ys2,xs3,ys3)


double xo,yo,xs1,ys1,xs2,ys2,xs3,ys3;


{
/************************************************************************/
/*                                                                      */
/*  This routine calculates the finite element contribution to the x    */
/*  component of the current or the y component for x or y excitation   */
/*  given the vertices of the source triangle xs1,ys1 xs2,ys2 xs3,ys3 and*/
/*  the observation point. The finite element approximating the current */
/*  varies linearly over the triangular subdomain and is assumed to be  */
/*  one at xs1,ys1 and zero at xs2,ys2 and xs3,ys3.                     */
/*                                                                      */
/************************************************************************/
/*                                                                      */
/*  xo  x coordinate of the observation point                           */
/*  yo  y coordinate of the observation point                           */
/*  xs1 x coordinate of vertice 1 of the source triangle                */
/*  ys1 y coordinate of vertice 1 of the source triangle                */
/*  xs2 x coordinate of vertice 2 of the source triangle                */
/*  ys2 y coordinate of vertice 2 of the source triangle                */
/*  xs3 x coordinate of vertice 3 of the source triangle                */
```

```
/*  ys3 y coordinate of vertice 3 of the source triangle                    */
/*  a  x coefficient of the finite element                                  */
/*  b  y coefficient of the finite element                                  */
/*  c  constant coefficient of the finite element                           */
/*                                                                          */
/****************************************************************************/
/*                                                                          */
/*  EXTERNAL VARIABLES                                                      */
/*                                                                          */
/****************************************************************************/
/*  dYxxr real part of the x component of the current for x excitation     */
/*  dYxxi imag part of the x component of the current for x excitation     */
/*  dYxyr real part of the y component of the current for x excitation     */
/*  dYxyi imag part of the y component of the current for x excitation     */
/*  dYyxr real part of the x component of the current for y excitation     */
/*  dYyxi imag part of the x component of the current for y excitation     */
/*  dYyyr real part of the y component of the current for y excitation     */
/*  dYyyi imag part of the y component of the current for y excitation     */
/*                                                                          */
/*    IT1 is the integral of 1/R over the triangle                          */
/*    IT2 is the integral of x/R over the triangle                          */
/*    IT3 is the integral of y/R over the triangle                          */
/*    IT4 is the integral of 1. over the triangle                           */
/*    IT5 is the integral of x over the triangle                            */
/*    IT6 is the integral of y over the triangle                            */
/*    IT7 is the integral of R over the triangle                            */
/*    IT8 is the integral of xR over thr triangle                           */
/*    IT9 is the integral of yR over the triangle                           */
/*    DX2IT1 is second derivative of IT1 with respect to x                  */
/*    DY2IT1 is second derivative of IT1 with respect to y                  */
/*    DXYIT1 is derivative of IT1 with respect to x and y                   */
/*    DX2IT2 is second derivative of IT2 with respect to x                  */
/*    DY2IT2 is second derivative of IT2 with respect to y                  */
/*    DXYIT2 is derivative of IT2 with respect to x and y                   */
/*    DX2IT3 is second derivative of IT3 with respect to x                  */
/*    DY2IT3 is second derivative of IT3 with respect to y                  */
/*    DXYIT3 is derivative of IT3 with respect to x and y                   */
/*    DX2IT7 is second derivative of IT7 with respect to x                  */
/*    DY2IT7 is second derivative of IT7 with respect to y                  */
/*    DXYIT7 is derivative of IT7 with respect to x and y                   */
/*    DX2IT8 is second derivative of IT8 with respect to x                  */
/*    DY2IT8 is second derivative of IT8 with respect to y                  */
/*    DXYIT8 is derivative of IT8 with respect to x and y                   */
/*    DX2IT9 is second derivative of IT9 with respect to x                  */
/*    DY2IT9 is second derivative of IT9 with respect to y                  */
/*    DXYIT9 is derivative of IT9 with respect to x and y                   */
/*    Rk_re real part of the rational kernel                                */
/*    Rk_im imag part of the rational kernel                                */
/*    Rkx_re real part of x times the rational kernel                       */
/*    Rkx_im imag part of x times the rational kernal                       */
/*    Rky_re real part of y times the rational kernal                       */
/*    Rky_im imag part of y times the rational kernel                       */
/*    Dx2rk_re real part of second derivative with respect to x of          */
/*            the rational kernel                                           */
/*    Dx2rk_im imag part of second derivative with respect to x of          */
/*            the rational kernel                                           */
```

```
/*      Dx2rkx_re real part of x times second derivative with respect to x */
/*               of the rational kernel                                    */
/*      Dx2rkx_im imag part of x times second derivative with respect to x */
/*               of the rational kernel                                    */
/*      Dx2rky_re real part of y times second derivative with respect to x */
/*               of the rational kernel                                    */
/*      Dx2rky_im imag part of y times second derivative with respect to x */
/*               of the rational kernel                                    */
/*      Dy2rk_re real part of second derivative with respect to y of       */
/*              the rational kernel                                        */
/*      Dy2rk_im imag part of second derivative with respect to y of       */
/*              the rational kernel                                        */
/*      Dy2rkx_re real part of x times second derivative with respect to y */
/*               of the rational kernel                                    */
/*      Dy2rkx_im imag part of x times second derivative with respect to y */
/*               of the rational kernel                                    */
/*      Dy2rky_re real part of y times second derivative with respect to y */
/*               of the rational kernel                                    */
/*      Dy2rky_im imag part of y times second derivative with respect to y */
/*               of the rational kernel                                    */
/*      Dxyrk_re real part of derivative with respect to x and y of        */
/*              the rational kernel                                        */
/*      Dxyrk_im imag part of derivative with respect to x and y of        */
/*              the rational kernel                                        */
/*      Dxyrkx_re real part of x times derivative with respect to x and y  */
/*               of the rational kernel                                    */
/*      Dxyrkx_im imag part of x times derivative with respect to x and y  */
/*               of the rational kernel                                    */
/*      Dxyrky_re real part of y times derivative with respect to x and y  */
/*               of the rational kernel                                    */
/*      Dxyrky_im imag part of y times derivative with respect to x and y  */
/*               of the rational kernel                                    */
/*      dYk wavenumber                                                     */
/*      dYZ impedance                                                      */
/*                                                                         */
/*************************************************************************/

double a,b,c,k,k2,PI;
double ikrnl_re,ikrnl_im,ixyk_re,ixyk_im,ix2k_re,ix2k_im,iy2k_re,iy2k_im;

k=dYk;
k2=k*k/2.;
PI=3.1415926535;

/*************************************************************************/
/*                                                                       */
/*      Call routines numer and analyt to calculate external variables   */
/*                                                                       */
/*************************************************************************/

analyt(xo,yo,xs1,ys1,xs2,ys2,xs3,ys3);

numer(xo,yo,xs1,ys1,xs2,ys2,xs3,ys3);

/*************************************************************************/
/*                                                                       */
```

```
/*    Calculate coefficients of finite element                          */
/*                                                                       */
/************************************************************************/

a=(ys2-ys3)/((xs1-xs2)*(ys2-ys3)-(xs2-xs3)*(ys1-ys2));

b=(xs3-xs2)/((xs1-xs2)*(ys2-ys3)-(xs2-xs3)*(ys1-ys2));

c=((xs2-xs3)*ys2-(ys2-ys3)*xs2)/((xs1-xs2)*(ys2-ys3)-(xs2-xs3)*(ys1-ys2));

/************************************************************************/
/*                                                                       */
/*    Calculate integrals of current over the triangle                  */
/*                                                                       */
/************************************************************************/


ikrnl_re=a*(IT2-k2*IT8)+b*(IT3-k2*IT9)+c*(IT1-k2*IT7)+a*Rkx_re+b*Rky_re
       +c*Rk_re;

ikrnl_im=k*(a*IT5+b*IT6+c*IT4)+a*Rkx_im+b*Rky_im+c*Rk_im;

ixyk_re=a*(DXYIT2-k2*DXYIT8)+b*(DXYIT3-k2*DXYIT9)+c*(DXYIT1-k2*DXYIT7)
      +a*Dxyrkx_re+b*Dxyrky_re+c*Dxyrk_re;

ixyk_im=a*Dxyrkx_im+b*Dxyrky_im+c*Dxyrk_im;

ix2k_re=a*(DX2IT2-k2*DX2IT8)+b*(DX2IT3-k2*DX2IT9)+c*(DX2IT1-k2*DX2IT7)
      +a*Dx2rkx_re+b*Dx2rky_re+c*Dx2rk_re;

ix2k_im=a*Dx2rkx_im+b*Dx2rky_im+c*Dx2rk_im;

iy2k_re=a*(DY2IT2-k2*DY2IT8)+b*(DY2IT3-k2*DY2IT9)+c*(DY2IT1-k2*DY2IT7)
      +a*Dy2rkx_re+b*Dy2rky_re+c*Dy2rk_re;

iy2k_im=a*Dy2rkx_im+b*Dy2rky_im+c*Dy2rk_im;

/************************************************************************/
/*                                                                       */
/*    Calculate contributions of the finite element currents           */
/*                                                                       */
/************************************************************************/

dYxxr=((-k*dYZ*ikrnl_im)+(-dYZ*ix2k_im/k))/(4.*PI);

dYxxi=((k*dYZ*ikrnl_re)+(dYZ*ix2k_re/k))/(4.*PI);

dYxyr=(-dYZ*ixyk_im/k)/(4.*PI);

dYxyi=(dYZ*ixyk_re/k)/(4.*PI);

dYyyr=((-k*dYZ*ikrnl_im)+(-dYZ*iy2k_im/k))/(4.*PI);

dYyyi=((k*dYZ*ikrnl_re)+(dYZ*iy2k_re/k))/(4.*PI);

dYyxr=dYxyr;
```

```
dYyxi=dYxyi;

}



analyt(xo,yo,xs1,ys1,xs2,ys2,xs3,ys3)

double xo,yo,xs1,ys1,xs2,ys2,xs3,ys3;

{
/*************************************************************************/
/*                                                                       */
/*  Calculate the contribution of a particular triangle given its vertices */
/*  x[1],y[1]  x[2],y[2]  x[3],y[3]  and the field point  xo,yo          */
/*                                                                       */
/*  List of Variables:                                                   */
/*                                                                       */
/*    x[n]    x coordinates of the vertices of the triangle              */
/*    y[n]    y coordinates of the vertices of the triangle              */
/*    xo      x coordinate of the field point                            */
/*    yo      y coordinate of the field point                            */
/*    ai[n]   slopes of lines through segments                           */
/*    alpi[n]  1./sqrt(1.+ai**2)                                         */
/*    gi[n]   ai[n]*qi[n]                                                */
/*    bi[n]   y-intercept of lines through segments                      */
/*    li[n]   length of ith segment                                      */
/*    ci[n]   constant related to the parameterization                   */
/*    exi[n]  indicates sense of x direction along integration path      */
/*    eyi[n]  indicate  sense of y direction along integration path      */
/*    Dxi[n]  derivative of x' with respect to path length parameter t   */
/*    Dyi[n]  derivative of y' with respect to path length parameter t   */
/*    A[n]    term of R                                                  */
/*    B[n]    term of R                                                  */
/*    C[n]    term of R                                                  */
/*    C2[n]   sqrt(C[n])                                                 */
/*    Rip[n]  distance from observation point to plus end of ith segment */
/*    Rim[n]  distance from observation point to minus end of ith segment */
/*    tip[n]  path length to plus end of ith segment                     */
/*    tim[m]  path length to minus end of ith segment                    */
/*    I1i[n]  integral of 1./R over a segment                            */
/*    I2i[n]  integral of t/R over a segment                             */
/*    I3i[n]  integral of t**2/R over a segment                          */
/*    I4i[n]  integral of R over a segment                               */
/*    I5i[n]  integral of tR over a segment                              */
/*    I6i[n]  integral of (t**2)R over a segment                         */
/*    sign    temporary variable                                         */
/*    t       temporary variable                                         */
/*    R       temporary variable                                         */
/*    xt[n]   temporary variable                                         */
/*    yt[n]   temporary variable                                         */
/*    ntrmx   temporary variable                                         */
/*    ntrmy   temporary variable                                         */
/*    dtrm    tempoaray variable                                         */
/*    delta[n]  (4AC-BB)/(8C)                                            */
```

```
/*     DXI1i is first derivative of I1i with respect to x                  */
/*     DYI1i is first derivative of I1i with respect to y                  */
/*     DX2I1i is second derivative of I1i with respect to x                */
/*     DY2I1i is second derivative of I1i with respect to y                */
/*     DXYI1i is derivative of I1i with respect to x and y                 */
/*     DXI2i is first derivative of I2i with respect to x                  */
/*     DYI2i is first derivative of I2i with respect to y                  */
/*     DX2I2i is second derivative of I2i with respect to x                */
/*     DY2I2i is second derivative of I2i with respect to y                */
/*     DXYI2i is derivative of I2i with respect to x and y                 */
/*     DXI3i is first derivative of I3i with respect to x                  */
/*     DYI3i is first derivative of I3i with respect to y                  */
/*     DX2I3i is second derivative of I3i with respect to x                */
/*     DY2I3i is second derivative of I3i with respect to y                */
/*     DXYI3i is derivative of I3i with respect to x and y                 */
/*     DXI4i is first derivative of I4i with respect to x                  */
/*     DYI4i is first derivative of I4i with respect to y                  */
/*     DX2I4i is second derivative of I4i with respect to x                */
/*     DY2I4i is second derivative of I4i with respect to y                */
/*     DXYI4i is derivative of I4i with respect to x and y                 */
/*     DXI5i is first derivative of I5i with respect to x                  */
/*     DYI5i is first derivative of I5i with respect to y                  */
/*     DX2I5i is second derivative of I5i with respect to x                */
/*     DY2I5i is second derivative of I5i with respect to y                */
/*     DXYI5i is derivative of I5i with respect to x and y                 */
/*     DXI6i is first derivative of I6i with respect to x                  */
/*     DYI6i is first derivative of I6i with respect to y                  */
/*     DX2I6i is second derivative of I6i with respect to x                */
/*     DY2I6i is second derivative of I6i with respect to y                */
/*     DXYI6i is derivative of I6i with respect to x and y                 */
/*     DXIT1 is first derivative of IT1 with respect to x                  */
/*     DYIT1 is first derivative of IT1 with respect to y                  */
/*     DXIT7 is first derivative of IT7 with respect to x                  */
/*     DYIT7 is first derivative of IT7 with respect to y                  */
/*     temp  temporary variable                                            */
/***************************************************************************/
/*                                                                         */
/*     EXTERNAL VARIABLES                                                  */
/*                                                                         */
/***************************************************************************/
/*     IT1 is the integral of 1/R over the triangle                        */
/*     IT2 is the integral of x/R over the triangle                        */
/*     IT3 is the integral of y/R over the triangle                        */
/*     IT4 is the integral of 1. over the triangle                         */
/*     IT5 is the integral of x over the triangle                          */
/*     IT6 is the integral of y over the triangle                          */
/*     IT7 is the integral of R over the triangle                          */
/*     IT8 is the integral of xR over thr triangle                         */
/*     IT9 is the integral of yR over the triangle                         */
/*     DX2IT1 is second derivative of IT1 with respect to x                */
/*     DY2IT1 is second derivative of IT1 with respect to y                */
/*     DXYIT1 is derivative of IT1 with respect to x and y                 */
/*     DX2IT2 is second derivative of IT2 with respect to x                */
/*     DY2IT2 is second derivative of IT2 with respect to y                */
/*     DXYIT2 is derivative of IT2 with respect to x and y                 */
/*     DX2IT3 is second derivative of IT3 with respect to x                */
```

