LA-14349
Approved for public release;
distribution is unlimited.

Slant Path Distances Through Cells in Cylindrical Geometry and an Application to the Computation of Isophotes

Los Alamos
NATIONAL LABORATORY
EST. 1943

Funding is provided by the Department of Energy, Nonproliferation Office.

Los Alamos National Laboratory, an affirmative action/ equal opportunity employer, is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396.

LA-14349
Issued: December 2007

Slant Path Distances Through Cells in Cylindrical Geometry and an Application to the Computation of Isophotes

Rodney Whitaker
Eugene Symbalisty

Los Alamos
NATIONAL LABORATORY
EST. 1943

Slant Path Distances Through Cells in Cylindrical Geometry and an Application to the Computation of Isophotes

by
Rodney Whitaker and Eugene Symbalisty

an updated version of the original unpublished report by
Henry G. Horak and John W. Kodis

Contents

1 Introduction **1**

2 Slant Path Geometry **1**
2.1 Right Circular Cone 4
2.2 Right Circular Cylinder 4
2.3 Plane 5

3 The Subroutine HOWFAR **5**

4 The Subroutine WHERE **6**

5 The Calculation of Isophotes **7**

6 Current Work **10**
6.1 Preliminaries 10
6.2 Isophote Application 11
6.3 Particular Cases 14

7 Summary **19**

8 Acknowledgements **19**

9 Bibliography **20**

10 Appendix **21**
10.1 Sample input file 21
10.2 Code listing 22
10.3 OpenMP make file on a MAC G5 50
10.4 Make file for serial processor such as Sun 50

List of Figures

1 The position vector, $\mathbf{a}$, and the direction vector, $\mathbf{\Omega}$. 2
2 A quadrilateral mesh cell. 2
3 The incident, $I_\nu(0, \mathbf{\Omega})$, and emergent, $I_\nu(s, \mathbf{\Omega})$, intensities. 7
4 The observer, O, is located at (a_0, c_0). 9
5 Isophotes for 10 kt with 200-m HOB at 2 s. 10
6 Isophotes for a 1-kt burst at 50-m HOB and a time of 0.4 s, horizontal path 12
7 Isophotes for a 1-kt burst at 50-m HOB and a time of 0.4 s, and 45^0 path. 13
8 The Si band power vs time curve for 45^0 look angle. 14
9 Si flux versus time for the 10 kt, 240-m HOB at three elevation angles. . . 15
10 Si flux versus time for the 10 kt, 0-m HOB at three elevation angles. 16
11 Isophotes for a 10 kt burst at 0-m HOB and 0^0 elevation angle. 16
12 Isophotes for a 10-kt burst at 0-m HOB and 90^0 elevation angle. 17
13 Si flux versus time for the 10 kt, 0-m and 240-m HOB showing the delayed minimum time for the 0-m HOB case. 18

1 Introduction

This report is composed of two parts. The first is a restoration of the hardcopy draft of the original report by Horak and Kodis from 1983 that did not get to final form. The algorithm documented in that draft was in routine use in weapon effects calculations at Los Alamos since the mid-1970s. That draft was converted to LaTex format by Rod Whitaker and Eugene Symbalisty in late 2006, and C. Flaming made electronic versions of the original figures. We give first the text and figures of the original report (Sections 1 through 5) with new results starting after Section 5.

Abstract

In computer programs involving two-dimensional cylindrical geometry, it is often necessary to calculate the slant path distance in a given direction from a point to the boundary of a mesh cell. A subroutine, *HOWFAR*, has been written that accomplishes this, and is very economical in computer time. An example of its use is given in constructing the isophotes for a low altitude nuclear fireball.

Computer programs that solve problems in two-dimensional cylindrical geometry often must calculate slant path distances through mesh cells (see, e.g., Amsden and Hirt, 1973; Anderson and Sandford, 1974, Horak et al., 1982, and Lathrop and Brinkley, 1973). This is particularly important in Monte Carlo computations that follow the random walks of numerous statistical particles. A subroutine, *HOWFAR* has been written that efficiently calculates the slant path distance in a given direction from a point to the boundary of a mesh cell, and allows for the many special cases. *HOWFAR* has become very valuable in several radiation-hydrodynamics codes that are used at Los Alamos to follow the evolution of nuclear explosions in the earth's atmosphere (Anderson and Sandford, 1974, Horak et al., 1982).

In Section V the use of *HOWFAR* is illustrated to find slant optical distances through cells, and by using the formal solution of the equation of transfer to calculate isophotes (radiance contours) for a low-altitude nuclear fireball.

2 Slant Path Geometry

The basic formulas for calculating slant path distances through mesh cells in cylindrical geometry are readily obtained by using vector algebra. Refer to Figures 1 and 2. The cross section of a cell in the XZ-plane is the area bounded by four straight-line segments; the entire cell is the volume described by rotating this figure in a circle about the OZ-axis. This quadrilateral is assumed to be convex, with the four vertices numbered consecutively i = 1,2,3,4 in the counterclockwise sense as viewed from a point on the negative Y-axis. In Lagrangian hydrodynamic calculations the mesh cells can be subject to severe distortion, although provision is usually provided to prevent cell boundaries from becoming concave.

The position vector, $\mathbf{a}$, of a moving point at a given instant of time can be written

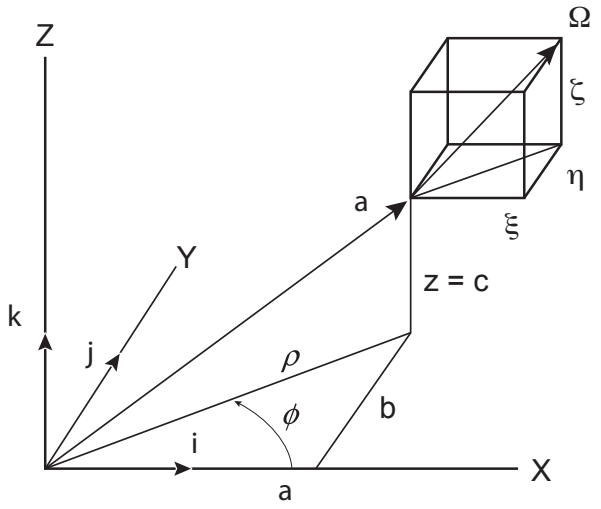

Figure 1: The position vector, $\mathbf{a}$, and the direction vector, $\mathbf{\Omega}$.

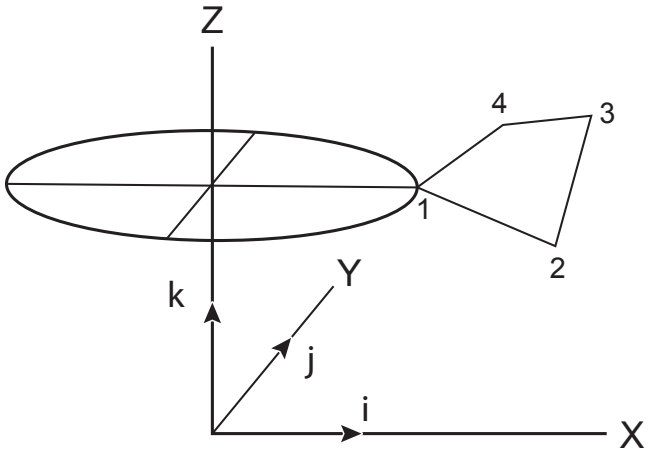

Figure 2: A quadrilateral mesh cell.

$$\mathbf{a} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k} \tag{1}$$

where **i**, **j**, and **k** are basic unit vectors in a right-handed orthogonal Cartesian coordinate system, and its direction of motion is given by the unit vector

$$\mathbf{\Omega} = \xi\mathbf{i} + \eta\mathbf{j} + \zeta\mathbf{k}. \tag{2}$$

The vector equation of the straight line segment originating at the point **a**, and extending in the direction $\mathbf{\Omega}$ is

$$\mathbf{r}(d) = \mathbf{a} + \mathbf{\Omega}d \qquad\qquad (d > 0), \tag{3}$$

where $\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ is the position vector of any point on the line segment, and d is the distance parameter. The geometric problem is to find the points of intersection of this line segment with the boundary surfaces of the mesh cells, viz., right circular cones and cylinders described about the OZ-axis, and planes perpendicular to the OZ-axis. It is assumed throughout that the moving point in question is located within a mesh cell, although it may be necessary to perform some computation in order to identify the precise cell (refer to Sec. IV).

Let (ρ, ϕ, z) be the cylindrical coordinates of a point P, and $\mathbf{x}_i, \mathbf{x}_i + \mathbf{dx}_i$ (i = 1,2,3,4) the position vectors in the XZ-plane of two consecutive vertices of a mesh cell, where

$\mathbf{x}_i = x_i\mathbf{i} + z_i\mathbf{k}$, and $\mathbf{dx}_i = dx_i\mathbf{i} + dz_i\mathbf{k}$. The equation of a right circular cone whose line of symmetry is the Z-axis, and which passes through the points $\mathbf{x}_i, \mathbf{x}_i + \mathbf{dx}_i$ is

$$\mathbf{r}(\rho, \phi) = \rho\ cos\phi\ \mathbf{i} + \rho\ sin\phi\ \mathbf{j} + (z_0 + \rho\ cot\alpha)\mathbf{k}, \tag{4}$$

where

$$z_0 = z_i - x_i\ cot\alpha, \tag{5}$$

and

$$cot\alpha = \frac{dz_i}{dx_i}. \tag{6}$$

α is the semivertex angle of the cone, and $z_0\mathbf{k}$ the position vector of the vertex. If $dx_i = 0$, the resulting equations describe a right circular cylinder:

$$\mathbf{r}(\phi, z) = \rho_0\ cos\phi\ \mathbf{i} + \rho_0\ sin\phi\ \mathbf{j} + z\ \mathbf{k}\ , \tag{7}$$

where

$$\rho_0 = x_i\ , \quad dx_i = 0. \tag{8}$$

The equation of a plane perpendicular to the z-axis, and that contains the points $\mathbf{x}_i, \mathbf{x}_i + dx_i\mathbf{i}$, is

$$\mathbf{r}(\rho, \phi) \cdot \mathbf{k} = z_0, \tag{9}$$

where

$$z_0 = z_i \ , \quad dz_i = 0. \tag{10}$$

The points of intersection $\mathbf{e}$ of the above surfaces with the straight line segment $\mathbf{r} = \mathbf{a} + \mathbf{\Omega} d$ are found by simultaneous solution, and the results are described below.

