ST-358

LA-5280

211
12-12-73

Polynomial Solutions of the Schrödinger Equation Applied to Photon Cross Sections in Atoms

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

In the interest of prompt distribution, this LAMS report was not edited by the Technical Information staff.

Printed in the United States of America. Available from
National Technical Information Service
U. S. *Department of Commerce*
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy $4.00; Microfiche $0.95

LA-5280
UC-32
ISSUED: August 1973

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

Polynomial Solutions of the Schrödinger Equation Applied to Photon Cross Sections in Atoms

by

A. L. Merts
Walter Matuska, Jr.

POLYNOMIAL SOLUTIONS OF THE SCHRÖDINGER EQUATION APPLIED TO PHOTON CROSS SECTIONS IN ATOMS

by

A. L. Merts and Walter Matuska, Jr.

ABSTRACT

Solutions of the Schrödinger equation with a realistic potential are carried out in detail. To check our methods, we have calculated a few bound-bound, bound-free, and free-free cross sections and compared our values with existing calculations and experimental data. These comparisons, along with a listing of the computer code in its bound-free form, are included.

I. INTRODUCTION

This report intends to show how the CDC-7600 code DEGA-A (Dense Electron Gas Approximation-Absorption) computes the bound-free, free-free, and bound-bound absorption coefficients, σ_{bf}, σ_{ff}, and σ_{bb}, respectively, as a function of photon energy, $h\upsilon$, from the given potential function V(r) and the electron occupancy of the atom. Assuming V(r) to be exact, the code will do its computation at any desired accuracy, within the machine limits, because the Schrödinger equation is solved using exact power-series expansions, not difference equations.

As the code is now written, any potential function can be used if it adheres to the three following requirements.

1. $\lim_{r \to 0} V(r) r^2 = 0$.

2. After some finite value of r, R_1, the potential must be $V(r) = -k/r$.

3. V(r) must be a negative, monotonically, increasing function.

Although it is not necessary to the internal structure of the code, we would also like V(r) to be a function of temperature, density, and the atomic number of the atom.

Throughout this report, $[0,R_1]$ is called Region 1 and $[R_1,\infty]$ Region 2.

The following V(r) is the one most often used in the code.

Experience has shown that a parameterized potential works well and gives energy levels close to those that Herman and Skillman[1] calculated using their Hartree-Fock-Slater method for isolated neutral atoms.

The potential form for an isolated atom is

$$V(r) = Ze/r(1 + \alpha r)^2 \quad . \tag{1}$$

where $\alpha = 0.6057Z^{1/3}$. However, the form actually used in this report, which is also valid for a compressed atom, is

$$V(r) = -(Z^*e)/R_1 \left(\frac{R_1}{r} + \frac{r^2}{2R_1^2} - 3/2\right) \tag{2}$$

for $r_0 \leq r \leq R_1$.

where

Z^* is the "effective number" of free electrons,

R_1 is the radius of the sphere representing the spherical atomic volume,

r_0 is the value of the radial distance for which Eq. (2) is equal to the value given in Eq. (3) below, and

e is the electronic charge.

In the inner region, the potential is given by

$$V(r) = -Ze/r(1 + \alpha r)^2 - Z^*e/r\left(\frac{r^2}{2R_1^2} - Bo\right) \tag{3}$$

for $0 \leq r \leq r_0$.

$$\mathrm{Bo} = \left(\frac{Z}{Z^*}\right)\left(\frac{R_1}{r_0}\right)/(1+\alpha r_0)^2 - \frac{R_1}{r_0} + 3/2 \quad . \tag{4}$$

$$Z^* = Z/(1+\alpha r_0)^2 \left[\left(\frac{2\alpha r_0}{1+\alpha r_0}\right) \cdot + 1\right] \quad . \tag{5}$$

After the form of the potential in the two regions is chosen, Eqs. (4) and (5) follow from the potential's continuity and its relation to charge density through Poisson's equation. In the potential outlined above, the value of R_1 is determined by the density of the material being considered. The value of Z^* is chosen by an iterative procedure so that at some finite temperature T, and for an atom occupying a spherical volume of radius R_1, we have

$$Z = N(Z^*) = \int_0^{R_1} \left[\int_0^{\infty} n(r, T, P)\, dP \right] 4\pi r^2\, dr \quad , \tag{6}$$

where n is the Fermi-Dirac distribution function, representing the number of electrons at point r, having momentum between P and P + dP. Using Eq. (5) we have also determined r_0.

We must correct the above potential for self-interaction. This is done in the simplest possible fashion. We replace Z in Eqs. (2) through (5) with $(Z-1)$ and add the term $-1/r$ to Eqs. (2) and (3). The potential for $r > R_1$ is defined as $-1/r$. This then represents our potential function used to calculate the one-electron energies and the one-electron orbitals from which we calculate the cross sections and f-values.

This potential can be shown to satisfy the three previously stated conditions.

The hydrogenic potential $V(r) = -1/r$ for $0 \leqslant r \leqslant \infty$ is often used to check parts of the code because analytic solutions for this potential are known.

The code considers the Schrödinger equation in the form

$$\varphi''(r) + \left[\lambda - 2V(r) - \frac{\ell(\ell+1)}{r^2}\right]\varphi(r) = 0,\ 0 \leqslant r \leqslant \infty \quad , \tag{7}$$

where

$r\varphi(r)$ is the radial wave function (however, $\varphi(r)$ will be called the wave function throughout the rest of this report),

λ is the energy eigenvalue, and

ℓ is the angular-momentum quantum number.

Equation (7) has an infinite (in some cases finite, but large) number of discrete bound solutions (negative λ) commonly denoted by 1s, 2s, 2p, 3s, 3p, 3d, etc. Of this infinite sequence, we calculate only the solutions allowed by the electron occupancy. σ_{bb} can be evaluated only at discrete values of $h\nu$, because $h\nu = |\lambda_1 - \lambda_2|$ where λ_1 and λ_2 both represent bound solutions. By using a line profile, we then distribute each σ_{bb} over a narrow range of $h\nu$. However, every positive λ, given ℓ, is an eigenvalue for a free state. This allows us to choose any finite number of positive λ's. Therefore, we can evaluate σ_{bf} at any energy $h\nu$ above the threshold, and σ_{ff} at any desired value of $h\nu$. Here $h\nu = |\lambda_1 - \lambda_2|$, λ_1 represents either a bound or a free solution, and λ_2 represents a free solution.

II. THE METHOD OF SOLUTION OF THE SCHRÖDINGER EQUATION

A. The Potential Approximation

To solve Eq. (7) on the computer, we choose a finite $R > R_1$ to approximate $r = \infty$, and divide $[0,R]$ into a finite number of intervals. For each of these intervals, r^2 times the potential is approximated by a parabola to some specified degree of accuracy, ξ. This series of fits is started at $r = 0$ with an interval of arbitrary length. In this first interval, we approximate the potential with

$$V(r) \approx c_1 + \frac{c_2}{r} + \frac{c_3}{r^2} \quad . \tag{8}$$

Let r_1 be the left end point of this interval, r_3 the right end point, and r_2 the midpoint. To evaluate the c's, we solve the set of equations

$$c_1 r_j^2 + c_2 r_j + c_3 = V(r_j) r_j^2 \ \text{for}\ j = 1, 2, 3. \tag{9}$$

These equations have no difficulty at $r = 0$, and they fit the potential exactly at r_1, r_2, and r_3. Because V(r) is a smooth, monotonic function, we can check our fit by checking the validity of

$$\left|c_1 + \frac{c_2}{r} + \frac{c_3}{r^2} - V(r)\right| < |V(r)|\xi \tag{10}$$

at several points between r_1 and r_3. If this inequality is not satisfied for all points, we decrease the length of the interval until inequality [Eq. (10)] is satisfied for all points checked. This process is continued until Region 1 is complete. Here c_1, c_2, and c_3 should actually be thought of as $c_{1,i}$, $c_{2,i}$, and $c_{3,i}$, where the subscript i denotes the ith interval.

In Region 2, obviously, $c_2 = -1$ and $c_1 = c_3 = 0$. The lengths of the intervals in this region are governed only by an additional condition discussed later. This condition applies to all intervals.

As the code now exists, for a given set of conditions, all solutions to the Schrödinger equation are found by using the same set of intervals over Region 1; however, each solution has its own set of intervals in Region 2. The code could also be written so that each solution would have its own set of intervals over the entire range of r. It would be impractical to require all solutions to have the same set of intervals in Region 2 because the maximum interval lengths allowed for the various solutions in Region 2 are so different.

B. The Expansion of the Radial Wave Function

In the expansion of the wave function, we will need the condition $c_{3,1} = 0$. This is satisfied if the first requirement on the potential is met.

We assume the wave function to have the form

$$\varphi(r) = \sum_{j=1}^{\infty} a_{ji} r^{j-1} \text{ for } i = 1 \tag{11}$$

in the first interval. Dropping the subscript i, we substitute the power series and its second derivative for $\varphi(r)$ and $\varphi''(r)$ and the approximation for V(r) in Eq. (8) into the Schrödinger equation to get

$$\begin{aligned}&\left\{-\left[2c_3 + \ell(\ell+1)\right]a_1\right\}/r^2\\ &+\left\{-2c_2a_1 - \left[2c_3 + \ell(\ell+1)\right]a_2\right\}/r\\ &+\sum_{j=3}^{\infty}\left\{(\lambda - 2c_1)a_{j-2} - 2c_2a_{j-1}\right.\\ &\left.+\left[(j-2)(j-1) - 2c_3 - \ell(\ell+1)\right]a_j\right\}r^{j-3} = 0 \quad .\end{aligned} \tag{12}$$

Here we assume that $c_3 = 0$ and $a_j = 0$ for $1 \leq j \leq \ell + 1$. Also $a_{\ell+2}$ is arbitrary because any constant times a solution eigenfunction $\varphi(r)$ is still a solution. As each term in Eq. (12) must be zero for every r in the interval, we now have the recursion relation

$$a_j = \frac{2c_2a_{j-1} - (\lambda - 2c_1)a_{j-2}}{(j-2)(j-1) - \ell(\ell+1)} \quad \text{for } j > \ell + 2 \quad . \tag{13}$$

The power-series expansions in all other intervals are also expanded about one of the interval's end points. If expanding about the left end point, we make the substitution $d = r - \rho_i$ and require that $0 \leq d \leq \rho_{i+1} - \rho_i$ where the ρ's are the end points of the intervals. When it is necessary to expand about the right end point, the roles of ρ_i and ρ_{i+1} are switched and $\rho_i - \rho_{i+1} \leq d \leq 0$. Again omitting the subscript i, the Schrödinger equation can now be written as

$$(\rho^2 + 2\rho d + d^2)\varphi''(d) + (B_1 + B_2 d + B_3 d^2)\varphi(d) = 0 \quad ,$$

where

$$B_1 = b_1\rho^2 + b_2\rho + b_3$$

$$B_2 = 2b_1\rho + b_2$$

$$B_3 = b_1$$

and (14)

$$b_1 = \lambda - 2c_1$$

$$b_2 = -2c_2$$

$$b_3 = -\left[2c_3 + \ell(\ell+1)\right] \quad .$$

Here we assume the wave function to have the form

$$\varphi(d) = \sum_{j=1}^{\infty} a_{ji} d^{j-1} \quad \text{for } i > 1 \quad . \tag{15}$$

Substituting the power series and its second derivative for $\varphi(d)$ and $\varphi''(d)$, we get

$$\begin{aligned}&\left[2\rho^2a_3 + B_1a_1\right] + \left[6\rho^2a_4 + 4\rho a_3 + B_1a_2 + B_2a_1\right]d\\ &+\sum_{j=5}^{\infty}\left[(j-2)(j-1)\rho^2a_j + (j-3)(j-2)2\rho a_{j-1}\right.\\ &+(j-4)(j-3)a_{j-2} + B_1a_{j-2}\\ &\left.+B_2a_{j-3} + B_3a_{j-4}\right]d^{j-3} = 0\end{aligned} \tag{16}$$

with the subscript i omitted. Again, because each term of Eq. (16) must be zero for all values of d, we have the recursion relations

$$a_3 = -B_1 a_1 / 2\rho^2$$

$$a_4 = -[B_2 a_1 + B_1 a_2 + 4\rho a_3]/6\rho^2$$

$$a_j = -\Big\{B_3 a_{j-4} + B_2 a_{j-3} + [B_1 + (j-4)(j-3)]a_{j-2} + 2(j-3)(j-2)\rho a_{j-1}\Big\}/(j-2)(j-1)\rho^2$$

$$\text{for } j > 4 \tag{17}$$

when a_1 and a_2 are known.

We can easily show that $a_1 = \varphi(d)$ and $a_2 = \varphi'(d)$ at $d = 0$. Therefore, a_1 and a_2 will always be known by the right-boundary condition or can be evaluated with the power-series expansion of the wave function in the previous interval.

In practice, we start with $a_j = 0$ for $1 \leq j \leq \ell + 1$ and $a_{\ell+2}$ an arbitrary nonzero constant to start the power-series expansion in the first interval. Then we evaluate this power-series and its first derivative at its right end point. This determines a_1 and a_2 for the second interval. This process is repeated until we reach some $\hat{r}$, $0 < \hat{r} < R$, where we evaluate $\varphi_f(\hat{r})$ and $\varphi_f'(\hat{r})$; the determination of $\hat{r}$ is discussed later. At the right boundary, R, $\varphi(R)$, and $\varphi'(R)$ are given by the boundary condition. This enables us to start our successive power-series expansions at R and work our way backward to $\hat{r}$ where we evaluate $\varphi_b(\hat{r})$ and $\varphi_b'(\hat{r})$. We have a solution when $\varphi_f'(\hat{r}) = \varphi_b'(\hat{r})$ for $\varphi_f(\hat{r}) = \varphi_b(\hat{r})$. $\varphi_f(\hat{r})$ is simply the value of the wave function at $\hat{r}$ found by forward expansions, and $\varphi_b(\hat{r})$ is the value of the same wave function at $\hat{r}$ found by backward expansions.

C. Determination of the Expansion Interval Length

In real life, these power series can have only a finite number of terms. The maximum number of terms in the present code is 50. This leads to the other condition regulating the maximum length of a particular interval. Let us define

$$\widetilde{V}(r) = \left[\lambda - 2V(r) - \frac{\ell(\ell+1)}{r^2}\right] \tag{18}$$

Using this definition, we can write the Schrödinger equation as

$$\varphi''(r) \pm h^2\varphi(r) = 0, \text{ where } h^2 = \pm\widetilde{V}(r) \tag{19}$$

choosing the sign so that $h^2 > 0$. If h is assumed to be constant locally, the solution is either

$$\varphi(r) = Ae^{-hr} + Be^{hr} \tag{20}$$

or

$$\varphi(r) = A \sin hr + B \cos hr, \tag{21}$$

depending upon the sign. Expanding the decreasing exponential solution in a Taylor series about an end point of the interval, we get

$$\varphi(r) = Ae^{-h\rho_i}\sum_{j=0}^{\infty} (-1)^j \frac{(h\Delta r)^j}{j!} \tag{22}$$

where $\Delta r = r - \rho_i$ is the maximum allowable interval length. With this series, we can estimate the largest value of $(h\Delta r)$ that allows the series to converge to a specified accuracy in the alloted number of terms. In Region 1, all solutions are considered in estimating the maximum interval length; however, in Region 2 only one solution at a time is considered. The code then plugs the largest encountered value of h into

$$\Delta r = \frac{C}{h} . \tag{23}$$

This C is an input parameter, and for a 50-term expansion and 8-place accuracy a conservative value for C is 6.

The sinusoidal and increasing exponential solutions have a series similar to Eq. (22), and the interval length is also estimated by Eq. (23).

D. Finding the Join Point $\hat{r}$ for Forward and Backward Integration

Let us consider the question of stability. Here again, Eqs. (20) and (21) can be thought of as local solutions to the Schrödinger equation. If an error is introduced in Eq. (20) during forward integration (increasing r) when the first term is the desired solution, this error will grow exponentially. However, any error introduced will diminish exponentially during backward integration. Likewise, when the second term is the desired solution, backward integration is unstable and forward integration causes any error to diminish exponentially. Equation (21) is considered stable for integration in either direction. Any error introduced into the integration will not grow exponentially; however, once an error is introduced it cannot be diminished as was the case with Eq. (20). Figure 1 shows the regions of stable and unstable integration for the four possible combinations of given conditions.

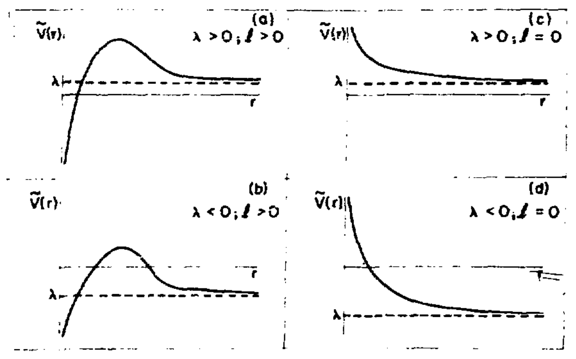

Fig. 1.
Integration is stable in both directions for $\tilde V(r) > 0$; integration is stable in only one direction for $\tilde V(r) < 0$. λ indicates the asymtotic limit of $\tilde V(r)$.

For the cases shown in Figs. 1a and 1b, $\hat r$ is chosen such that $\tilde V(\hat r)$ is the maximum value of $\tilde V(r)$. For the case shown in Fig. 1c, $\hat r$ is the center of the rightmost interval, and for the case shown in Fig. 1d, $\hat r$ is chosen such that $\tilde V(\hat r) = 0$. For positive λ, the value of λ is known and is used to find $\hat r$; however, for negative λ one has only a maximum and minimum guess at its value at this stage, so the minimum guess is used in evaluating $\hat r$.

E. Boundary Condition at R

1. Bound States–Negative λ. The two right-boundary conditions used in the code are discussed below; however, this method of solving the Schrödinger equation is not limited to these two conditions. For the bound state, the Schrödinger equation becomes

$$\varphi''(r) - |\lambda|\varphi(r) = 0 \text{ as } r \to \infty \quad . \tag{24}$$

Note that this is true only in Region 2. Because $\varphi(R)$ is arbitrary and

$$\varphi(r) = Ae^{-\sqrt{|\lambda|}\, r} \tag{25}$$

is the desired solution to Eq. (24), we derive the boundary condition

$$\varphi'(R) = -\sqrt{|\lambda|}\;\varphi(R) \quad . \tag{26}$$

Although we have no exact relation between R and the accuracy desired, we regulate R by the following method. First, we find the largest root of $\tilde V(r)$. R is then taken as some constant (typically 25) times this root. Then, after the Schrödinger equation is solved using this R, we check the validity of

$$|\varphi(r)|_{\max} < \gamma|\varphi(R)| \tag{27}$$

(10^{-15} is a typical value for γ). If this condition is not satisfied, we simply keep increasing R until it is. A quick method for getting a value near $|\varphi(r)|_{\max}$ is to check $\varphi(r)$ at the end points of the intervals.

2. Free States–Positive λ. The sinusoidal condition given by Carson, Mayers, and Stibbs[2] is used on the right boundary in the free case. This boundary condition also applies only in Region 2.

For large r, the solution may be written

$$\varphi(r) = M(r) \cos\left[kr + \delta(r)\right] \tag{28}$$

in which $k = \sqrt{\lambda}$,

$$\lim_{r\to\infty} M(r) = M, \quad \text{and} \quad \lim_{r\to\infty} \frac{\delta(r)}{r} = 0 \quad . \tag{29}$$

M fixes the scale of the solution. We shall require the normalization $M = \sqrt{2/\pi k}$.