```
/*     DY2IT3 is second derivative of IT3 with respect to y              */
/*     DXYIT3 is derivative of IT3 with respect to x and y               */
/*     DX2IT7 is second derivative of IT7 with respect to x              */
/*     DY2IT7 is second derivative of IT7 with respect to y              */
/*     DXYIT7 is derivative of IT7 with respect to x and y               */
/*     DX2IT8 is second derivative of IT8 with respect to x              */
/*     DY2IT8 is second derivative of IT8 with respect to y              */
/*     DXYIT8 is derivative of IT8 with respect to x and y               */
/*     DX2IT9 is second derivative of IT9 with respect to x              */
/*     DY2IT9 is second derivative of IT9 with respect to y              */
/*     DXYIT9 is derivative of IT9 with respect to x and y               */
/*************************************************************************/

   double ai[4],alpi[4],gi[4],bi[4],li[4],ci[4],exi[4],eyi[4],Dxi[4],Dyi[4];
   double Rim[4],tip[4],tim[4],x[5],y[5],Rip[4],A[4],B[4],C[4],C2[4];
   double I1i[4],I2i[4],I3i[4],I4i[4],I5i[4],I6i[4],di[4];
   double DXI1i[4],DYI1i[4],DX2I1i[4],DY2I1i[4],DXYI1i[4];
   double DXI2i[4],DYI2i[4],DX2I2i[4],DY2I2i[4],DXYI2i[4];
   double DXI3i[4],DYI3i[4],DX2I3i[4],DY2I3i[4],DXYI3i[4];
   double DXI4i[4],DYI4i[4],DX2I4i[4],DY2I4i[4],DXYI4i[4];
   double DXI5i[4],DYI5i[4],DX2I5i[4],DY2I5i[4],DXYI5i[4];
   double DXI6i[4],DYI6i[4],DX2I6i[4],DY2I6i[4],DXYI6i[4];
   double DXIT1,DYIT1,DXIT7,DYIT7;
   double sign,t[3],R[3],xt[3],yt[3],ntrmx,ntrmy,dtrm,delta[4],temp;
   double sqrt(),log(),fabs();
   int n,m;

/*************************************************************************/
/*                                                                       */
/*   Calculate ai[n]. If segment is parallel to the y-axis, ai[n] would go */
/*   infinity; consequently, set ai[n] in this case to zero. This will not */
/*   affect any calculations since this feature of ai[n] has been accounted */
/*   for.                                                                */
/*                                                                       */
/*************************************************************************/

   x[1]=xs1;
   y[1]=ys1;
   x[2]=xs2;
   y[2]=ys2;
   x[3]=xs3;
   y[3]=ys3;
   x[4]=x[1];
   y[4]=y[1];

   for(n=1;n<=3;n++)
      {
      if(x[n+1] == x[n])
         ai[n]=0.;
      else
         ai[n]=(y[n+1]-y[n])/(x[n+1]-x[n]);
      }

/*************************************************************************/
/*                                                                       */
/*   Calculate alpi[n]. Note when ai[n] goes to infinity, qi[n] goes to zero. */
```

```
/*                                                                          */
/****************************************************************************/

  for(n=1;n<=3;n++)
     {
     if(x[n+1] == x[n])
        alpi[n]=0.;
     else
        alpi[n]=1.0/sqrt(1.0+ai[n]*ai[n]);
     }

/****************************************************************************/
/*                                                                          */
/*   Calculate gi[n]. Note when ai[n] goes to infinity, gi[n] goes to one.  */
/*                                                                          */
/****************************************************************************/

  for(n=1;n<=3;n++)
     {
     if(x[n+1] == x[n])
        gi[n]=1.;
     else
        gi[n]=ai[n]*alpi[n];
     }

/****************************************************************************/
/*                                                                          */
/*   Calculate bi[n],li[n],C[n],B[n],A[n],ci[n],di[n],tip[n].tim[n],Rip[n]  */
/*             Rim[n]                                                       */
/*                                                                          */
/****************************************************************************/

  for(n=1;n<=3;n++)
     {

     bi[n]=y[n]-ai[n]*x[n];

     li[n]=sqrt((x[n+1]-x[n])*(x[n+1]-x[n])+(y[n+1]-y[n])*(y[n+1]-y[n]));

     C[n]=alpi[n]*alpi[n]+gi[n]*gi[n];
     C2[n]=sqrt(C[n]);

     }

  ci[1]=x[1];
  ci[2]=x[2]-alpi[2]*li[1];
  ci[3]=x[3]-alpi[3]*(li[1]+li[2]);

  di[1]=y[1];
  di[2]=y[2]-gi[2]*li[1];
  di[3]=y[3]-gi[3]*(li[1]+li[2]);

  tip[1]=li[1];
  tip[2]=li[1]+li[2];
  tip[3]=li[1]+li[2]+li[3];
```

```
    tim[1]=0.;
    tim[2]=tip[1];
    tim[3]=tip[2];

    for(n=1;n<=3;n++)
       {

       B[n]=(-2.0)*(alpi[n]*(xo-ci[n])+gi[n]*(yo-di[n]));

       A[n]=(xo-ci[n])*(xo-ci[n])+(yo-di[n])*(yo-di[n]);

       Rip[n]=sqrt(C[n]*tip[n]*tip[n]+B[n]*tip[n]+A[n]);

       Rim[n]=sqrt(C[n]*tim[n]*tim[n]+B[n]*tim[n]+A[n]);

       }

  /****************************************************************************/
  /*                                                                          */
  /*   Calculate Dxi[n],Dyi[n]                                                */
  /*                                                                          */
  /****************************************************************************/

    for(n=1;n<=3;n++)
       {

       if( x[n+1] >= x[n] )
            exi[n]=1.0;
       if( x[n+1] < x[n] )
            exi[n]=(-1.0);
       if( y[n+1] >= y[n] )
            eyi[n]=1.0;
       if( y[n+1] < y[n] )
            eyi[n]=(-1.0);

       Dxi[n]=exi[n]*alpi[n];

       Dyi[n]=eyi[n]*fabs(gi[n]);
       }

  /****************************************************************************/
  /*                                                                          */
  /*   Calculate integral of 1./R over a segment              I11             */
  /*   Calculate integral of t/R over a segment               I21             */
  /*   Calculate integral of t**2/R over a segment            I31             */
  /*   Calculate integral of R over a segment                 I41             */
  /*   Calculate integral of tR over a segment                I51             */
  /*   Calculate integral of (t**2)R over a segment           I61             */
  /*                                                                          */
  /****************************************************************************/

    for(n=1;n<=3;n++)
       {

       I11[n]=(log(2.*C2[n]*Rip[n]+2.*C[n]*tip[n]+B[n])-log(2.*C2[n]*Rim[n]+2.
              *C[n]*tim[n]+B[n]))/C2[n];
```

```

      I21[n]=(R1p[n]-R1m[n])/C2[n]-B[n]*I11[n]/(2.*C[n]);

      I31[n]=(((t1p[n]-3.*B[n]/(2.*C[n]))*R1p[n]-(t1m[n]-3.*B[n]/(2.*C[n]))
              *R1m[n])+(3.*B[n]*B[n]/(4.*C[n])-A[n])*I11[n])/(2.*C[n]);

      I41[n]=((2.*C[n]*t1p[n]+B[n])*R1p[n]-(2.*C[n]*t1m[n]+B[n])*R1m[n])/(4.
              *C[n])+(4.*A[n]*C[n]-B[n]*B[n])*I11[n]/(8.*C[n]);

      I51[n]=(powr(R1p[n],3)-powr(R1m[n],3))/(3.*C[n])-B[n]*I41[n]/(2.*C[n]);

      I61[n]=(((t1p[n]-5.*B[n]/(6.*C[n]))*powr(R1p[n],3)-(t1m[n]-5.*B[n]/(6.*C[n]
              ))*powr(R1m[n],3))+(5.*B[n]*B[n]/(4.*C[n])-A[n])*I41[n])/(4.*C[n]);
      }

/***************************************************************************/
/*                                                                         */
/*               Calculate integral of 1/R over the triangle         IT1   */
/*                                                                         */
/***************************************************************************/

   IT1=0.;
   for(n=1;n<=3;n++)
      IT1=IT1+(Dy1[n]*(alp1[n]*I21[n]-(xo-c1[n])*I11[n])-Dx1[n]*(g1[n]*I21[n]
              -(yo-d1[n])*I11[n]));

/***************************************************************************/
/*                                                                         */
/*               Calculate integral of x/R over the triangle         IT2   */
/*                                                                         */
/***************************************************************************/

   IT2=0.;
   for(n=1;n<=3;n++)
      IT2=IT2+(Dy1[n]*(alp1[n]*alp1[n]*I31[n]+alp1[n]*(2.*c1[n]-xo)*I21[n]+c1[n]
               *(c1[n]-xo)*I11[n])-Dx1[n]*(g1[n]*alp1[n]*I31[n]+(g1[n]*c1[n]
               +alp1[n]*(d1[n]-yo))*I21[n]+c1[n]*(d1[n]-yo)*I11[n]));
   IT2=IT2/2.0+xo*IT1/2.0;

/***************************************************************************/
/*                                                                         */
/*               Calculate integral of y/R over the triangle         IT3   */
/*                                                                         */
/***************************************************************************/

   IT3=0.;
   for(n=1;n<=3;n++)
      IT3=IT3+(Dy1[n]*(alp1[n]*g1[n]*I31[n]+(g1[n]*(c1[n]-xo)+alp1[n]*d1[n])
               *I21[n]+d1[n]*(c1[n]-xo)*I11[n])-Dx1[n]*(g1[n]*g1[n]*I31[n]+g1[n]
               *(2.*d1[n]-yo)*I21[n]+d1[n]*(d1[n]-yo)*I11[n]));
   IT3=IT3/2.0+yo*IT1/2.0;

/***************************************************************************/
/*                                                                         */
/*               Calculate integral of 1 over the triangle           IT4   */
/*                                                                         */
```

```
/******************************************************************************/

  IT4=0.;
  for(n=1;n<=3;n++)
     IT4=IT4+((alpi[n]*Dyi[n]-gi[n]*Dxi[n])*(tip[n]*tip[n]-tim[n]*tim[n])/2.0
            +(Dyi[n]*ci[n]-Dxi[n]*di[n])*(tip[n]-tim[n]))/2.0;

/******************************************************************************/
/*                                                                            */
/*              Calculate integral of x over the triangle             IT5     */
/*                                                                            */
/******************************************************************************/

  IT5=0.;
  for(n=1;n<=3;n++)
     IT5=IT5+(Dyi[n]*(alpi[n]*alpi[n]*(powr(tip[n],3)-powr(tim[n],3))/3.0
        +alpi[n]*ci[n]*(tip[n]*tip[n]-tim[n]*tim[n])+ci[n]*ci[n]*(tip[n]
        -tim[n])))/2.0;

/******************************************************************************/
/*                                                                            */
/*              Calculate integral of y over the triangle             IT6     */
/*                                                                            */
/******************************************************************************/

  IT6=0.;
  for(n=1;n<=3;n++)
     IT6=IT6+(Dxi[n]*(gi[n]*gi[n]*(powr(tip[n],3)-powr(tim[n],3))/3.0+gi[n]
        *di[n]*(tip[n]*tip[n]-tim[n]*tim[n])+di[n]*di[n]*(tip[n]-tim[n])))/2.0;

/******************************************************************************/
/*                                                                            */
/*              Calculate integral of R over the triangle             IT7     */
/*                                                                            */
/******************************************************************************/

  IT7=0.;
  for(n=1;n<=3;n++)
     IT7=IT7+(Dyi[n]*(alpi[n]*I5i[n]-(xo-ci[n])*I4i[n])-Dxi[n]*(gi[n]*I5i[n]
            -(yo-di[n])*I4i[n]))/3.0;

/******************************************************************************/
/*                                                                            */
/*              Calculate integral of xR over the triangle            IT8     */
/*                                                                            */
/******************************************************************************/

  IT8=0.;
  for(n=1;n<=3;n++)
     IT8=IT8+(Dyi[n]*(alpi[n]*alpi[n]*I6i[n]+alpi[n]*(2.*ci[n]-xo)*I5i[n]+ci[n]
            *(ci[n]-xo)*I4i[n])-Dxi[n]*(gi[n]*alpi[n]*I6i[n]+(gi[n]*ci[n]
            +alpi[n]*(di[n]-yo))*I5i[n]+ci[n]*(di[n]-yo)*I4i[n]));
  IT8=IT8/4.0+xo*IT7/4.0;

/******************************************************************************/
/*                                                                            */
```

```
/*                Calculate integral of yR over the triangle            IT9      */
/*                                                                               */
/*********************************************************************************/