2.1 Right Circular Cone

For a right circular cone, we have

$$\begin{aligned} \mathbf{e} &= e\mathbf{i} + f\mathbf{j} + g\mathbf{k} && (11) \\ &= (a + \xi d)\mathbf{i} + (b + \eta d)\mathbf{j} + (c + \zeta d)\mathbf{k}, && (12) \end{aligned}$$

where

$$\begin{aligned} d &= \frac{-Q \pm \sqrt{(Q^2 - PR)}}{P} && (13) \\ P &= \xi^2 + \eta^2 - F^2 && (14) \\ Q &= a\xi + b\eta - EF && (15) \\ R &= a^2 + b^2 - E^2 && (16) \\ E &= (c - z_0)\ tan\alpha && (17) \\ F &= \xi\ tan\alpha && (18) \\ z_o &= z_i - x_i\ cot\alpha && (19) \\ cot\alpha &= \frac{dz_i}{dx_i}. && (20) \end{aligned}$$

2.2 Right Circular Cylinder

For the case of the right circular cylinder, the above equations for $\mathbf{e}$ and d apply with

$$\begin{aligned} P &= \xi^2 + \eta^2 && (21) \\ Q &= a\xi + b\eta && (22) \\ R &= a^2 + b^2 - x_i^2. && (23) \end{aligned}$$

2.3 Plane

For the case of the plane, equation 11 applies with

$$d = \frac{z_i - c}{\zeta}. \tag{24}$$

There are special cases that can arise and must be properly treated in the program; some of these will be discussed subsequently.

3 The Subroutine HOWFAR

A FORTRAN listing of the subroutine *HOWFAR* is given in the appendix. It will be necessary for the user to provide statements regarding the memory storage and location of coordinates, etc., and to set up the appropriate common blocks. Comments have been liberally inserted, but certain aspects require more discussion.

A test is made at the outset whether the point P with position vector $\mathbf{a}$ really lies inside the cell in which it is surmised to be located. Again let $\mathbf{x_i} = x_i\mathbf{i} + z_i\mathbf{k}$, and $\mathbf{x_i} + \mathbf{dx_i}$, where $\mathbf{dx_i} = dx_i\mathbf{i} + dz_i\mathbf{k}$, be the position vectors of any two consecutive mesh cell vertices in the XZ-plane. The plane POZ can be rotated about axis OZ into coincidence with the XOZ reference plane; thus P maps onto P' so that the position vector of P' is $\mathbf{a}' = \rho\mathbf{i} + c\mathbf{k}$, where $\rho = \sqrt{a^2 + b^2}$. The scalar triple product

$$C = [(\mathbf{a}' - \mathbf{x_i}) \times \mathbf{dx_i}] \cdot \mathbf{j}, \tag{25}$$

serves to indicate in which half of the XZ-plane, as divided by the line $\mathbf{x_i} + t\ \mathbf{dx_i}$ (t is a variable scalar), the point P' is to be found. If C is positive, P' is said to lie in the *left* half plane. If P' is to the *left* of all four cell sides ($\mathbf{dx_i} = \mathbf{x_2} - \mathbf{x_1}, \mathbf{x_3} - \mathbf{x_2}, \mathbf{x_4} - \mathbf{x_3}, \mathbf{x_1} - \mathbf{x_4}$ respectively), it is clearly inside the cell. If C is not positive for any one of the cell sides, then P', and therefore P, must lie outside the cell. In this latter case, the subroutine *WHERE* (see the next section) is called to identify the proper cell.

In order to find the distance from $P(\mathbf{a})$ in the direction $\mathbf{\Omega}$ to the emergence point, it is necessary to find the minimum positive distance among the intersections with the four cell surfaces. In doing this, whenever imaginary values of D occur, they need only be identified, and not calculated. The choice between two positive real roots is complicated because of a possible intersection with the *false cone*. The latter is that half of the cone not containing the two given mesh points, but nevertheless defined by the same second degree equation. There is a simple procedure that can be used in computations to separate true and false cone solutions. An intersection point $\mathbf{e}$ of the straight line and cone, obtained from equations 11 and 13, is given by $\mathbf{e} = \mathbf{a} + \mathbf{\Omega}d = e\mathbf{i} + f\mathbf{j} + g\mathbf{k} = \rho\mathbf{i}' + g\mathbf{k}$, where $\rho\mathbf{i}' = e\mathbf{i} + f\mathbf{j}$ and $\rho = \sqrt{e^2 + f^2}$. Also, the cone line-element through $\mathbf{e}$ passes through the point $\mathbf{P_i} = x_i\mathbf{i}' + z_i\mathbf{k}$ in the direction $\mathbf{dp_i} = dx_i\mathbf{i}' + dz_i\mathbf{k}$. The condition that $\mathbf{e}$ lies on this line-element is $(\mathbf{e} - \mathbf{p_i}) \times \mathbf{dp_i} \cdot \mathbf{j}' = 0$ with $\mathbf{j}' = \mathbf{k} \times \mathbf{i}'$, or simply

$$(\rho - x_i) - (g - z_i)(\frac{dx_i}{dz_i}) = 0 \quad , \tag{26}$$

which is equivalent to

$$(c - z_i + \zeta d)\ dx_i\ dz_i + x_i dz_i^2 = \rho dz_i^2. \tag{27}$$

Now the vector $\mathbf{i}'$ is always chosen for each intersection point in such a way that $\rho = \mathbf{e} \cdot \mathbf{i}'$ is positive, and as a consequence it can be verified that equation 27 is not satisfied by false cone solutions. In practice the most difficult cases arise when the semi-cone angle α is near 90 degrees, and the tolerances become severe.

If the direction vector $\mathbf{\Omega}$ lies in a plane through $\mathbf{a}$ containing the Z-axis, the condition being $\mathbf{a} \cdot \mathbf{\Omega} \times \mathbf{k} = a\eta - b\xi = 0$, the problem becomes that of finding the intersection of the given line with other straight lines (cone elements) in this plane. It can be readily shown that the distances from $\mathbf{a}$ to the two possible intersecting points with the cone elements are given by

$$d = \frac{(z_0 - c)\ cos\phi\ tan\alpha + \sqrt{a^2 + b^2}}{\zeta\ cos\phi\ tan\alpha - \mathbf{\Omega} \cdot \mathbf{P}}, \tag{28}$$

$$cos\phi = \pm 1, \tag{29}$$

where

$$\mathbf{a} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k} = \sqrt{a^2 + b^2}\mathbf{P} + c\mathbf{k} \tag{30}$$

$$\mathbf{P} = \frac{a\mathbf{i} + b\mathbf{j}}{\sqrt{a^2 + b^2}} \tag{31}$$

$$\mathbf{\Omega} = (\mathbf{\Omega} \cdot \mathbf{P})\mathbf{P} + \zeta\mathbf{k} \tag{32}$$

$$z_0 = z_i - x_i\ cot\alpha \tag{33}$$

$$tan\alpha = \frac{dx_i}{dz_i}. \tag{34}$$

False solutions can be discarded as described previously.

4 The Subroutine WHERE

The subroutine *WHERE* calculates the identity of the mesh cell (cell number or number pair) within which a point with given coordinates is located. *WHERE* carries out a systematic search procedure beginning with some cell in which the point is surmised to be located; the test based on equation 25 is used, although it is applied to triangles rather than quadrilaterals.

Consider a mesh quadrilateral in the XZ-plane with vertices $\mathbf{x_i}$ (i=1,2,3,4), and corresponding coordinates (x_i, z_i). The image point P' (position vector $\mathbf{a}'$) corresponding to the given point P (position vector $\mathbf{a} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$) has coordinates (ρ, c), where $\rho = \sqrt{a^2 + b^2}$. Construct the diagonal of the quadrilateral, $\mathbf{x_1} - \mathbf{x_3}$, and form the product $D_{13} = (\mathbf{x_1} - \mathbf{x_3}) \times (\mathbf{a}' - \mathbf{x_3}) \cdot \mathbf{j}$. If D_{13} is negative, $\mathbf{a}'$ lies outside triangle 1,3,4; then, if

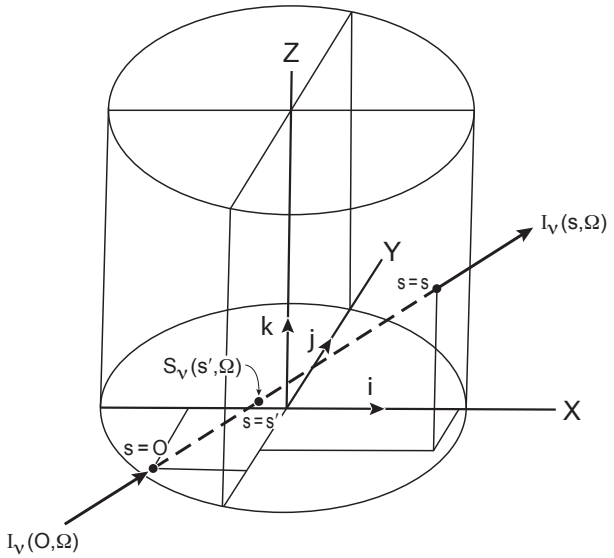

Figure 3: The incident, $I_\nu(0, \mathbf{\Omega})$, and emergent, $I_\nu(s, \mathbf{\Omega})$, intensities.