In Region 2 where $V(r) = -c/r$ for $c > 0$, we assume a solution of the form

$$\varphi(r) = A(r)\cos\left(kr + \frac{c}{k}\ln r\right) + B(r)\sin\left(kr + \frac{c}{k}\ln r\right) \quad . \tag{30}$$

Now substitute Eq. (30) into the Schrödinger equation. Then asymptotic expansions for A(r) and B(r) exist in the form

$$A(r) = \sum a_n/r^n,\; B(r) = \sum b_n/r^n \quad , \tag{31}$$

where

$$a_{n+1} = \left\{\left[\ell(\ell+1) + \frac{c^2}{k^2} - n(n+1)\right] b_n - \frac{c}{k}(2n+1)a_n\right\} / 2k(n+1)$$

$$b_{n+1} = \left\{\left[n(n+1) - \frac{c^2}{k^2} - \ell(\ell+1)\right] a_n - \frac{c}{k}(2n+1)b_n\right\} / 2k(n+1) \tag{32}$$

and

$$a_o = M, b_o = 0 \quad . \qquad (33)$$

The sums in Eq. (31) are only semiconvergent. Therefore A(r) and B(r) must be evaluated at an r large enough that the sums converge to the desired accuracy.

Equation (30) can also be written as

$$\varphi(r) = M(r) \cos \theta(r) \quad , \qquad (34)$$

where

$$M(r) = \sqrt{A(r)^2 + B(r)^2} \qquad (35)$$

and

$$\theta(r) = kr + \frac{c}{k} + \ln r + \tan^{-1} \left[\frac{A(r)}{B(r)}\right]$$

The asymptotic series expansion for the solution of $\varphi(r)$ given by Eq. (34) is uniquely determined up to some constant phase α_ℓ; therefore, we can write

$$\varphi(r) = M(r) \cos [\theta(r) + \alpha_\ell] \quad , \qquad (36)$$

instead of Eq. (34), and we also have

$$\varphi'(r) = M'(r) \cos [\theta(r) + \alpha_\ell] - M(r)\theta'(r) \sin [\theta(r) + \alpha_\ell] \quad . \qquad (37)$$

For sufficiently large R, $\varphi(R)$ and $\varphi'(R)$ are the desired right-boundary conditions. The guesses at R may have to be increased several times before Eq. (31) is satisfied.

The phase α_ℓ is determined by an iteration process that is explained later.

III. DETERMINATION OF THE EIGENVALUES AND PHASE FACTORS

A. The Eigenvalue

As stated previously, for the bound state, λ is varied until we find a $[\varphi(\lambda,r),\lambda]$ that solves the Schrödinger equation. We start with a minimum and a maximum guess at λ. This difference in λ is divided into a specified number of logarithmically equal λ-intervals. We define

$$F(\lambda) = \frac{\varphi_b(\lambda,\hat{r})}{\varphi_f(\lambda,\hat{r})} \varphi_f'(\lambda,\hat{r}) - \varphi_b'(\lambda,\hat{r}) \qquad (38)$$

at the end points of these λ-intervals and look for a sign change in $F(\lambda)$ in each. If we detect a sign change in an interval, we use the Regula Falsa[3] method to find the root of $F(\lambda)$ in that interval. As $F(\lambda)$ also changes sign through poles, we check to reject these intervals.

When we find a $[\varphi(\lambda,r),\lambda]$ solution, we must determine whether it is the desired one for specified quantum numbers n and ℓ, where $\ell \leqslant n-1$. We have the desired solution when

$$I = n - \ell - 1 \quad , \qquad (39)$$

where I is the number of roots in $\varphi(r)$. I is found using a Sturm Sequence.[3] If Eq. (39) is not satisfied, we reject this solution and continue our search. The value of λ so obtained is called the eigenvalue.

B. Phase Determination

In the free state, λ is specified and the phase α_ℓ at the right boundary is varied over the range $0 \leqslant \alpha_\ell \leqslant \pi$ until we find a $[\varphi(\alpha_\ell,r),\alpha_\ell]$ that solves the Schrödinger equation. To find this solution, we again use the Regula Falsa method to find the root of $F(\alpha_\ell)$ where

$$F(\alpha_\ell) = \frac{\varphi_b(\alpha_\ell,\hat{r})}{\varphi_f(\alpha_\ell,\hat{r})} \varphi_f'(\alpha_\ell,\hat{r}) - \varphi_b'(\alpha_\ell,\hat{r}) \quad . \qquad (40)$$

Here, we need not check for poles or unwanted solutions as was necessary for the bound states.

C. Normalization

Even though the wave functions need not be normalized to be solutions to the Schrödinger equation, they must be normalized to produce correct absorption coefficients. We define $\varphi(r)$ as normalized when, for bound states:

$$\int_0^\infty \varphi(r)\varphi(r)\, dr = 1 \quad , \qquad (41)$$

and, for free states:

$$\int_0^\infty \varphi_\lambda \varphi_\lambda' \, dr = \delta(\lambda - \lambda') \quad . \qquad (42)$$

The free states have this normality built into the right-boundary condition. However, for each bound state, we evaluate

$$\int_0^{\infty} \varphi(r)\varphi(r)\, dr = \beta^2 \quad , \tag{43}$$

and the normalized wave function is then $\varphi(r)/\beta$. Equation (43) is evaluated by polynomial multiplication and integration over each interval of r.

IV. EVALUATION OF MATRIX ELEMENTS

Now that we have found all of the necessary solutions to the Schrödinger equation, we compute the matrix elements, H_{mn}, used in evaluating the absorption coefficients. The matrix elements are found by

$$H_{mn} = 2\int_0^{\infty} \varphi_m(r) \frac{\partial V(r)}{\partial r} \varphi_n(r)\, dr \quad , \tag{44}$$

where m and n refer to eigen solutions of the Schrödinger equation. If one wave function is bound and the other is free, H_{mn} is used in evaluating σ_{bf}. Likewise, if both are bound, the result is σ_{bb}, and if both are free, the result is σ_{ff}.

Equation (44) is evaluated by summing the integrals calculated in each expansion interval. Because $\partial V(r)/\partial r$ decreases as r^{-2} and $\varphi(r)$ decreases exponentially for bound states, this integral converges rapidly to a specified accuracy at some finite value of r, provided at least one $\varphi(r)$ represents a bound state. However, if both wave functions represent free states, the convergence is much slower, because each bound wave function asymptotically approaches a constant amplitude. Here, convergence is achieved only through the r^{-2} decrease in $\partial V(r)/\partial r$.

In the first interval we have

$$\frac{\partial V(r)}{\partial r} = -c_{2,1} r^{-2} \quad , \tag{45}$$

multiplied by the polynomial representations of $\varphi_m(r)$ and $\varphi_n(r)$ that can be integrated easily. In the rest of the intervals, we have

$$\frac{\partial V(d)}{\partial d} = -\frac{c_{2,i}(\rho_i + d) + 2c_{3,i}}{(\rho_i + d)^3} \quad , \tag{46}$$

which, by continued long division, can be written as a converging power series in d when $|d/\rho_i| < 1$. Now, multiplying these three polynomials, we get a polynomial that can be integrated easily.

V. EXAMPLES

A. Bound-Free Absorptions

In the code, λ, r, V(r), and H_{mn} are dimensionless parameters. Both λ and V(r), when multiplied by one Rydberg, are energies expressed in Rydbergs, and r, when multiplied by one Bohr radius, is a length expressed in Bohr radii.

The bound-free absorption coefficient is given by

$$\sigma_{bf}(h\nu) = 10.756 \times 10^6 \sum \frac{\ell_{max} H_{mn}^2 \eta}{|\lambda_m - \lambda_n|^3\, g} \quad , \tag{47}$$

where

ℓ_{max} is the maximum of ℓ_m and ℓ_n.

η is the number of electrons in the bound state that can make this transition.

g is the maximum possible degeneracy given by $2(2\ell + 1)$ for the ℓ of the bound state.

Here λ_m, λ_n, and H_{mn} are dimensionless numbers and σ_{bf} is expressed in barns per atom. Also when $h\nu = 13.605\, |\lambda_m - \lambda_n|$, $h\nu$ is expressed in electron volts. The sum in Eq. (47) is over all possible bound-free transitions at the given $h\nu$.

Figures 2 through 7 show examples of bound-free absorptions for cold, normal-density beryllium, carbon, aluminum, iron, copper, and lead as computed by DEGA-A. In these figures, the continuous line was computed by DEGA-A and the X's are experimental data given by Storm and Israel.[4] In Fig. 5, the three experimental points at the m edge for iron were given by Carter and Givens.[5] In Fig. 4, DEGA-A was compared with calculations by Barfield, Koontz, and Huebner[6] for aluminum at low photon energies.

Even though the bound-free cross sections have been computed down to the lowest edge in these examples, we make no claims about the accuracy of the copper and lead cross sections at these low photon energies.

None of the cross sections in this report include electron spin or relativistic effect.

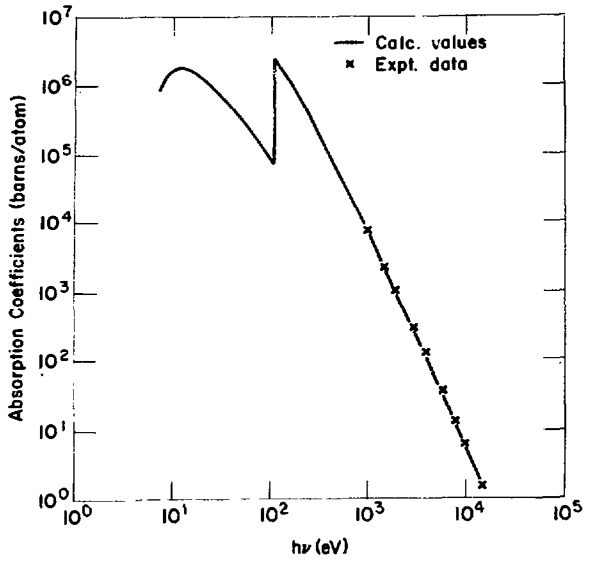

Fig. 2.
Bound-free absorption coefficients for cold, normal density (1.845 g/cm^3) Be. (Z = 4, r_0 = 1.406, R_1 = 2.355)

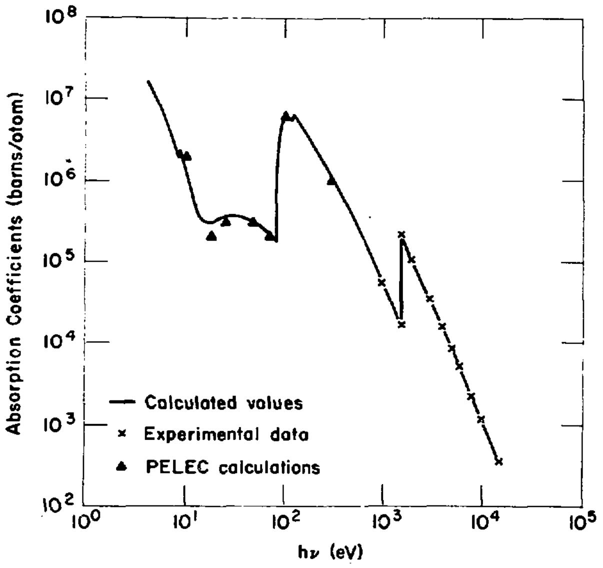

Fig. 4.
Bound-free absorption coefficients for cold, normal density (2.699 g/cm^3) Al. (Z = 13, r_0 = 1.699, R_1 = 2.990)

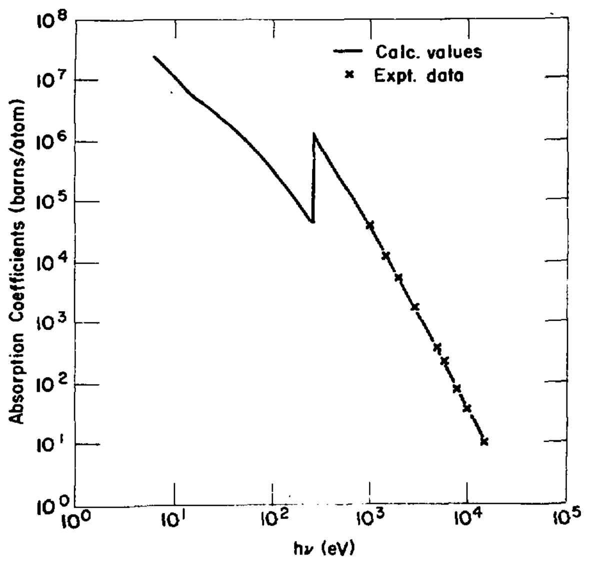

Fig. 3.
Bound-free absorption coefficients for cold, normal density (2.25 g/cm^3) C. (Z = 6, r_0 = 1.442, R_1 = 2.426)

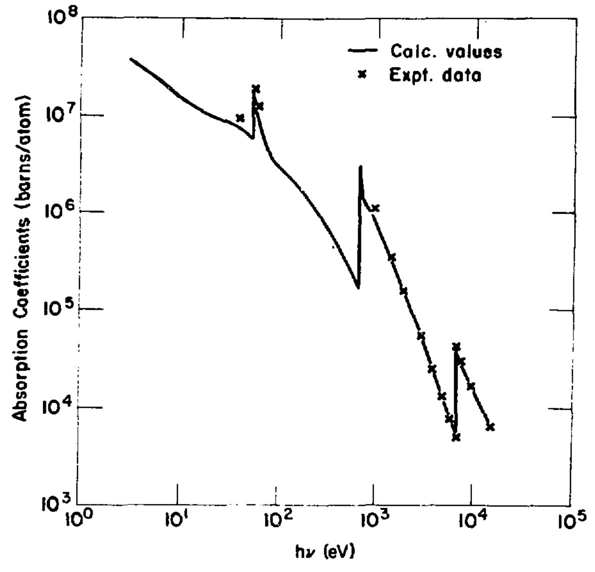

Fig. 5.
Bound-free absorption coefficients for cold, normal density (7.85 g/cm^3) Fe. (Z = 26, r_0 = 1.512, R_1 = 2.670)

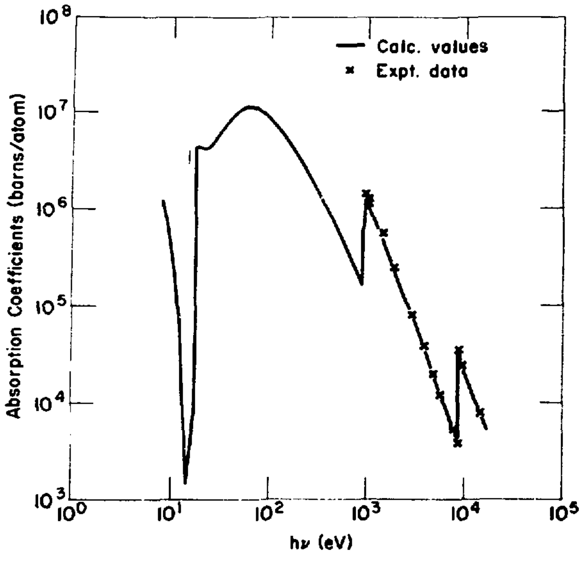

Fig. 6.
Bound-free absorption coefficients for cold, normal density (8.89 g/cm^3) Cu. (Z = 29, r_0 = 1.517, R_1 = 2.674)

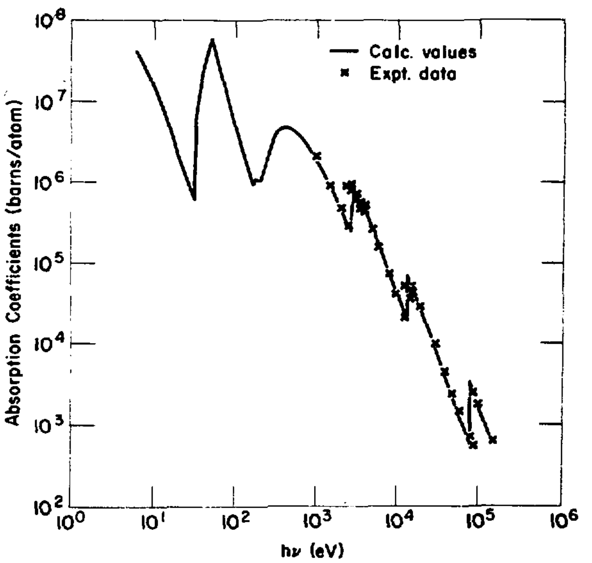

Fig. 7.
Bound-free absorption coefficients for cold, normal density (11.34 g/cm^3) Pb. (Z = 82, r_0 = 2.003, R_1 = 3.656)

B. Bound-Bound Absorptions

In checking the bound-bound case, we consider hydrogen, lithium, beryllium, sodium, potassium, and strontium. A formula similar to Eq. (47) can be derived for $\sigma_{bb}(h\nu)$.[2] However, because the data found in the literature[7] appeared in a different form, we used the formula

$$\tau = \frac{4H_{mn}^2}{|\lambda_m - \lambda_n|^4 \, (4\ell_{max}^2 - 1)} \tag{48}$$

to make our numbers directly comparable. See Table I.

Squares of dipole moments for hydrogen were computed by

$$M^2 = \frac{4H_{mn}^2}{|\lambda_m - \lambda_n|^4} \quad . \tag{49}$$

These results checked with the values of the squares of dipole moments* given by Bethe and Salpeter.[8] Note that Eq. (49) is not defined by the method presented in this report when $\lambda_m = \lambda_n$.

To check the code to more digits than given by Bethe and Salpeter (the CDC 7600 is a 14-digit machine), $H_{1s,2p}$ was computed to the maximum possible accuracy using the hydrogenic potential and was compared with its analytic solution. We know that for hydrogen

$$\varphi_{1s} = 2re^{-r}$$

and (50)

$$\varphi_{2p} = \frac{1}{2\sqrt{6}} r^2 e^{-r/2}$$

When these analytic expressions for φ are used in Eq. (44) along with $V(r) = -1/r$, we obtain

$$H_{1s,2p} = \frac{8}{9\sqrt{6}} = 0.36288736930121$$

and DEGA-A computed

$$H_{1s,2p} = 0.362887369301 18 \quad .$$

*Compare with Table 13, page 264, Bethe and Salpeter.[8]

TABLE I

**TRANSITION INTEGRAL τ
FOR COLD, NORMAL DENSITY ELEMENTS**

		Transition Integral τ			
Element	Transition	Screened-Hydrogenic	Coulomb Approx	Self-Cons Field	DEGA-A
Li I[a]	2p-2s	5.96	5.42	5.5-5.6	4.26
	3p-2s	0.011	0.016	0.011-.020	0.071
	3s-2p	1.72	2.39		1.75
	4s-2p	0.105	0.177		0.148
	5s-2p	0.029	0.056		0.044
	6s-2p	0.013	0.025		0.020
	3d-2p	1.28	1.14		0.748
	4d-2p	0.19	0.18		0.148
Be I[b]	$2s2p$-$2s^2$	2.27	2.03	1.86	3.29
Na I[c]	3p-3s	3.41	6.0	6.7	3.63
	4p-3s	0.211	0.047	0.051	0.026
	4s-3p	2.06	6.09	6.2	3.60
K I[d]	4p-4s	4.61	8.05	9.05	7.77
Sr I[e]	5d-5p	0.483	0.42		1.39

[a]$Z=3$, $r_0=1.928$, and $R_1=3.263$.

[b]$Z=4$, $r_0=1.406$, and $R_1=2.355$.

[c]$Z=11$, $r_0=2.264$, and $R_1=3.986$.

[d]$Z=19$, $r_0=2.714$, and $R_1=4.950$.

[e]$Z=38$, $r_0=2.475$, and $R_1=4.518$.

C. Free-Free Absorptions

1. Discussion of Gaunt Factors. A formula similar to Eq. (47)[2] can be derived for σ_{ff} (hν) in terms of Gaunt factors g_{ff}; however, here we only compute Gaunt factors for hydrogen and sodium and compare some of these results with Karzas and Latter.[9] Gaunt factors can be evaluated with

$$g_{ff}(\lambda_a,h\nu) = \frac{\pi\sqrt{3}}{8(Z^{eff})^2} \sum_{\ell=0}^{L'} \left[(\ell+1)H_{\ell+1,\ell}(\lambda_a,h\nu)^2 + \ell H_{\ell-1,\ell}(\lambda_a,h\nu)^2\right] \quad (51)$$

when the sum converges rapidly enough to be practical. Here λ_a is the initial energy of the electron, $h\nu$ is the photon energy, Z^{eff} is the effective number of free electrons in the atom under consideration, and L' is a finite integer approximation to infinity. The matrix elements are still defined by Eq. (44), but we rewrite the equation as

$$H_{mn}(\lambda_a,h\nu) = 2\int_0^\infty \varphi_m(\lambda_a,r)\,\frac{\partial V(r)}{\partial r}\,\varphi_n(\lambda_a + h\nu,r)\,dr \quad (52)$$

to define the association between energy levels and quantum numbers.