   IT9=0.;
   for(n=1;n<=3;n++)
      IT9=IT9+(Dyi[n]*(alpi[n]*gi[n]*I6i[n]+(gi[n]*(ci[n]-xo)+alpi[n]*di[n])
            *I5i[n]+di[n]*(ci[n]-xo)*I4i[n])-Dxi[n]*(gi[n]*gi[n]*I6i[n]+gi[n]
            *(2.*di[n]-yo)*I5i[n]+di[n]*(di[n]-yo)*I4i[n]));
   IT9=IT9/4.0+yo*IT7/4.0;

/*********************************************************************************/
/*                                                                               */
/*      Calculate derivatives of I11                                             */
/*      DXI11 is first derivative of I11 with respect to x                       */
/*      DYI11 is first derivative of I11 with respect to y                       */
/*      DX2I11 is second derivative of I11 with respect to x                     */
/*      DY2I11 is second derivative of I11 with respect to y                     */
/*      DXYI11 is derivative of I11 with respect to x and y                      */
/*                                                                               */
/*********************************************************************************/

      for(n=1;n<=3;n++)
         {
         DXI11[n]=0.0;
         DYI11[n]=0.0;
         DX2I11[n]=0.0;
         DY2I11[n]=0.0;
         DXYI11[n]=0.0;

         t[1]=tip[n];
         t[2]=tim[n];

         R[1]=Rip[n];
         R[2]=Rim[n];

         sign=1.0;

         for(m=1;m<=2;m++)
            {
            xt[m]=xo-(alpi[n]*t[m]+ci[n]);
            yt[m]=yo-(gi[n]*t[m]+di[n]);
            ntrmx=2.0*alpi[n]*C2[n]*R[m]-2.0*C[n]*xt[m];
            dtrm=2.0*C2[n]*R[m]+2.0*C[n]*t[m]+B[n];
            ntrmy=2.0*gi[n]*C2[n]*R[m]-2.0*C[n]*yt[m];

            if(m==2)
               sign=(-1.0);

            DXI11[n]=DXI11[n]-sign*ntrmx/(C[n]*R[m]*dtrm);

            DYI11[n]=DYI11[n]-sign*ntrmy/(C[n]*R[m]*dtrm);

            DX2I11[n]=DX2I11[n]+sign*(-(ntrmx*ntrmx)/(C[n]*sqrt(C[n])*R[m]
                        *R[m]*dtrm*dtrm)+(2.0/dtrm)*(1.0/R[m]-(xt[m]*xt[m])/
                        powr(R[m],3)));
```

```
                DY2I11[n]=DY2I11[n]+sign*(-(ntrmy*ntrmy)/(C[n]*sqrt(C[n])*R[m]
                        *R[m]*dtrm*dtrm)+(2.0/dtrm)*(1.0/R[m]-(yt[m]*yt[m])/
                        powr(R[m],3)));

                DXYI11[n]=DXYI11[n]+sign*(-(ntrmx*ntrmy)/(C[n]*sqrt(C[n])*R[m]
                        *R[m]*dtrm*dtrm)-(2.0*xt[m]*yt[m])/(dtrm*powr(R[m],3)));
                }

/*************************************************************************/
/*                                                                       */
/*      Calculate derivatives of I21                                     */
/*      DXI21 is first derivative of I21 with respect to x               */
/*      DYI21 is first derivative of I21 with respect to y               */
/*      DX2I21 is second derivative of I21 with respect to x             */
/*      DY2I21 is second derivative of I21 with respect to y             */
/*      DXYI21 is derivative of I21 with respect to x and y              */
/*                                                                       */
/*************************************************************************/

            DXI21[n]=(xt[1]/R[1]-xt[2]/R[2])/sqrt(C[n])-B[n]*DXI11[n]/(2.*C[n])
                    +alp1[n]*I11[n]/C[n];

            DYI21[n]=(yt[1]/R[1]-yt[2]/R[2])/sqrt(C[n])-B[n]*DYI11[n]/(2.*C[n])
                    +g1[n]*I11[n]/C[n];

            DX2I21[n]=((1.0/R[1]-xt[1]*xt[1]/powr(R[1],3))-(1.0/R[2]-xt[2]*xt[2]/
                     powr(R[2],3)))/sqrt(C[n])-B[n]*DX2I11[n]/(2.*C[n])+2.
                     *alp1[n]*DXI11[n]/C[n];

            DY2I21[n]=((1.0/R[1]-yt[1]*yt[1]/powr(R[1],3))-(1.0/R[2]-yt[2]*yt[2]/
                     powr(R[2],3)))/sqrt(C[n])-B[n]*DY2I11[n]/(2.*C[n])+2.*g1[n]
                     *DYI11[n]/C[n];

            DXYI21[n]=(xt[2]*yt[2]/powr(R[2],3)-xt[1]*yt[1]/powr(R[1],3))/C2[n]
                     +g1[n]*DXI11[n]/C[n]+alp1[n]*DYI11[n]/C[n]-B[n]*DXYI11[n]/
                     (2.*C[n]);

/*************************************************************************/
/*                                                                       */
/*      Calculate derivatives of I31                                     */
/*      DXI31 is first derivative of I31 with respect to x               */
/*      DYI31 is first derivative of I31 with respect to y               */
/*      DX2I31 is second derivative of I31 with respect to x             */
/*      DY2I31 is second derivative of I31 with respect to y             */
/*      DXYI31 is derivative of I31 with respect to x and y              */
/*                                                                       */
/*************************************************************************/

            DXI31[n]=(((t[1]-3.*B[n]/(2.*C[n]))*xt[1]/R[1]+3.*alp1[n]*R[1]/C[n])
                    -((t[2]-3.*B[n]/(2.*C[n]))*xt[2]/R[2]+3.*alp1[n]*R[2]/C[n]))
                    /(2.*C[n]);
            DXI31[n]=DXI31[n]+(3.*B[n]*B[n]/(4.*C[n])-A[n])*DXI11[n]/(2.*C[n])
                    -(3.*B[n]*alp1[n]/(2.*C[n])+(xo-c1[n]))*I11[n]/C[n];

            DYI31[n]=(((t[1]-3.*B[n]/(2.*C[n]))*yt[1]/R[1]+3.*g1[n]*R[1]/C[n])
```

```
                    -((t[2]-3.*B[n]/(2.*C[n]))*yt[2]/R[2]+3.*gi[n]*R[2]/C[n]))
                    /(2.*C[n]);
          DYI3i[n]=DYI3i[n]+(3.*B[n]*B[n]/(4.*C[n])-A[n])*DYI1i[n]/(2.*C[n])
                    -(3.*B[n]*gi[n]/(2.*C[n])+(yo-di[n]))*I1i[n]/C[n];

          DX2I3i[n]=(3.*alpi[n]*xt[1]/(C[n]*C[n]*R[1])+(t[1]-3.*B[n]/(2.*C[n]))
                    *(1./R[1]-xt[1]*xt[1]/powr(R[1],3))/(2.*C[n]))-(3.*alpi[n]
                    *xt[2]/(C[n]*C[n]*R[2])+(t[2]-3.*B[n]/(2.*C[n]))*(1./R[2]
                    -xt[2]*xt[2]/powr(R[2],3))/(2.*C[n]));
          DX2I3i[n]=DX2I3i[n]+(3.*B[n]*B[n]/(4.*C[n])
                    -A[n])*DX2I1i[n]/(2.*C[n])-(3.*B[n]*alpi[n]/C[n]-2.*(xo
                    -ci[n]))*DXI1i[n]/C[n]+(3.*alpi[n]*alpi[n]/C[n]-1.)*I1i[n]/
                    C[n];

          DY2I3i[n]=(3.*gi[n]*yt[1]/(C[n]*C[n]*R[1])+(t[1]-3.*B[n]/(2.*C[n]))
                    *(1./R[1]-yt[1]*yt[1]/powr(R[1],3))/(2.*C[n]))-(3.*gi[n]
                    *yt[2]/(C[n]*C[n]*R[2])+(t[2]-3.*B[n]/(2.*C[n]))*(1./R[2]
                    -yt[2]*yt[2]/powr(R[2],3))/(2.*C[n]));
          DY2I3i[n]=DY2I3i[n]+(3.*B[n]*B[n]/(4.*C[n])
                    -A[n])*DY2I1i[n]/(2.*C[n])-(3.*B[n]*gi[n]/C[n]-2.*(yo
                    -di[n]))*DYI1i[n]/C[n]+(3.*gi[n]*gi[n]/C[n]-1.)*I1i[n]/C[n];

          DXYI3i[n]=3.*alpi[n]*yt[1]/(2.*C[n]*C[n]*R[1])-(t[1]-3.*B[n]/(2.
                    *C[n]))*yt[1]*xt[1]/(2.*C[n]*powr(R[1],3))+3.*gi[n]*xt[1]/(2.
                    *C[n]*C[n]*R[1]);
          DXYI3i[n]=DXYI3i[n]-(3.*alpi[n]*yt[2]/(2.*C[n]*C[n]*R[2])-(t[2]
                   -3.*B[n]/(2.*C[n]))*yt[2]*xt[2]/(2.*C[n]*powr(R[2],3))+3.
                   *gi[n]*xt[2]/(2.*C[n]*C[n]*R[2]));
          DXYI3i[n]=DXYI3i[n]-(3.*B[n]*alpi[n]/(2.*C[n])
                   +xo-ci[n])*DYI1i[n]/C[n]+(3.*B[n]*B[n]/(4.*C[n])-A[n])
                   *DXYI1i[n]/(2.*C[n]);
          DXYI3i[n]=DXYI3i[n]-(3.*B[n]*gi[n]/(2.*C[n])+yo-di[n])
                   *DXI1i[n]/C[n]+3.*gi[n]*gi[n]*I1i[n]/(C[n]*C[n]);
/*************************************************************************/
/*                                                                       */
/*      Calculate derivatives of I4i                                     */
/*      DXI4i is first derivative of I4i with respect to x               */
/*      DYI4i is first derivative of I4i with respect to y               */
/*      DX2I4i is second derivative of I4i with respect to x             */
/*      DY2I4i is second derivative of I4i with respect to y             */
/*      DXYI4i is derivative of I4i with respect to x and y              */
/*                                                                       */
/*************************************************************************/

         delta[n]=(4.*A[n]*C[n]-B[n]*B[n])/(8.*C[n]);

         DXI4i[n]=(((2.0*C[n]*t[1]+B[n])*xt[1]/R[1]-2.*alpi[n]*R[1])-((2.0*C[n]
                  *t[2]+B[n])*xt[2]/R[2]-2.0*alpi[n]*R[2]))/(4.0*C[n])+(xo
                  -ci[n]+B[n]*alpi[n]/(2.0*C[n]))*I1i[n]+delta[n]*DXI1i[n];

         DYI4i[n]=(((2.0*C[n]*t[1]+B[n])*yt[1]/R[1]-2.*gi[n]*R[1])-((2.0*C[n]
                  *t[2]+B[n])*yt[2]/R[2]-2.0*gi[n]*R[2]))/(4.0*C[n])+(yo-di[n]
                  +B[n]*gi[n]/ (2.0*C[n]))*I1i[n]+delta[n]*DYI1i[n];

         DX2I4i[n]=(((2.0*C[n]*t[1]+B[n]-4.0*alpi[n]*xt[1])/R[1]-(2.0*C[n]*t[1]
```

```
                      +B[n])*xt[1]*xt[1]/powr(R[1],3))-((2.0*C[n]*t[2]
                      +B[n]-4.0*alpi[n]*xt[2])/R[2]-(2.0*C[n]*t[2]+B[n])*xt[2]
                      *xt[2]/powr(R[2],3)))/(4.0*C[n]);
            DX2I41[n]=DX2I41[n]+(1.0-alpi[n]*alpi[n]/C[n])
                      *I11[n]+(2.0*(xo-ci[n])+B[n]*alpi[n]/C[n])*DXI11[n]+delta[n]
                      *DX2I11[n];

            DY2I41[n]=(((2.0*C[n]*t[1]+B[n]-4.0*gi[n]*yt[1])/R[1]-(2.0*C[n]*t[1]
                      +B[n])*yt[1]*yt[1]/powr(R[1],3))-((2.0*C[n]*t[2]
                      +B[n]-4.0*gi[n]*yt[2])/R[2]-(2.0*C[n]*t[2]+B[n])*yt[2]
                      *yt[2]/powr(R[2],3)))/(4.0*C[n]);
            DY2I41[n]=DY2I41[n]+(1.0-gi[n]*gi[n]/C[n])
                      *I11[n]+(2.0*(yo-di[n])+B[n]*gi[n]/C[n])*DYI11[n]+delta[n]
                      *DY2I11[n];

            DXYI41[n]=((2.0*(gi[n]*xt[2]+alpi[n]*yt[2])/R[2]+(2.0
                      *C[n]*t[2]+B[n])*xt[2]*yt[2]/powr(R[2],3))-(2.0*(gi[n]*xt[1]
                      +alpi[n]*yt[1])/R[1]+(2.0*C[n]*t[1]+B[n])*xt[1]*yt[1]
                      /powr(R[1],3)))/(4.0*C[n]);
            DXYI41[n]=DXYI41[n]-gi[n]*alpi[n]*I11[n]/C[n]
                      +(xo-ci[n]+B[n]*alpi[n]/(2.0*C[n]))*DYI11[n]+(yo-di[n]+gi[n]
                      *B[n]/(2.0*C[n]))*DXI11[n]+delta[n]*DXYI11[n];

/***************************************************************************/
/*                                                                         */
/*      Calculate derivatives of I51                                       */
/*      DXI51 is first derivative of I51 with respect to x                 */
/*      DYI51 is first derivative of I51 with respect to y                 */
/*      DX2I51 is second derivative of I51 with respect to x               */
/*      DY2I51 is second derivative of I51 with respect to y               */
/*      DXYI51 is derivative of I51 with respect to x and y                */
/*                                                                         */
/***************************************************************************/

            DXI51[n]=(R[1]*xt[1]-R[2]*xt[2]-B[n]*DXI41[n]/2.0+alpi[n]*I41[n])
                     /C[n];