$D_{23} = (\mathbf{x_2} - \mathbf{x_3}) \times (\mathbf{a}' - \mathbf{x_3}) \cdot \mathbf{j}$ is negative, P' lies outside the quadrilateral, and the search proceeds to the adjoining cell that also contains side 3,2. If D_{23} is positive, it is necessary to form $D_{12} = (\mathbf{a}' - \mathbf{x_1}) \times (\mathbf{x_2} - \mathbf{x_1}) \cdot \mathbf{j}$; if D_{12} is negative, P' lies outside, and the search proceeds to the adjoining cell that also contains side 1,2. If D_{13} is positive, then the product $D_{34} = (\mathbf{a}' - \mathbf{x_3}) \times (\mathbf{x_4} - \mathbf{x_3}) \cdot \mathbf{j}$ is formed, and finally $D_{14} = (\mathbf{a}' - \mathbf{x_4}) \times (\mathbf{x_1} - \mathbf{x_4}) \cdot \mathbf{j}$, if required.

5 The Calculation of Isophotes

Consider the problem of calculating the emergent radiance field for a low-altitude nuclear explosion. The geometric form of such a fireball is initially spherical at very early times, evolving later to a toroid with a vertical axis. Radiation-Hydrodynamic codes using cylindrical geometry have been written at Los Alamos that compute the evolution of such explosions. At each time step in the evolution, the physical variables, temperature, density, hot-air absorption coefficients, etc., are calculated for each mesh cell. Such information is stored only for certain preselected times and is later recovered for post-processing. The program *ISOPHOT*, given in the appendix, is usually run in this mode. The radiance calculations can be accomplished with the same method of radiative transfer used in evolving the fireball, such as discrete ordinates; however, because *HOWFAR* is available to calculate the slant distance, s, through a mesh cell, the formal solution of the equation of transfer can be easily applied and should give the most accurate results possible with the given constraints created by the grid structure. Furthermore this gives an independent check on the transfer method adopted for the evolution computation. The

emergent radiation is given by (refer to Fig. 3):

$$I_\nu(s,\mathbf{\Omega}) = I_\nu(0,\mathbf{\Omega})e^{-\int_0^s k_\nu(s')ds'} + \int_0^s S_\nu(s',\mathbf{\Omega})e^{(-\int_{s'}^s k_\nu(s'')ds'')}k_\nu(s')ds' \tag{35}$$

where $I_\nu(0,\mathbf{\Omega})$ is the radiance of light of frequency ν in the direction $\mathbf{\Omega}$ at $s=0$, $k_\nu(s')$ is the absorption coefficient at s', and $S_\nu(s',\mathbf{\Omega})$ is the source function at s' in direction $\mathbf{\Omega}$. For thermal radiation the source function is the Planck function $B_\nu(T)$, and

$$S_\nu(s',\mathbf{\Omega}) = B_\nu[T(s')] = \frac{2h\nu^3}{c^2}\frac{1}{e^{h\nu/kT}-1} \quad , \tag{36}$$

where T is the absolute temperature. Equation 35 can be applied to a single mesh cell within which the source function is assumed to be constant, giving

$$I_\nu(s,\mathbf{\Omega}) = I_\nu(0,\mathbf{\Omega})e^{-k_\nu s} + B_\nu(T)(1-e^{-k_\nu s}) \quad , \tag{37}$$

where T and k_ν are appropriate for the cell, s is the slant distance through the cell calculated by *HOWFAR* and $I_\nu(0,\mathbf{\Omega})$ the radiance incident on the cell in direction $\mathbf{\Omega}$. This procedure can be applied along a chosen ray through the entire mesh, cell by cell. In order to obtain isophote contours, it is necessary to find emergent radiances in this fashion along many rays through the fireball to the observer's position (we routinely use about 1000 to 3000 rays). A contour-line plotting routine can then be used to produce isophote graphs. Finally, by integration over the field of isophotes the irradiance, D_ν, at the observer can be found,

$$D_\nu = \int_0^{\omega_0} I_\nu \; cos\theta \; d\omega. \tag{38}$$

I_ν is the radiance emitted in the direction of the observer by a surface area element, $d\Sigma$, of the source, ω_0 is the solid angle subtended by the source and $d\omega$ that subtended by $d\Sigma$ as viewed from the observer's position. Theta is the angle between the collimation axis of the camera, or photometer, and the axis of $d\omega$. Any convenient cubature formula can be used to evaluate the integral.

In the program *ISOPHOT*, given in the appendix, the following procedure is adopted (refer to Fig. 4). Let $\mathbf{a_0} = a_0\mathbf{i} + c_0\mathbf{k}$ be the position vector of the observer in the XZ-plane with respect to the initial center, C, of the mesh. The code first searches the mesh for the cell with the highest temperature. The vertex of this cell has coordinates (x_h, z_h) in the XZ-plane. A line is drawn from the observer $\mathbf{a_0}(a_o, c_0)$ to the point $C:\ \mathbf{H}(0, z_h)$ in the direction

$$\mathbf{\Omega}^* = \frac{(\mathbf{H_0} - \mathbf{a_0})}{|\mathbf{H_0} - \mathbf{a_0}|}. \tag{39}$$

Next, a plane is constructed through the point $\mathbf{H}(0, z_h)$ perpendicular to $\mathbf{\Omega}^*$. A rectangular cartesian coordinate system (coordinates y, l) is formed in this plane with the origin at C and axes in directions $\mathbf{j} = \mathbf{k} \times \mathbf{i}$ and $\mathbf{l} = \mathbf{j} \times \mathbf{\Omega}^*$. A grid is created in the JL-plane from $y = 0$ to $+R$ along the J-axis and from $l = -R$ to R along the L-axis, where $2R$

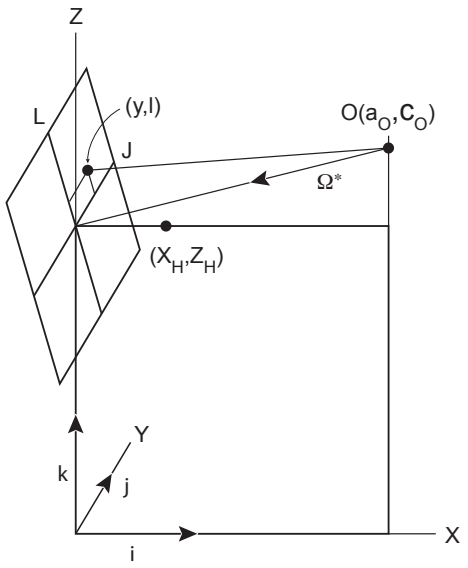

Figure 4: The observer, O, is located at (a_0, c_0).

is the maximum linear dimension of the fireball. The radiance values display symmetry with respect to the L-axis, because the physical parameters possess axial symmetry about the Z-axis. Therefore, it is only necessary to calculate radiances on the positive side of the L-axis; for this purpose, a grid can be used with $y_j/R = 0, 1, ...n(j = 1, ..., N + 1)$ and $l_k/R = -N, -N + 1, ... - 1, -, 1, ..., N(k = 1, ..., 2N + 1)$. The integration to find the irradiance D_ν can be performed in two parts over positive and negative l values: thus,

$$D_\nu = \frac{2A}{r_H^2} \Sigma_{j=1}^{N+1} \Sigma_{k=1}^{2N+1} \; w_j \; w_k \; I_{j,k}(\nu) \; cos^4\theta_{j,k} \tag{40}$$

where

$$r_H = |\mathbf{H} - \mathbf{a_0}| = \sqrt{a_0^2 + (c_0 - z_h)^2}, \tag{41}$$

A is the area of a grid cell,

$$cos\theta_{j,k} = \frac{r_H}{\sqrt{y_j^2 + l_k^2 + r_H^2}}, \tag{42}$$

and $I_{j,k}$ is the radiance in the direction of the observer at point (y_j, l_k), and w_j, w_k are appropriate quadrature weights.

Fig. 5 shows the isophotes computed in the fashion just described for a nominal 10-kt near-surface (altitude 200 m) nuclear explosion at the evolution time of 2 seconds. The fireball is beginning to develop into the form of a ring-shaped vortex, or toroid, under the action of the ground reflected shock. The observer is located at a large distance horizontally from the fireball. The two points of maximum radiance are labelled O,

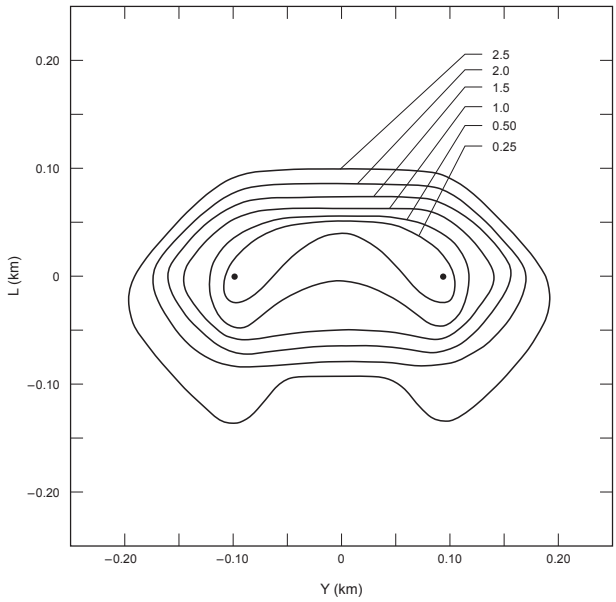

Figure 5: Isophotes for 10 kt with 200-m HOB at 2 s.

and the isophote contours are assigned relative values in the stellar-magnitude scale, $m = -2.5log[I(m)/I(0)]$, where m is is the magnitude associated with radiance $I(m)$, and $I(0)$ is the maximum radiance arbitrarily assigned magnitude zero.