The sum in Eq. (51) is not always converging rapidly. Results for other than a coulomb potential can often be obtained in these cases by making use of the formula[10]

$$g_{ff}(\lambda_a,h\nu) = g^c_{ff}(\lambda_a,h\nu)$$
$$+ \frac{\pi\sqrt{3}}{8} \sum_{\ell=0}^{L} \left\{(\ell+1)\left[H_{\ell+1,\ell}(\lambda_a,h\nu)^2 - H^c_{\ell+1,\ell}(\lambda_a,h\nu)^2\right]\right.$$
$$\left. + \ell\left[H_{\ell-1,\ell}(\lambda_a,h\nu)^2 - H^c_{\ell-1,\ell}(\lambda_a,h\nu)^2\right]\right\} , \quad (53)$$

where g^c_{ff} is the coulombic Gaunt factor for the initial energy and photon energy under consideration, H^c_{mn} is a coulombic matrix element defined by Eq. (52) using $V(r) = -1/r$, and L is an integer sufficiently large so that the sum has converged to a predetermined accuracy. A graph of $g^c_{ff}(\lambda_a,h\upsilon)$ is given by Karzas and Latter who circumvented the slow convergence problem in Eq. (51) by using hypergeometric functions. Applications of this procedure are limited to the coulomb potential.

The method for obtaining a noncoulombic $\sigma_{ff}(h\upsilon)$, as described in this report, never requires the evaluation of a $Z^{eff}(\lambda_a,h\upsilon)$. In terms of coulombic Gaunt factors, $g^c_{ff}(\lambda_a,h\upsilon)$, $Z^{eff}(\lambda_a,h\upsilon)$ must also be evaluated and then $Z^{eff}(\lambda_a,h\upsilon)\, g^c_{ff}(\lambda_a,h\upsilon)$ must be used as the desired Gaunt factor. However, Eqs. (51) and (53) give the desired noncoulombic $g_{ff}(\lambda_a,h\upsilon)$ for $Z^{eff} = 1$ when the matrix elements are computed by DEGA-A using a realistic potential.

2. Checks on Method. For the first check on the code, we computed several Gaunt factors with Eq. (51) using the coulomb potential, $V(r) = -1/r$, and compared these results with the Karzas and Latter graph. Here comparisons were made up to, at most, three significant figures, which is the maximum accuracy for reading the graph. Table II shows that for several cases we were able to reproduce the Karzas and Latter values. In the remaining cases, Eq. (51) had not converged for $\ell = 34$, the maximum value of ℓ used in these calculations.

Cold, normal-density sodium ($Z = 11$, $r_0 = 2.264$, $R_1 = 3.986$) was arbitrarily used in the next check (Fig. 8). Here, for low photon energies and low initial energies, the Gaunt factors using the sodium potential and $Z^{eff} = 1$ should approach the coulombic Gaunt factors. Also, for large photon energies and large initial energies, the Gaunt factors using the sodium potential and $Z^{eff} = 11$ should approach the coulombic Gaunt factors. In both cases, the sodium Gaunt factor for $(\lambda_a,h\upsilon)$ is compared with the coulombic Gaunt factor for $(\lambda_a/(Z^{eff})^2, h\upsilon/(Z^{eff})^2)$.

3. Discussion of Screening. The general screening effects are noted for the small electron kinetic energy and the small photon energy shown in Fig. 8. As the photon energy is held fixed and the electron energy is increased, the calculated Gaunt factor increases faster than would be expected from the coulombic case. The qualitative reason for this follows. As the electron energy is increased, the electron wave function samples in greater and greater detail the structure of the atom (the shielding), and as a result the effective charge Z^{eff} increases with increasing energy. The result is that, relative to a coulomb potential, the cross section is raised. However, as the energy of the electron and the photon increase, we eventually arrive at a

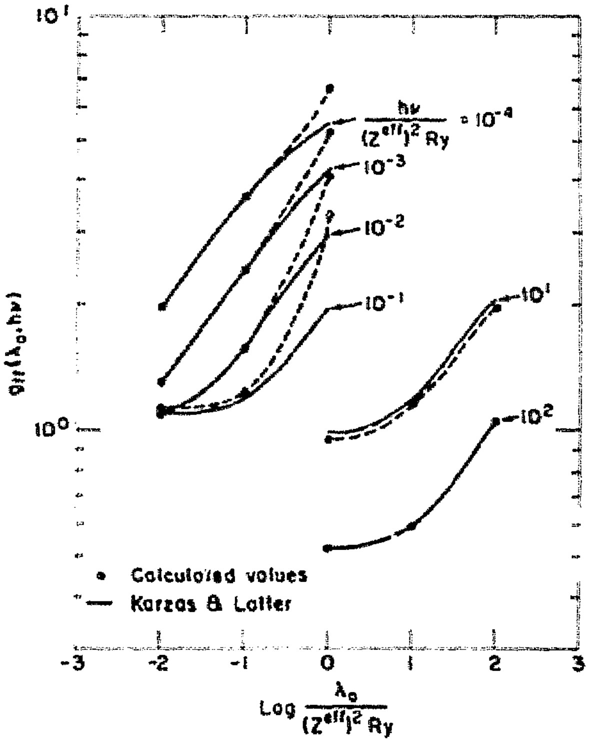

Fig. 8.
A comparison of coulombic and shielded potential (NaI) Gaunt factors.

point where the wave function of the electron oscillates sufficiently rapidly that it is essentially seeing the coulomb field of the nucleus. The effective charge is then the nuclear charge and we have agreement with the coulomb field results. This corresponds to the lower right-hand curves of Fig. 8.

4. Numerical Results. The purpose of Table II is to illustrate a trend in the convergence rate for each of Eqs. (51) and (53) as a function of $(\lambda_a,h\upsilon)$. An integer in parentheses indicates the value of ℓ for which the series was terminated. For example, where $(\lambda_a,h\upsilon) = (1.0, 0.01)$, Eq. (51) has summed to 3.07 by 34 terms, whereas Eq. (53) has summed to within 2% of the expected (converged) value of 4.10 by three terms. Here convergence means no change in the sum to the accuracy given in Table II. Also, for the coulombic case at this energy pair, Eq. (51) has not converged for $\ell = 34$. For the (1.0, 10) entry, both Eqs. (51) and (53) have converged by $\ell = 3$ for the sodium case, and Eq. (51) has converged by $\ell = 3$ in the coulombic case.

Table II also shows that neither Eq. (51) nor (53) has converged for the entries labeled with footnote b. The use

TABLE II

CALCULATIONS OF COLD, NORMAL DENSITY SODIUM AND COULOMBIC GAUNT FACTORS[a]

λ_a(Ry)	$h\nu$(Ry)	$g_{ff}^{c}(\lambda_a,h\nu)$ from Karzas and Latter	$g_{ff}^{c}(\lambda_a,h\nu)$ Calculated Using Eq. (51)	$g_{ff}(\lambda_a,h\nu)$ for Na Calculated Using Eq. (51)	$g_{ff}(\lambda_a,h\nu)$ for Na Calculated Using Eq. (53)		
0.01	0.0001	1.97	0.83(34)	0.83(34)	1.97(3)	1.97(34)	
	0.001	1.32	1.20(34)	1.20(34)	1.32(3)	1.32(34)	
	0.01	1.11	1.11(29)	1.09(29)	1.09(3)	1.09(29)	
	0.1	1.08	1.08(14)	1.11(14)	1.11(3)	1.11(14)	
	1.0	1.10	1.10(3)	1.51(3)	1.51(3)	1.51(9)	
0.1	0.0001	3.65	1.34(34)	1.34(34)	3.64(3)	3.65(34)	
	0.001	2.46	1.41(34)	1.40(34)	2.45(3)	2.45(34)	
	0.01	1.58	1.53(34)	1.53(34)	1.58(3)	1.58(34)	
	0.1	1.21	1.21(24)	1.24(24)	1.24(3)	1.24(24)	
	1.0	1.13	1.13(3)	1.99(3)	1.99(3)	1.99(11)	
1.0	0.0001	5.5	1.92(34)	3.01(34)	6.50(3)	6.59(25)	6.59(34)
	0.001	4.2	1.92(34)	3.01(34)	5.20(3)	5.29(25)	5.29(34)
	0.01	3.0	1.97(34)	3.07(34)	4.02(3)	4.10(25)	4.10(34)
	0.1	1.95	1.89(34)	3.23(34)	3.22(3)	3.29(25)	3.29(34)
	1.0	1.31	1.31(15)	5.50(15)	5.44(3)	5.50(15)	5.50(22)
	10.	0.97	0.97(3)	37.4(3)	37.4(3)	37.4(7)	
10.0	0.01	4.55	2.23(34)	27.5(34)	26.7(3)	29.8(25)	29.8(34)
	0.1	3.25	2.19(34)	27.6(34)	25.5(3)	28.6(25)	28.7(34)
	1.0	2.09	2.06(34)	29.6(34)	26.6(3)	29.6(25)	29.6(34)
	10.	1.20	1.20(10)	50.6(10)	49.4(3)	50.6(10)	50.6(16)
	100.	0.59	0.59(3)	106.7(3)	106.7(3)	106.7(7)	
100.0	0.1	4.55	2.28(34)	102.2(34)	82.6(3)	104.3(25)	104.5(34)
	1.	3.30	2.31(34)	102.4(34)	81.7(3)	103.2(25)	103.4(34)
	10.	2.08	2.05(34)	106.1(34)	85.8(3)	106.0(25)	106.1(34)
	100.	1.06	1.06(10)	125.4(10)	121.0(3)	125.4(10)	125.4(15)
	1000.	0.42	0.42(3)	117.7(3)	117.7(3)	117.7(7)	
1000.0	10.0[b]	3.30	2.31(34)	187.3(34)	122.4(3)	185.7(25)	188.3(34)
	100.[b]	2.07	1.89(17)	187.7(34)	129.0(3)	172.8(10)	183.1(17)
	1000.	0.99	0.99(10)	141.3(10)	135.2(3)	141.3(10)	141.3(15)
	10,000.	0.36	0.36(3)	74.1(3)	74.1(3)	74.1(7)	
10,000.0	100.0[b]	3.30	2.31(34)	244.8(34)	132.1(3)	235.4(25)	245.8(34)
	1000.[b]	2.06	2.03(34)	232.2(34)	139.5(3)	228.1(25)	232.2(34)
	10,000.	0.98	0.98(10)	128.1(10)	122.4(3)	128.1(10)	128.1(15)
	100,000.	0.35	0.35(3)	51.8(3)	51.8(3)	51.8(7)	

[a]All values in this table, with the exception of the Karzas and Latter entries, were computed with the DEGA-A code. An integer in parentheses is the value of ℓ for which either Eq. (51) or (53) was terminated to obtain the Gaunt factor. These integers illustrate the speed of convergence of Eqs. (51) and (53).

[b]Neither Eq. (51) nor (53) has converged.

of a geometric sum enables the extrapolation of Eq. (51) to results that should represent a lower bound to the infinite sum. Numerical checks indicate that this extrapolation is good to 10% or better. To derive this extrapolation formula, let us simplify Eq. (51) by setting

$$Y_\ell^1 = (\ell + 1)\, H_{\ell+1,\ell}(\lambda_a, h\nu)^2$$

and (54)

$$Y_\ell^2 = \ell H_{\ell-1,\ell}(\lambda_a, h\nu)^2 \quad .$$

From calculations, we note that both $(\log Y_\ell^1, \ell)$ and $(\log Y_\ell^2, \ell)$ approximate straight lines for large values of ℓ. (The $(\log Y_\ell, \ell)$ pair is not to be confused with the $(\lambda_a, h\nu)$ pair.) Figure 9 illustrates this point for (1000, 100). The straight line for Y_ℓ^1 is written as

$$\log Y_\ell^1 = s(\ell - \ell') + q \quad , \tag{55}$$

and two known points, $(\log Y_{\ell'}^1, \ell')$ and $(\log Y_{\ell'+1}^1, \ell' + 1)$, are chosen to evaluate s and q. Equation (55) can also be written as

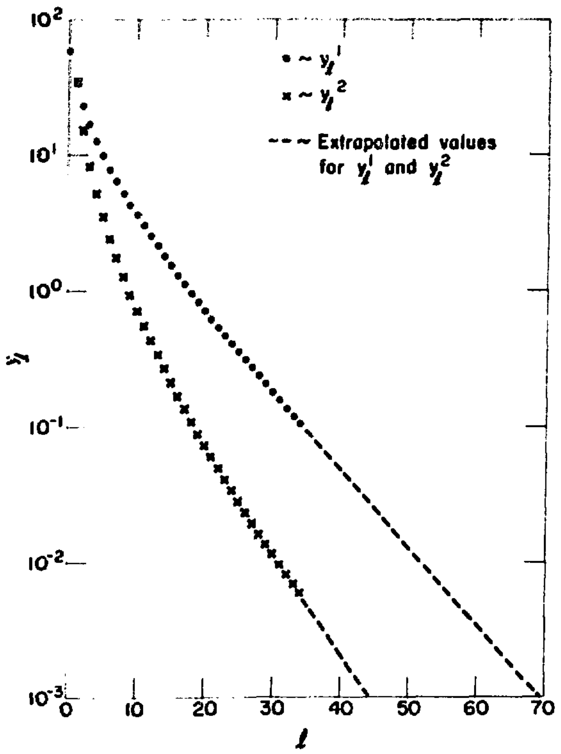

Fig. 9.
A graphic representation of the terms in Eq. (51) and the extrapolation indicated in Eq. (56).

$$Y_\ell^1 = e^q (e^s)^{\ell - \ell'} \quad . \tag{56}$$

Now using Eq. (56) to take the infinite sum for all $\ell \geq \ell'$, we have the extrapolation formula

$$\sum_{\ell=\ell'}^{\infty} Y_\ell^1 = e^q/(1 - e^s) \quad . \tag{57}$$

The above procedure is repeated using Y_ℓ^2. These two extrapolated values are added to the number obtained by summing Eq. (51) to $\ell = \ell' - 1$. The results obtained by using this extrapolation formula are given in Table III.

TABLE III

EXTRAPOLATED VALUES OF GAUNT FACTORS

λ_a	$h\nu$	$g_{ff}(\lambda_a, h\nu)$ for Na Calculated Using Eq. (51); ℓ in Parentheses	$g_{ff}(\lambda_a, h\nu)$ for Na Calculated Using Extrapolated Form of Eq. (51); ℓ' in Parentheses	
1000.0	10.0	187.3 (34)	189.5 (27)	189.9 (34)
	100.	187.7 (34)	188.3 (27)	188.3 (34)
10,000.0	100.0	244.8 (34)	259.0 (27)	261.7 (34)
	1000.	232.2 (34)	234.2 (27)	234.3 (34)

REFERENCES

1. F. Herman and S. Skillman, *Atomic Structure Calculations* (Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1963).

2. T. R. Carson, D. F. Mayers, and D. W. N. Stibbs, "The Calculation of Stellar Radiative Opacity," Mon. Not. R. Astr. Soc. 140, 483 (1968).

3. A. S. Householder, *Principles of Numerical Analysis* (McGraw-Hill Book Company Inc., New York, 1953).

4. E. Storm and H. I. Israel, "Photo Cross Sections from 0.001 to 100 MeV for Elements 1 through 100," Los Alamos Scientific Laboratory report LA-3753 (November 15, 1967).

5. D. E. Carter and M. P. Givens "Soft X-Ray Absorption of Thin Films of Iron and Iron Oxide," Phys. Rev. 101, 1469 (1956).

6. W. D. Barfield, G. D. Koontz, and W. F. Huebner, "Fits to New Calculations of Photoionization Cross Sections for Low Z-Elements," J. Quant. Spectrosc. Radiat. Transfer 12, 1409 (1972).

7. A. M. Naqvi, "Calculations and Applications of Screened Hydrogenic Wave Functions," J. Quant. Spectrosc. Radiat. Transfer 4, 597 (1964).

8. H. A. Bethe and E. E. Salpeter, *Quantum Mechanics of One- and Two-Electron Atoms* (Academic Press Inc., New York, 1957).

9. W. J. Karzas and R. Latter, "Electron Radiative Transitions in a Coulomb Field," Apl. J., Suppl. 6, 167 (1961). Also see "Free-Free Gaunt Factors," The Rand Corporation, Santa Monica, California, report RM-2010-AEC (November 8, 1957).

10. R. R. Johnston, "Free-Free Radiative Transitions–A Survey of Theoretical Results," J. Quant. Spectrosc. Radiat. Transfer 7, 815 (1967).

APPENDIX
DEGA-A

DEGA-A in its present form is a research and not a production code. The version, whose listing follows, only calculates bound-free absorptions. The code lists these absorption coefficients and makes plots as seen in the previous section. With slight modifications in the coding, bound-bound and free-free absorptions can be calculated.

There are several free parameters that regulate the accuracy of the code. Most of these parameters are set at the beginning of the code; however, a few are found throughout the code. The values in the listing will give at least four-place accuracy. The user should feel free to vary these parameters as he pleases and at his own risk.

Data are read into the bound-free version of DEGA-A with the following FORTRAN statements.

```
  2 FORMAT (8F10.3)
  7 FORMAT (16I5)
133 FORMAT (F15.5,2I5)
    READ 2, Z, RR2, RR
    IF (Z.LT.0.0) end job
    READ 7, ISSMAX
    READ 7, (NSUBSHL (ISS), ISS = 1, ISSMAX)
    READ 7, KWKB
    DO 132 K = 1, KWKB
132 READ 133, XLAMWKB(K), LWKB(K), NWKB(K)
    READ 7, IHNUMAX
    READ 2, (HNUVEC(IHNU), ACOFVEC(IHNU),
              IHNU = 1, IHNUMAX)
```

Z	is the atomic number of the element under consideration.
RR2	is the same as r_o on page 1.
RR	is the same as R_1 on page 1. Units for RR2 and RR are number of Bohr Radii.
ISSMAX	is the number of subshells under consideration.
NSUBSHL(ISS)	is an array that contains the number of electrons in each subshell. The entries are read into this array in the order 1s, 2s, 2p, 3s, 3p, 3d, etc., up to the last occupied subshell. If any previous subshell is vacant, it must be assigned the value zero.
KWKB	is the number of bound wave functions to be computed. The necessary free wave functions are generated internally by the code.
XLAMWKB(K)	is an array that contains guesses at the eigenvalues of the bound states. Units are number of Rydbergs.
LWKB(K)	is an array that contains the quantum numbers ℓ. $\ell = 0$ for s - states, $\ell = 1$ for p - states, $\ell = 2$ for d - states, etc.
NWKB(K)	is an array that contains the quantum numbers n. n = 1 for the 1s state, n = 2 for the 2s and 2p states, n = 3 for the 3s, 3p, and 3d states, etc.

For each K, XLAMWKB(K), LWKB(K), and NWKB(K) should be consisent with Eq. (39). These guesses at bound eigenvalues can be read in any order. NSUBSHL(ISS) should be defined for each of these guesses at a bound eigenvalue.

IHNUMAX	is the number of ($h\nu$, $\sigma_{bf}(h\nu)$) pairs from a separate source that one wants to compare with the results of DEGA-A. This option is illustrated by the X's in Figs. 2 through 7. If IHNUMAX equals zero, no ($h\nu$, $\sigma_{bf}(h\nu)$) pairs will be read.
HNUVEC(IHNU)	is the array that contains the $h\nu$'s given in electron-volts.
ACOFVEC(IHNU)	is the array that contains the $\sigma_{bf}(h\nu)$'s given in barns/atom.