            DYI51[n]=(R[1]*yt[1]-R[2]*yt[2]-B[n]*DYI41[n]/2.0+gi[n]*I41[n])/C[n];

            DX2I51[n]=(R[1]+xt[1]*xt[1]/R[1]-R[2]-xt[2]*xt[2]/R[2]-B[n]*DX2I41[n]
                     /2.0+2.0*alpi[n]*DXI41[n])/C[n];

            DY2I51[n]=(R[1]+yt[1]*yt[1]/R[1]-R[2]-yt[2]*yt[2]/R[2]-B[n]*DY2I41[n]
                     /2.0+2.0*gi[n]*DYI41[n])/C[n];

            DXYI51[n]=(xt[1]*yt[1]/R[1]-xt[2]*yt[2]/R[2]+gi[n]*DXI41[n]-B[n]
                     *DXYI41[n]/2.0+alpi[n]*DYI41[n])/C[n];

/***************************************************************************/
/*                                                                         */
/*      Calculate derivatives of I61                                       */
/*      DXI61 is first derivative of I61 with respect to x                 */
/*      DYI61 is first derivative of I61 with respect to y                 */
/*      DX2I61 is second derivative of I61 with respect to x               */
/*      DY2I61 is second derivative of I61 with respect to y               */
/*      DXYI61 is derivative of I51 with respect to x and y                */
```

```
/*                                                                          */
/****************************************************************************/

            DXI61[n]=(5.*alpi[n]*powr(R[1],3)/(12.*C[n]*C[n])+(t[1]/(4.*C[n])-5.
                 *B[n]/(24.*C[n]*C[n]))*3.*R[1]*xt[1])-(5.*alpi[n]*powr(R[2],3)/
                  (12.*C[n]*C[n])+(t[2]/(4.*C[n])-5.*B[n]/(24.*C[n]*C[n]))*3.*R[2]
                       *xt[2]);
            DXI61[n]=DXI61[n]-(5.*B[n]*alpi[n]/(4.*C[n]*C[n])+(xo-ci[n])/(2.*C[n]))
                       *I41[n]+(5.*B[n]*B[n]/(16.*C[n]*C[n])-A[n]/(4.*C[n]))
                       *DXI41[n];

            DYI61[n]=(5.*gi[n]*powr(R[1],3)/(12.*C[n]*C[n])+(t[1]/(4.*C[n])-5.
                  *B[n]/(24.*C[n]*C[n]))*3.*R[1]*yt[1])-(5.*gi[n]*powr(R[2],3)/
                  (12.*C[n]*C[n])+(t[2]/(4.*C[n])-5.*B[n]/(24.*C[n]*C[n]))*3.*R[2]
                       *yt[2]);
            DYI61[n]=DYI61[n]-(5.*B[n]*gi[n]/(4.*C[n]*C[n])+(yo-di[n])/(2.*C[n]))
                       *I41[n]+(5.*B[n]*B[n]/(16.*C[n]*C[n])-A[n]/(4.*C[n]))
                       *DYI41[n];

            DX2I61[n]=(5.*alpi[n]*R[1]*xt[1]/(2.*C[n]*C[n])+3.*(t[1]/(4.*C[n])-5.
                       *B[n]/(24.*C[n]*C[n]))*(R[1]+xt[1]*xt[1]/R[1]));
            DX2I61[n]=DX2I61[n]-(5.*alpi[n]
                       *R[2]*xt[2]/(2.*C[n]*C[n])+3.*(t[2]/(4.*C[n])-5.*B[n]/(24.
                       *C[n]*C[n]))*(R[2]+xt[2]*xt[2]/R[2]));
            DX2I61[n]=DX2I61[n]-2.*(5.*B[n]*alpi[n]/
                      (4.*C[n]*C[n])+(xo-ci[n])/(2.*C[n]))*DXI41[n]-(1.-5.*alpi[n]
                       *alpi[n]/C[n])*I41[n]/(2.*C[n]);
            DX2I61[n]=DX2I61[n]+(5.*B[n]*B[n]/(16.*C[n]*C[n])
                       -A[n]/(4.*C[n]))*DX2I41[n];

            DY2I61[n]=(5.*gi[n]*R[1]*yt[1]/(2.*C[n]*C[n])+3.*(t[1]/(4.*C[n])-5.
                       *B[n]/(24.*C[n]*C[n]))*(R[1]+yt[1]*yt[1]/R[1]));
            DY2I61[n]=DY2I61[n]-(5.*gi[n]
                       *R[2]*yt[2]/(2.*C[n]*C[n])+3.*(t[2]/(4.*C[n])-5.*B[n]/(24.
                       *C[n]*C[n]))*(R[2]+yt[2]*yt[2]/R[2]));
            DY2I61[n]=DY2I61[n]-2.*(5.*B[n]*gi[n]/(4.
                       *C[n]*C[n])+(yo-di[n])/(2.*C[n]))*DYI41[n]-(1.-5.*gi[n]*gi[n]
                       /C[n])*I41[n]/(2.*C[n]);
            DY2I61[n]=DY2I61[n]+(5.*B[n]*B[n]/(16.*C[n]*C[n])-A[n]/
                       (4.*C[n]))*DY2I41[n];

            DXYI61[n]=(5.*gi[n]*R[1]*xt[1]/(4.*C[n]*C[n])+5.*alpi[n]*R[1]*yt[1]/
                       (4.*C[n]*C[n])+3.*(t[1]/(4.*C[n])-5.*B[n]/(24.*C[n]*C[n]))
                       *xt[1]*yt[1]/R[1]);
            DXYI61[n]=DXYI61[n]-(5.*gi[n]*R[2]*xt[2]/(4.*C[n]*C[n])+5.
                       *alpi[n]*R[2]*yt[2]/(4.*C[n]*C[n])+3.*(t[2]/(4.*C[n])-5.
                       *B[n]/(24.*C[n]*C[n]))*xt[2]*yt[2]/R[2]);
            DXYI61[n]=DXYI61[n]-5.*alpi[n]*gi[n]
                       *I41[n]/(2.*C[n]*C[n])-(5.*B[n]*gi[n]/(2.*C[n])+yo-di[n])
                       *DXI41[n]/(2.*C[n]);
            DXYI61[n]=DXYI61[n]-(5.*B[n]*alpi[n]/(2.*C[n])+xo-ci[n])
                       *DYI41[n]/(2.*C[n])+(5.*B[n]*B[n]/(16.*C[n]*C[n])-A[n]/(4.
                       *C[n]))*DXYI41[n];

            }
```

```
/****************************************************************************/
/*                                                                          */
/*      Calculate derivatives of IT1                                        */
/*      DXIT1 is first derivative of IT1 with respect to x                  */
/*      DYIT1 is first derivative of IT1 with respect to y                  */
/*      DX2IT1 is second derivative of IT1 with respect to x                */
/*      DY2IT1 is second derivative of IT1 with respect to y                */
/*      DXYIT1 is derivative of IT1 with respect to x and y                 */
/*                                                                          */
/****************************************************************************/

      DXIT1=0.0;
      DYIT1=0.0;
      DX2IT1=0.0;
      DY2IT1=0.0;
      DXYIT1=0.0;

      for(n=1;n<=3;n++)
        {
         DXIT1=DXIT1+Dy1[n]*(alp1[n]*DXI21[n]+(c1[n]-xo)*DXI11[n]-I11[n])
                   -Dx1[n]*(g1[n]*DXI21[n]+(d1[n]-yo)*DXI11[n]);

         DYIT1=DYIT1+Dy1[n]*(alp1[n]*DYI21[n]+(c1[n]-xo)*DYI11[n])
                   -Dx1[n]*(g1[n]*DYI21[n]+(d1[n]-yo)*DYI11[n]-I11[n]);

         DX2IT1=DX2IT1+Dy1[n]*(alp1[n]*DX2I21[n]+(c1[n]-xo)*DX2I11[n]-2.
                     *DXI11[n])-Dx1[n]*(g1[n]*DX2I21[n]+(d1[n]-yo)*DX2I11[n]);

         DY2IT1=DY2IT1+Dy1[n]*(alp1[n]*DY2I21[n]+(c1[n]-xo)*DY2I11[n])
                    -Dx1[n]*(g1[n]*DX2I21[n]+(d1[n]-yo)*DX2I11[n]-2.*DYI11[n]);


         DXYIT1=DXYIT1+Dy1[n]*(alp1[n]*DXYI21[n]+(c1[n]-xo)*DXYI11[n]-DYI11[n])
                    -Dx1[n]*(g1[n]*DXYI21[n]+(d1[n]-yo)*DXYI11[n]-DXI11[n]);
        }

/****************************************************************************/
/*                                                                          */
/*      Calculate derivatives of IT2                                        */
/*      DX2IT2 is second derivative of IT2 with respect to x                */
/*      DY2IT2 is second derivative of IT2 with respect to y                */
/*      DXYIT2 is derivative of IT2 with respect to x and y                 */
/*                                                                          */
/****************************************************************************/

      DX2IT2=0.0;
      DY2IT2=0.0;
      DXYIT2=0.0;

      for(n=1;n<=3;n++)
        {
         DX2IT2=DX2IT2+Dy1[n]*(alp1[n]*alp1[n]*DX2I31[n]+alp1[n]*(2.*c1[n]-xo)
                *DX2I21[n]-2.*alp1[n]*DXI21[n]+c1[n]*(c1[n]-xo)*DX2I11[n]-2.
                *c1[n]*DXI11[n]);
         DX2IT2=DX2IT2-Dx1[n]*(alp1[n]*g1[n]*DX2I31[n]+(g1[n]*c1[n]
                +alp1[n]*(d1[n]-yo))*DX2I21[n]+c1[n]*(d1[n]-yo)*DX2I11[n]);
```

```
            DY2IT2=DY2IT2+Dyi[n]*(alpi[n]*alpi[n]*DY2I3i[n]+alpi[n]*(2.*ci[n]-xo)
                    *DY2I2i[n]+ci[n]*(ci[n]-xo)*DY2I1i[n]);
            DY2IT2=DY2IT2-Dxi[n]*(gi[n]*alpi[n]
                    *DY2I3i[n]-2.*alpi[n]*DYI2i[n]+(gi[n]*ci[n]+alpi[n]*(di[n]-yo))
                    *DY2I2i[n]-2.*ci[n]*DYI1i[n]+ci[n]*(di[n]-yo)*DY2I1i[n]);

            DXYIT2=DXYIT2+Dyi[n]*(alpi[n]*alpi[n]*DXYI3i[n]+alpi[n]*(2.*ci[n]-xo)
                    *DXYI2i[n]-alpi[n]*DYI2i[n]+ci[n]*(ci[n]-xo)*DXYI1i[n]-ci[n]
                    *DYI1i[n]);
            DXYIT2=DXYIT2-Dxi[n]*(alpi[n]*gi[n]*DXYI3i[n]+(gi[n]*ci[n]+alpi[n]
                    *(di[n]-yo))*DXYI2i[n]-alpi[n]*DXI2i[n]+ci[n]*(di[n]-yo)
                    *DXYI1i[n]-ci[n]*DXI1i[n]);
            }

        DX2IT2=DX2IT2/2.0+xo*DX2IT1/2.0+DXIT1;
        DY2IT2=DY2IT2/2.0+xo*DY2IT1/2.0;
        DXYIT2=DXYIT2/2.0+DYIT1/2.0+xo*DXYIT1/2.0;

/**************************************************************************/
/*                                                                        */
/*      Calculate derivatives of IT3                                      */
/*      DX2IT3 is second derivative of IT3 with respect to x              */
/*      DY2IT3 is second derivative of IT3 with respect to y              */
/*      DXYIT3 is derivative of IT3 with respect to x and y               */
/*                                                                        */
/**************************************************************************/

        DX2IT3=0.0;
        DY2IT3=0.0;
        DXYIT3=0.0;

        for(n=1;n<=3;n++)
            {
            DX2IT3=DX2IT3+Dyi[n]*(alpi[n]*gi[n]*DX2I3i[n]+(gi[n]*(ci[n]-xo)+di[n])
                    *DX2I2i[n]-2.*gi[n]*DXI2i[n]+di[n]*(ci[n]-xo)*DX2I1i[n]-2.*di[n]
                    *I1i[n]);
            DX2IT3=DX2IT3-Dxi[n]*(gi[n]*gi[n]*DX2I3i[n]+gi[n]*(2.*di[n]-yo)
                    *DX2I2i[n]+di[n]*(di[n]-yo)*DX2I1i[n]);

            DY2IT3=DY2IT3+Dyi[n]*(gi[n]*alpi[n]*DY2I3i[n]+(gi[n]*(ci[n]-xo)+di[n])
                    *DY2I2i[n]+di[n]*(ci[n]-xo)*DY2I1i[n]);
            DY2IT3=DY2IT3-Dxi[n]*(gi[n]*gi[n]
                    *DY2I3i[n]+gi[n]*(2.*di[n]-yo)*DY2I2i[n]-2.*gi[n]*DYI2i[n]+di[n]
                    *(di[n]-yo)*DY2I1i[n]-2.*di[n]*DYI1i[n]);

            DXYIT3=DXYIT3+Dyi[n]*(alpi[n]*gi[n]*DXYI3i[n]+(gi[n]*(ci[n]-xo)+di[n])
                    *DXYI2i[n]-gi[n]*DYI2i[n]+di[n]*(ci[n]-xo)*DXYI1i[n]-di[n]
                    *DYI1i[n]);
            DXYIT3=DXYIT3-Dxi[n]*(gi[n]*gi[n]*DXYI3i[n]+gi[n]*(2.*di[n]-yo)
                    *DXYI2i[n]-gi[n]*DXI2i[n]+di[n]*(di[n]-yo)*DXYI1i[n]-di[n]
                    *DXI1i[n]);

            }

        DX2IT3=DX2IT3/2.0+yo*DX2IT1/2.0;
```

```
        DY2IT3=DY2IT3/2.0+yo*DY2IT1/2.0+DYIT1;
        DXYIT3=DXYIT3/2.0+DXIT1/2.0+yo*DXYIT1/2.0;