6 Current Work

6.1 Preliminaries

In the work here I_ν, B_ν, and S_ν all have units of energy per unit area, per unit time, per steradian, and per Hz and have names of specific monochromatic intensity, the Planck function and the source function, respectively. The term radiance is also used for I_ν. The irradiance, D_ν, is the integral of I_ν over solid angle and has the units of a flux, energy per unit area, per unit time, per Hz. The intensity, I_ν, is the basic dependent variable in the radiative transfer equation.

In the late 1970s, the computational approach had been incorporated into the existing atmospheric effects code SnYAQUI that combined the finite difference hydrodynamic code YAQUI, Amsden and Hirt, (1973), with the discrete ordinate Sn algorithm for radiative transfer, (Lathrop, 1972, Lathrop and Brinkely, 1973). Over time, as computational resources shifted from a central computing facility with large mainframes, to distributed computing with increasingly powerful desktops, we have migrated our atmospheric effects codes. Along the way, we adopted the newer hydro package CAVEAT, Addessio et al. (1992) and coupled it to the Sn discrete ordinate code to make SnCAVEAT. In making

SnCAVEAT we included some new developments in the Sn method, Hill and Patternoster (1982). SnCAVEAT now runs on Sun Solaris Unix, with the f90 compiler; Mac OS X, with the Absoft and g95 compilers; and a Los Alamos National Laboratory high performance computing cluster, with the Portland Group f90 compiler.

The code SnCAVEAT is a 2-D program usually run in cylindrical geometry, (r, z). Not every calculation of interest needs 2-D; often a 1-D program can be used. Once the ground surface is encountered or when bouyant deformation begins, then a calculation must transition from 1-D to at least 2-D. (One could argue that everything should be 3-D, but we are not there yet.) A strategy often followed is to perform the calculation with a 1-D, spherically symmetric, code until just before 2-D effects begin. Then one maps the 1-D problem into the the 2-D mesh and continues the evolution with the 2-D code. The same equation of state and multi-group opacity tables are used for the 1-D and 2-D codes. Our 1-D code is based on the work of Zinn (1973) as programmed in the code HYCHEM. Additional information on the code can be found in the other Zinn references in the bibliography, as well as Symbalisty et al. (1995). Another version of the 1-D code, called RADFLO, is described in Horak and Kodis (1983). RADFLO does not include the detailed atmospheric chemistry capability found in HYCHEM; otherwise, they are quite similar.

The Appendix provides a sample input file, a code listing, and two sample makefiles. There are some parts of the Isophote processor that are specific to our SnCAVEAT program. For use with outputs from other numerical applications, a few comments may be helpful. One needs the numerical mesh coordinates, the specific internal energy (energy/mass), the mass density (mass/unit volume), opacity data, planck functions and equation of state (EOS) data. In our application, the coordinates are in the variable xv, specific internal energy in sie, mass density in rho, opacity data in uk, Planck functions in plb and equation of state data in gt and fp. In the listing, the include files *comdeck* and *comdeck1* hold a number of SnCAVEAT variables including those needed by the isophote code. These would be different with other codes, and the details of reading the binary dumps would be different as well. In the listing, common block *rlc1* holds the opacity and planck functions while block *state* holds the equation of state (EOS) data.

The details of the EOS and opacity data can be found in Symbalisty et al. (1995) and Horak and Kodis (1983); however, we will provide a brief summary here. Let T be temperature in ev, P be pressure, E be specific internal energy and ρ mass density. Then gt holds T/E as a function of E for 100 temperatures and seven densities. The array fp holds $P/(\rho E)$ as a function of E for 100 temperatures and 7 densities. The array uk holds the opacity data as Rosseland means, cm^2/gm, for 51 frequency bands, for 100 temperatures and seven densities.

6.2 Isophote Application

An isophote post processor was written that processes a SnCaveat binary dump file. This post processor follows the algorithm documented in Sections 1—5. With the post

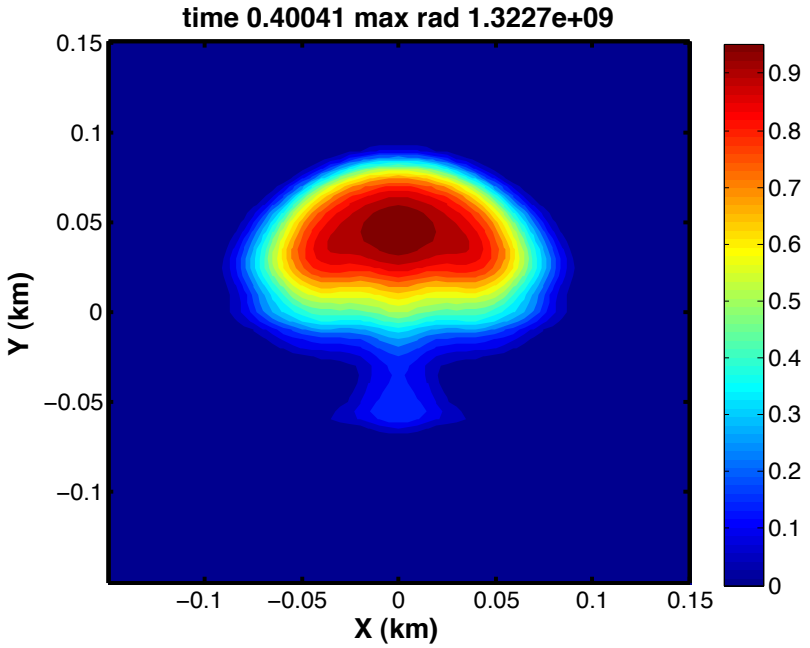

Figure 6: Isophotes for a 1-kt burst at 50-m HOB and a time of 0.4 s, horizontal path

processor code, one can easily calculate contours for a series of SnCAVEAT dump files and change the observer look angle. The first example of the current version is given in Fig. 6 for a - kt burst at 50-m HOB with a horizontal look angle, at a nominal distance of 20000 km, while Fig. 7 is the same except for a 45^0 look angle. In these plots, the time and maximum radiance are given at the top of the plots. The contours are for radiance values normalized by the maximum radiance. The calculation was initialized at 1.5 ms from a 1-D HYCHEM run. Fig. 8 then gives the Si irradiance (Eq. 43) as a function of time for the 45^0 look angle.

For a given observer position, and a given set of isophote contours, Eq. 38, for D_ν gives the energy per unit area, per unit time (a flux), at the observer position. One could then, for some detector with some area A, determine the total energy per second in the detector by AD_ν.

The irradiance values in the different bands can be summed with appropriate weights to calculate specific responses. For a silicon detector, we determine the irradiance with the following summation:

$$\begin{aligned} siirad &= 0.17 * hkirad(7) + 0.95 * hkirad(8) + 0.89 * hkirad(9) \\ &+ 0.62 * hkirad(10) + 0.37 * hkirad(11) + 0.12 * hkirad(12), \end{aligned} \tag{43}$$

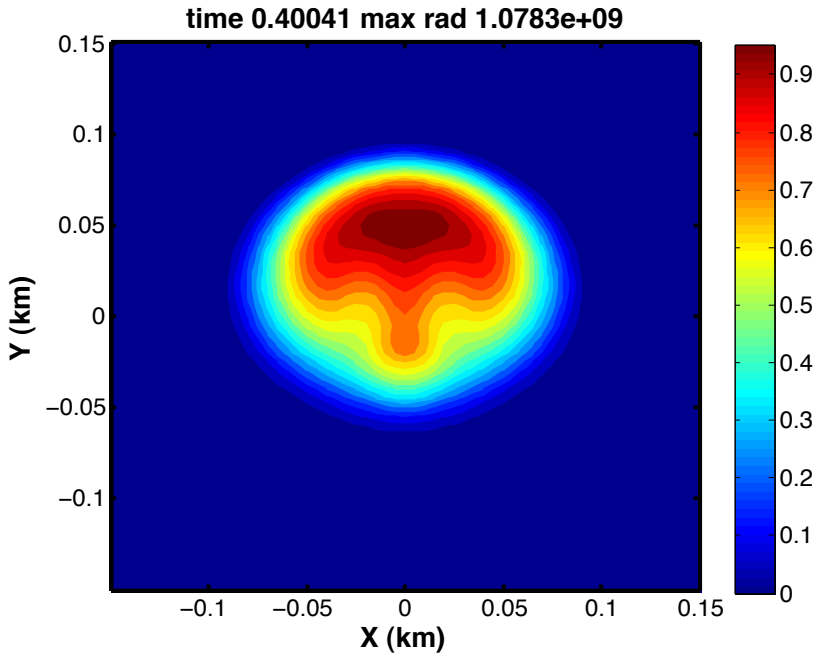

Figure 7: Isophotes for a 1-kt burst at 50-m HOB and a time of 0.4 s, and 45^0 path.

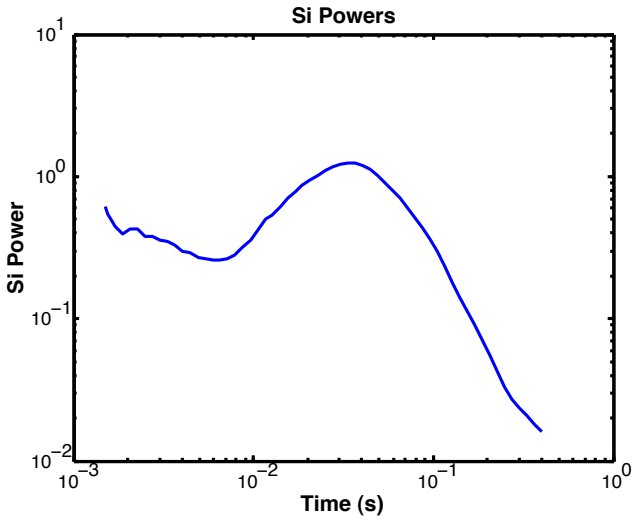

Figure 8: The Si band power vs time curve for 45^0 look angle.

where the hkirad values are the fluxes in the bands 7 to 12, and the coefficients give the relative weight for calculating the bhangmeter-weighted Si band. This is what is plotted in Fig 8. Although *siirad* has the units of a flux, it could also be viewed as the power in a unit area.