Data decks may be stacked one behind the other. The job terminates normally when it encounters a negative Z.

```
      PROGRAM DEGAA(INP, OUT, FILM)
      DIMENSION XLAMARY(100), LARY(101), ALFAARY(100), IMAXARY(100)
     1      ,NSUBSHL(28), NSHELL(100)
      COMMON/SCRATCH/SCRATCH(1204)
      DIMENSION AM1S(401), AM2S(401), JMAXS(401)
      EQUIVALENCE (SCRATCH(1), AM1S(1)), (SCRATCH(402), AM2S(1)),
     1      (SCRATCH(803), JMAXS(1))
      DIMENSION AROOT(2)
      DIMENSION PDUM(2)
      COMMON/CB3/ Z, ZM1, RR2, RR, A0
      COMMON/AIMAX/  AF(51,402), JMAXF(402), IMAX,
     1      IMAXF, IMAXFP1, IMAXB, IMAXBP1, C(3,400),
     2      CF(3,402), ATOP(401), ATOPF(402)
      DIMENSION AB(51,402), JMAXB(402), CB(3,402), ATOPB(402)
      EQUIVALENCE(AF(1), AB(1)), (JMAXF(1), JMAXB(1)),
     1      (CF(1), CB(1)), (ATOPF(1), ATOPB(1))
      COMMON/EPS/EPSCONV, FBMAX, EM16, EMATELE, EMDVDD, MAXDIM
      DIMENSION AXLAMB(100)
      DIMENSION XLAMWKB(100), LWKB(101), NWKB(100)
      COMMON/PI/PI, TWOSPI
      COMMON/CB1/ HNUVEC(500), ACOFVEC(500), NOPTS, IHNUMAX, MM(100),
     1      NN(100), MNMAX, I4MIN, I4MAX
      COMMON/METHODS/METHOD, TEMPLAM, CASF, CAPFP, THETA, THETAP
      PI = 3.1415926535898   $   TWOSPI = 2.0/PI
      MAXITER = 100
      MAXDIM = 100
      EPSCONV = 1.0E-5
      FBMAX = 1.0E200
      EM16 = 1.0E-8
      EMATELE = 1.0E-5
      EMDVDD = 1.0E-8
      RMAXFAC = 25.0
      EPHI = 1.0E-15
      DRMIN = 1.0E-8
      DVMAX = 1.0E-4
      ATOPFAC = .5
      DHNUI = .1
      ZZZFAC = 10.0
      NODIVNG = 70
      XLAMFC1 = 3.0
      XLAMFC2 = .2
C      MAKE XLAMFC1 .GT. 1.0 AND XLAMFC2 .LT. 1.0
      CALL ADV(15)
    1 FORMAT(E24.14, 2I5)
    2 FORMAT(8F10.3)
    3 FORMAT(5E25.14)
    7 FORMAT(16I5)
    9 FORMAT(8F10.3)
  997 CONTINUE
      MFAC = 1
      NOPTS = 500
      KPHIMAX = 0
      IERROR = 0
      RMIN = 0.0
      READ 2, Z, RR2, RR
      IF(Z .LT. 0.0) GO TO 998
      RMAX = RR
      PRINT 3, Z, RR2, RR
      ZM1 = Z - 1.0
      A0 = .6057*Z**.333333
      READ  7, ISSMAX
      PRINT 7, ISSMAX
      READ  7, (NSUBSHL(ISS), ISS=1, ISSMAX)
      PRINT 7, (NSUBSHL(ISS), ISS=1, ISSMAX)
      READ  7, KWKB
      PRINT 7, KWKB
  133 FORMAT(F15.5, 2I5)
      DO 132 K=1, KWKB
      READ 133, XLAMWKB(K), LWKB(K), NWKB(K)
      PRINT  1, XLAMWKB(K), LWKB(K), NWKB(K)
      XLAMARY(K) = XLAMWKB(K)
      LARY(K) = LWKB(K)
  132 NSHELL(K) = NWKB(K)
      LARY(KWKB + 1) = -1
      IF(RR2 .GE. 0.0) SCRATCH(1) = V1(RR2)
      CALL YYY(RMIN,RMAX,DRMIN,DVMAX,ATOPFAC,I2,IERROR,
     1      XLAMWKB, LWKB, KWKB, ZZZFAC)
      IF(IERROR .NE. 0) GO TO 99
      I2P1 = I2 + 1
      ISKIP = 3*MAXDIM + 4
      IECS = 1 - ISKIP
      DO 118 I=1, I2
      IECS = IECS + ISKIP
      CALL ECWR(ATOP(I+1), IECS, 1, IE)
  118 CALL ECWR(C(1,I), IECS+1, 3, IE)
      PRINT 7, IMAX , I2
      IMAXP1 = IMAX + 1
      IF(IMAXP1 .GT. 400) GO TO 109
      DO 108 I=IMAXP1, 400
      C(1,I)= 0.0
      C(2,I) = -1.0
  108 C(3,I) = 0.0
  109 ATOP(1) = 0.0
      DO 10 I=1, IMAX
   10 PRINT3, C(1,I), C(2,I), C(3,I), ATOP(I+1)
      GO TO 49
  135 CONTINUE
      CALL ZZZ(XLAMARY, LARY, NSHELL, KPHIMAX, HNUVEC, NOPTS,
     1      IHNUMAX, DHNUI, ZZZFAC)
  136 FORMAT(///)
      PRINT 136
      DO 137 KPHI=1, KPHIMAX
  137 PRINT 133, XLAMARY(KPHI), LARY(KPHI), NSHELL(KPHI)
      PRINT 136
      PRINT 3, (HNUVEC(IHNU), IHNU=1, IHNUMAX)
      PRINT 136
      DO 138 IHNU=1, IHNUMAX
  138 ACOFVEC(IHNU) = 0.0
      KWKBMAX = 0
      DO 139 KPHI=1, KPHIMAX
      KWKBMAX = KWKBMAX + 1
      NWKB(KWKBMAX) = NSHELL(KPHI)
      LWKB(KWKBMAX) = 1000*(LARY(KPHI) + 1) + LARY(KPHI)
      XLAMWKB(KWKBMAX) = XLAMARY(KPHI)
      IF(LARY(KPHI) .EQ. 0) GO TO 139
      KWKBMAX = KWKBMAX + 1
      NWKB(KWKBMAX) = NSHELL(KPHI)
      LWKB(KWKBMAX) = 1000*(LARY(KPHI) - 1) + LARY(KPHI)
      XLAMWKB(KWKBMAX) = XLAMARY(KPHI)
  139 CONTINUE
      KWKB = 0
      I4MIN = 1
  140 KWKB = KWKB + 1
      XLAMARY(1) = XLAMWKB(KWKB)
      NSHELL(1) = NWKB(KWKB)
      LPOS = LWKB(KWKB)/1000
      LARY(1) = LWKB(KWKB) - LPOS*1000
  141 IF(1.0000001*ABS(XLAMARY(1)) .LT. HNUVEC(I4MIN)) GO TO 142
      I4MIN = I4MIN + 1
      GO TO 141
  142 I4MAX = IHNUMAX
      MNMAX = 0
      DO 143 I4=I4MIN , I4MAX
      MNMAX = MNMAX + 1
      MNMAXP1 = MNMAX + 1
      MM(MNMAX) = 1
      NN(MNMAX) = MNMAXP1
      XLAMARY(MNMAXP1) = HNUVEC(I4) + XLAMARY(1)
      LARY(MNMAXP1) = LPOS
  143 NSHELL(MNMAXP1) = 0
      LARY(MNMAXP1 + 1) = -2
      KPHIMAX = 0
   49 CONTINUE
      KPHIMP1 = KPHIMAX + 1
      L = LARY(KPHIMP1)
      IF(L .EQ. -1) GO TO 135
      IF(L .EQ. -2) GO TO 99
      METHOD = 2
      IF(XLAMARY(KPHIMP1) .LT. 0.0) METHOD = 1
      XL = L
      XLLP1 = L*(L+1)
      XLLM1 = XL*(XL-1.0)
      TWOXL = 2.0*XL
      LM2 = L-2
      LM1 = L-1
      LP1 = L+1
      XLP1 = LP1
      IF (METHOD .EQ. 1) GO TO 107
      XLAMBDA = XLAMARY(KPHIMP1)
      SMALLK = SQRT(XLAMBDA)
      RMAX = (XLLP1 + 1.0/XLAMBDA - 30.0)*SMALLK
      RTMP = 10.0/XLAMBDA
      IF(RTMP .GT. RMAX) RMAX = RTMP
      IF(ATOP(I2+1) .GT. RMAX) RMAX = 1.05*ATOP(I2+1)
      IF(RMAX .LT. 1.01) RMAX = 1.01
      GO TO 122
  146 RMAX = 1.2*RMAX
      IERROR = 0
      GO TO 110
  107 AROOT(1) = 0.0
      AROOT(2) = 0.0
      IMAX = I2 + 1
      ATOP(IMAX+1) = 1000.0
      CALL ROOTDIV(XLAMARY(KPHIMP1), XLLP1, AROOT, ICNTR, ICNTD, ICASE,
     1   IERROR, 1)
      PRINT 3, XLAMARY(KPHIMP1), AROOT(1), AROOT(2)
      IF(IERROR .EQ. 0) GO TO 151
      PRINT 152, IERROR
  152 FORMAT(I5, *  RMAX SET TO 1.1ATOP(I2+1)*)
      IERROR = 0
      RMAX = 1.1*ATOP(I2+1)
      GO TO 122
  151 RMAX = AROOT(2)
      IF(AROOT(1) .GT. RMAX) RMAX = AROOT(1)
      RMAX = RMAXFAC*RMAX
      IF(RMAX .LT. 1.01*ATOP(I2+1))RMAX = 1.01*ATOP(I2+1)
      PRINT 153, RMAX                                                   DEGA    00188
  153 FORMAT(*  RMAX = *, E20.10)                                       DEGA    00189
  122 IMAX = I2                                                         DEGA    00190
  110 DR = ATOPFAC*ATOP(IMAX+1)                                         DEGA    00191
      RTMP= ATOP(IMAX + 1)                                              DEGA    00192
      SMALLK = SQRT(ABS(XLAMARY(KPHIMP1) + (2.0 - XLLP1/RTMP)/RTMP))    DEGA    00193
      RTMP = ATOP(IMAX+1) + .5*(ATOP(IMAX+1) - ATOP(IMAX))              DEGA    00194
      SMALLK1 = SQRT(ABS(XLAMARY(KPHIMP1) + (2.0 - XLLP1/RTMP)/RTMP))   DEGA    00195
      IF(SMALLK1 .GT. SMALLK) SMALLK = SMALLK1                          DEGA    00196
      DRMAX = 6.28/SMALLK                                               DEGA    00197
      IF(DR .GT. DRMAX) DR = DRMAX                                      DEGA    00198
      IMAX = IMAX + 1                                                   DEGA    00199
      ATOP(IMAX+1) = ATOP(IMAX) + DR                                    DEGA    00200
      IF(IMAX .EQ. 400) 124, 123                                        DEGA    00201
  124 PRINT 125                                                         DEGA    00202
  125 FORMAT(*      400 INTERVALS WILL NOT SPAN (0,RMAX)*)              DEGA    00203
      GO TO 999                                                         DEGA    00204
  123 IF(ATOP(IMAX+1) .GE. RMAX) 111, 110                               DEGA    00205
  111 ATOP(IMAX+1) = RMAX                                               DEGA    00206
      CAPR = RMAX                                                       DEGA    00207
      ATOPTMP = ATOP(IMAX + 1)                                          DEGA    00208
      ATOP(IMAX+1) = CAPR                                               DEGA    00209
      GO TO (68,69), METHOD                                             DEGA    00210
   68 NODIV = NODIVNG                                                   DEGA    00211
      NODIVP1 = NODIV + 1                                               DEGA    00212
      AXLAMB(1)          = XLAMFC1*XLAMARY(KPHIMP1)                     DEGA    00213
      AXLAMB(NODIVP1) = XLAMFC2*XLAMARY(KPHIMP1)                        DEGA    00214
      DAXLAMB = ABS(ALOG(ABS(AXLAMB(1)))                                DEGA    00215
     1     - ALOG(ABS(AXLAMB(NODIVP1))))/NODIV                          DEGA    00216
      IF(ABS(AXLAMB(1)) .GT. ABS(AXLAMB(NODIVP1))) DAXLAMB = -DAXLAMB   DEGA    00217
      DO 60 I=2, NODIV                                                  DEGA    00218
   60 AXLAMB(I) = -EXP(ALOG(ABS(AXLAMB(I-1))) + DAXLAMB)                DEGA    00219
      XLAMBDA = AXLAMB(1)                                               DEGA    00220
      GO TO 70                                                          DEGA    00221
   69 TEMPLAM = XLAMBDA                                                 DEGA    00222
      NODIV = 4     $    NODIVP1 = NODIV + 1                            DEGA    00223
      AXLAMB(1) = 0.0     $    DXLAM = PI/4.0                           DEGA    00224
      DO 71 I=1, NODIV                                                  DEGA    00225
   71 AXLAMB(I+1) = AXLAMB(I) + DXLAM                                   DEGA    00226
      CALL CARSON(XLAMBDA, C(2,IMAX), MFAC, CAPR, XLLP1,                DEGA    00227
     1     CAPF, CAPFP, THETA, THETAP, IERROR)                          DEGA    00228
      IF(IERROR .EQ. 12) GO TO 146                                      DEGA    00229
   70 CONTINUE                                                          DEGA    00230
      AROOT(1) = 0.0    $    AROOT(2) = 0.0                             DEGA    00231
      CALL        ROOTDIV(XLAMBDA, XLLP1, AROOT, ICNTR, ICNTD, ICASE,   DEGA    00232
     1              IERROR, 0)                                          DEGA    00233
      PRINT 3, XLAMBDA, AROOT(1), AROOT(2)                              DEGA    00234
      J25SET = 1                                                        DEGA    00235
   80 J25FST = 1                                                        DEGA    00236
      J25MIN = J25SET                                                   DEGA    00237
      J25MAX = NODIVP1                                                  DEGA    00238
      GO TO 90                                                          DEGA    00239
   81 J25FST = 2                                                        DEGA    00240
      J25MIN = NODIVP1+1                                                DEGA    00241
      J25MAX = NODIVP1 + 20                                             DEGA    00242
      D25 = (XLAMSTP - XLAMSTT)/19.0                                    DEGA    00243
      AXLAMB(J25MIN) = XLAMSTT                                          DEGA    00244
      AXLAMB(J25MAX) = XLAMSTP                                          DEGA    00245
      I79MIN = J25MIN+1                                                 DEGA    00246
      I79MAX = J25MAX - 1                                               DEGA    00247
      DO 79 I79=I79MIN, I79MAX                                          DEGA    00248
   79 AXLAMB(I79) = AXLAMB(I79 - 1) + D25                               DEGA    00249
   90 DO 25 J= J25MIN, J25MAX
      XLAMSTP = AXLAMB(J)
      CALL TAYLORF(XL, XLAMSTP, IERROR, AF, JMAXF, IMAXF, CF, ATOPF)
      IBKW = IMAXFP1 + 1
      CALL TAYLORB(XL, XLAMSTP, IERROR, AB(1, IBKW),
     1     JMAXB(IBKW), IMAXB, CB(1,IBKW), ATOPB(IBKW))
      CALL BOUNDRY(IFCONV, FBSTP)
C          TROUBLE MAY ARISE (MISS AN EIGENVALUE) IF FBSTP=0 TWICE
C     IN A ROW.
C     *****************
C          HERE, LATER, FIND NEW ROOTS AND CHECK STABILITY CONDITIONS.
      IF(IFCONV .EQ. 1) 73, 74
   73 CAPLAMB = XLAMSTP
      GO TO 75
   74 IF(J .EQ. J25MIN) 26, 27
   27 IF(FBSTT*FBSTP .LE. 0.0) 28, 26
   28 XLAMMID = .5*(XLAMSTP + XLAMSTT)
      CALL TAYLORF(XL, XLAMMID, IERROR, AF, JMAXF, IMAXF, CF, ATOPF)
      IBKW = IMAXFP1 + 1
      CALL TAYLORB(XL, XLAMMID, IERROR, AB(1,IBKW),
     1     JMAXB(IBKW), IMAXB, CB(1,IBKW), ATOPB(IBKW))
      CALL BOUNDRY(IFCONV, FBMID)
      IF(FBSTT .GT. FBSTP) 29, 30
   29 FBTOP = FBSTT    $    FBBOT = FBSTP    $    GO TO 31
   30 FBTOP = FBSTP    $    FBBOT = FBSTT
   31 IF((FBTOP .GT. FBMID) .AND. (FBMID .GT. FBBOT)) 32, 57
   57 PRINT 58
      PRINT 58
      PRINT 59
      PRINT 58
      PRINT 58
   58 FORMAT(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)
   59 FORMAT(*  FBMID IS NOT BETWEEN FBTOP AND FBBOT.  MAKE SURE A VALUE
     1 OF LAMBDA WAS NOT MISSED.*)
      GO TO 87
   32 XLAMB0 = XLAMSTT    $    XLAMB1 = XLAMSTP
      FB0 = FBSTT    $    FB1 = FBSTP
C          REGULA DOES NOT CHECK STABILITY CONDITIONS.
      CALL REGULA(XLAMB0,XLAMB1,FB0,FB1,CAPLAMB,MAXITER,XL,IERROR)
C          HERE, LATER, FIND NEW ROOTS AND CHECK STABILITY CONDITIONS.
      IF(IERROR .NE. 0) GO TO 87
      IF(XLAMSTT .LT. XLAMSTP) GO TO 105
      XLAMB0 = XLAMSTP
      XLAMB1 = XLAMSTT
      GO TO 104
  105 XLAMB0 = XLAMSTT
      XLAMB1 = XLAMSTP
  104 IF(XLAMB0 .LE. CAPLAMB .AND. CAPLAMB .LE. XLAMB1) 75, 86
   86 PRINT 58
      PRINT 58
      PRINT 88
   88 FORMAT(*  CAPLAMB IS NOT BETWEEN XLAMSTT AND XLAMSTP.  XLAMSTT, CA
     1PLAMB, AND XLAMSTP ARE *)
      PRINT 3, XLAMSTT, CAPLAMB, XLAMSTP
      PRINT 58
      PRINT 58
   87 IF(J25FST .EQ. 1) 82, 84
   82 J25SET = J
      PRINT 83
   83 FORMAT(*  DIVIDE THIS INTERVAL INTO 19 EQUAL INTERVALS AND TRY AGA
     1IN.  *)
      PRINT 58
      PRINT 58
      GO TO 81
   84 PRINT 85
   85 FORMAT(*  THIS INTERVAL FAILED FOR THE SECOND TIME.  FORGET IT AND
     1 GO TO THE NEXT INTERVAL. *)
      PRINT 58
      PRINT 58
      GO TO 80
   75 GO TO(65, 66), METHOD
   66 ALPHA = CAPLAMB    $  CAPLAMB = XLAMBDA
      PRINT 67, ALPHA, CAPLAMB
   67 FORMAT(*  ALPHA =*,  E24.14, *      FOR XLAMBDA =*,E24.14)
      GO TO 127
   65 CONTINUE
      IV1 = 0
      D = ATOPF(2)
      DLP1 = D**LP1
      JMAXBOT = JMAXF(1) - 1
      PDUM(1) = AF(1,2)/DLP1
      PDUM(2) = (AF(2,2) - XLP1*AF(1,2)/D)/DLP1
      CALL STURMSQ(AF(2,1), JMAXBOT, PDUM, D, IV2, IERROR)
      IV1 = IV1 + IV2
      IF(IMAXF .EQ. 1) GO TO 98