/***************************************************************************/
/*                                                                         */
/*      Calculate derivatives of IT7                                       */
/*      DXIT7 is first derivative of IT7 with respect to x                 */
/*      DYIT7 is first derivative of IT7 with respect to y                 */
/*      DX2IT7 is second derivative of IT7 with respect to x               */
/*      DY2IT7 is second derivative of IT7 with respect to y               */
/*      DXYIT7 is derivative of IT7 with respect to x and y                */
/*                                                                         */
/***************************************************************************/

        DXIT7=0.0;
        DYIT7=0.0;
        DX2IT7=0.0;
        DY2IT7=0.0;
        DXYIT7=0.0;

        for(n=1;n<=3;n++)
           {
           DXIT7=DXIT7+Dy1[n]*(alp1[n]*DXI51[n]+(c1[n]-xo)*DXI41[n]-I41[n])
                      -Dx1[n]*(g1[n]*DXI51[n]+(d1[n]-yo)*DXI41[n]);

           DYIT7=DYIT7+Dy1[n]*(alp1[n]*DYI51[n]+(c1[n]-xo)*DYI41[n])
                      -Dx1[n]*(g1[n]*DYI51[n]+(d1[n]-yo)*DYI41[n]-I41[n]);

           DX2IT7=DX2IT7+Dy1[n]*(alp1[n]*DX2I51[n]+(c1[n]-xo)*DX2I41[n]-2.
                      *DXI41[n])-Dx1[n]*(g1[n]*DX2I51[n]+(d1[n]-yo)*DX2I41[n]);

           DY2IT7=DY2IT7+Dy1[n]*(alp1[n]*DY2I51[n]+(c1[n]-xo)*DY2I41[n])
                      -Dx1[n]*(g1[n]*DX2I51[n]+(d1[n]-yo)*DX2I41[n]-2.*DYI41[n]);


           DXYIT7=DXYIT7+Dy1[n]*(alp1[n]*DXYI51[n]+(c1[n]-xo)*DXYI41[n]-DYI41[n])
                      -Dx1[n]*(g1[n]*DXYI51[n]+(d1[n]-yo)*DXYI41[n]-DXI41[n]);

           }
        DXIT7=DXIT7/3.0;
        DYIT7=DYIT7/3.0;
        DX2IT7=DX2IT7/3.0;
        DY2IT7=DY2IT7/3.0;
        DXYIT7=DXYIT7/3.0;

/***************************************************************************/
/*                                                                         */
/*      Calculate derivatives of IT8                                       */
/*      DX2IT8 is second derivative of IT8 with respect to x               */
/*      DY2IT8 is second derivative of IT8 with respect to y               */
/*      DXYIT8 is derivative of IT8 with respect to x and y                */
/*                                                                         */
/***************************************************************************/

        DX2IT8=0.0;
        DY2IT8=0.0;
```

MISSING
PAGE

```
DXYIT8=0.0;

for(n=1;n<=3;n++)
   {
   DX2IT8=DX2IT8+Dyi[n]*(alpi[n]*alpi[n]*DX2I6i[n]+alpi[n]*(2.*ci[n]-xo)
          *DX2I5i[n]-2.*alpi[n]*DXI5i[n]+ci[n]*(ci[n]-xo)*DX2I4i[n]-2.
          *ci[n]*DXI4i[n]);
   DX2IT8=DX2IT8-Dxi[n]*(alpi[n]*gi[n]*DX2I6i[n]+(gi[n]*ci[n]
          +alpi[n]*(di[n]-yo))*DX2I5i[n]+ci[n]*(di[n]-yo)*DX2I4i[n]);

   DY2IT8=DY2IT8+Dyi[n]*(alpi[n]*alpi[n]*DY2I6i[n]+alpi[n]*(2.*ci[n]-xo)
          *DY2I5i[n]+ci[n]*(ci[n]-xo)*DY2I4i[n]);
   DY2IT8=DY2IT8-Dxi[n]*(gi[n]*alpi[n]
          *DY2I6i[n]-2.*alpi[n]*DYI5i[n]+(gi[n]*ci[n]+alpi[n]*(di[n]-yo))
          *DY2I5i[n]-2.*ci[n]*DYI4i[n]+ci[n]*(di[n]-yo)*DY2I4i[n]);

   DXYIT8=DXYIT8+Dyi[n]*(alpi[n]*alpi[n]*DXYI6i[n]+alpi[n]*(2.*ci[n]-xo)
          *DXYI5i[n]-alpi[n]*DYI5i[n]+ci[n]*(ci[n]-xo)*DXYI4i[n]-ci[n]
          *DYI4i[n]);
   DXYIT8=DXYIT8-Dxi[n]*(alpi[n]*gi[n]*DXYI6i[n]+(gi[n]*ci[n]+alpi[n]
          *(di[n]-yo))*DXYI5i[n]-alpi[n]*DXI5i[n]+ci[n]*(di[n]-yo)
          *DXYI4i[n]-ci[n]*DXI4i[n]);
   }

DX2IT8=DX2IT8/4.0+xo*DX2IT7/4.0+DXIT1/2.0;
DY2IT8=DY2IT8/4.0+xo*DY2IT7/4.0;
DXYIT8=DXYIT8/4.0+DYIT7/4.0+xo*DXYIT1/4.0;


/***************************************************************************/
/*                                                                         */
/*     Calculate derivatives of IT9                                        */
/*     DX2IT9 is second derivative of IT9 with respect to x                */
/*     DY2IT9 is second derivative of IT9 with respect to y                */
/*     DXYIT9 is derivative of IT9 with respect to x and y                 */
/*                                                                         */
/***************************************************************************/

DX2IT9=0.0;
DY2IT9=0.0;
DXYIT9=0.0;

for(n=1;n<=3;n++)
   {
   DX2IT9=DX2IT9+Dyi[n]*(alpi[n]*gi[n]*DX2I6i[n]+(gi[n]*(ci[n]-xo)+di[n])
          *DX2I5i[n]-2.*gi[n]*DXI5i[n]+di[n]*(ci[n]-xo)*DX2I4i[n]-2.*di[n]
          *I4i[n]);
   DX2IT9=DX2IT9-Dxi[n]*(gi[n]*gi[n]*DX2I6i[n]+gi[n]*(2.*di[n]-yo)
          *DX2I5i[n]+di[n]*(di[n]-yo)*DX2I4i[n]);

   DY2IT9=DY2IT9+Dyi[n]*(gi[n]*alpi[n]*DY2I6i[n]+(gi[n]*(ci[n]-xo)+di[n])
          *DY2I5i[n]+di[n]*(ci[n]-xo)*DY2I4i[n]);
   DY2IT9=DY2IT9-Dxi[n]*(gi[n]*gi[n]
          *DY2I6i[n]+gi[n]*(2.*di[n]-yo)*DY2I5i[n]-2.*gi[n]*DYI5i[n]+di[n]
          *(di[n]-yo)*DY2I4i[n]-2.*di[n]*DYI4i[n]);


   DY2IT9=(-1)*DY2IT9;
   DXYIT9=(-1)*DXYIT9;
   }
```

```
}

powr(x,n)

double x;
int n;

{
 double p,pow();

 if(x == 0.)
   {
   p=0.;
   return(p);
   }
 else
   {
   p=pow(x,n);
   return(p);
   }
}






numer(xo,yo,xs1,ys1,xs2,ys2,xs3,ys3)

double xo,yo,xs1,ys1,xs2,ys2,xs3,ys3;

{
/**********************************************************************/
/*                                                                    */
/*  This routine performs the numerical evaluations over the triangles  */
/*                                                                    */
/*  xo  x coordinate of the field point                               */
/*  yo  y coordinate of the field point                               */
/*  xs1 x coordinate of vertice 1 of source triangle                  */
/*  ys1 y coordinate of vertice 1 of source triangle                  */
/*  xs2 x coordinate of vertice 2 of source triangle                  */
/*  ys2 y coordinate of vertice 2 of source triangle                  */
/*  xs3 x coordinate of vertice 3 of source triangle                  */
/*  ys3 y coordinate of vertice 3 of source triangle                  */
/*  a   weight factor                                                 */
/*  b   weight factor                                                 */
/*  c   weight factor                                                 */
/*  r   evaluation point constant                                     */
/*  s   evaluation point constant                                     */
/*  ctrdx x coordinate of the centroid                                */
/*  ctrdy y coordinate of the centroid                                */
/*  xp[n] x coordinates of the evaluation points                      */
/*  yp[n] y coordinates of the evaluation points                      */
/*  area  area of the triangle                                        */
/*  k     wavenumber                                                  */
/*  rkr[n] real part of the rational kernel                           */
```

```
/*  rki[n] imag part of the rational kernel                                    */
/*  rkrx[n] real part of x times the rational kernel                           */
/*  rkix[n] imag part of x times the rational kernal                           */
/*  rkry[n] real part of y times the rational kernal                           */
/*  rkiy[n] imag part of y times the rational kernel                           */
/*  dx2rkr[n] real part of second derivative with respect to x of              */
/*            the rational kernel                                              */
/*  dx2rki[n] imag part of second derivative with respect to x of              */
/*            the rational kernel                                              */
/*  dx2rkrx[n] real part of x times second derivative with respect to x of*/
/*             the rational kernel                                             */
/*  dx2rkix[n] imag part of x times second derivative with respect to x of*/
/*             the rational kernel                                             */
/*  dx2rkry[n] real part of y times second derivative with respect to x of*/
/*             the rational kernel                                             */
/*  dx2rkiy[n] imag part of y times second derivative with respect to x of*/
/*             the rational kernel                                             */
/*  dy2rkr[n] real part of second derivative with respect to y of              */
/*            the rational kernel                                              */
/*  dy2rki[n] imag part of second derivative with respect to y of              */
/*            the rational kernel                                              */
/*  dy2rkrx[n] real part of x times second derivative with respect to y of*/
/*             the rational kernel                                             */
/*  dy2rkix[n] imag part of x times second derivative with respect to y of*/
/*             the rational kernel                                             */
/*  dy2rkry[n] real part of y times second derivative with respect to y of*/
/*             the rational kernel                                             */
/*  dy2rkiy[n] imag part of y times second derivative with respect to y of*/
/*             the rational kernel                                             */
/*  dxyrkr[n] real part of derivative with respect to x and y of               */
/*            the rational kernel                                              */
/*  dxyrki[n] imag part of derivative with respect to x and y of               */
/*            the rational kernel                                              */
/*  dxyrkrx[n] real part of x times derivative with respect to x and y of */
/*             the rational kernel                                             */
/*  dxyrkix[n] imag part of x times derivative with respect to x and y of */
/*             the rational kernel                                             */
/*  dxyrkry[n] real part of y times derivative with respect to x and y of */
/*             the rational kernel                                             */
/*  dxyrkiy[n] imag part of y times derivative with respect to x and y of */
/*             the rational kernel                                             */
/*****************************************************************************/
/*                                                                             */
/*  EXTERNAL VARIABLES                                                         */
/*                                                                             */
/*****************************************************************************/
/*  Rk_re real part of the rational kernel                                     */
/*  Rk_im imag part of the rational kernel                                     */
/*  Rkx_re real part of x times the rational kernel                            */
/*  Rkx_im imag part of x times the rational kernal                            */
/*  Rky_re real part of y times the rational kernal                            */
/*  Rky_im imag part of y times the rational kernel                            */
/*  Dx2rk_re real part of second derivative with respect to x of               */
/*           the rational kernel                                               */
/*  Dx2rk_im imag part of second derivative with respect to x of               */
/*           the rational kernel                                               */
```

```
/*  Dx2rkx_re real part of x times second derivative with respect to x of */
/*            the rational kernel                                         */
/*  Dx2rkx_im imag part of x times second derivative with respect to x of */
/*            the rational kernel                                         */
/*  Dx2rky_re real part of y times second derivative with respect to x of */
/*            the rational kernel                                         */
/*  Dx2rky_im imag part of y times second derivative with respect to x of */
/*            the rational kernel                                         */
/*  Dy2rk_re real part of second derivative with respect to y of          */
/*           the rational kernel                                          */
/*  Dy2rk_im imag part of second derivative with respect to y of          */
/*           the rational kernel                                          */
/*  Dy2rkx_re real part of x times second derivative with respect to y of */
/*            the rational kernel                                         */
/*  Dy2rkx_im imag part of x times second derivative with respect to y of */
/*            the rational kernel                                         */
/*  Dy2rky_re real part of y times second derivative witn respect to y of */
/*            the rational kernel                                         */
/*  Dy2rky_im imag part of y times second derivative with respect to y of */
/*            the rational kernel                                         */
/*  Dxyrk_re real part of derivative with respect to x and y of           */
/*           the rational kernel                                          */
/*  Dxyrk_im imag part of derivative with respect to x and y of           */
/*           the rational kernel                                          */
/*  Dxyrkx_re real part of x times derivative with respect to x and y of  */
/*            the rational kernel                                         */
/*  Dxyrkx_im imag part of x times derivative with respect to x and y of  */
/*            the rational kernel                                         */
/*  Dxyrky_re real part of y times derivative with respect to x and y of  */
/*            the rational kernel                                         */
/*  Dxyrky_im imag part of y times derivative with respect to x and y of  */
/*            the rational kernel                                         */
/*  dYk wavenumber                                                        */
/**************************************************************************/

double x[4],y[4],a,b,c,r,s,k,ctrdx,ctrdy,xp[8],yp[8],area,R[8];
double rkr[8],rki[8],rkrx[8],rkix[8],rkry[8],rkiy[8];
double dx2rkr[8],dx2rki[8],dx2rkrx[8],dx2rkix[8],dx2rkry[8],dx2rkiy[8];
double dy2rkr[8],dy2rki[8],dy2rkrx[8],dy2rkix[8],dy2rkry[8],dy2rkiy[8];
double dxyrkr[8],dxyrki[8],dxyrkrx[8],dxyrkix[8],dxyrkry[8],dxyrkiy[8];
double sqrt(),pow();
int n;

area=IT4;
k=dYk;

x[1]=xs1;
y[1]=ys1;
x[2]=xs2;
y[2]=ys2;
x[3]=xs3;
y[3]=ys3;

/**************************************************************************/
/*                                                                        */
/*   Calculate the centroid of the triangle                               */
```