6.3 Particular Cases

One reason for using a post-processing code, such as isophote, is to calculate fluxes from different look angles (expressed as either elevation angle or zenith angle). For lower heights of burst (HOB), where there is ground interaction, 2-D effects become important, and the fireball evolution departs from that for a 1-D free air burst. For a burst on an ideal perfectly reflecting surface, HOB = 0, where downward-directed energy is reflected back into the fireball, there is a well known factor of two effect on the observed yield, for blast as well as optical. On the other hand, for a higher HOB, one would expect a 1-D free-air behavior with a 2-D code.

We have performed a series of calculations of a 10-kt burst at HOBs of (meters): 0, 30, 60, 120, and 240. These were chosen in part to test for the surface burst factor of two in yield as measured by minimum time in the silicon power time curve. The fireball evolution was performed with our 2-D code SNCAVEAT. Binary mesh dumps are made periodically over the course of the calculation. The isophote post-processor then reads the dumps and performs the radiative transport through the fireball structure. These calculations were performed with Sn order 4. A new initialization scheme was used in SnCAVEAT that follows the philosophy that is used in our 1-D code, HYCHEM, Zinn

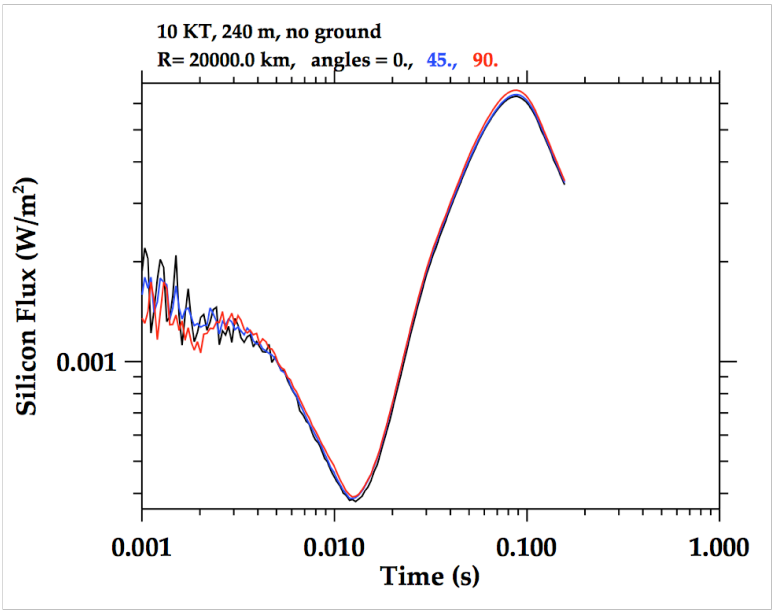

Figure 9: Si flux versus time for the 10 kt, 240-m HOB at three elevation angles.

(1973). In the following figures we show data from the isophote post processor for an observer at 20000 km and at various elevation angles.

In Fig. 9, for the 240-m HOB, we show Si-weighted fluxes for the elevation angles of 0^0, 45^0 and 90^0. In this calculation, there was no ground in the calculation. As expected the curves overlay nicely, with a slight difference at second max, showing no look angle dependence, the expected result for no ground interaction. Next, in Fig. 10 for the 0-m HOB we show Si fluxes at three elevation angles. Here, the ground interaction is immediate, the fireball structure is not the same as seen from different angles, and the large differences in the Si flux curves can be understood by reference to the isophote contours. In Fig. 11, for the 0-m HOB, we first show the contours for the horizontal look angle, followed by Fig. 12 for the vertical look angle. Both are plotted on the same scale, and the integration over the contours of Fig. 12 will clearly give a larger value than that for Fig. 11.

The HOB effect on minimum time is illustrated in Fig. 13 where we show Si flux versus time for the 240-m HOB and the 0-m HOB, both with a horizontal look angle. In this figure, we also show one curve from the Los Alamos version of Hychem. The surface burst has a later minimum time than the 240-m HOB, 0.0149 s as opposed to 0.0120 s. The value for the 0m HOB is exactly that for a free air 20-kt burst as given by the scaling law in Symbalisty et al. (1995). This demonstrates the HOB effect rather nicely for optical parameters. The 240-m values are larger consistent with the contours shown in Figs. 11 and 12. The early time, through first maximum, radiative output is difficult to calculate because of the large optical depths in the very hot shock that would require

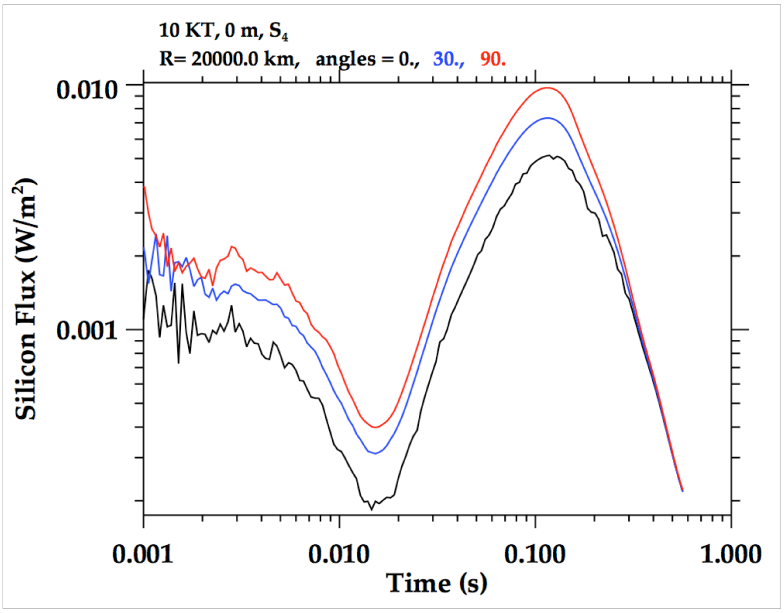

Figure 10: Si flux versus time for the 10 kt, 0-m HOB at three elevation angles.

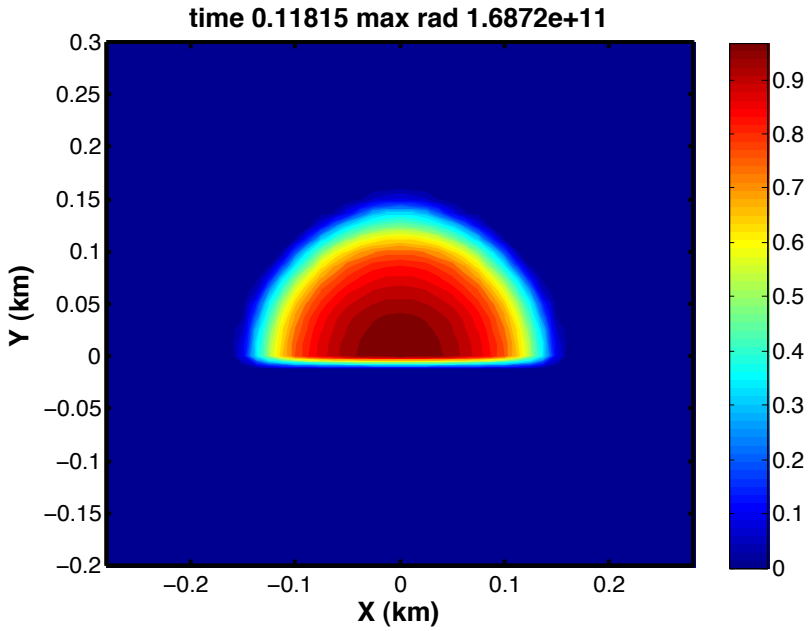

Figure 11: Isophotes for a 10 kt burst at 0-m HOB and 0^0 elevation angle.

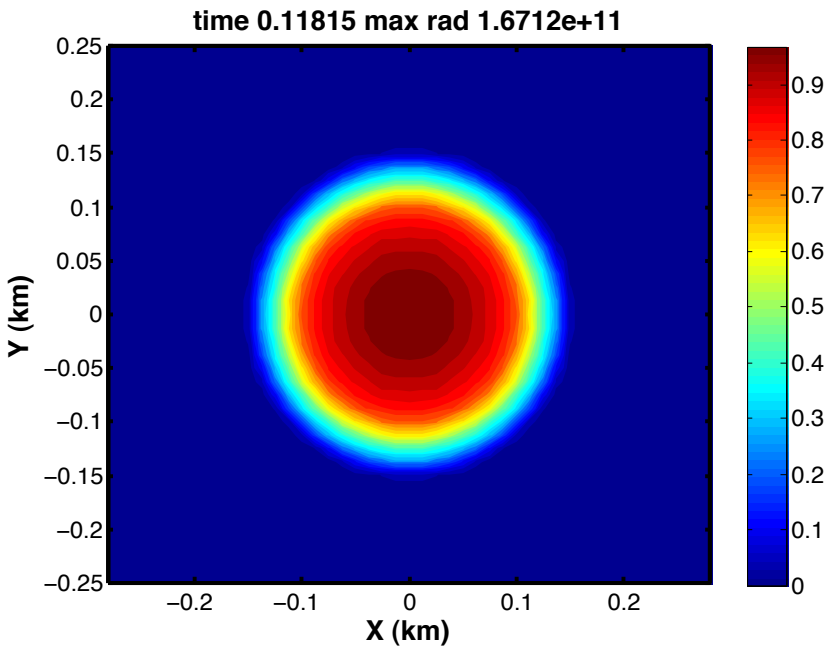

Figure 12: Isophotes for a 10-kt burst at 0-m HOB and 90^0 elevation angle.

prohibitively small cells. In this regime, HYCHEM uses tables of shock brightness versus shock speed obtained from steady state shock theory, Symbalisty et al. (1995) and Zinn and Sutherland (1981). This avoids the oscillations in brightness such as are shown in the early part of the SnCAVEAT curves until the radiative scales are compatible with the cell sizes.