      DO 91 IF=2, IMAXF
      D = ATOPF(IF+1) - ATOPF(IF)
      CALL STURMSQ(AF(1,IF), JMAXF(IF), AF(1,IF+1), D, IV2, IERROR)
   91 IV1 = IV1 + IV2
   98 CONTINUE
      DO 95 IB=1, IMAXB
      IBKW = IMAXFP1 + IB
      D = ATOPB(IBKW + 1) - ATOPB(IBKW)
      CALL STURMSQ(AB(1,IBKW), JMAXB(IBKW), AB(1,IBKW+1),
     1     D, IV2, IERROR)
   95 IV1 = IV1 + IV2
      PRINT 128, IV1
  128 FORMAT(* THIS WAVE FUNCTION HAS*,I5, * CROSSINGS.*)
  127 IBKW = IMAXFP1 + IMAXBP1
      FAC = AB(1,IBKW)/AF(1,IMAXFP1)
      DO 33 I=1, IMAXF
      JJJ = JMAXF(I)
      DO 33 JF=1, JJJ
   33 AF(JF,I) = FAC*AF(JF,I)
      IF(CAPLAMB .LT. 0.0) CALL NORMPHI(CAPLAMB, L)
      PRINT 3, CAPLAMB
      KPHIMAX = KPHIMAX + 1
      GO TO(129,130), METHOD
  129 IV1L1 = IV1 + L + 1
      IF(NSHELL(KPHIMAX) .EQ. IV1L1) GO TO 154
      IV1 = NSHELL(KPHIMAX) - L - 1
      KPHIMAX = KPHIMAX - 1
      PRINT 134, IV1
  134 FORMAT(*  THE PREVIOUS WAVE FUNCTION SHOULD HAVE HAD*, I5,
     1   * CROSSINGS.*/*  FORGET THE LAST EIGENVALUE AND TRY SOMEMORE.*)
      GO TO 26
  154 IBKW = IMAXFP1 + 1
      PHIMIN9 = ABS(AB(1,IBKW))
      PHIMAX9 = PHIMIN9
      IF(IMAXB .EQ. 1) GO TO 155
      DO 156 IB=2, IMAXB
      IBKW = IMAXFP1 + IB
      PHI9 = ABS(AB(1,IBKW))
      IF(PHI9 .GT. PHIMAX9) PHIMAX9 = PHI9
  156 CONTINUE
  155 DO 157 IF=1, IMAXF
      PHI9 = ABS(AF(1,IF))
      IF(PHI9 .GT. PHIMAX9) PHIMAX9 = PHI9
  157 CONTINUE
      PRINT 158, PHIMAX9, PHIMIN9
  158 FORMAT(*  PHI MAX AND MIN ARE*, 2E20.10)
      IF(PHIMIN9 .LT. PHIMAX9*EPHI) GO TO 131
      PRINT 159
  159 FORMAT(*  RMAX IS NOT BIG ENOUGH*)
      KPHIMAX = KPHIMAX - 1
      GO TO 146
  130 NSHELL(KHPIMAX) = 0
  131 IMAXARY(KPHIMAX) = IMAX
      XLAMARY(KPHIMAX) = CAPLAMB
      LARY(KPHIMAX) = L
      IF(CAPLAMB .LT. 0.0) 112, 113
  112 ALFAARY(KPHIMAX) = 0.0
      GO TO 114
  113 ALFAARY(KPHIMAX) = ALPHA
  114 CONTINUE
      JMAXS(1) = JMAXF(1)
      DO 115 I=1, IMAXF
      AM1S(I) = AF(1,I)
  115 AM2S(I) = AF(2,I)
      IC = IMAXBP1
      ID = IMAXF
      DO 116 I=1, IMAXB
      IC = IC-1
      ID = ID+1
      IBKW = IMAXFP1 + IC
      AM1S(ID) = AB(1,IBKW)
      AM2S(ID) = AB(2,IBKW)
  116 JMAXS(ID) = JMAXB(IBKW)
      DO 117 I=2, IMAXFP1
  117 JMAXS(I) = 0
      IMAXBF = IMAXB + IMAXF
      IECS = KPHIMAX + 4
      IECS1 = IECS + MAXDIM
      IECS2 = IECS1 + MAXDIM
      CALL ECWR(JMAXS(1), IECS, 1, IE)
      CALL ECWR(AM1S(1), IECS1, 1, IE)
      CALL ECWR(AM2S(1), IECS2, 1, IE)
      DO 121 I=3, I2P1
      IECS = IECS + ISKIP
      IECS1 = IECS + MAXDIM
      IECS2 = IECS1 + MAXDIM
      CALL ECWR(JMAXS(I), IECS, 1, IE)
      CALL ECWR(AM1S(I), IECS1, 1, IE)
  121 CALL ECWR(AM2S(I), IECS2, 1, IE)
      IF(IMAX .EQ. I2) GO TO 126
      IECS = 4*(400 - I2)
      IECS = IECS*(KPHIMAX - 1) + 1
      IECS = IECS + I2*ISKIP
      IJUMP = 400 - I2
      INUM = IMAX - I2
      I2P2 = I2+2
      CALL ECWR(ATOP(I2P2), IECS, INUM, IE)
      IECS = IECS + IJUMP
      CALL ECWR(JMAXS(I2P2), IECS, INUM, IE)
      IECS = IECS + IJUMP
      CALL ECWR(AM1S(I2P2), IECS, INUM, IE)
      IECS = IECS + IJUMP
      CALL ECWR(AM2S(I2P2), IECS, INUM, IE)
  126 GO TO 61
   26 XLAMSTT = XLAMSTP    $    FBSTT = FBSTP
   25 CONTINUE
      IF(J25FST .EQ. 2) 80, 998
   61 ATOP(IMAX+1) = ATOPTMP
      GO TO  49
   99 IF(KPHIMAX .EQ. 0) GO TO 999
      DO 120 KPHI = 1, KPHIMAX
      PRINT 7, LARY(KPHI)
  120 PRINT 3, XLAMARY(KPHI) , ALFAARY(KPHI)
      CALL          MATELE(KPHIMAX, XLAMARY, LARY, ALFAARY, IMAXARY, I2,
     1      NSURSHL, ISSMAX, NSHELL)
  999 CONTINUE
      IF(KWKB .LT. KWKBMAX) GO TO 140
  145 FORMAT(15X, * EV*, 10X, * BARNS/ATOM*)
      PRINT 145
      YTOP = ALOG10(ACOFVEC(1))
      YBOT = YTOP
      DO 144 IHNU=1, IHNUMAX
      HNUVEC(IHNU) = 13.605*HNUVEC(IHNU)
      PRINT 3, HNUVEC(IHNU), ACOFVEC(IHNU)
      ACOFVEC(IHNU) = ALOG10(ACOFVEC(IHNU))
      IF(ACOFVEC(IHNU) .GT. YTOP) YTOP = ACOFVEC(IHNU)
      IF(ACOFVEC(IHNU) .LT. YBOT) YBOT = ACOFVEC(IHNU)
  144 HNUVEC(IHNU) = ALOG10(HNUVEC(IHNU))
      CALL ADV(2)
      CALL DGA(120, 980, 50, 910, HNUVEC(1), HNUVEC(IHNUMAX), YTOP,YBOT)
      CALL DLGLG
      CALL SBLOG
      CALL SLLOG
      CALL PLOT(IHNUMAX, HNUVEC, 1, ACOFVEC, 1, 42, 1)
      READ 7, IHNUMAX
      PRINT 7, IHNUMAX
      IF(IHNUMAX .EQ. 0) GO TO 997
      READ 2, (HNUVEC(IHNU), ACOFVEC(IHNU), IHNU=1, IHNUMAX)
      DO 150 IHNU=1, IHNUMAX
      PRINT 3, HNUVEC(IHNU), ACOFVEC(IHNU)
      HNUVEC(IHNU) = ALOG10(HNUVEC(IHNU))
  150 ACOFVEC(IHNU) = ALOG10(ACOFVEC(IHNU))
      CALL PLOT(IHNUMAX, HNUVEC, 1, ACOFVEC, 1, 55, 0)
      GO TO 997
  998 CONTINUE
      CALL ADV(15)
      CALL EMPTY
      END
      SUBROUTINE ROOTDIV(XLAMBDA, XLLP1, AROOT, ICNTR, ICNTD, ICASE,
     1              IERROR, IFSTOP)
      DIMENSION AROOT(2)
      COMMON/AIMAX/  AF(51,402), JMAXF(402), IMAX,
     1      IMAXF, IMAXFP1, IMAXB, IMAXBP1, C(3,400),
     2      CF(3,402), ATOP(401), ATOPF(402)
      DIMENSION AB(51,402), JMAXB(402), CB(3,402), ATOPB(402)
      EQUIVALENCE(AF(1), AB(1)), (JMAXF(1), JMAXB(1)),
     1      (CF(1), CB(1)), (ATOPF(1), ATOPB(1))
      ICNTR = 0    $   ICNTD = 0    $    IERROR = 0
      DO 10 I=1, IMAX
      II=I
      CALL BINOM (XLAMBDA, XLLP1, II, NOROOTS, ROOT1, ROOT2, NODIV,DIV1)
      NOROOT1 = NOROOTS + 1
      GO TO(14, 11, 12),NOROOT1
```

```
 11 IF((ATOP(I) .LE. ROOT1) .AND. (ROOT1 .LT. ATOP(I+1)))13,14        DEGA      00498
 13 ICNTR = ICNTR + 1                                                  DEGA      00499
    AROOT(ICNTR) = ROOT1                                               DEGA      00500
    IF(ICNTR .EQ. 2)15, 14                                             DEGA      00501
 12 IF(ROOT1 .LT. ROOT2) 28, 27                                        DEGA      00502
 27 TEMP = ROOT1    $    ROOT1 = ROOT2    $    ROOT2 = TEMP            DEGA      00503
 28 IF((ATOP(I) .LE. ROOT1) .AND.(ROOT1 .LT. ATOP(I+1)))16,17         DEGA      00504
 16 ICNTR = ICNTR + 1                                                  DEGA      00505
    AROOT(ICNTR) = ROOT1                                               DEGA      00506
    IF(ICNTR .EQ. 2)15, 17                                             DEGA      00507
 17 IF((ATOP(I) .LE. ROOT2) .AND. (ROOT2 .LT. ATOP(I+1)))18,14        DEGA      00508
 18 ICNTR = ICNTR + 1                                                  DEGA      00509
    AROOT(ICNTR) = ROOT2                                               DEGA      00510
    IF(ICNTR .EQ. 2)15, 14                                             DEGA      00511
 15 IF(ICNTD .EQ. 1)19, 20                                             DEGA      00512
 20 IF(NODIV .EQ. 1)21, 19                                             DEGA      00513
 21 IF((ATOP(I).LE. DIV1) .AND. (DIV1 .LT. ATOP(I+1)))22, 19          DEGA      00514
 22 ICNTD = 1   $    ADIV = DIV1    $    IDIV = I    $    GO TO 19    DEGA      00515
 14 IF(ICNTD .EQ. 1)10, 23                                             DEGA      00516
 23 IF(NODIV .EQ. 1)24, 10                                             DEGA      00517
 24 IF((ATOP(I).LE. DIV1) .AND. (DIV1 .LT. ATOP(I+1)))25, 10          DEGA      00518
 25 ICNTD = 1    $    ADIV = DIV1    $    IDIV = I                     DEGA      00519
 10 CONTINUE                                                           DEGA      00520
 19 IF((ICNTD .EQ. 1) .AND. (ICNTR .EQ.2))29, 44                       DEGA      00521
 44 IF((ICNTD .EQ.1).AND. (ICNTR .EQ. 0)) 45, 26                       DEGA      00522
 45 ICASE = 5        $    GO TO 31                                     DEGA      00523
 29 ICASE = 4                                                          DEGA      00524
    IF((AROOT(1) .LT. ADIV) .AND. (ADIV .LT. AROOT(2)))31, 30          DEGA      00525
 30 IERROR = 2                                                         DEGA      00526
    PRINT 2                                                            DEGA      00527
  2 FORMAT(*  IERROR=2, ICASE=4, MAX IS NOT BETWEEN THE TOW ROOTS,*)  DEGA      00528
    RETURN                                                             DEGA      00529
 26 IF((ICNTD.EQ. 0) .AND. (ICNTR .EQ. 0))34, 35                       DEGA      00530
 34 ICASE = 1    $    IDIV = IMAX                                      DEGA      00531
    ADIV = (ATOP(IMAX) + ATOP(IMAX+1))/2.0                             DEGA      00532
    GO TO 31                                                           DEGA      00533
 35 IF((ICNTD .EQ. 0) .AND.(ICNTR .EQ. 1))37, 40                       DEGA      00534
 37 ICASE = 2                                                          DEGA      00535
    IF(XLLP1 .LT. .25) 47, 46                                          DEGA      00536
 47 IMAXP1 = IMAX + 1                                                  DEGA      00537
    DO 48 I=2, IMAXP1                                                  DEGA      00538
    IF(AROOT(1) .LT. ATOP(I)) 49, 48                                   DEGA      00539
 49 IDIV = I-1                                                         DEGA      00540
    ADIV = AROOT(1)                                                    DEGA      00541
    GO TO 31                                                           DEGA      00542
 48 CONTINUE                                                           DEGA      00543
    IDIV = IMAX                                                        DEGA      00544
    GO TO 38                                                           DEGA      00545
 46 IDIV = IMAX                                                        DEGA      00546
    IF(AROOT(1) .LT. ATOP(IMAX))38, 39                                 DEGA      00547
 38 ADIV = (ATOP(IMAX) + ATOP(IMAX+1))/2.0                             DEGA      00548
    GO TO 31                                                           DEGA      00549
 39 ADIV = (AROOT(1) + ATOP(IMAX+1))/2.0                               DEGA      00550
    GO TO 31                                                           DEGA      00551
 40 IF((ICNTD .EQ. 1) .AND.(ICNTR .EQ. 1))41, 42                       DEGA      00552
 42 IERROR = 1                                                         DEGA      00553
    PRINT 1, ICNTD, ICNTR                                              DEGA      00554
  1 FORMAT(*  IERROR=1,   ICNTD =*, I2, *,   ICNTR =*, I2)            DEGA      00555
    RETURN                                                             DEGA      00556
 41 ICASE = 3                                                          DEGA      00557
    IF(AROOT(1) .LT. ADIV) 31, 43                                      DEGA      00558
 43 IERROR = 3                                                         DEGA      00559
```

```
      PRINT 3, AROOT(1), ADIV
    3 FORMAT(*  IERROR=3, ICASE=3, AROOT(1).GE.ADIV, AROOT(1) =*,E20.10,
     1        *, ADIV =*, E20.10)
      RETURN
   31 IF(IFSTOP .EQ. 1) RETURN
      IMAXF = IDIV   $   IMAXFP1 = IMAXF + 1
      IMAXB = IMAX + 1 - IDIV   $   IMAXBP1 = IMAXB + 1
      DO 32 I=1, IMAXF
      ATOPF(I) = ATOP(I)
      CF(1,I) = C(1,I)   $   CF(2,I) = C(2,I)
   32 CF(3,I) = C(3,I)
      ATOPF(IMAXFP1) = ADIV
      II = IMAX + 1   $   III = IMAX + 2
      DO 33 I=1, IMAXB
      III = III-1
      IBKW = IMAXFP1 + I
      ATOPB(IBKW) = ATOP(III)
      II = II - 1
      CB(1,IBKW) = C(1,II)   $   CB(2,IBKW) = C(2,II)
   33 CB(3,IBKW) = C(3,II)
      IBKW = IMAXFP1 + IMAXBP1
      ATOPB(IBKW) = ADIV
      RETURN
      END
      SUBROUTINE BINOM(XLAMBDA, XLLP1, I, NOROOTS, ROOT1, ROOT2,
     1         NODIV, DIV1)
      COMMON/AIMAX/  AF(51,402), JMAXF(402), IMAX,
     1       IMAXF, IMAXFP1, IMAXB, IMAXBP1, C(3,400),
     2       CF(3,402), ATOP(401), ATOPF(402)
      DIMENSION AB(51,402), JMAXB(402), CB(3,402), ATOPB(402)
      EQUIVALENCE(AF(1), AB(1)), (JMAXF(1), JMAXB(1)),
     1       (CF(1), CB(1)), (ATOPF(1), ATOPB(1))
      B1 = -(2.0*C(3,I) + XLLP1)
      B2 = -2.0*C(2,I)
      B3 = XLAMBDA - 2.0*C(1,I)
      ABSB1 = ABS(B1)
      IF(ABSB1 .LT. ABS(B2)*1.0E-12) 1, 2
    2 IF(ABSB1 .LT. ABS(B3)*1.0E-12) 1, 3
    1 ROOT1 = -B2/B3
      NOROOTS = 1
      NODIV = 0
      GO TO 6
    3 RAD = B2*B2 - 4.0*B1*B3
      TWOB1 = 2.0*B1
      NODIV = 1
      DIV1 = -TWOB1/B2
      IF(RAD .LT. 0.0) 4, 5
    4 NOROOTS = 0
      GO TO 6
    5 NOROOTS = 2
      RAD = SQRT(RAD)
      T1 = ABS(TWOB1)
      T2 = ABS(B2 + RAD)
      IF(T2 .LT. T1*1.0E-20) 7, 8
    9 FORMAT( I5, 3E20.10, *  XXXXX*)
    7 I1 = 1
      PRINT 9, I1, TWOB1, B2, RAD
      ROOT1 = 1.0E100
      GO TO 10
    8 CONTINUE
      ROOT1 = -TWOB1/(B2 + RAD)
   10 T1 = ABS(TWOB1)
```

```
      T2 = ABS(-B2 + RAD)
      IF(T2 .LT. T1*1.0E-20) 11, 12
   11 I1 = 2
      PRINT 9, I1, TWOB1, B2, RAD
      ROOT2 = 1.0E100
      GO TO 6
   12 CONTINUE
      ROOT2 = TWOB1/(-B2 + RAD)
    6 RETURN
      END
      SUBROUTINE REGULA(XLAMB0,XLAMB1,FB0,FB1,CAPLAMB,MAXITER,XL,IERROR)
      COMMON/AIMAX/  AF(51,402), JMAXF(402), IMAX,
     1      IMAXF, IMAXFP1, IMAXB, IMAXBP1, C(3,400),
     2     CF(3,402), ATOP(401), ATOPF(402)
      DIMENSION AB(51,402), JMAXB(402), CB(3,402), ATOPB(402)
      EQUIVALENCE(AF(1), AB(1)), (JMAXF(1), JMAXB(1)),
     1      (CF(1), CB(1)), (ATOPF(1), ATOPB(1))
      COMMON/EPS/ EPSCONV, FBMAX
    1 FORMAT(2E24.14, 4E20.10)
      IERROR = 0
      DO 14 ITER = 1, MAXITER
      IF(FB1 .EQ. FB0) 20, 21
   20 IBKW = IMAXFP1 + IMAXBP1
      AFAC = AB(1,IBKW)/AF(1,IMAXFP1)*AF(2,IMAXFP1)
      FBOUNDC = FBOUND/ABS(AFAC)
      CAPLAMB = .5*(XLAMB1 + XLAMB0)
      PRINT 22
      PRINT 22
      PRINT 23, FBOUNDC, EPSCONV
      PRINT 24, CAPLAMB, XLAMB1, XLAMB0
      PRINT 22
      PRINT 22
      RETURN
   22 FORMAT(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)
   23 FORMAT(* FB1=FB0,  FBOUNDC = *,E20.10,*    EPSCONV =*,E20.10)
   24 FORMAT(* CAPLAMB =*,E24.14,4H = (,E24.14,2H +,E24.14,4H )/2)
   21 ONEGAMA = (XLAMB1 - XLAMB0)/(FB1 - FB0)
      XLAMB2 = XLAMB1 - ONEGAMA*FB1
      IF(XLAMB2.EQ.XLAMB1.OR.XLAMB2.EQ.XLAMB0) GO TO 15