```
/*                                                                      */
/************************************************************************/

ctrdx=(x[1]+x[2]+x[3])/3.0;
ctrdy=(y[1]+y[2]+y[3])/3.0;

/************************************************************************/
/*                                                                      */
/*    If the field point and the centroid are coincident use the numerical*/
/*    method described in Hammer for a quadratic function to avoid the    */
/*    singularity problem                                                 */
/*                                                                      */
/************************************************************************/

if( xo == ctrdx && yo == ctrdy )
     {
     r=0.5;

     for(n=1;n<=3;n++)
        {
        xp[n]=r*x[n]+(1.0-r)*ctrdx;
        yp[n]=r*y[n]+(1.0-r)*ctrdy;
        }
     for(n=1;n<=3;n++)
        R[n]=sqrt((xo-xp[n])*(xo-xp[n])+(yo-yp[n])*(yo-yp[n]));

     for(n=1;n<=3;n++)
      {

      rkr[n]=cos(k*R[n])/R[n]-1.0/R[n]+k*k*R[n]/2.0;
      rki[n]=sin(k*R[n])/R[n]-k;
      rkrx[n]=xp[n]*(cos(k*R[n])/R[n]-1.0/R[n]+k*k*R[n]/2.0);
      rkix[n]=xp[n]*(sin(k*R[n])/R[n]-k);
      rkry[n]=yp[n]*(cos(k*R[n])/R[n]-1.0/R[n]+k*k*R[n]/2.0);
      rkiy[n]=yp[n]*(sin(k*R[n])/R[n]-k);


      dx2rkr[n]=(3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0-k*k
                *(xo-xp[n])*(xo-xp[n]))*cos(k*R[n])/pow(R[n],3)
                -(1.-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k
                *sin(k*R[n])/(R[n]*R[n]);

      dx2rki[n]=(1.0-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k*cos(k*R[n])
                /(R[n]*R[n])+(3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0
                -k*k*(xo-xp[n])*(xo-xp[n]))*sin(k*R[n])/pow(R[n],3);

      dx2rkrx[n]=xp[n]*((3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0-k*k
                *(xo-xp[n])*(xo-xp[n]))*cos(k*R[n])/pow(R[n],3)
                -(1.-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k
                *sin(k*R[n])/(R[n]*R[n]));

      dx2rkix[n]=xp[n]*((1.0-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k*cos(k
                *R[n])/(R[n]*R[n])+(3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0
                -k*k*(xo-xp[n])*(xo-xp[n]))*sin(k*R[n])/pow(R[n],3));

      dx2rkry[n]=yp[n]*((3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0-k*k
```

```
                *(xo-xp[n])*(xo-xp[n]))*cos(k*R[n])/pow(R[n],3)
                -(1.-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k
                *sin(k*R[n])/(R[n]*R[n]));

        dx2rkiy[n]=yp[n]*((1.0-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k*cos(k
                *R[n])/(R[n]*R[n])+(3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0
                -k*k*(xo-xp[n])*(xo-xp[n]))*sin(k*R[n])/pow(R[n],3));



        dy2rkr[n]=(3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0-k*k*(yo-yp[n])
                *(yo-yp[n]))*cos(k*R[n])/pow(R[n],3)-(1.-3.*(yo-yp[n])*
                (yo-yp[n])/(R[n]*R[n]))*k*sin(k*R[n])/(R[n]*R[n]);

        dy2rki[n]=(1.0-3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n]))*k*cos(k*R[n])
                /(R[n]*R[n])+(3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0
                -k*k*(yo-yp[n])*(yo-yp[n]))*sin(k*R[n])/pow(R[n],3);

        dy2rkrx[n]=xp[n]*((3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0-k*k*(yo-
                yp[n])*(yo-yp[n]))*cos(k*R[n])/pow(R[n],3)-(1.-3.*(yo-yp[n])*
                (yo-yp[n])/(R[n]*R[n]))*k*sin(k*R[n])/(R[n]*R[n]));

        dy2rkix[n]=xp[n]*((1.0-3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n]))*k*cos(k
                *R[n])/(R[n]*R[n])+(3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0
                -k*k*(yo-yp[n])*(yo-yp[n]))*sin(k*R[n])/pow(R[n],3));

        dy2rkry[n]=yp[n]*((3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0-k*k*(yo-
                yp[n])*(yo-yp[n]))*cos(k*R[n])/pow(R[n],3)-(1.-3.*(yo-yp[n])*
                (yo-yp[n])/(R[n]*R[n]))*k*sin(k*R[n])/(R[n]*R[n]));

        dy2rkiy[n]=yp[n]*((1.0-3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n]))*k*cos(k
                *R[n])/(R[n]*R[n])+(3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0
                -k*k*(yo-yp[n])*(yo-yp[n]))*sin(k*R[n])/pow(R[n],3));




        dxyrkr[n]=((3.-k*k*R[n]*R[n])*cos(k*R[n])+3.*k*R[n]*sin(k*R[n]))
                *(xo-xp[n])*(yo-yp[n])/pow(R[n],5);

        dxyrki[n]=((3.-k*k*R[n]*R[n])*sin(k*R[n])-3.*k*R[n]*cos(k*R[n]))
                *(xo-xp[n])*(yo-yp[n])/pow(R[n],5);

        dxyrkrx[n]=xp[n]*(((3.-k*k*R[n]*R[n])*cos(k*R[n])+3.*k*R[n]*sin(k
                *R[n]))*(xo-xp[n])*(yo-yp[n])/pow(R[n],5));

        dxyrkix[n]=xp[n]*(((3.-k*k*R[n]*R[n])*sin(k*R[n])-3.*k*R[n]*cos(k
                *R[n]))*(xo-xp[n])*(yo-yp[n])/pow(R[n],5));

        dxyrkry[n]=yp[n]*(((3.-k*k*R[n]*R[n])*cos(k*R[n])+3.*k*R[n]*sin(k
                *R[n]))*(xo-xp[n])*(yo-yp[n])/pow(R[n],5));

        dxyrkiy[n]=yp[n]*(((3.-k*k*R[n]*R[n])*sin(k*R[n])-3.*k*R[n]*cos(k
                *R[n]))*(xo-xp[n])*(yo-yp[n])/pow(R[n],5));

        }
```

```
        Rk_re=(rkr[1]+rkr[2]+rkr[3])*area/3.0;

        Rk_im=(rki[1]+rki[2]+rki[3])*area/3.0;

        Rkx_re=(rkrx[1]+rkrx[2]+rkrx[3])*area/3.0;

        Rkx_im=(rkix[1]+rkix[2]+rkix[3])*area/3.0;

        Rky_re=(rkry[1]+rkry[2]+rkry[3])*area/3.0;

        Rky_im=(rkiy[1]+rkiy[2]+rkiy[3])*area/3.0;


        Dx2rk_re=(dx2rkr[1]+dx2rkr[2]+dx2rkr[3])*area/3.0;

        Dx2rk_im=(dx2rki[1]+dx2rki[2]+dx2rki[3])*area/3.0;

        Dx2rkx_re=(dx2rkrx[1]+dx2rkrx[2]+dx2rkrx[3])*area/3.0;

        Dx2rkx_im=(dx2rkix[1]+dx2rkix[2]+dx2rkix[3])*area/3.0;

        Dx2rky_re=(dx2rkry[1]+dx2rkry[2]+dx2rkry[3])*area/3.0;

        Dx2rky_im=(dx2rkiy[1]+dx2rkiy[2]+dx2rkiy[3])*area/3.0;


        Dy2rk_re=(dy2rkr[1]+dy2rkr[2]+dy2rkr[3])*area/3.0;

        Dy2rk_im=(dy2rki[1]+dy2rki[2]+dy2rki[3])*area/3.0;

        Dy2rkx_re=(dy2rkrx[1]+dy2rkrx[2]+dy2rkrx[3])*area/3.0;

        Dy2rkx_im=(dy2rkix[1]+dy2rkix[2]+dy2rkix[3])*area/3.0;

        Dy2rky_re=(dy2rkry[1]+dy2rkry[2]+dy2rkry[3])*area/3.0;

        Dy2rky_im=(dy2rkiy[1]+dy2rkiy[2]+dy2rkiy[3])*area/3.0;


        Dxyrk_re=(dxyrkr[1]+dxyrkr[2]+dxyrkr[3])*area/3.0;

        Dxyrk_im=(dxyrki[1]+dxyrki[2]+dxyrki[3])*area/3.0;

        Dxyrkx_re=(dxyrkrx[1]+dxyrkrx[2]+dxyrkrx[3])*area/3.0;

        Dxyrkx_im=(dxyrkix[1]+dxyrkix[2]+dxyrkix[3])*area/3.0;

        Dxyrky_re=(dxyrkry[1]+dxyrkry[2]+dxyrkry[3])*area/3.0;

        Dxyrky_im=(dxyrkiy[1]+dxyrkiy[2]+dxyrkiy[3])*area/3.0;

    }

/*************************************************************************/
/*                                                                       */
/*    If the field point and the centroid are not coincident use the     */
```

```
/*    numerical method described in Hammer for a quintic function           */
/*                                                                          */
/****************************************************************************/

else
  {
  r=(1.0+sqrt(15.0))/7.0;
  s=(1.0-sqrt(15.0))/7.0;

  a=(155.-sqrt(15.0))*area/1200.;
  b=(155.+sqrt(15.0))*area/1200.;
  c=9.0*area/40.0;

  xp[1]=ctrdx;
  yp[1]=ctrdy;

  for(n=1;n<=3;n++)
    {
    xp[n+1]=r*x[n]+(1.0-r)*ctrdx;
    yp[n+1]=r*y[n]+(1.0-r)*ctrdy;
    xp[n+4]=s*x[n]+(1.0-s)*ctrdx;
    yp[n+4]=s*y[n]+(1.0-s)*ctrdy;
    }

  for(n=1;n<=7;n++)
    R[n]=sqrt((xo-xp[n])*(xo-xp[n])+(yo-yp[n])*(yo-yp[n]));

  for(n=1;n<=7;n++)
    {


      rkr[n]=cos(k*R[n])/R[n]-1.0/R[n]+k*k*R[n]/2.0;
      rki[n]=sin(k*R[n])/R[n]-k;
      rkrx[n]=xp[n]*(cos(k*R[n])/R[n]-1.0/R[n]+k*k*R[n]/2.0);
      rkix[n]=xp[n]*(sin(k*R[n])/R[n]-k);
      rkry[n]=yp[n]*(cos(k*R[n])/R[n]-1.0/R[n]+k*k*R[n]/2.0);
      rkiy[n]=yp[n]*(sin(k*R[n])/R[n]-k);


      dx2rkr[n]=(3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0-k*k
               *(xo-xp[n])*(xo-xp[n]))*cos(k*R[n])/pow(R[n],3)
               -(1.-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k
               *sin(k*R[n])/(R[n]*R[n]);

      dx2rki[n]=(1.0-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k*cos(k*R[n])
               /(R[n]*R[n])+(3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0
               -k*k*(xo-xp[n])*(xo-xp[n]))*sin(k*R[n])/pow(R[n],3);

      dx2rkrx[n]=xp[n]*((3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0-k*k
               *(xo-xp[n])*(xo-xp[n]))*cos(k*R[n])/pow(R[n],3)
               -(1.-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k
               *sin(k*R[n])/(R[n]*R[n]));

      dx2rkix[n]=xp[n]*((1.0-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k*cos(k
               *R[n])/(R[n]*R[n])+(3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0
               -k*k*(xo-xp[n])*(xo-xp[n]))*sin(k*R[n])/pow(R[n],3));
```

```
	dx2rkry[n]=yp[n]*((3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0-k*k
		*(xo-xp[n])*(xo-xp[n]))*cos(k*R[n])/pow(R[n],3)
		-(1.-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k
		*sin(k*R[n])/(R[n]*R[n]));

	dx2rkiy[n]=yp[n]*((1.0-3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n]))*k*cos(k
		*R[n])/(R[n]*R[n])+(3.*(xo-xp[n])*(xo-xp[n])/(R[n]*R[n])-1.0
		-k*k*(xo-xp[n])*(xo-xp[n]))*sin(k*R[n])/pow(R[n],3));



	dy2rkr[n]=(3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0-k*k*(yo-yp[n])
		*(yo-yp[n]))*cos(k*R[n])/pow(R[n],3)-(1.-3.*(yo-yp[n])*
		(yo-yp[n])/(R[n]*R[n]))*k*sin(k*R[n])/(R[n]*R[n]);

	dy2rki[n]=(1.0-3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n]))*k*cos(k*R[n])
		/(R[n]*R[n])+(3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0
		-k*k*(yo-yp[n])*(yo-yp[n]))*sin(k*R[n])/pow(R[n],3);

	dy2rkrx[n]=xp[n]*((3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0-k*k*(yo-
		yp[n])*(yo-yp[n]))*cos(k*R[n])/pow(R[n],3)-(1.-3.*(yo-yp[n])*
		(yo-yp[n])/(R[n]*R[n]))*k*sin(k*R[n])/(R[n]*R[n]));

	dy2rkix[n]=xp[n]*((1.0-3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n]))*k*cos(k
		*R[n])/(R[n]*R[n])+(3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0
		-k*k*(yo-yp[n])*(yo-yp[n]))*sin(k*R[n])/pow(R[n],3));

	dy2rkry[n]=yp[n]*((3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0-k*k*(yo-
		yp[n])*(yo-yp[n]))*cos(k*R[n])/pow(R[n],3)-(1.-3.*(yo-yp[n])*
		(yo-yp[n])/(R[n]*R[n]))*k*sin(k*R[n])/(R[n]*R[n]));

	dy2rkiy[n]=yp[n]*((1.0-3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n]))*k*cos(k
		*R[n])/(R[n]*R[n])+(3.*(yo-yp[n])*(yo-yp[n])/(R[n]*R[n])-1.0
		-k*k*(yo-yp[n])*(yo-yp[n]))*sin(k*R[n])/pow(R[n],3));




	dxyrkr[n]=((3.-k*k*R[n]*R[n])*cos(k*R[n])+3.*k*R[n]*sin(k*R[n]))
		*(xo-xp[n])*(yo-yp[n])/pow(R[n],5);

	dxyrki[n]=((3.-k*k*R[n]*R[n])*sin(k*R[n])-3.*k*R[n]*cos(k*R[n]))
		*(xo-xp[n])*(yo-yp[n])/pow(R[n],5);

	dxyrkrx[n]=xp[n]*(((3.-k*k*R[n]*R[n])*cos(k*R[n])+3.*k*R[n]*sin(k
		*R[n]))*(xo-xp[n])*(yo-yp[n])/pow(R[n],5));

	dxyrkix[n]=xp[n]*(((3.-k*k*R[n]*R[n])*sin(k*R[n])-3.*k*R[n]*cos(k
		*R[n]))*(xo-xp[n])*(yo-yp[n])/pow(R[n],5));

	dxyrkry[n]=yp[n]*(((3.-k*k*R[n]*R[n])*cos(k*R[n])+3.*k*R[n]*sin(k
		*R[n]))*(xo-xp[n])*(yo-yp[n])/pow(R[n],5));

	dxyrkiy[n]=yp[n]*(((3.-k*k*R[n]*R[n])*sin(k*R[n])-3.*k*R[n]*cos(k
		*R[n]))*(xo-xp[n])*(yo-yp[n])/pow(R[n],5));
```