Table 1 gives the full set of minimum times for the different HOB calculations.

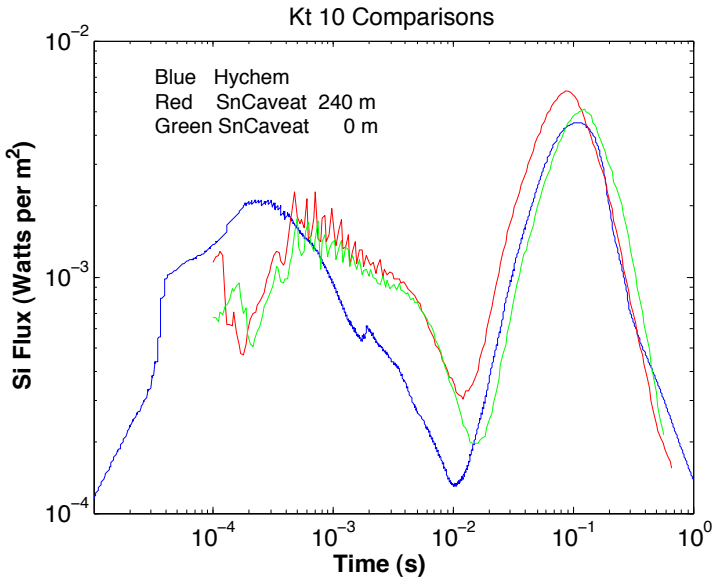

Figure 13: Si flux versus time for the 10 kt, 0-m and 240-m HOB showing the delayed minimum time for the 0-m HOB case.

Table 1: Minimum times for the 10-kt HOB series.

HOB (m)	T min (s)
0	0.0149
30	0.0133
60	0.0121
120	0.0120
240	0.0120

7 Summary

In this report, we have illustrated the application of our isophote post-processor program, isopost.f, on some SnCaveat 2D fireball simulations. In addition, we have shown the delay in minimum time, compared to a free air burst, for lower heights of burst, including a surface burst. We recover the well known factor of two in apparent yield for the surface burst. The current version does require a cylindrical (r,z) mesh. With this constraint, the code could be applied to other CFD code outputs with some changes in reading in the CFD output and having an appropriate set of opacity and equation-of-state data.

A similar and independent capability was developed at Sandia National Laboratory and is discussed in Dreike et al. (2006).

8 Acknowledgements

The original draft report was prepared by Henry Horak and John W. Kodis (deceased). Much of the supporting work was done by several individuals working on atmospheric nuclear effects in various groups within LANL. In addition to Horak and Kodis, important contributions to the overall effort in effects work were made by Brook Sanford, Eric Jones, Richard Anderson, and John Zinn. We have benefited from productive collaboration with the LANL numerical hydrodynamics group, T-3. Tom Hill made significant contributions in ensuring that our Sn radiative transfer algorithms were efficiently incorporated and in assisting with new applications. Christy Flaming prepared the figures in Sections 1-5. We thank Laurie Triplett for reviewing the manuscript. Support, over the years, has come from the Defense Nuclear Agency (now the Defense Threat Reduction Agency), the Department of Energy, the Air Force and LANL.

9 Bibliography

1. Addessio, Frank L., John R. Baumgardner, John K. Dukowicz, Norman L. Johnson, Bryan A. Kashiwa, Rick M. Rauenzahn and Charles Zemach, *CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip*, Los Alamos National Laboratory technical report LA-10613-MS, Rev. 1 (1992).

2. Amsden, A. and C. Hirt, *YAQUI: An Arbitrary Lagrangian-Eulerian Computer Program for Fluid Flow at All Speeds*, Los Alamos Scientific Laboratory report LA-5100 (1973).

3. Anderson, R. and M. Sandford II, *YOKIFER: A Two-Dimensional Hydrodynamics and Radiation Transport Program*, Los Alamos Scientific Laboratory report LA-5704-MS (1974).

4. Dreike, Philip L., Raymond L. Bell, Gary D. Cable, William Hilbun, Michael L. Hobbs, Anne R. Moats, J. Randy Weatherby and Robert J. Weir, *Progress on Modeling Optical Signals from Nuclear Detonations in Multi-Dimensional Emplacements*, Sandia National Laboratory report SAND2006-7510 (2006). (Report is OUO.)

5. Hill, Thomas R. and Richard R. Paternoster, *Two-Dimensional Spatial-Discretization Methods on a Lagrangian Mesh*, Los Alamos National Laboratory technical report, LA-UR-82-1055 (1982).

6. Horak, H., E. Jones, J. Sandford II, R. Whitaker, R. Anderson, and J. Kodis, *Two-Dimensional Radiation Hydrodynamic Calculations for a Nominal 1-Mt Nuclear Explosion Near the Ground*, Los Alamos National Laboratory report, LA-9137 (1982).

7. Horak, H.G. and J. Kodis, *RADFLO - A User's Manual*, Los Alamos National Laboratory report LA-9245-M May (1983).

8. Lathrop, K.D., *Discrete-Ordinates Methods for the Numerical Solution of the Transport Equation*, **Reactor Technology**, 15, 107-133 (1972).

9. Lathrop, K. and F. Brinkley, *TWOTRAN II: An Interfaced, Exportable Version of the TWOTRAN Code for Two-Dimensional Transport*, Los Alamos Scientific Laboratory report LA-4848-MS (1973).

10. Symbalisty, E.M.D., J. Zinn, and R.W. Whitaker, *RADFLO Physics and Algorithms,* Los Alamos National Laboratory report LA-12988-MS (1995).

11. Zinn, J., *A Finite Difference Scheme for Time-Dependent Spherical Radiation-Hydrodynamics Problems,* **J. Comput. Phys.**, **13**, 569 (1973).

10 Appendix

10.1 Sample input file

Below we show the input file for running the isophote code that is namelist based.

```
$input
 nangl    = 2,
 angl(1:2)= 0.00, 45.00,
 zevent   = 5.0e+01,
 robs     = 20000.0,
 ndumps   = 2,
 binpath ='/scratch/wphenom/ISO_MP/',
 tstrt    = 0.0000,
 tend     = 3.00e-05,
 istellar = 0,
 irect    = 0,
$

Number of grid points will be (2*nstk-1)(2*nstk-1)
Maximum number of rays is     (2*nstk-1)*nstk
zevent   = height of burst in km
robs     = distance to observer in km
nangl    = number of different angles
angl     = array of angles in degrees
istellar = 1  implies stellar magnitudes
```