      CALL TAYLORF(XL, XLAMB2, IERROR, AF, JMAXF, IMAXF, CF, ATOPF)
      IF(IERROR .NE. 0) RETURN
      IBKW = IMAXFP1 + 1
      CALL TAYLORB(XL, XLAMB2, IERROR, AB(1,IBKW),
     1      JMAXB(IBKW), IMAXB, CB(1,IBKW), ATOPB(IBKW))
      IF(IERROR .NE. 0) RETURN
      CALL BOUNDRY(IFCONV, FBOUND)
      IF(IFCONV .EQ. 1) 15, 16
   15 CAPLAMB = XLAMB2   $   RETURN
   16 IF(ABS(FBOUND) .GT. FBMAX) 17, 19
   17 IERROR = 7
      PRINT 7
    7 FORMAT(*  IERROR = 7, FBOUND .GT. FBMAX*)
      RETURN
   19 XLAMB0 = XLAMB1    $    XLAMB1 = XLAMB2
      FB0 = FB1
   14 FB1 = FBOUND
      IERROR = 8
      PRINT 8, MAXITER
    8 FORMAT (*  PROBLEM DOES NOT CONVERGE WITHIN*,I5,* ITERATIONS*)
      RETURN
      END
      SUBROUTINE TAYLORF(XL, ALPHLAM, IERROR, A, JMAX, IMAX, C, ATOP)
      DIMENSION A(51,1), JMAX(1), C(3,1), ATOP(1)
      COMMON/METHODS/METHOD, TEMPLAM, CAPF, CAPFP, THETA, THETAP
      COMMON/EPS/EPSCONV, FUMAX, EM16
      GO TO (60, 61), METHOD
   60 XLAMBDA = ALPHLAM
      GO TO 62
   61 ALPHA = ALPHLAM
      XLAMBDA = TEMPLAM
   62 CONTINUE
      DO 30 I=1, IMAX
   30 JMAX(I) = 0
      IERROR = 0
      D = ATOP(2)
C*****************************************************************
C             LEFT BOUNDARY CONDITIONS.  THESE ARE ONLY TRUE FOR
C        PHI(0)=0 WHERE PHI(R) IS THE EIGENFUNCTION.
C*****************************************************************
      A(1,1) = 0.0
      A(2,1) = 1.0
C*****************************************************************
      CALL TAYLOR1(A(1,1), JMAX(1), D, XLAMBDA,
     1      XL, C(1,1), C(2,1), EM16, IERROR)
      IF(IERROR .EQ. 9) RETURN
      RR = ATOP(2)
      CALL POLY0P(A(1,1), JMAX(1), RR, P)
      A(1,2)= P*RR**XL
      CALL POLY1P(A(1,1), JMAX(1), RR, DERIVP)
      A(2,2) = (DERIVP*RR + P*XL)*RR**(XL-1.0)
      IF(IMAX .EQ. 1) GO TO 37
      IBOT = 2
      GO TO 58
      ENTRY TAYLORB
      DO 54 I=1, IMAX
   54 JMAX(I) = 0
      GO TO (65, 66), METHOD
   65 XLAMBDA = ALPHLAM
      GO TO 67
   66 ALPHA = ALPHLAM
      XLAMBDA = TEMPLAM
   67 CONTINUE
      IERROR = 0
      IBOT = 1
******************************************************************
             RIGHT BOUNDARY ,R.B. CONDITIONS.  HERE A(1,1) IS AN
        ARBITRARY CONSTANT, THE MAGNITUDE OF PHI(R.B.),  A(2,1) IS
        THE DERIVATIVE OF PHI(R.B.) NORMILIZED TO A(1,1).
******************************************************************
      GO TO (63, 64) , METHOD
   63 CONTINUE
      A(1,1) = 1.0E-140
      A(2,1) = -SQRT(ABS(XLAMBDA))*A(1,1)
      GO TO 58
   64 THETA1 = THETA + ALPHA
      COSTHE1 = COS(THETA1)
      A(1,1) = CAPF*COSTHE1
      A(2,1) = CAPFP*COSTHE1 - CAPF*THETAP*SIN(THETA1)
******************************************************************
   58 XLL = XL*(XL+1.0)
      DO 25 I=IBOT, IMAX
      D = ATOP(I+1) - ATOP(I)
      CALL TAYLORS(A(1,I), JMAX(I), ATOP(I), D, XLAMBDA,
     1      XLL, C(1,I), C(2,I), C(3,I), EM16, IERROR)
```

```
      IF(IERROR .EQ. 10) RETURN
      CALL POLY0P(A(1,I), JMAX(I), D, A(1,I+1))
      CALL POLY1P(A(1,I), JMAX(I), D, A(2,I+1))
   25 CONTINUE
   37 RETURN
      END
      SUBROUTINE BOUNDRY(IFCONV, FBOUND)
      COMMON/AIMAX/  AF(51,402), JMAXF(402), IMAX,
     1     IMAXF, IMAXFP1, IMAXB, IMAXBP1, C(3,400),
     2     CF(3,402), ATOP(401), ATOPF(402)
      DIMENSION AB(51,402), JMAXB(402), CB(3,402), ATOPB(402)
      EQUIVALENCE(AF(1), AB(1)), (JMAXF(1), JMAXB(1)),
     1     (CF(1), CB(1)), (ATOPF(1), ATOPB(1))
      COMMON/EPS/EPSCONV, FBMAX
C          YOU ARE LIVING DANGEROUSLY IF YOU LET IMAXF=1.  THIS MAY RESU
C     IN AN UNDETECTED DIVISION BY ZERO OR AN UNDETECTED LOSS OF ACCURAC
      IF(IMAXF .EQ. 1) 1, 2
    2 IF(ABS(AF(1,IMAXFP1)) .LT. ABS(AF(1,IMAXF))*1.0E-4) 3, 1
    1 IBKW = IMAXFP1 + IMAXB
      IBKWP1 = IBKW + 1
      IF(ABS(AB(1,IBKWP1)) .LT. ABS(AB(1,IBKW))*1.0E-4)3,5
    3 PRINT 4
    4 FORMAT(*    IN DANGER OF DIVIDING BY ZERO, OR ATLEAST LOOSING ACCU
     1RACY.  *)
    5 IBKW = IMAXFP1 + IMAXBP1
      FAC = AB(1,IBKW)/AF(1,IMAXFP1)
      AFAC = FAC*AF(2,IMAXFP1)
      FBOUND = AFAC - AB(2,IBKW)
      EPSC = ABS(AFAC)*EPSCONV
      IF(ABS(FBOUND) .LT. EPSC) 6, 7
    6 IFCONV = 1    $    GO TO 8
    7 IFCONV = 0
    8 RETURN
      END
      SUBROUTINE STURMSQ(A1, JMAX1, A2, D, IV2, IERROR)
      DIMENSION A1(2), A2(2), CPI( 51), CPIP1( 51)
      COMMON/SCRATCH/SCRATCH(604)
      EQUIVALENCE(SCRATCH(1), CPI(1)), (SCRATCH(52), CPIP1(1))
      IERROR = 0
      JMAXI = JMAX1
      IF(D .LT. 0.0) 11, 12
   11 ISIGN = -1
      GO TO 13
   12 ISIGN = 1
   13 IV2 = 0
      J1 = JMAXI+1
      DO 14 J=1, JMAXI
      J1 = J1 - 1
      CPI(J) = A1(J1)
   14 CPIP1(J) = (J1-1)*CPI(J)
      JMAXIP1 = JMAXI-1
      P0 = CPI(JMAXI)
      PD = A2(1)
      P10 = CPIP1(JMAXIP1)
      P1D = A2(2)
      IF(P10 .EQ. 0.0) GO TO 15
      IF(P0*P10 .LT. 0.0) 15, 16
   15 IV2 = IV2 + 1
   16 IF(P1D .EQ. 0.0) GO TO 17
      IF(PD*P1D .LT. 0.0) 17, 18
   17 IV2 = IV2 - 1
   18 FAC = CPI(1)/CPIP1(1)
```

```
      ZERO = ABS(CPI(1))*1.0E-11
      DO 19 J1=2, JMAXIP1
   19 CPI(J1) = CPI(J1) - FAC*CPIP1(J1)
      NOZERO = 1
      DO 20 J=2, JMAXI
      IF(ABS(CPI(J)) .LT. ZERO) 20, 21
   20 NOZERO = NOZERO + 1
      GO TO 99
   21 JMAXI = JMAXI - NOZERO
      DO 22 J=1, JMAXI
   22 CPI(J) = CPI(J+NOZERO)
      IF(JMAXI .LT. JMAXIP1) 23, 18
   23 JMIN = JMAXI + 1
      DO 24 J=JMIN, JMAXIP1
   24 CPI(J) = 0.0
      DO 25 J=1, JMAXIP1
      TEMP = CPI(J)
      CPI(J) = CPIP1(J)
   25 CPIP1(J) = -TEMP
      JTEMP = JMAXI
      JMAXI = JMAXIP1
      JMAXIP1 = JTEMP
      P0 = P10
      PD = P1D
      P10 = CPIP1(JMAXIP1)
      P1D = CPIP1(1)
      IF(JMAXIP1 .EQ. 1) 28, 26
   26 DO 27 J1=2, JMAXIP1
   27 P1D = P1D*D + CPIP1(J1)
   28 IF(P10 .EQ. 0.0) GO TO 29
      IF(P0*P10 .LT. 0.0) 29, 30
   29 IV2 = IV2 + 1
   30 IF(P1D .EQ. 0.0) GO TO 31
      IF(PD*P1D .LT. 0.0) 31, 35
   31 IV2 = IV2 - 1
   35 IF(JMAXIP1 .EQ. 1) 99, 18
   99 IV2 = ISIGN*IV2
      IF(IV2 .LT. 0) 32, 34
   32 IERROR = 5
      PRINT 33
   33 FORMAT( *    IV2 IS LESS THAN 0.   *)
   34 RETURN
      END
      SUBROUTINE YYY(RMIN, RMAX, DRMIN, DVMAX, ATOPFAC, I2, IERROR,
     1     XLAMWKB, LWKB, KWKB, ZZZFAC)
      DIMENSION XLAMWKB(1), LWKB(1)
      COMMON/AIMAX/  AF(51,402), JMAXF(402), IMAX,
     1     IMAXF, IMAXFP1, IMAXB, IMAXBP1, C(3,400),
     2     CF(3,402), ATOP(401), ATOPF(402)
      DIMENSION AB(51,402), JMAXB(402), CB(3,402), ATOPB(402)
      EQUIVALENCE(AF(1), AB(1)), (JMAXF(1), JMAXB(1)),
     1     (CF(1), CB(1)), (ATOPF(1), ATOPB(1))
      DIMENSION A(3, 4)
      DIMENSION XLAMDIM(10), XLDIM(10)
      COMMON/CB3/ Z, ZM1, RR2, RR, A0
    2 FORMAT(/////)
      IERROR = 0
      XLAMMAX = 0.0
      LMAX = 0
      DO 65 I=1, 10
      XLAMDIM(I) = 0.0
   65 XLDIM(I) = I*(I-1)
```

```
      DO 66 I=1, KWKB
      IF(XLAMWKB(I) .LT. XLAMMAX) XLAMMAX = XLAMWKB(I)
      IF(LWKB(I) .GT. LMAX) LMAX = LWKB(I)
      J = LWKB(I) + 1
      IF(XLAMWKB(I) .LT. XLAMDIM(J)) XLAMDIM(J) = XLAMWKB(I)
   66 CONTINUE
      LMAXP1 = LMAX + 1
      XLDIM(LMAX + 2) = LMAXP1*(LMAX + 2)
      LMAX = LMAX + 3
      XLAMDIM(LMAX) = ZZZFAC*ABS(XLAMMAX)
      XLDIM(LMAX) = 0.0
   68 FORMAT(2E20.10)
      DO 69 L=1, LMAX
   69 PRINT 68, XLAMDIM(L), XLDIM(L)
      IF(RR2 .GT. 0.0) GO TO 62
      IMAX = 1
      DR = ABS(RR2)
      ATOP(2) = DR
      C(1,1) = 0.0
      C(2,1)= -1.0
      C(3,1) = 0.0
   63 DR = ATOPFAC*ATOP(IMAX+1)
      IF(DR .GT. DRMAX) DR = DRMAX
      IMAX = IMAX + 1
      ATOP(IMAX+1) = ATOP(IMAX) + DR
      C(1,IMAX) = 0.0
      C(2,IMAX) = -1.0
      C(3,IMAX) = 0.0
      IF(ATOP(IMAX+1) .GE. RMAX) 64, 63
   64 ATOP(IMAX+1) = RMAX
      I2 = IMAX
      RETURN
   62 IMAX = 0
      I2 = 0
      IREG1 = 2
      IREG2 = 1
      DRAF = 1.0 + ATOPFAC
      RBOT = RMIN
      IF(RR2 .GE. RMAX) 50, 51
   50 RTOP = RMAX
      IFINISH = 1
      GO TO 37
   51 RTOP = RR2
      IFINISH = 0
   37 DRTB = RTOP - RBOT
      R1 = .3*DRTB
      R2 = .6*DRTB
      CALL DRMAXSB(DRMAX, R1, R2, XLAMDIM, XLDIM, LMAX)
      IF(DRTB .LE. DRMAX) GO TO 21
      RTOP = RBOT + DRMAX
      IFINISH = 0
   21 IF(IREG2 .EQ. 3) 60, 59
   60 IF(I2 .EQ. 0) I2 = IMAX
      IMAX = IMAX + 1
      C(1, IMAX ) = 0.0
      C(2,IMAX) = -1.0
      C(3,IMAX) = 0.0
      GO TO 61
   59 DR = RTOP - RBOT
      DDR = DR/9.0
      IF(DR .LT. DRMIN) 22, 23
   22 PRINT 3
```

```
    3 FORMAT(* AN INTERVAL ALONG THE R-AXIS GOT TOO SMALL.*)
      PRINT 4, DR, DRMIN
    4 FORMAT(* DR =*,E20.10,*,*,10X,*DRMIN =*,E20.10)
      IERROR = 21
    6 FORMAT (*  IERROR =*, I5)
      PRINT 6, IERROR
      PRINT 2
      GO TO 99
   23 R1 = RBOT    $   R3 = RTOP    $   R2 = (R1 + R3)/2.0
      IF(R1 .LT. 1.0E-200) 33, 34
   33 A(3,4) = 0.0    $   GO TO 35
   34 A(3,4) = R1*R1*V(R1)
   35 A(2,4) = R2*R2*V(R2)
      A(1,4) = R3*R3*V(R3)
      A(1,1) = R3*R3    $   A(2,1) = R2*R2    $   A(3,1) = R1*R1
      A(1,2) = R3    $   A(2,2) = R2    $   A(3,2) = R1
      A(1,3) = 1.0    $   A(2,3) = 1.0    $   A(3,3) = 1.0
      CALL MATPAC(-1, A, 3, 1, DET, 0.0, IF SING)
      CHAT1 = A(1,4)    $   CHAT2 = A(2,4)    $   CHAT3 = A(3,4)
C**********************************************************************
C**********************************************************************
C        R=0 IS NEVER USED.  WE WILL ONLY BE WORKING WITH VS
C      WHERE V(R) .NE. 0 FOR R .NE. 0.
C**********************************************************************
      R = RBOT
      DO 24 J=1, 8
      R = R + DDR
      VR = V(R)
      P = CHAT1 + CHAT2/R + CHAT3/(R*R)
      XJ = J
C   1 FORMAT(4E20.10)
C     PRINT 1, XJ, R, VR, P
      IF(ABS((VR - P)/VR) .LT. DVMAX) 24, 25
   24 CONTINUE
C**********************************************************************
      IREG2 = IREG1
      IMAX = IMAX + 1
      C(1,IMAX) = CHAT1    $   C(2,IMAX) = CHAT2    $   C(3,IMAX) = CHAT3
   61 ATOP(IMAX + 1) = RTOP
      IF (IFINISH .EQ. 1) 99, 31
   31 IF(IMAX .LT. 400) 28, 32
   32 PRINT 7
    7 FORMAT(* THE MAXIMUM NUMBER OF INTERVALS WILL NOT SPAN (RMIN, RMAX
     1).*)
      IERROR = 22
      PRINT 6, IERROR
      PRINT 2
      GO TO 99
   28 RBOT = RTOP
      IF(IREG1 .EQ. 3) 58, 57
   58 DRTOP = ATOPFAC*RTOP
      R1 = RTOP
      R2 = RTOP + .5*(RTOP - ATOP(IMAX))
      CALL DRMAXSB(DRMAX, R1, R2, XLAMDIM, XLDIM, LMAX)
      IF(DRTOP .GT. DRMAX) DRTOP = DRMAX
      RTOP = RTOP + DRTOP
      GO TO 52
C**********************************************************************
C**********************************************************************
C           THIS CARD DETERMINES MAXIMUM INTERVAL LENGTH.
C**********************************************************************
   57 IF(IMAX .EQ. 1) 40, 41
   40 DR1 = DR
      DR2 = DR
      DR3 = DR
      DR4 = DR
      DRTMP = DR
      GO TO 42
   41 DR5 = DR4
      DR4 = DR3
      DR3 = DR2
      DR2= DR1
      DR1 = DR
      DRTMP = (DR1 + DR2 + DR3 + DR4 + DR5)/5.0
   42 IF(2.0*DRTMP .LT. ATOPFAC*RTOP) 46, 47
   46 RTOP = RTOP + 2.0*DRTMP
      GO TO 48
   47 RTOP = OPAF*RTOP
   48 R1 = RBOT
      R2 = RBOT + .5*(RBOT - ATOP(IMAX))
      CALL DRMAXSB(DRMAX, R1, R2, XLAMDIM, XLDIM, LMAX)
      IF(RTOP - RBOT .GT. DRMAX) RTOP = RBOT + DRMAX
      GO TO (54, 53) , IREG1
   54 IF(RTOP .GE. RR2) 55, 52
   55 RTOP = RR2
      IREG1 = 2
      GO TO 52
   53 IF(RTOP .GE. RR) 56, 52
   56 RTOP = RR
      IREG1 = 3
C***********************************************************************
   52 IF(RMAX .LE. RTOP) 29, 26
   29 RTOP = RMAX   $   IFINISH = 1
      GO TO 21
   25 IREG1 = IREG2
      IF(IMAX .EQ. 0) 43, 44
   44 IF( DR .LT. .5*DRTMP) 43, 45
   43 RTOP = R2
      GO TO 26
   45 RTOP = RBOT + .75*DR
   26 IFINISH = 0
      GO TO 21
   99 IF(I2 .EQ. 0) I2 = IMAX
      RETURN
      END
      SUBROUTINE DRMAXSB(DRMAX, R1, R2, XLAMDIM, XLDIM, LMAX)
      DIMENSION XLAMDIM(1), XLDIM(1)
      SMALLK = 0.0
      R12 = 1.0/R1**2
      R22 = 1.0/R2**2
      TVR1 = 2.0*V(R1)
      TVR2 = 2.0*V(R2)
      DO 70 L=1, LMAX
      SK = ABS(XLAMDIM(L) - TVR1 - XLDIM(L)*R12)
      IF(SK .GT. SMALLK) SMALLK = SK
      SK = ABS(XLAMDIM(L) - TVR2 - XLDIM(L)*R22)
      IF(SK .GT. SMALLK) SMALLK = SK
   70 CONTINUE
      DRMAX = 6.28/SQRT(SMALLK)
      RETURN
      END
      FUNCTION V1(R)
      COMMON/CB3/ Z, ZM1, RR2, RR, A0
      FNZZ = 1.0/(1.0 + A0*RR2)
      FNZ =  FNZZ**2*(2.0*A0*RR2*FNZZ + 1.0)*ZM1                         DEGA     01056
      A1 = RR/RR2*(ZM1/FNZ*FNZZ**2 - 1.0) + 1.5                          DEGA     01057
      VV = ZM1/RR                                                        DEGA     01058
      A0RR = A0*RR                                                       DEGA     01059
      FNZSRRM = -FNZ/RR                                                  DEGA     01060
      ENTRY V                                                            DEGA     01061
      IF(R .GT. RR) GO TO 3                                              DEGA     01062
      X1 = R/RR                                                          DEGA     01063
      IF(R .GT. RR2) GO TO 2                                             DEGA     01064
      V1 = VV/(X1*(1.0 + A0RR*X1)**2)                                    DEGA     01065
      V1 = -V1 + FNZSRRM*(X1**2/2.0 - A1) - 1.0/R                        DEGA     01066
      RETURN                                                             DEGA     01067
    2 V1 = FNZSRRM*(1.0/X1 + X1**2/2.0 - 1.5) - 1.0/R                    DEGA     01068
      RETURN                                                             DEGA     01069
    3 V1 = -1.0/R                                                        DEGA     01070
      RETURN                                                             DEGA     01071
      END                                                                DEGA     01072
      SUBROUTINE MATPAC (IJOB, A, N, M, DET, EP, IF SING)                DEGA     01073
      DIMENSION A(3, 4)                                                  DEGA     01074
      IF SING = 0                                                        DEGA     01075