```
    }

Rk_re=c*rkr[1]+a*(rkr[2]+rkr[3]+rkr[4])+b*(rkr[5]+rkr[6]+rkr[7]);
Rk_im=c*rki[1]+a*(rki[2]+rki[3]+rki[4])+b*(rki[5]+rki[6]+rki[7]);
Rkx_re=c*rkrx[1]+a*(rkrx[2]+rkrx[3]+rkrx[4])+b*(rkrx[5]+rkrx[6]+rkrx[7]);
Rkx_im=c*rkix[1]+a*(rkix[2]+rkix[3]+rkix[4])+b*(rkix[5]+rkix[6]+rkix[7]);
Rky_re=c*rkry[1]+a*(rkry[2]+rkry[3]+rkry[4])+b*(rkry[5]+rkry[6]+rkry[7]);


Dx2rk_re=c*dx2rkr[1]+a*(dx2rkr[2]+dx2rkr[3]+dx2rkr[4])+b
        *(dx2rkr[5]+dx2rkr[6]+dx2rkr[7]);

Dx2rk_im=c*dx2rki[1]+a*(dx2rki[2]+dx2rki[3]+dx2rki[4])+b
        *(dx2rki[5]+dx2rki[6]+dx2rki[7]);

Dx2rkx_re=c*dx2rkrx[1]+a*(dx2rkrx[2]+dx2rkrx[3]+dx2rkrx[4])+b
        *(dx2rkrx[5]+dx2rkrx[6]+dx2rkrx[7]);

Dx2rkx_im=c*dx2rkix[1]+a*(dx2rkix[2]+dx2rkix[3]+dx2rkix[4])+b
        *(dx2rkix[5]+dx2rkix[6]+dx2rkix[7]);

Dx2rky_re=c*dx2rkry[1]+a*(dx2rkry[2]+dx2rkry[3]+dx2rkry[4])+b
        *(dx2rkry[5]+dx2rkry[6]+dx2rkry[7]);

Dx2rky_im=c*dx2rkiy[1]+a*(dx2rkiy[2]+dx2rkiy[3]+dx2rkiy[4])+b
        *(dx2rkiy[5]+dx2rkiy[6]+dx2rkiy[7]);



Dy2rk_re=c*dy2rkr[1]+a*(dy2rkr[2]+dy2rkr[3]+dy2rkr[4])+b
        *(dy2rkr[5]+dy2rkr[6]+dy2rkr[7]);

Dy2rk_im=c*dy2rki[1]+a*(dy2rki[2]+dy2rki[3]+dy2rki[4])+b
        *(dy2rki[5]+dy2rki[6]+dy2rki[7]);

Dy2rkx_re=c*dy2rkrx[1]+a*(dy2rkrx[2]+dy2rkrx[3]+dy2rkrx[4])+b
        *(dy2rkrx[5]+dy2rkrx[6]+dy2rkrx[7]);

Dy2rkx_im=c*dy2rkix[1]+a*(dy2rkix[2]+dy2rkix[3]+dy2rkix[4])+b
        *(dy2rkix[5]+dy2rkix[6]+dy2rkix[7]);

Dy2rky_re=c*dy2rkry[1]+a*(dy2rkry[2]+dy2rkry[3]+dy2rkry[4])+b
        *(dy2rkry[5]+dy2rkry[6]+dy2rkry[7]);

Dy2rky_im=c*dy2rkiy[1]+a*(dy2rkiy[2]+dy2rkiy[3]+dy2rkiy[4])+b
        *(dy2rkiy[5]+dy2rkiy[6]+dy2rkiy[7]);



Dxyrk_re=c*dxyrkr[1]+a*(dxyrkr[2]+dxyrkr[3]+dxyrkr[4])+b
        *(dxyrkr[5]+dxyrkr[6]+dxyrkr[7]);

Dxyrk_im=c*dxyrki[1]+a*(dxyrki[2]+dxyrki[3]+dxyrki[4])+b
        *(dxyrki[5]+dxyrki[6]+dxyrki[7]);
Dxyrkx_re=c*dxyrkrx[1]+a*(dxyrkrx[2]+dxyrkrx[3]+dxyrkrx[4])+b
```

```
            *(dxyrkrx[5]+dxyrkrx[6]+dxyrkrx[7]);

    Dxyrkx_im=c*dxyrkix[1]+a*(dxyrkix[2]+dxyrkix[3]+dxyrkix[4])+b
            *(dxyrkix[5]+dxyrkix[6]+dxyrkix[7]);

    Dxyrky_re=c*dxyrkry[1]+a*(dxyrkry[2]+dxyrkry[3]+dxyrkry[4])+b
            *(dxyrkry[5]+dxyrkry[6]+dxyrkry[7]);

    Dxyrky_im=c*dxyrkiy[1]+a*(dxyrkiy[2]+dxyrkiy[3]+dxyrkiy[4])+b
            *(dxyrkiy[5]+dxyrkiy[6]+dxyrkiy[7]);
    }
}
C NAASA  2.1.042 CGECO    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
      SUBROUTINE CGECO(A,LDA,N,IPVT,RCOND,Z)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LDA,N,IPVT(1)
      COMPLEX*16 A(LDA,1),Z(1)
      REAL*8 RCOND
C
C     CGECO FACTORS A COMPLEX MATRIX BY GAUSSIAN ELIMINATION
C     AND ESTIMATES THE CONDITION OF THE MATRIX.
C
C     IF  RCOND  IS NOT NEEDED, CGEFA IS SLIGHTLY FASTER.
C     TO SOLVE  A*X = B , FOLLOW CGECO BY CGESL.
C     TO COMPUTE  INVERSE(A)*C , FOLLOW CGECO BY CGESL.
C     TO COMPUTE  DETERMINANT(A) , FOLLOW CGECO BY CGEDI.
C     TO COMPUTE  INVERSE(A) , FOLLOW CGECO BY CGEDI.
C
C     ON ENTRY
C
C        A       COMPLEX(LDA, N)
C                THE MATRIX TO BE FACTORED.
C
C        LDA     INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C     ON RETURN
C
C        A       AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
C                WHICH WERE USED TO OBTAIN IT.
C                THE FACTORIZATION CAN BE WRITTEN  A = L*U  WHERE
C                L  IS A PRODUCT OF PERMUTATION AND UNIT LOWER
C                TRIANGULAR MATRICES AND  U  IS UPPER TRIANGULAR.
C
C        IPVT    INTEGER(N)
C                AN INTEGER VECTOR OF PIVOT INDICES.
C
C        RCOND   REAL
C                AN ESTIMATE OF THE RECIPROCAL CONDITION OF  A .
C                FOR THE SYSTEM  A*X = B , RELATIVE PERTURBATIONS
C                IN  A  AND  B  OF SIZE  EPSILON  MAY CAUSE
C                RELATIVE PERTURBATIONS IN  X  OF SIZE  EPSILON/RCOND .
C                IF  RCOND  IS SO SMALL THAT THE LOGICAL EXPRESSION
```

```
C                      1.0 + RCOND .EQ. 1.0
C               IS TRUE, THEN  A  MAY BE SINGULAR TO WORKING
C               PRECISION.  IN PARTICULAR,  RCOND  IS ZERO  IF
C               EXACT SINGULARITY IS DETECTED OR THE ESTIMATE
C               UNDERFLOWS.
C
C        Z      COMPLEX(N)
C               A WORK VECTOR WHOSE CONTENTS ARE USUALLY UNIMPORTANT.
C               IF  A  IS CLOSE TO A SINGULAR MATRIX, THEN  Z  IS
C               AN APPROXIMATE NULL VECTOR IN THE SENSE THAT
C               NORM(A*Z) = RCOND*NORM(A)*NORM(Z) .
C
C     LINPACK. THIS VERSION DATED 07/14/77 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C     SUBROUTINES AND FUNCTIONS
C
C     LINPACK CGEFA
C     BLAS CAXPY,CDOTC,CSSCAL,SCASUM
C     FORTRAN DABS,DIMAG,DMAX1,DCMPLX,DCONJG,REAL
C
C     INTERNAL VARIABLES
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 CDOTC,EK,T,WK,WKM
      REAL*8 ANORM,S,SCASUM,SM,YNORM
      INTEGER INFO,J,K,KB,KP1,L
C
      COMPLEX*16 ZDUM,ZDUM1,ZDUM2,CSIGN1
      REAL*8 CABS1
      CABS1(ZDUM) = DABS(DREAL(ZDUM)) + DABS(DIMAG(ZDUM))
      CSIGN1(ZDUM1,ZDUM2) = CABS1(ZDUM1)*(ZDUM2/CABS1(ZDUM2))
C
C     COMPUTE 1-NORM OF A
C
      ANORM = 0.0D0
      DO 10 J = 1, N
         ANORM = DMAX1(ANORM,SCASUM(N,A(1,J),1))
   10 CONTINUE
C
C     FACTOR
C
      CALL CGEFA(A,LDA,N,IPVT,INFO)
C
C     RCOND = 1/(NORM(A)*(ESTIMATE OF NORM(INVERSE(A)))) .
C     ESTIMATE = NORM(Z)/NORM(Y) WHERE  A*Z = Y  AND  CTRANS(A)*Y = E .
C     CTRANS(A)  IS THE CONJUGATE TRANSPOSE OF A .
C     THE COMPONENTS OF  E  ARE CHOSEN TO CAUSE MAXIMUM LOCAL
C     GROWTH IN THE ELEMENTS OF W  WHERE  CTRANS(U)*W = E .
C     THE VECTORS ARE FREQUENTLY RESCALED TO AVOID OVERFLOW.
C
C     SOLVE CTRANS(U)*W = E
C
      EK = DCMPLX(1.0D0,0.0D0)
      DO 20 J = 1, N
         Z(J) = DCMPLX(0.0D0,0.0D0)
```

```
   20 CONTINUE
      DO 100 K = 1, N
         IF (CABS1(Z(K)) .NE. 0.0D0) EK = CSIGN1(EK,-Z(K))
         IF (CABS1(EK-Z(K)) .LE. CABS1(A(K,K))) GO TO 30
            S = CABS1(A(K,K))/CABS1(EK-Z(K))
            CALL CSSCAL(N,S,Z,1)
            EK = DCMPLX(S,0.0D0)*EK
   30    CONTINUE
         WK = EK - Z(K)
         WKM = -EK - Z(K)
         S = CABS1(WK)
         SM = CABS1(WKM)
         IF (CABS1(A(K,K)) .EQ. 0.0D0) GO TO 40
            WK = WK/DCONJG(A(K,K))
            WKM = WKM/DCONJG(A(K,K))
         GO TO 50
   40    CONTINUE
            WK = DCMPLX(1.0D0,0.0D0)
            WKM = DCMPLX(1.0D0,0.0D0)
   50    CONTINUE
         KP1 = K + 1
         IF (KP1 .GT. N) GO TO 90
            DO 60 J = KP1, N
               SM = SM + CABS1(Z(J)+WKM*DCONJG(A(K,J)))
               Z(J) = Z(J) + WK*DCONJG(A(K,J))
               S = S + CABS1(Z(J))
   60       CONTINUE
            IF (S .GE. SM) GO TO 80
               T = WKM - WK
               WK = WKM
               DO 70 J = KP1, N
                  Z(J) = Z(J) + T*DCONJG(A(K,J))
   70          CONTINUE
   80       CONTINUE
   90    CONTINUE
         Z(K) = WK
  100 CONTINUE
      S = 1.0D0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
C
C     SOLVE CTRANS(L)*Y = V
C
      DO 120 KB = 1, N
         K = N + 1 - KB
         IF (K .LT. N) Z(K) = Z(K) + CDOTC(N-K,A(K+1,K),1,Z(K+1),1)
         IF (CABS1(Z(K)) .LE. 1.0D0) GO TO 110
            S = 1.0D0/CABS1(Z(K))
            CALL CSSCAL(N,S,Z,1)
  110    CONTINUE
         L = IPVT(K)
         T = Z(L)
         Z(L) = Z(K)
         Z(K) = T
  120 CONTINUE
      S = 1.0D0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
```