10.2 Code listing

Now we provide a listing of the isophote processor. This version does incorporate OpenMP for shared memory, multi-processor machines. The OpenMP instructions appear as compiler directives that are just seen as comments for a serial processor computer.

```
cemds  1  Aug 2007   Converted to run parallel using OpenMP
cemds                setenv OMP_NUM_THREADS #

cemds  04 Apr 2007   Added loop over observer locations
cemds                nangl   =  number of different observer angles
cemds                angl    =  array of observer angles (measured from ground up), degrees
cemds                robs    =  distance to observer in km
cemds                zevent  =  altitude of event in km

cemds  13 Mar 2007   Added istellar switch
cemds                Added irect switch
cemds                Added ptiso.txt output file
cemds                Added siechk.txt output file
cemds                The number of possible rays is (nstk * (2*nstk-1)),
cemds                   but only the rays with nonzero intensity are
cemds                   included in the count.

!  isopost.f  program to read sncaveat binary dump files
!  and generate contours of brightest as seen by an
!  observer at x = a01 (cm) and z = a03 (cm)
!  the size of the plane of isophotes is a grid of
!  x and z points of size (2*nstk-1, 2*nstk-1)
!  based on Horak and Kodis, 1983.

!  generates isodat.txt file
!  the input file  iniso

      program isopost
      parameter (nangmax=4, nbands=6, nstk=51, nlstk=2*nstk-1)
      parameter (nlstsq=nlstk*nlstk, nsxnl=nstk*nlstk)

      include 'comdeck'
      include 'comdeck1'

      common/rlc1/uk(100,7,51), plb (100,51), freq(52), freqd(51)
      common /state/ gt(100,7), fp(100,7)
      common /eqstk/ dk1,d2,d3,d4

      common /mesh/ nxp,nyp,nx,ny
      common /iando/ binfile,binpath
      common/plotk/ikpt,jkpt
```

```
      common/esstuff/ irect, istellar
      character dstart*9, dfin*9, tstart*10, tfin*10

      dimension angl(nangmax), siband(nangmax), siirad(nangmax),
     &          hkirad(51,nangmax), xk(nlstk,nangmax),
     &          yk(nlstk,nangmax), hkint(nlstsq,nbands,nangmax),
     &          hwk(nlstk)

      character*10 binfile
      character*70 binpath

      namelist /input/ zevent,nangl,robs,ndumps,binpath,
     &                 angl,irect,istellar, tstrt,tend

      call date_and_time(dstart, tstart)

      open (7,file='iniso',form='formatted')
      open (9,file='isopht.owt',form='formatted')
      open(10, file='isodat.txt',form='formatted')
      open(12, file='siechk.txt',form='formatted')
      open(14, file='ptiso.txt',form='formatted')

      do i=1,4
        angl(i) = float(i-1) * 30.
      enddo
      irect    = 0
      istellar = 0
      nangl    = 4
      ndumps   = 3
      robs     = 20000.
      tend     = 100.
      tstrt    = 0.
      zevent   = 0.060
      degtorad = acos(-1.0)/180.

      read (7,input)

c     convert from km to cm

      robs     = robs * 1.e5
      zevent   = zevent * 1.e5

      write(14,*) binpath
      write(14,'(i4,"  =  nstk")') nstk
      write(14,'(i4,"  =  nangl")') nangl
      write(14,'(10(1x,f5.1))') (angl(i),i=1,nangl)
```

```
      write(14,'(i5,"  = number of time points")') ndumps

      write(10,'(i4,"  = ndumps")') ndumps
      write(10,'(2(i4,1x)," nstk, nlstk")') nstk, nlstk
      write(10,'(i4,"  = # of angles")') nangl
      write(10,'(10(1x,f5.1))') (angl(i),i=1,nangl)

      do i=1,4
        angl(i) = degtorad * angl(i)
      enddo

      call rdaesop

      open(11, file=trim(binpath)//'binlist',form='formatted',
     1  status='old')

cemds  gu(,1)    = jt
cemds  gu(,2)    = tfrc
cemds  gu(,3)    = jr
cemds  gu(,4)    = rfrc

      call hwkset(hwk,nlstk,nstk)

      do 550 ii=1,ndumps

       read (11,'(a10)')binfile
       call idlrd

       if (t.ge.tstrt.and.t.le.tend) then

        nx           = n1(1) + 1
        ny           = n2(1) + 1
        nxp          = nx + 2
        nyp          = ny + 2
        iblk         = 1
        n1p3         = n1(iblk) + 3
        n2p2         = n2(iblk) + 2
        n2p1         = n2(iblk) + 1
        ifrst(iblk)  = n1p3 + 2
        lastv(iblk)  = n1p3*n2p2 - 1
        lastc(iblk)  = n1p3*n2p1 - 2
        msizv(iblk)  = n1p3*n2p1 - 2
        tsn          = t
        ms           = msz(iblk)
        m1(iblk)     = 1
```

```
        do i=1,lastv(iblk)
          sie(i) = te(i) - 0.5*(uc(i)**2 + uc(msz(iblk)+i)**2)
        enddo
        write(12,'(/,"t =",1pe11.4)') t

      call siechk(sie,n1(1),n2(1))

      call dblknt(rho(1), sie(1), gu(1), gu(1+ms), gu(1+2*ms),
     &            gu(1+3*ms), n1(1), n2(1), ms)

      mkfst = 7
      mklst = 12

!$omp parallel do
      do iang =1,nangl
        call setxkyk(sie,xv(1),xv(1+ms),xk(1,iang),yk(1,iang),
     &               ikpt,jkpt,ms,nlstk,nstk,nx,nxp)
      enddo

!$omp parallel do private(iang, a01, a03)
!$omp+ shared(n1,n2,xv,rho,sie,gu,hkint,siband,siirad,hkirad,xk,yk)
      do 500 iang = 1, nangl

        a01    =          robs * cos(angl(iang))
        a03    = zevent + robs * sin(angl(iang))

          call isopht(a01,a03,n1(1),n2(1),xv(1),xv(ms+1),rho(1),
     &             sie(1),gu(1),gu(1+ms),gu(1+2*ms),gu(1+3*ms),
     &             ms,ncyc,nstk,nlstk,nlstsq,nsxnl, hkint(1,1,iang),
     &             siband(iang), siirad(iang), hkirad(1,iang), mkfst,
     &             mklst, xk(1,iang), yk(1,iang), nbands, hwk,
     &             uk, plb, nxp, nyp, nx, ny, ikpt, jkpt)

 500  continue

      do iang=1,nangl

        a01    =          robs * cos(angl(iang))
        a03    = zevent + robs * sin(angl(iang))

        call owtvar1(xk(1,iang), yk(1,iang), hkint(1,1,iang),
     &              siirad(iang), hkirad(1,iang), a01, a03,
     &              tsn, ncyc, nlstk, mkfst, mklst, nbands)

      enddo
```

```
      write (6,'(" cycle = ",i6,"   time =",e13.5)') ncyc, t

      elseif(t.gt.tend) then
        go to 600
      endif

 550  continue
 600  continue

      call date_and_time(dfin, tfin)

      write(*,'(/,"DATE AND TIME END   = ",a8,2x,a10,
     &          /,"DATE AND TIME START = ",a8,2x,a10,/)')
     &   dfin, tfin, dstart, tstart
      write(9,'(/,"DATE AND TIME END   = ",a8,2x,a10,
     &          /,"DATE AND TIME START = ",a8,2x,a10,/)')
     &   dfin, tfin, dstart, tstart

      close(3)
      close(7)
      close(10)
      close(11)
      close(14)
      close(31)
      end
c =====================================================================

      subroutine siechk(sie,n1,n2)
      dimension sie(0:n1+2,0:n2+2)

      do j=1,n2
      do i=1,n1
        if(sie(i,j) .lt. 0.0) then
          write(12,'("sie(",i3,",",i3,")=",1pe11.4)') i,j,sie(i,j)
        endif
      enddo
      enddo

      return
      end

c =====================================================================
      subroutine idlrd

c     This routine writes a dump containing the contents of most
c     of the labeled common blocks into file DP2D.
```

```
c     Called by:  HYDROOUT
c     Calls    :  none

c ======================================================================
      include 'comdeck'
      include 'comdeck1'
      common /iando/ binfile,binpath
c ======================================================================
      character*10 binfile
      character*70 binpath

      open(31,file=trim(binpath)//trim(binfile),status='old',
     1     form='unformatted')

      read(31)  t,ncyc,nblks,alecoef,dthydro,handed,rpl

      do iblk=1,nblks
        read(31) n1(iblk)
        read(31) n2(iblk)
        read(31) msz(iblk)
        m1(iblk) = 1
        m2(iblk) = 1
        call idlbin3(xv(m2(iblk)),   uc(m2(iblk)),
     &               pr(m1(iblk)),   te(m1(iblk)),
     &              rho(m1(iblk)), temp(m1(iblk)),
     &                  n1(iblk),       n2(iblk))
      enddo

      read(31) grav(1)

      if(grav(1) .gt. 0) then
        read(31) nzeq
        read(31) (prequ(i),i=1,nzeq)
        read(31) (roequ(i),i=1,nzeq)
        read(31) (tequ(i), i=1,nzeq)
        read(31) (zeq(i),  i=1,nzeq)
      endif

   10 format(1x," idl dump read ",i3," at t=",1pe12.5," cycle=",i5)
  130 format(a10)
      end

c ======================================================================
      subroutine idlbin3(xv,uc,pr,te,rho,temp,n1,n2)
```

```
c ======================================================================

c     This routine writes arrays to binary dump

c     Called by:  idlbin
c     Calls    :  none

c ======================================================================
      dimension xv(n1+3,n2+3,2), uc(n1+3,n2+3,2),  pr(n1+3,n2+3),
     &          te(n1+3,n2+3),  rho(n1+3,n2+3),  temp(n1+3,n2+3)

      read (31) ((xv(i,j,1),i=1,n1+3),j=1,n2+3)
      read (31) ((xv(i,j,2),i=1,n1+3),j=1,n2+3)
      read (31) ((uc(i,j,1),i=1,n1+3),j=1,n2+3)