      DET = 1.                                                           DEGA     01076
      NP1 = N+1                                                          DEGA     01077
      NPM = N+M                                                          DEGA     01078
      NM1 = N-1                                                          DEGA     01079
      IF(IJOB) 2, 1, 2                                                   DEGA     01080
    1 DO 3 I=1, N                                                        DEGA     01081
      NPI = N+I                                                          DEGA     01082
      A(I,NPI) = 1.                                                      DEGA     01083
      IP1 = I+1                                                          DEGA     01084
      IF(N - IP1) 2, 19, 19                                              DEGA     01085
   19 DO 3 J=IP1, N                                                      DEGA     01086
      NPJ = N+J                                                          DEGA     01087
      A(I, NPJ) = 0.                                                     DEGA     01088
    3 A(J, NPI) = 0.                                                     DEGA     01089
    2 DO 4 J=1, NM1                                                      DEGA     01090
      C =  ABS(A(J,J))                                                   DEGA     01091
      JP1 = J+1                                                          DEGA     01092
      DO 5 I=JP1, N                                                      DEGA     01093
      D =  ABS(A(I,J))                                                   DEGA     01094
      IF(C-D) 6,5,5                                                      DEGA     01095
    6 DET = -DET                                                         DEGA     01096
      DO 7 K=J, NPM                                                      DEGA     01097
      B = A(I,K)                                                         DEGA     01098
      A(I,K) = A(J,K)                                                    DEGA     01099
    7 A(J,K) = B                                                         DEGA     01100
      C = D                                                              DEGA     01101
    5 CONTINUE                                                           DEGA     01102
      IF( ABS(A(J,J))-EP) 14, 15, 15                                     DEGA     01103
   14 DET = 0.                                                           DEGA     01104
      IF(IJOB) 16, 16, 17                                                DEGA     01105
   16 IF SING = 1                                                        DEGA     01106
   17 RETURN                                                             DEGA     01107
   15 DO 4 I= JP1, N                                                     DEGA     01108
      CONST = A(I,J)/A(J,J)                                              DEGA     01109
      DO 4 K= JP1, NPM                                                   DEGA     01110
    4 A(I,K) = A(I,K) - CONST*A(J,K)                                     DEGA     01111
      IF( ABS(A(N,N)) - EP) 14, 18, 18                                   DEGA     01112
   18 DO 11 I=1, N                                                       DEGA     01113
   11 DET = DET*A(I,I)                                                   DEGA     01114
      IF(IJOB) 10, 10, 17                                                DEGA     01115
   10 DO 12 I=1, N                                                       DEGA     01116
      K = N-I+1                                                          DEGA     01117
      KP1 = K+1                                                         DEGA   01118
      DO 12 L=NP1, NPM                                                  DEGA   01119
      S= 0.                                                             DEGA   01120
      IF( N - KP1) 12, 20, 20                                           DEGA   01121
   20 DO 13 J=KP1, N                                                    DEGA   01122
   13 S = S+A(K,J)*A(J,L)                                               DEGA   01123
   12 A(K,L) = (A(K,L)-S)/A(K,K)                                        DEGA   01124
      RETURN                                                            DEGA   01125
      END                                                               DEGA   01126
      SUBROUTINE NORMPHI(CAPLAMB, L)                                    DEGA   01127
      COMMON/AIMAX/  AF(51,402), JMAXF(402), IMAX,                      DEGA   01128
     1      IMAXF, IMAXFP1, IMAXB, IMAXBP1, C(3,400),                   DEGA   01129
     2      CF(3,402), ATOP(401), ATOPF(402)                            DEGA   01130
      DIMENSION AB(51,402), JMAXB(402), CB(3,402), ATOPB(402)           DEGA   01131
      EQUIVALENCE(AF(1), AB(1)), (JMAXF(1), JMAXB(1)),                  DEGA   01132
     1      (CF(1), CB(1)), (ATOPF(1), ATOPB(1))                        DEGA   01133
      DIMENSION CC(101)                                                 DEGA   01134
      COMMON/SCRATCH/SCRATCH(604)                                       DEGA   01135
      EQUIVALENCE (SCRATCH(1), CC(1))                                   DEGA   01136
      SQAB = SQRT(ABS(CAPLAMB))                                         DEGA   01137
      IBKW = IMAXFP1 + 1                                                DEGA   01138
      EXPSQ = EXP(-SQAB*ATOPB(IBKW))                                    DEGA   01139
      CAPA = AB(1,IBKW)/EXPSQ                                           DEGA   01140
      GUNDA2 = (CAPA*EXPSQ)**2/(2.0*SQAB)                               DEGA   01141
      D = ATOPF(2)                                                      DEGA   01142
      CALL POLYMUL(AF(1,1), JMAXF(1), AF(1,1), JMAXF(1), CC, NC)        DEGA   01143
      L2 = 2*L                                                          DEGA   01144
      LD = L2 + NC                                                      DEGA   01145
      SUM = CC(NC)/LD                                                   DEGA   01146
      N = NC                                                            DEGA   01147
      NCM1 = NC - 1                                                     DEGA   01148
      DO 26 NN=1, NCM1                                                  DEGA   01149
      N = N - 1                                                         DEGA   01150
      LD = LD - 1                                                       DEGA   01151
   26 SUM = SUM*D + CC(N)/LD                                            DEGA   01152
      GUNDA2 = GUNDA2 + SUM*D**(L2 + 1)                                 DEGA   01153
      IF(IMAXF .EQ. 1) GO TO 27                                         DEGA   01154
      DO 20 I=2, IMAXF                                                  DEGA   01155
      D = ATOPF(I+1) - ATOPF(I)                                         DEGA   01156
      CALL POLYMUL(AF(1,I), JMAXF(I), AF(1,I), JMAXF(I), CC, NC)        DEGA   01157
      SUM = CC(NC)/NC                                                   DEGA   01158
      N = NC                                                            DEGA   01159
      NCM1 = NC - 1                                                     DEGA   01160
      DO 21 NN=1, NCM1                                                  DEGA   01161
      N = N - 1                                                         DEGA   01162
   21 SUM = SUM*D + CC(N)/N                                             DEGA   01163
   20 GUNDA2 = GUNDA2 + SUM*D                                           DEGA   01164
   27 DO 22 I=1, IMAXB                                                  DEGA   01165
      IBKW = IMAXFP1 + I                                                DEGA   01166
      D = ATOPB(IBKW + 1) - ATOPB(IBKW)                                 DEGA   01167
      CALL POLYMUL(AB(1,IBKW), JMAXB(IBKW),                             DEGA   01168
     1      AB(1,IBKW), JMAXB(IBKW), CC, NC)                            DEGA   01169
      SUM = -CC(NC)/NC                                                  DEGA   01170
      N = NC                                                            DEGA   01171
      NCM1 = NC - 1                                                     DEGA   01172
      DO 23 NN=1, NCM1                                                  DEGA   01173
      N = N - 1                                                         DEGA   01174
   23 SUM = SUM*D - CC(N)/N                                             DEGA   01175
   22 GUNDA2 = GUNDA2 + SUM*D                                           DEGA   01176
      GUNDA = SQRT(GUNDA2)                                              DEGA   01177
      DO 24 I=1, IMAXF                                                  DEGA   01178
      JJJ = JMAXF(I)                                                    DEGA   01179
      DO 24 J=1, JJJ
   24 AF(J,I) = AF(J,I)/GUNDA
      DO 25 I=1, IMAXB
      IBKW = IMAXFP1 + I
      JJJ = JMAXB(IBKW)
      DO 25 J=1, JJJ
   25 AB(J,IBKW) = AB(J,IBKW)/GUNDA
      RETURN
      END
      SUBROUTINE POLYMUL (A, LM, B, LN, C, LL)
      DIMENSION A(1), B(1), C(1)
      LL = LM + LN - 1
      MMIN = 1
      DO 1 L=1 ,LL
      IF(L .GT. LM) 3, 2
    2 MMAX = L
    3 IF(L .GT. LN) 5, 4
    4 NMAXP1 = L + 1
      GO TO 6
    5 MMIN = MMIN + 1
    6 C(L) = 0.0
      N = NMAXP1
      DO 1 M= MMIN , MMAX
      N = N - 1
    1 C(L) = C(L) + A(M)*B(N)
      RETURN
      END
      SUBROUTINE  MATELE(KPHIMAX, XLAMARY, LARY, ALFAARY, IMAXARY, I2,
     1      NSUBSHL, ISSMAX, NSHELL)
      DIMENSION JMAX(400), AM1S(400), AM2S(400)
      EQUIVALENCE (SCRATCH(1), ATOPIP1), (SCRATCH(2), C1),
     1      (SCRATCH(3), C2), (SCRATCH(4),C3), (SCRATCH(5), JMAX(1)),
     2      (SCRATCH(405),AM1S(1)), (SCRATCH(805), AM2S(1)).
      DIMENSION XLAMARY(1), LARY(1), ALFAARY(1), IMAXARY(1)
     1      ,NSUBSHL(1), NSHELL(1)
      COMMON/AIMAX/ A(51,100), CAPO(100), IFSTOP(100),
     1      XLLP1(100), S1(300), S2(300), DVPOLY(70), SCRATCH(1204)
     2      , A1M(400), A2M(400), AMT(51), AMS(51)
     3      , A1N(400), A2N(400), ANT(51), ANS(51)
     4      , ATOPP1M(400), ATOPP1N(400), JMAXM(400), JMAXN(400)
      COMMON/CB1/ HNUVEC(500), ACOFVEC(500), NOPTS, IHNUMAX, MM(100),
     1      NN(100), MNMAX, I4MIN, I4MAX
      COMMON/EPS/EPSCONV, FBMAX, EM16, EMATELE, EMDVDD, MAXDIM
    1 FORMAT(16I5)
    2 FORMAT (25I5)
    3 FORMAT(5E25.14)
    4 FORMAT(2E25.14, I5)
    5 FORMAT(/)
    6 FORMAT (4E25.14, 2I5)
      PRINT1, MNMAX
      PRINT 1, (MM(MN), NN(MN), MN=1, MNMAX)
      KPHIM4 = KPHIMAX + 4
      MAXDIM4 = MAXDIM + 4
      ISKIP = 3*MAXDIM + 4
      IECS = 1
      IECS1 = IECS + MAXDIM4
      IECS2 = IECS1+ MAXDIM
      CALL ECRD (SCRATCH(1), IECS, KPHIM4, IE)
      CALL ECRD(SCRATCH(405), IECS1, KPHIMAX, IE)
      CALL ECRD(SCRATCH(805), IECS2, KPHIMAX, IE)
      DO 31 KPHI=1, KPHIMAX
      A(1,KPHI) = AM1S(KPHI)
```

```
 31 A(2,KPHI) = AM2S(KPHI)
    DO 21 KPHI=1, KPHIMAX
    XL = LARY(KPHI)
    IERROR = 0
    CALL TAYLORI (A(1,KPHI), JMAX(KPHI), ATOPIP1,
   1      XLAMARY(KPHI), XL, C1, C2, EM16, IERROR)
    JMAXKP = JMAX(KPHI)
 21 XLLP1(KPHI) = XL*(XL + 1.0)
    DO 20 MN=1, MNMAX
    M = MM(MN)
    N = NN(MN)
    LNLM1 = LARY(N)+ LARY(M) + 1
    DEOM = LNLM1 - 1
    CALL POLYMUL (A(2,M), JMAX(M)-1, A(2,N), JMAX(N)-1, S2, IS2)
    DO 30 J=1, IS2
    DEOM = DEOM + 1.0
 30 S2(J) = S2(J)/DEOM
    CALL POLYOP (S2, IS2, ATOPIP1, P)
 20 CAPO(MN) = -C2*P*ATOPIP1**LNLM1
    DO 36 MN=1, MNMAX
 36 IFSTOP(MN) = 0
    DO 24 I=2, I2
    ATOP1 = ATOPIP1
    IECS = IECS + ISKIP
    IECS1 = IECS + MAXDIM4
    IECS2 = IECS1 + MAXDIM
    CALL ECRD (SCRATCH(1), IECS, KPHIM4, IE)
    CALL ECRD(SCRATCH(405), IECS1,KPHIMAX, IE)
    CALL ECRD(SCRATCH(805), IECS2, KPHIMAX, IE)
    DO 25 KPHI=1, KPHIMAX
    A(1, KPHI) = AM1S(KPHI)
 25 A(2, KPHI) = AM2S(KPHI)
    D = ATOP1 - ATOPIP1
    DO 27 KPHI=1, KPHIMAX
    IERROR = 0
 27 CALL TAYLORS(A(1,KPHI), JMAX(KPHI), ATOPIP1,
   1      D, XLAMARY(KPHI), XLLP1(KPHI), C1, C2, C3, EM16, IERROR)
    CALL DVDD(ATOPIP1, D, C2, C3, DVPOLY, IDVDD, EMDVDD)
    MLAST = 0
    DO 28 MN=1, MNMAX
    IF(IFSTOP(MN) .EQ. 5) GO TO 28
    M= MM(MN)
    N=NN(MN)
    IF(M .EQ. MLAST) GO TO 33
    MLAST = M
    CALL POLYMUL(A(1,M), JMAX(M), DVPOLY, IDVDD, S1, IS1)
 33 CALL POLYMUL(A(1,N), JMAX(N), S1, IS1, S2, IS2)
    CALL POLYINT(S2, IS2, D, P)
    CAPO(MN) = CAPO(MN) - P
    IF(ABS(P) .LT. ABS(CAPO(MN))*EMATELE) 34, 35
 34 IFSTOP(MN) = IFSTOP(MN) + 1
    GO TO 28
 35 IFSTOP(MN) = 0
 28 CONTINUE
 24 CONTINUE
    ATOP12 = ATOPIP1
    ISKIP = 3*MAXDIM + 4
    IJUMP = 400 - I2
    MLAST = 0
    NLAST = 0
    DO 40 MN=1, MNMAX
    IF(IFSTOP(MN) .EQ. 5) GO TO 40
```

```
      M = MM(MN)
      N = NN(MN)
      IF(M .EQ. MLAST) 42, 41
   41 MLAST = M
      IMMAX = IMAXARY(M) - I2
      IF(IMMAX .EQ. 0) GO TO 60
      IECS = 4*(400 - I2)
      IECS = IECS*(M - 1) + 1
      IECS = IECS + I2*ISKIP
      CALL ECRD(ATOPP1M(1), IECS, IMMAX, IE)
      IECS = IECS + IJUMP
      CALL ECRD(JMAXM(1), IECS, IMMAX, IE)
      IECS = IECS + IJUMP
      CALL ECRD(A1M(1), IECS, IMMAX, IE)
      IECS = IECS + IJUMP
      CALL ECRD(A2M(1), IECS, IMMAX, IE)
   60 CONTINUE
   42 IF(N .EQ. NLAST) 45, 44
   44 NLAST = N
      INMAX = IMAXARY(N) - I2
      IF(INMAX .EQ. 0) GO TO 61
      IECS = 4*(400 - I2)
      IECS = IECS*(N - 1) + 1
      IECS = IECS + I2*ISKIP
      CALL ECRD(ATOPP1N(1), IECS, INMAX, IE)
      IECS = IECS + IJUMP
      CALL ECRD(JMAXN(1), IECS, INMAX, IE)
      IECS = IECS + IJUMP
      CALL ECRD(A1N(1), IECS, INMAX, IE)
      IECS = IECS + IJUMP
      CALL ECRD(A2N(1), IECS, INMAX, IE)
   61 CONTINUE
   45 IM = 1
      IN = 1
      ATOP1P1 = ATOPI2
      ALASTM = ATOPI2
      ALASTN = ATOPI2
      IMFIRST = 0
      INFIRST = 0
      CALL FINDATP(IM, IMMAX, XLLP1(M), ALASTM, ATOPP1M,