```
C
      YNORM = 1.0D0
C
C     SOLVE L*V = Y
C
      DO 140 K = 1, N
         L = IPVT(K)
         T = Z(L)
         Z(L) = Z(K)
         Z(K) = T
         IF (K .LT. N) CALL CAXPY(N-K,T,A(K+1,K),1,Z(K+1),1)
         IF (CABS1(Z(K)) .LE. 1.0D0) GO TO 130
            S = 1.0D0/CABS1(Z(K))
            CALL CSSCAL(N,S,Z,1)
            YNORM = S*YNORM
  130    CONTINUE
  140 CONTINUE
      S = 1.0D0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
      YNORM = S*YNORM
C
C     SOLVE  U*Z = V
C
      DO 160 KB = 1, N
         K = N + 1 - KB
         IF (CABS1(Z(K)) .LE. CABS1(A(K,K))) GO TO 150
            S = CABS1(A(K,K))/CABS1(Z(K))
            CALL CSSCAL(N,S,Z,1)
            YNORM = S*YNORM
  150    CONTINUE
         IF (CABS1(A(K,K)) .NE. 0.0D0) Z(K) = Z(K)/A(K,K)
         IF (CABS1(A(K,K)) .EQ. 0.0D0) Z(K) = DCMPLX(1.0D0,0.0D0)
         T = -Z(K)
         CALL CAXPY(K-1,T,A(1,K),1,Z(1),1)
  160 CONTINUE
C     MAKE ZNORM = 1.0
      S = 1.0D0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
      YNORM = S*YNORM
C
      IF (ANORM .NE. 0.0D0) RCOND = YNORM/ANORM
      IF (ANORM .EQ. 0.0D0) RCOND = 0.0D0
      RETURN
      END
C NAASA  2.1.044 CGESL     FTN-A 05-02-78      THE UNIV OF MICH COMP CTR
      SUBROUTINE CGESL(A,LDA,N,IPVT,B,JOB)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LDA,N,IPVT(1),JOB
      COMPLEX*16 A(LDA,1),B(1)
C
C     CGESL SOLVES THE COMPLEX SYSTEM
C     A * X = B  OR  CTRANS(A) * X = B
C     USING THE FACTORS COMPUTED BY CGECO OR CGEFA.
C
C     ON ENTRY
C
```

```
C        A       COMPLEX(LDA, N)
C                THE OUTPUT FROM CGECO OR CGEFA.
C
C        LDA     INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C        IPVT    INTEGER(N)
C                THE PIVOT VECTOR FROM CGECO OR CGEFA.
C
C        B       COMPLEX(N)
C                THE RIGHT HAND SIDE VECTOR.
C
C        JOB     INTEGER
C                = 0         TO SOLVE  A*X = B ,
C                = NONZERO   TO SOLVE  CTRANS(A)*X = B  WHERE
C                            CTRANS(A)  IS THE CONJUGATE TRANSPOSE.
C
C     ON RETURN
C
C        B       THE SOLUTION VECTOR  X .
C
C     ERROR CONDITION
C
C        A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS A
C        ZERO ON THE DIAGONAL.  TECHNICALLY THIS INDICATES SINGULARITY
C        BUT IT IS OFTEN CAUSED BY IMPROPER ARGUMENTS OR IMPROPER
C        SETTING OF LDA .  IT WILL NOT OCCUR IF THE SUBROUTINES ARE
C        CALLED CORRECTLY AND IF CGECO HAS SET RCOND .GT. 0.0
C        OR CGEFA HAS SET INFO .EQ. 0 .
C
C     TO COMPUTE  INVERSE(A) * C  WHERE  C  IS A MATRIX
C     WITH  P  COLUMNS
C           CALL CGECO(A,LDA,N,IPVT,RCOND,Z)
C           IF (RCOND IS TOO SMALL) GO TO ...
C           DO 10 J = 1, P
C              CALL CGESL(A,LDA,N,IPVT,C(1,J),0)
C        10 CONTINUE
C
C     LINPACK. THIS VERSION DATED 07/14/77 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS CAXPY,CDOTC
C     FORTRAN DCONJG
C
C     INTERNAL VARIABLES
C
      COMPLEX*16 CDOTC,T
      INTEGER K,KB,L,NM1
C
      NM1 = N - 1
      IF (JOB .NE. 0) GO TO 50
```

```
C
C         JOB = 0 , SOLVE  A * X = B
C         FIRST SOLVE  L*Y = B
C
          IF (NM1 .LT. 1) GO TO 30
          DO 20 K = 1, NM1
             L = IPVT(K)
             T = B(L)
             IF (L .EQ. K) GO TO 10
                B(L) = B(K)
                B(K) = T
   10        CONTINUE
             CALL CAXPY(N-K,T,A(K+1,K),1,B(K+1),1)
   20     CONTINUE
   30     CONTINUE
C
C         NOW SOLVE  U*X = Y
C
          DO 40 KB = 1, N
             K = N + 1 - KB
             B(K) = B(K)/A(K,K)
             T = -B(K)
             CALL CAXPY(K-1,T,A(1,K),1,B(1),1)
   40     CONTINUE
       GO TO 100
   50 CONTINUE
C
C         JOB = NONZERO, SOLVE  CTRANS(A) * X = B
C         FIRST SOLVE  CTRANS(U)*Y = B
C
          DO 60 K = 1, N
             T = CDOTC(K-1,A(1,K),1,B(1),1)
             B(K) = (B(K) - T)/DCONJG(A(K,K))
   60     CONTINUE
C
C         NOW SOLVE CTRANS(L)*X = Y
C
          IF (NM1 .LT. 1) GO TO 90
          DO 80 KB = 1, NM1
             K = N - KB
             B(K) = B(K) + CDOTC(N-K,A(K+1,K),1,B(K+1),1)
             L = IPVT(K)
             IF (L .EQ. K) GO TO 70
                T = B(L)
                B(L) = B(K)
                B(K) = T
   70        CONTINUE
   80     CONTINUE
   90     CONTINUE
  100 CONTINUE
      RETURN
      END
C NAASA  2.1.043 CGEFA    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
      SUBROUTINE CGEFA(A,LDA,N,IPVT,INFO)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LDA,N,IPVT(1),INFO
```

```
      COMPLEX*16 A(LDA,1)
C
C     CGEFA FACTORS A COMPLEX MATRIX BY GAUSSIAN ELIMINATION.
C
C     CGEFA IS USUALLY CALLED BY CGECO, BUT IT CAN BE CALLED
C     DIRECTLY WITH A SAVING IN TIME IF  RCOND  IS NOT NEEDED.
C     (TIME FOR CGECO) = (1 + 9/N)*(TIME FOR CGEFA) .
C
C     ON ENTRY
C
C        A       COMPLEX(LDA, N)
C                THE MATRIX TO BE FACTORED.
C
C        LDA     INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C     ON RETURN
C
C        A       AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
C                WHICH WERE USED TO OBTAIN IT.
C                THE FACTORIZATION CAN BE WRITTEN  A = L*U  WHERE
C                L  IS A PRODUCT OF PERMUTATION AND UNIT LOWER
C                TRIANGULAR MATRICES AND  U  IS UPPER TRIANGULAR.
C
C        IPVT    INTEGER(N)
C                AN INTEGER VECTOR OF PIVOT INDICES.
C
C        INFO    INTEGER
C                = 0  NORMAL VALUE.
C                = K  IF  U(K,K) .EQ. 0.0 .  THIS IS NOT AN ERROR
C                     CONDITION FOR THIS SUBROUTINE, BUT IT DOES
C                     INDICATE THAT CGESL OR CGEDI WILL DIVIDE BY ZERO
C                     IF CALLED.  USE  RCOND  IN CGECO FOR A RELIABLE
C                     INDICATION OF SINGULARITY.
C
C     LINPACK. THIS VERSION DATED 07/14/77 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS CAXPY,CSCAL,ICAMAX
C     FORTRAN DABS,DIMAG,DCMPLX,DREAL
C
C     INTERNAL VARIABLES
C
      COMPLEX*16 T
      INTEGER ICAMAX,J,K,KP1,L,NM1
C
      COMPLEX*16 ZDUM
      REAL*8 CABS1
      CABS1(ZDUM) = DABS(DREAL(ZDUM)) + DABS(DIMAG(ZDUM))
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
```

```
C
      INFO = 0
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 70
      DO 60 K = 1, NM1
         KP1 = K + 1
C
C        FIND L = PIVOT INDEX
C
         L = ICAMAX(N-K+1,A(K,K),1) + K - 1
         IPVT(K) = L
C
C        ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
         IF (CABS1(A(L,K)) .EQ. 0.0D0) GO TO 40
C
C           INTERCHANGE IF NECESSARY
C
            IF (L .EQ. K) GO TO 10
               T = A(L,K)
               A(L,K) = A(K,K)
               A(K,K) = T
   10       CONTINUE
C
C           COMPUTE MULTIPLIERS
C
            T = -DCMPLX(1.0D0,0.0D0)/A(K,K)
            CALL CSCAL(N-K,T,A(K+1,K),1)
C
C           ROW ELIMINATION WITH COLUMN INDEXING
C
            DO 30 J = KP1, N
               T = A(L,J)
               IF (L .EQ. K) GO TO 20
                  A(L,J) = A(K,J)
                  A(K,J) = T
   20          CONTINUE
               CALL CAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
   30       CONTINUE
         GO TO 50
   40    CONTINUE
            INFO = K
   50    CONTINUE
   60 CONTINUE
   70 CONTINUE
      IPVT(N) = N
      IF (CABS1(A(N,N)) .EQ. 0.0D0) INFO = N
      RETURN
      END
C NAASA  1.1.014 CAXPY     FTN-A 05-02-78      THE UNIV OF MICH COMP CTR
      SUBROUTINE CAXPY(N,CA,CX,INCX,CY,INCY)
C
C     CONSTANT TIMES A VECTOR PLUS A VECTOR.
C     JACK DONGARRA, LINPACK, 6/17/77.
C
      IMPLICIT REAL*8(A-H,O-Z)
```

```
      COMPLEX*16 CX(1),CY(1),CA
      INTEGER I,INCX,INCY,IX,IY,N
C
      IF(N.LE.0)RETURN
      IF (DABS(DREAL(CA)) + DABS(DIMAG(CA)) .EQ. 0.0D0 ) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GOTO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        CY(IY) = CY(IY) + CA*CX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
   20 DO 30 I = 1,N
        CY(I) = CY(I) + CA*CX(I)
   30 CONTINUE
      RETURN
      END
C NAASA  1.1.012 CDOTC     FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
      COMPLEX*16 FUNCTION CDOTC(N,CX,INCX,CY,INCY)
C
C     FORMS THE DOT PRODUCT OF TWO VECTORS, CONJUGATING THE FIRST
C     VECTOR.
C     JACK DONGARRA, LINPACK,  6/17/77.
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 CX(1),CY(1),CTEMP
      INTEGER I,INCX,INCY,IX,IY,N
C
      CTEMP = DCMPLX(0.0D0,0.0D0)
      CDOTC = DCMPLX(0.0D0,0.0D0)
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GOTO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        CTEMP = CTEMP + DCONJG(CX(IX))*CY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
```

```
      CDOTC = CTEMP
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
   20 DO 30 I = 1,N
        CTEMP = CTEMP + DCONJG(CX(I))*CY(I)
   30 CONTINUE
      CDOTC = CTEMP
      RETURN
      END
C NAASA  1.1.018 CSSCAL   FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
      SUBROUTINE  CSSCAL(N,SA,CX,INCX)
C
C     SCALES A COMPLEX VECTOR BY A REAL CONSTANT.
C     JACK DONGARRA, LINPACK, 6/17/77.
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 CX(1)
      REAL*8 SA
      INTEGER I,INCX,N,NINCX
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        CX(I) = DCMPLX(SA*DREAL(CX(I)),SA*DIMAG(CX(I)))
   10 CONTINUE
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
   20 DO 30 I = 1,N
        CX(I) = DCMPLX(SA*DREAL(CX(I)),SA*DIMAG(CX(I)))
   30 CONTINUE
      RETURN
      END
C NAASA  1.1.010 SCASUM   FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
      REAL*8 FUNCTION SCASUM(N,CX,INCX)
C
C     TAKES THE SUM OF THE ABSOLUTE VALUES OF A COMPLEX VECTOR AND
C     RETURNS A SINGLE PRECISION RESULT.
C     JACK DONGARRA, LINPACK, 6/17/77.
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 CX(1)
      REAL*8 STEMP
      INTEGER I,INCX,N,NINCX
C
      SCASUM = 0.0D0
      STEMP = 0.0D0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
```

```
C
C         CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        STEMP = STEMP + DABS(DREAL(CX(I))) + DABS(DIMAG(CX(I)))
   10 CONTINUE
      SCASUM = STEMP
      RETURN
C
C         CODE FOR INCREMENT EQUAL TO 1
C
   20 DO 30 I = 1,N
        STEMP = STEMP + DABS(DREAL(CX(I))) + DABS(DIMAG(CX(I)))
   30 CONTINUE
      SCASUM = STEMP
      RETURN
      END
C NAASA  1.1.019 CSCAL     FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
      SUBROUTINE  CSCAL(N,CA,CX,INCX)
C
C     SCALES A VECTOR BY A CONSTANT.
C     JACK DONGARRA, LINPACK,  6/17/77.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 CA,CX(1)
      INTEGER I,INCX,N,NINCX
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        CX(I) = CA*CX(I)
   10 CONTINUE
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
   20 DO 30 I = 1,N
        CX(I) = CA*CX(I)
   30 CONTINUE
      RETURN
      END
C NAASA  1.1.021 ICAMAX    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
      INTEGER FUNCTION ICAMAX(N,CX,INCX)
C
C     FINDS THE INDEX OF ELEMENT HAVING MAX. ABSOLUTE VALUE.
C     JACK DONGARRA, LINPACK, 6/17/77.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 CX(1)
      REAL*8 SMAX
      INTEGER I,INCX,IX,N
```

```
      COMPLEX*16 ZDUM
      REAL*8 CABS1
      CABS1(ZDUM) = DABS(DREAL(ZDUM)) + DABS(DIMAG(ZDUM))
C
      ICAMAX = 1
      IF(N.LE.1)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      IX = 1
      SMAX = CABS1(CX(1))
      IX = IX + INCX
      DO 10 I = 2,N
         IF(CABS1(CX(IX)).LE.SMAX) GO TO 5
         ICAMAX = I
         SMAX = CABS1(CX(IX))
    5    IX = IX + INCX
   10 CONTINUE
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
   20 SMAX = CABS1(CX(1))
      DO 30 I = 2,N
         IF(CABS1(CX(I)).LE.SMAX) GO TO 30
         ICAMAX = I
         SMAX = CABS1(CX(I))
   30 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCC
C     DREAL DOESN'T SEEM TO WORK, SO THIS FUNCTION IS A SUBSTITUTE
      REAL*8 FUNCTION DREAL(X)
      COMPLEX*16 X,X2
      REAL*8 XA(2)
      EQUIVALENCE (X2,XA(1))
      X2=X
      DREAL=XA(1)
      RETURN
      END
```