     1     XLAMARY(M), IERROR)
      IF(IERROR .EQ. 14) GO TO 100
      CALL FINDATP(IN, INMAX, XLLP1(N), ALASTN, ATOPP1N,
     1     XLAMARY(N), IERROR)
      IF(IERROR .EQ. 14) GO TO 100
   47 CONTINUE
      ATOPI = ATOPIP1
      IF(ABS(ALASTM-ALASTN) .LT. (ALASTM+ALASTN)*.5E-12 ) 48, 49
   48 INFIRST = 0
      IMFIRST = 0
      ATOPIP1 = .5*(ALASTM + ALASTN)
      CALL FINDA12(IM, IMMAX, XLAMARY(M), ALFAARY(M), XLLP1(M),
     1     ATOPIP1, A1M, A2M, JMAXM, AMS, JMSMAX, IERROR)
      IF(IERROR .EQ. 12) GO TO 100
      CALL FINDA12(IN, INMAX, XLAMARY(N), ALFAARY(N), XLLP1(N),
     1     ATOPIP1, A1N, A2N, JMAXN, ANS, JNSMAX, IERROR)
      IF(IERROR .EQ. 12) GO TO 100
      IM = IM + 1
      IN = IN + 1
      CALL FINDATP(IM, IMMAX, XLLP1(M), ALASTM, ATOPP1M,
     1     XLAMARY(M), IERROR)
      IF(IERROR .EQ. 14) GO TO 100
      CALL FINDATP(IN, INMAX, XLLP1(N), ALASTN, ATOPP1N,
     1     XLAMARY(N), IERROR)
      IF(IERROR .EQ. 14) 100, 50
   49 IF(ALASTM .GT. ALASTN) 51, 52
   51 INFIRST = 0
      IMFIRST = IMFIRST + 1
      ATOPIP1 = ALASTN
      DM = ATOPIP1 - ALASTM
      IF(IMFIRST .EQ. 1) 53, 54
   53 CALL FINDA12(IM, IMMAX, XLAMARY(M), ALFAARY(M),
     1     XLLP1(M), ALASTM, A1M, A2M, JMAXM, AMT, JMTMAX, IERROR)
      IF(IERROR .EQ. 12) GO TO 100
      IERROR = 0
      CALL TAYLORS(AMT, JMTMAX, ALASTM, DM, XLAMARY(M),
     1     XLLP1(M), 0.0, -1.0, 0.0, EM16, IERROR)
      IF(IERROR .EQ. 10) 100, 54
   54 CALL POLY0P(AMT, JMTMAX, DM, AMS(1))
      CALL POLY1P(AMT, JMTMAX, DM, AMS(2))
      JMSMAX = 0
      CALL FINDA12(IN, INMAX, XLAMARY(N), ALFAARY(N), XLLP1(N),
     1     ATOPIP1, A1N, A2N, JMAXN, ANS, JNSMAX, IERROR)
      IF(IERROR .EQ. 12) GO TO 100
      IN = IN + 1
      CALL FINDATP(IN, INMAX, XLLP1(N), ALASTN, ATOPP1N,
     1     XLAMARY(N), IERROR)
      IF(IERROR .EQ. 14) 100, 50
   52 IMFIRST = 0
      INFIRST = INFIRST + 1
      ATOPIP1 = ALASTM
      DN = ATOPIP1 - ALASTN
      IF(INFIRST .EQ. 1) 55, 56
   55 CALL FINDA12(IN, INMAX, XLAMARY(N), ALFAARY(N),
     1     XLLP1(N), ALASTN, A1N, A2N, JMAXN, ANT, JNTMAX, IERROR)
      IF(IERROR .EQ. 12) GO TO 100
      IERROR = 0
      CALL TAYLORS(ANT, JNTMAX, ALASTN, DN, XLAMARY(N),
     1     XLLP1(N), 0.0, -1.0, 0.0, EM16, IERROR)
      IF(IERROR .EQ. 10) 100, 56
   56 CALL POLY0P(ANT, JNTMAX, DN, ANS(1))
      CALL POLY1P(ANT, JNTMAX, DN, ANS(2))
      JNSMAX = 0
      CALL FINDA12(IM, IMMAX, XLAMARY(M), ALFAARY(M), XLLP1(M),
     1     ATOPIP1, A1M, A2M, JMAXM, AMS, JMSMAX, IERROR)
      IF(IERROR .EQ. 12) GO TO 100
      IM = IM + 1
      CALL FINDATP(IM, IMMAX, XLLP1(M), ALASTM,
     1     ATOPP1M, XLAMARY(M), IERROR)
      IF(IERROR .EQ. 14) 100, 50
   50 D = ATOPI - ATOPIP1
      IERROR = 0
      CALL TAYLORS(AMS, JMSMAX, ATOPIP1, D, XLAMARY(M),
     1     XLLP1(M), 0.0, -1.0, 0.0, EM16, IERROR)
      IF(IERROR .EQ. 10) GO TO 100
      CALL TAYLORS(ANS, JNSMAX, ATOPIP1, D, XLAMARY(N),
     1     XLLP1(N), 0.0, -1.0, 0.0, EM16, IERROR)
      IF(IERROR .EQ. 10) GO TO 100
      CALL DVDD(ATOPIP1, D, -1.0, 0.0, DVPOLY, IDVDD, EMDVDD)
      CALL POLYMUL(AMS, JMSMAX, DVPOLY, IDVDD, S1, IS1)
      CALL POLYMUL(ANS, JNSMAX, S1, IS1, S2, IS2)
      CALL POLYINT(S2, IS2, D, P)
      CAPO(MN) = CAPO(MN) - P
      IF(ABS(P) .LT. ABS(CAPO(MN))*EMATELE) 58, 57

      PRINT 7, ISSMAX
```

```
 58 IFSTOP(MN) = IFSTOP(MN) + 1
    IF(IFSTOP(MN) .EQ. 5) 40, 47
 57 IFSTOP(MN) = 0
    GO TO 47
100 PRINT 101
101 FORMAT(*     ERROR IN REGION 3*)
 40 CONTINUE
    DO 29 MN=1, MNMAX
    CAPO(MN) = 2.0*CAPO(MN)
 29 PRINT 3, CAPO(MN)
 64 FORMAT(E24.14, * EV*, I5, E24.14, * EV* , I5,
  1       E24.14, * EV*, E24.14, * BARNS/ATOM*)
 65 FORMAT(E24.14, * BARNS/ATOM*)
 66 FORMAT(/)
    DO 62 MN=1, MNMAX
    M = MM(MN)
    N = NN(MN)
    DE = ABS(XLAMARY(M) - XLAMARY(N))
    HNU = DE*13.605
    DE3 = DE**3
    LMAX = LARY(M)
    IF(LARY(N) .GT. LMAX) LMAX = LARY(N)
THE GA AND NOE ARE ONLY GOOD HERE FOR THE BOUND-FREE CASE.
    MNSS = N
    IF(XLAMARY(M) .LT. 0.0) MNSS = M
    ISS = NSHELL(MNSS)*(NSHELL(MNSS) - 1)/2 + LARY(MNSS) + 1
    NOE = NSUBSHL(ISS)
    GA = 2*(2*LARY(MNSS) + 1)
    DSIGMA = 10.756E+6*NOE*LMAX*CAPO(MN)**2/(DE3*GA)
    I4 = I4MIN + MN - 1
    ACOFVEC(I4) = ACOFVEC(I4) + DSIGMA
    EM = XLAMARY(M)*13.605
    EN = XLAMARY(N)*13.605
 62 CONTINUE
    RETURN
    END
    SUBROUTINE ZZZ(XLAMWKB, LWKB, NWKB, KWKB, HNUVEC,
  1       NOPTS, IHNUMAX, DHNUI, ZZZFAC)
    COMMON/SCRATCH/SCRATCH(1204)
    DIMENSION XLAMWKB(1), LWKB(1), NWKB(1), HNUVEC(1)
    FAC = ALOG(10.0)
    DHNU = DHNUI
    DO 6 K=1, KWKB
  6 XLAMWKB(K) = ABS(XLAMWKB(K))
    CALL SORT1(KWKB, 2, XLAMWKB, SCRATCH, LWKB, NWKB)
    XLAMWKB(KWKB+1) = ZZZFAC*XLAMWKB(KWKB)
  3 IHNUMAX = 0
    DO 1 I=1, KWKB
    EPS = .01
    XNUM = ALOG10(ABS(XLAMWKB(I+1))) - ALOG10(ABS(XLAMWKB(I) + EPS))
    NOINT = XNUM/DHNU
    IF(NOINT .EQ. 0) NOINT = 1
    NOINT = NOINT + 1
    SDHNU = XNUM/NOINT
    IHNUMAX = IHNUMAX + 1
    IF(IHNUMAX .GT. NOPTS) 4, 5
  4 DHNU = 1.2*DHNU
    GO TO 3
  5 HNUVEC(IHNUMAX) = XLAMWKB(I) + EPS
    NOINTM1 = NOINT - 1
    DO 2 INT=1, NOINTM1
    IHNUMAX = IHNUMAX + 1
```

```
      IF(IHNUMAX .GT. NOPTS) GO TO 4
      XNUM = FAC*(ALOG10(ABS(XLAMWKB(I) + EPS)) + INT*SDHNU)
    2 HNUVEC(IHNUMAX) =  EXP(XNUM)
      IHNUMAX = IHNUMAX + 1
      IF(IHNUMAX .GT. NOPTS) GO TO 4
    1 HNUVEC(IHNUMAX) = XLAMWKB(I+1)
      DO 7 K=1, KWKB
    7 XLAMWKB(K) = -XLAMWKB(K)
      RETURN
      END
      SUBROUTINE CARSON(XLAMBDA, C2, MFAC, CAPR, XLLP1,
     1     CAPF, CAPFP, THETA, THETAP, IERROR)
      COMMON/PI/PI, TWOSPI
      IERROR = 0
      SMALLC = ABS(C2)
      SMALLK = SQRT(XLAMBDA)
      CAPM = SQRT(TWOSPI/SMALLK)
      CAPM = MFAC*CAPM
      AN = CAPM    $    BN = 0.0
      CSK = SMALLC/SMALLK    $    CSK2 = CSK*CSK
      ACON = XLLP1 + CSK2    $    BCON = -(CSK2 + XLLP1)
      DEOM = 1.0   $    TWOK = 2.0*SMALLK
      CAPA = AN    $    CAPB = BN    $    CAPAP = 0.0    $    CAPBP = 0.0
      DO 63 NP1 = 1, 15
      DEOM = DEOM*CAPR
      N = NP1 - 1
      XNP1 = NP1
      XNNP1 = N*XNP1
      TWONP1 = (2.0*N + 1.0)*CSK
      XDEOM = TWOK*XNP1
      ANP1 = ((ACON - XNNP1)*BN - TWONP1*AN)/XDEOM
      BNP1 = ((XNNP1 + BCON)*AN -TWONP1*BN)/XDEOM
      AADD = ANP1/DEOM    $    BADD = BNP1/DEOM
      APADD = XNP1*AADD    $    BPADD = XNP1*BADD
      CAPA = CAPA + AADD    $    CAPB = CAPB + BADD
      CAPAP = CAPAP + APADD    $    CAPBP = CAPBP + BPADD
      CAPAB = ABS(CAPA) + ABS(CAPB)
      CAPABP = ABS(CAPAP) + ABS(CAPBP)
      IF( ABS(AADD) .LT. CAPAB*1.0E-4 .AND.
     1     ABS(BADD) .LT. CAPAB*1.0E-4 .AND.
     2     ABS(APADD) .LT. CAPABP*1.0E-4 .AND.
     3     ABS(BPADD) .LT. CAPABP*1.0E-4) GO TO 62
      AN = ANP1
   63 BN = BNP1
      PRINT 64
   64 FORMAT(*   CAPA, CAPB, CAPAP, AND CAPBP DO NOT CONVERGE.*)
      IERROR = 12
      RETURN
C*************************************************************************
C      RUN THIS CODE FOR CAPR .GT. 1.0
C*************************************************************************
   62 CAPAP = -CAPAP/CAPR
      CAPBP = -CAPBP/CAPR
      CAPF2 = CAPA*CAPA + CAPB*CAPB
      CAPF = SQRT(CAPF2)
      CAPFP = (CAPA*CAPAP + CAPB*CAPBP)/CAPF
      THETA = SMALLK*CAPR + CSK*ALOG(CAPR) + ATAN2(CAPA, CAPB)
      THETAP = SMALLK + CSK/CAPR + (CAPAP*CAPB - CAPBP*CAPA)/CAPF2
      RETURN
      END
      SUBROUTINE TAYLOR1(A, JMAX, D, XLAMBDA, XL, C1, C2, EM16, IERROR)
      DIMENSION A(1)

      PRIN! 7, ISSMAX
      IERROR = 0
      DEOM = 0.0
      DDEOM = 2.0*XL
      B1 = 2.0*C1 - XLAMBDA
      B2 = 2.0*C2
      IF(JMAX .GT. 2) GO TO 40
      IFCONV = 0
      FAC = 1.0
      AMAX = A(2)
      AMAXM16 = AMAX*EM16
      DO 20 J=3, 50
      DDEOM = DDEOM + 2.0
      DEOM = DEOM + DDEOM
      JM1 = J-1
      A(J) = (B1*A(J-2) + B2*A(JM1))/DEOM
      FAC = FAC*D
      ABSA = JM1*ABS(A(J))*FAC
      IF(ABSA .GT. AMAX) 30, 31
   30 AMAX = ABSA
      AMAXM16 = AMAX*EM16
      GO TO 32
   31 IF(ABSA .LT. AMAXM16) 33, 32
   32 IFCONV = 0
      GO TO 20
   33 IF(IFCONV .EQ. 1) 35, 34
   35 JMAX = J
      RETURN
   34 IFCONV = 1
   20 CONTINUE
      IERROR = 9
      PRINT 9, IERROR
    9 FORMAT (I5,* TAYLOR SERIES DOES NOT CONVERGE IN SUBROUTINE TAYLORI
     1, DO-LOOP 20.*)
      RETURN
   40 DO 41 J=3, JMAX
      DDEOM = DDEOM + 2.0
      DEOM = DEOM + DDEOM
      JM1 = J-1
   41 A(J) = (B1*A(J-2) + B2*A(JM1))/DEOM
      RETURN
      END
      SUBROUTINE TAYLORS(A, JMAX, RR, D, XLAMBDA,
     1      XLLP1, C1, C2, C3, EM16, IERROR)
      DIMENSION A(1)
      IERROR = 0
      RR2 = RR*RR
      TWORR = 2.0*RR
      SB1= XLAMBDA - 2.0*C1
      SB2= -2.0*C2
      SB3= -(2.0*C3     + XLLP1)
      B1 = SB1*RR2 + SB2*RR + SB3
      B2 = SB1*TWORR + SB2
      B3 = SB1
      A(3)   = -B1*A(1)  /(2.0*RR2)
      A(4)   = -(B2*A(1)  +B1*A(2)  +2.0*TWORR*A(3)  )/(6.0*RR2)
      IF(JMAX .GT. 5) GO TO 60
      ABSD = ABS(D)
      FAC = 1.0
      AMAX = ABS(A(2))
      AMAXM16 = AMAX*EM16
      FAC = FAC*ABSD
      ABS3 = 2.0*ABS(A(3))*FAC
      IF(ABS3        .GT. AMAX) 54, 55
   54 AMAX = ABS3
      AMAXM16 = AMAX*EM16
   55 FAC = FAC*ABSD
      ABS4 = 3.0*ABS(A(4))*FAC
      IF(ABS4        .GT. AMAX) 56, 57
   56 AMAX = ABS4
      AMAXM16 = AMAX*EM16
   57 DO 26 J=6, 50 , 2
      JM1=J-1    $    JM2 = J-2    $    JM3 = J-3
      JM4 = J-4    $    JM5 = J-5
      DEOM = JM3*JM2*RR2
      A(JM1)    = -(B3*A(JM5)   +B2*A(JM4)   +(B1+JM5*JM4)*A(JM3)
     1              +JM4*JM3*TWORR*A(JM2)   )/DEOM
      DEOM = JM2*JM1*RR2
      A(J)    = -(B3*A(JM4)   +B2*A(JM3)   +(B1+JM4*JM3)*A(JM2)
     1             +JM3*JM2*TWORR*A(JM1)   )/DEOM
      FAC = FAC*ABSD
      ABSA1 = JM2*ABS(A(JM1))*FAC
      FAC = FAC*ABSD
      ABSA = JM1*ABS(A(J))*FAC
      IF(ABSA .GT. AMAX) 45, 44
   45 AMAX = ABSA
      AMAXM16 = AMAX*EM16
      IF(ABSA1 .GT. AMAX)46, 26
   46 AMAX = ABSA1    $    AMAXM16 = AMAX*EM16      $    GO TO 26
   44 IF(ABSA1 .GT. AMAX) 46, 47
   47 IF(ABSA .LT. AMAXM16) 48, 26
   48 IF(ABSA1 .LT. AMAXM16) 49, 26
   49 JMAX    = J
      RETURN
   26 CONTINUE
      IERROR = 10
      PRINT 10, IERROR, I
   10 FORMAT (2I5,* TAYLOR SERIES DOES NOT CONVERGE IN SUBROUTINE TAYLOR
     1S. DO-LOOP 26 IN LOOP 25.*)
      JMAX = 50
      IERROR = 0
    1 FORMAT(4E24.14)
      PRINT 1, AMAX, AMAXM16, ABSA, ABSA1
      RETURN
   60 DO 27 J=6, JMAX, 2
      JM1=J-1    $    JM2 = J-2    $    JM3 = J-3
      JM4 = J-4    $    JM5 = J-5
      DEOM = JM3*JM2*RR2
      A(JM1)    = -(B3*A(JM5)   +B2*A(JM4)   +(B1+JM5*JM4)*A(JM3)
     1              +JM4*JM3*TWORR*A(JM2)   )/DEOM
      DEOM = JM2*JM1*RR2
   27 A(J)    = -(B3*A(JM4)   +B2*A(JM3)   +(B1+JM4*JM3)*A(JM2)
     1             +JM3*JM2*TWORR*A(JM1)   )/DEOM
      RETURN
      END
      SUBROUTINE POLYOP( A, JMAX, D, P)
      DIMENSION A(1)
      M = JMAX
      P = A(M)
      JUP = M - 1
      IF(JUP .LT. 1) RETURN
      DO 1 J=1, JUP
      M = M- 1
    1 P = P*D + A(M)
      RETURN

      PRINT 7, ISSMAX                                          DEGA  00063
      ENTRY POLY1P
      M = JMAX
      P = (M-1)*A(M)
      JUP = M-2
      IF(JUP .LT. 1) RETURN
      DO 2 J=1, JUP
      M = M-1
    2 P = P*D + (M-1)*A(M)
      RETURN
      ENTRY POLYINT
      M = JMAX
      P = A(M)/M
      JUP = M-1
      IF(JUP .LT. 1) GO TO 4
      DO 3 J=1, JUP
      M = M-1
    3 P = P*D + A(M)/M
    4 P = P*D
      RETURN
      END
      SUBROUTINE DVDD( ATOPI, D, C2, C3, DVPOLY, IDVDD, EMDVDD)
      COMMON/SCRATCH/SCRATCH(604)
      DIMENSION DVPOLY(1)
      OATOPI = -1.0/ATOPI
      DN = 1.0
      IF(C3 .EQ. 0.0) 10, 20
   10 IDVDD = 1
      DVPOLY(1) = -C2/ATOPI**2
      EPSILON = ABS(DVPOLY(1))*EMDVDD
      TERMLST = DVPOLY(1)
   30 IDVDD = IDVDD + 1
      TERMLST = TERMLST*OATOPI
      DN = DN*D
      DVPOLY(IDVDD) = IDVDD*TERMLST
      IF(ABS(DVPOLY(IDVDD)*DN) .LT. EPSILON) 40, 30
   20 ISCH = 1
      ICNT1 = 1
      ICNT2 = 1
      SCRATCH(1) = -1.0/ATOPI**3
      EPSILON = ABS(SCRATCH(1))*EMDVDD
      TERMLST = SCRATCH(1)
   50 ISCH = ISCH + 1
      ICNT2 = ICNT 2 + 1
      ICNT1 = ICNT1 + ICNT2
      TERMLST = TERMLST*OATOPI
      DN = DN*D
      SCRATCH(ISCH) = ICNT1*TERMLST
      IF(ABS(SCRATCH(ISCH)*DN) .LT. EPSILON) 60, 50
   60 SCRATCH(499) = C2*ATOPI + 2.0*C3
      SCRATCH(500) = C2
      CALL POLYMUL(SCRATCH(1), ISCH, SCRATCH(499), 2, DVPOLY, IDVDD)
   40 RETURN
      END
      SUBROUTINE FINDA12(I, IMAX, XLAM, ALFA, XLLP1, ATOPIPI,
     1     A1, A2, JMAX, AS, JSMAX, IERROR)
      DIMENSION A1(1), A2(1), JMAX(1), AS(2)
      IERROR = 0
      IF(I .GT. IMAX) 2, 1
    1 AS(1) = A1(I)
      AS(2) = A2(I)
      JSMAX = JMAX(I)
      RETURN
```

```
2 CALL CARSON(XLAM, -1.0, 1, ATOPIP1, XLLP1,
 1      CAPF, CAPFP, THETA, THETAP, IERROR)
  IF(IERROR .EQ. 12) RETURN
  THETA1 = THETA + ALFA
  COSTHE1 = COS(THETA1)
  AS(1) = CAPF*COSTHE1
  AS(2) = CAPFP*COSTHE1 - CAPF*THETAP*SIN(THETA1)
  JSMAX = 0
  RETURN
  END
  SUBROUTINE FINDATP(IM, IMAX, XLLP1, ALAST, ATOPP1, XLAMB, IERROR)
  DIMENSION ATOPP1(1)
  IERROR = 0
  IF(IM .GT. IMAX) 2, 1
1 ALAST = ATOPP1(IM)
  RETURN
2 IF(XLAMB .GT. 0.0) 4, 3
3 IERROR = 14
  RETURN
4 DR = .2*ALAST
  R1 = 1.05*ALAST
  CALL DRMAXSB(DRMAX, ALAST, R1, XLAMB, XLLP1, 1)
  IF(DR .GT. DRMAX) DR = DRMAX
  ALAST = ALAST + DR
  RETURN
  END
```

```
PRINT 7, ISSMAX
```

```
 01/09/73 +LASL MCH1  .16 01/09/73 MACH. 1 ECS ON
18.56.54.J15WM2W  01/09/73
18.56.54.$JOB(NAME=J15WM, AC=J15D, USF=DEGAA, TL=
18.56.54.1, SC=156000, PL=200, MX=66,
18.56.54.$1     CL=U, PR=6, CAT=6, UA=8715C131)
18.56.55.ASSIGNMT, OLDPL(PLB,LF223L00,SHB)
18.56.55.65 ASSIGNED
18.56.55.LF223L00
18.56.55.ASSIGNMT, NEWPL(NLB,,SHB)
18.56.56.66 ASSIGNED
18.56.56.LD274L00
18.56.56.UPDATE(F,S)
18.57.07. DECK STRUCTURE CHANGED
18.57.07. DECK STRUCTURE CHANGED
18.57.14. UPDATING FINISHED
18.57.15.REWIND(SOURCE)
18.57.16.UPDATE(N,I=SOURCE)
18.57.38. UPDATING FINISHED
18.57.38.COPYSBF(COMPILE,OUTPUT)
18.57.43.CP  00005.336 SEC.
18.57.43.PP  00057.626 SEC.
18.57.43.SS  00127.002 SEC.
18.57.43.
18.57.43.
18.57.43.
18.57.43.          2W         2W         2W
```