      read (31) ((uc(i,j,2),i=1,n1+3),j=1,n2+3)
      read (31) ((te(i,j)  ,i=1,n1+3),j=1,n2+3)
      read (31) ((pr(i,j)  ,i=1,n1+3),j=1,n2+3)
      read (31) ((rho(i,j) ,i=1,n1+3),j=1,n2+3)
      read (31) ((temp(i,j),i=1,n1+3),j=1,n2+3)
      end

c ======================================================================
      subroutine contrj(z,nzx,nzy)
      dimension z(nzx,nzy)

!crww  write out array z

      do jy=1,nzy
       write(10,'(5e14.6)') (z(i,jy),i=1,nzx)
      enddo

      end
c     ======================================================================
      subroutine dblknt(ro, sie, jt, tfrc, jr, rfrc, n1,n2,ms)

      dimension  ro(ms), sie(ms)
      dimension  jt(1:n1+3,1:n2+3), tfrc(1:n1+3,1:n2+3)
      dimension  jr(1:n1+3,1:n2+3), rfrc(1:n1+3,1:n2+3)

      common /state/ gt(100,7), fp(100,7)
      common /eqstk/ dk1,d2,d3,d4


      common /mesh/ nxp,nyp,nx,ny
      common/plotk/ikpt,jkpt
```

```
      tmax = 0.0

      do 30 jj=2,ny
      do 20 ii=2,nx

      ij          = (jj-1)*nxp+ii
      fj          = d4*log(ro(ij))+dk1
      j           = max(1,min(int(fj),6))
      jr(ii,jj)   = j
      rfr         = fj - float(j)
      rfrc(ii,jj) = rfr
      rfr1        = 1.-rfr
      fi          = (log(sie(ij))-d2)*d3
      i           = max(1,min(int(fi),99))
      efr         = fi-float(i)
      efr1        = 1.-efr
      tmp  = ((gt(i,j  )*efr1 + gt(i+1,j  )*efr)*rfr1 +
     1        (gt(i,j+1)*efr1 + gt(i+1,j+1)*efr)*rfr)*sie(ij)
      if (tmp.lt.tmax) go to 10
        tmax = tmp
        ikpt = ii
        jkpt = jj

   10 tk        = 6.7808525*log(tmp) + 26.290555
      jt(ii,jj) = max(1,min(int(tk),99))
      tfr       = tk - float(jt(ii,jj))

c      below prevents negative opacities or plankb,s

      if (tfr.lt.0.0) tfr = 0.0
      tfrc(ii,jj) = tfr

   20 continue
   30 continue

      return
      end
c     ================================================================
      subroutine howfar(x,y,ms,iray,sintsq,a,omega,dcell,inew,jnew,
     &      iold,jold,xd,yd,iko,a1a2sq,rhop,nko)

      dimension x(ms),y(ms), a(3), omega(3), xd(5), yd(5)

c      form geometry for tests of all 4 sides

      nhen=0
```

```
      if (rhop.eq.0.0) go to 99
    1 a12om=a(1)*omega(1)+a(2)*omega(2)
      a21om=a(1)*omega(2)-a(2)*omega(1)
   20 dcell=1.e+38
      i=1
      iko=0
      inew=iold
      jnew=jold
  2   n2=i+1
      y1=yd(i)
      y2 =yd(n2)
      dy=y2-y1
      x1=xd(i)
      x2=xd(n2)
      dx=x2-x1
      d2=1.e+38
      if (dx.eq.0.0) go to 11
      yma3=y1-a(3)
      ad3=x1*dy-yma3*dx
      celtst=rhop*dy-ad3
      if(celtst.ge.0.0)go to 100
      if((dy*omega(3)).lt.0.0) go to 7
      yma3=yma3+dy

c      test below eliminates abt 25 percent of cases

  7   if((yma3*omega(3)).lt.0.0) go to 19
      dysq=dy*dy
      if(dysq.lt.1.e-16) go to 60
      if(a12om.lt.0.0) go to 30
      if(x2.ge.x1) go to 29
      x2=x1

c      x2=xmax     nxt statement diverts about 19 percent

 29   if( rhop.ge.x2) go to 19
 30   omg3dx=omega(3)*dx
      if(a21om.eq.0.0) go to 80
      p=sintsq*dysq-omg3dx**2
      r=a1a2sq*dysq-ad3**2
      q=a12om*dysq-ad3*omg3dx
      pxr =p*r
      if(pxr.lt.0.0)go to 43

c      if above true, only 1 pos. root exists as disc**0.5.gt.qpinv
```

```
      if((q*p).ge.0.0) go to 19
      pinv=1./p
      qsq=q*q
      if(qsq. ge.(1.e3*pxr))go to 35

c       abt 6 percent come thru here

      disc=qsq-pxr
      if(disc.lt.0.0) go to 19
      qpinv=-q*pinv
      disc=sqrt(disc*pinv**2)

c       always 2 positive roots d1 is the smallest

      d1=qpinv-disc
      d2= qpinv+disc
      go to 46
   11 r1=rhop-x1

c        dx=0.0 routes

      if ((r1*dy).ge.0.0) go to 100
      r2=rhop+x1
      if (sintsq.eq.0.0) go to 19
      r=r1*r2
      qsq=a12om**2
      if (a21om.eq.0.0) go to 82
      pxr=sintsq*r
      qsqfr=1.e-03*qsq
      if (r.lt.0.0) go to 12
      if (a12om.ge.0.0) go to 19
      if((qsqfr-pxr).ge.0.0) go to 21
      disc=qsq-pxr
      if (disc.lt.0.0) go to 19
      d1= -(a12om+sqrt(qsq-pxr))/sintsq
      go to 5
  21  qinv=1./a12om
      d1= -r*qinv*(0.5+0.125*pxr*qinv**2)
      go to 5
   12 if ((qsqfr+pxr).ge.0.0) go to 13

c      r=neg routes  fall thru here means q=0.0

      d1=(-a12om+sqrt(qsq-pxr))/sintsq
      go to 5
   13 if (a12om.lt.0.0) go to 15
```

```
      d1= (-r/a12om)*(0.5+0.125*pxr/qsq)
      go to 5
   15 prdq2=pxr/qsq
      d1=(a12om/sintsq)*(-2.+prdq2*(0.5+0.125*prdq2))
      go to 5
 35   prdqsq=pxr/qsq

c        less than .04 percent come here
c       use taylor series expansion to prevent underflow.

      qpinv=-q*pinv
      dadd=2.*qpinv
      d1=prdqsq*qpinv*(0.5+0.125*prdqsq)
      d2=dadd-d1
      go to 46

c       use taylor series expansion to prevent underflow.

  43  pinv=1./p

c       p*r negative. sqrt(disc). gt. -q/p

      qsq=q*q
      if(qsq.ge.1.e+3*(-pxr)) go to 44
      disc=qsq-pxr
      d1=sqrt(disc*pinv**2)-q*pinv
      go to 46
 44   prdqsq=pxr/qsq
      qpinv=-q*pinv
      dadd=2.*qpinv
      d1=prdqsq*qpinv*(0.5+0.125*prdqsq)
      if(dadd.lt.0.0)go to 46
      d1=dadd-d1
   46 ztest=ad3+d1*omg3dx
      if ((ztest*dy).ge.0.0) go to 5
      if ((ad3+d2*omg3dx)*dy.lt.0.0)d2=1.e38
      d1=d2
 5    if(d1.ge.dcell) go to 19
      dcell=d1
      iko=i
   19 i=i+1
      if (i.le.4) go to 2
      if(iko.eq.0) go to 100
      inew=iold+3-iko
      if(iko.eq.1)inew=iold
      if(inew.lt.1) inew=1
```

```
      jnew=jold+iko-2
      if(iko.eq.4)jnew=jold
      return
  60  if(omega(3).eq.0.0) go to 19

c        dy=small route

      d1=yma3/omega(3)
      go to 5
  80  if(dx.eq.0.0)go to 82

c        rte for motion in plane of axis of symmetry

      denom=rhop*omg3dx-a12om*dy
      if(denom.ge.0.0) go to 19
      d1=celtst*rhop/denom
      go to 5
 82   if(a12om*dy.le.0.0)go to 19
      d1=rhop*(x1-rhop)/a12om
      go to 5
   99 a(1)=a(1)+1.e-06
      a1a2sq=a(1)**2+a(2)**2
      rhop=sqrt(a1a2sq)
      go to 1

  100 call whrko(kko,x,y,ms,rhop,a(3),inew,jnew,iold,jold,
     &   xd(1),xd(2),xd(3),xd(4),xd(5), yd(1),yd(2),yd(3),yd(4),yd(5))

c       kko=1 means no chng in cell.   kko=0  pt out of mesh
c        kko=-1=  pt not found

      nko=nko+1
      if (kko.gt.1) go to 20
      if (kko.lt.0) go to 140
      if (kko.eq.0) go to 138
      if (nhen.eq.1) go to 140

c     below corrects points lying on cylinder

cemds      write (6,160) iold,jold,rhop,a(3)
      nhen=1
      dx=xd(n2)-xd(i)
      dy=yd(n2)-yd(i)
      if (a(1).eq.0.0 .and. a(2).eq.0.0) a(1)=1.e-06

c      we increment 1.e-06*rhop perp to sfc on whc pt. lies
```

```
      csddr=1.e-06/(sqrt(dx**2+dy**2))
      if ((dy*a12om-rhop*omega(3)*dx).lt.0.0) go to 134
      a(1)=a(1)*(1.+dy*csddr)
      a(2)=a(2)*(1.+dy*csddr)
      a(3)=a(3)-dx*csddr*rhop
      go to 136
  134 a(1)=a(1)*(1.-dy*csddr)
      a(2)=a(2)*(1.-dy*csddr)
      a(3)=a(3)+dx*csddr*rhop
  136 a1a2sq=a(1)**2+a(2)**2
      rhop=sqrt(a1a2sq)

      call whrko(kko,x,y,ms,rhop,a(3),inew,jnew,iold,jold,
     &   xd(1),xd(2),xd(3),xd(4),xd(5), yd(1),yd(2),yd(3),yd(4),yd(5))

      go to 1
  138 iko=-1
      return
  140 write (6,170) iray,iold,jold,rhop,a(3)
      iko=0
      return

  160 format (14h pt on cyl  i=,i3,3h j=,i3,6h rhop=,e10.4,4h a3=,e10.4)
  170 format (1x,7hpt unkn,3h l=,i4,3h i=,i3,3h j=,i3,4h xp=,e10.4,4h yp
     1=,e10.4)
      end
c =====================================================================
      subroutine isopht(a01,a03,n11,n21,x,y,ro,sie,jt,tfrc,
     &                  jr,rfrc,ms,ncyc,nstk,nlstk,nlstsq,nsxnl, hkint,
     &                  siband, siirad, hkirad,mkfst, mklst, xk, yk,
     &                  nbands,hwk,uk,plb,nxp,nyp,nx,ny, ikpt, jkpt)

      dimension a(3), omega(3), xd(5), yd(5)

      dimension  ro(ms),     sie(ms), x(ms), y(ms),
     &        jt(1:n11+3,1:n21+3),  tfrc(1:n11+3,1:n21+3),
     &        jr(1:n11+3,1:n21+3),  rfrc(1:n11+3,1:n21+3),
     &        hkint(nlstsq,nbands), dkinv(nsxnl), xk(nlstk), yk(nlstk),
     &        xkpd2(nlstk),   hkmag(nlstsq),  hwk(nlstk)
      dimension hkirad(51)
      dimension uk(100,7,51), plb(100,51)

      nst    = nstk
      nlst   = nlstk
      dist1  = 1.0000002
```

```
      distin = 2.e-06
      zbot   = y(nxp+2)
      xmax   = x(nxp+nx+1)
      ztop   = y(ny*nxp+nx+1)

      xmaxsq = xmax*xmax
      ijkp   = (jkpt-1)*nxp+ikpt
      z0     = y(ijkp)
      sietst = min(1.0e+09,0.5*sie(ijkp))
      do 30 i=ikpt,nx
        ijkpt = (jkpt-1)*nxp+i
        pimax = x(ijkpt)
        if(sie(ijkpt) .lt. sietst) go to 40
   30 continue
   40 continue

      jkp1=jkpt+1
      do 42 jj=1,jkp1
        ij = (jj-1)*nxp+ikpt
        if(sie(ij) .ge. sietst) go to 43
   42 continue
   43 zb = y(ij)

      do 44 jj=jkp1,ny
        ij=(jj-1)*nxp+ikpt
        if(sie(ij) .ge. sietst) go to 45
   44 continue
   45 zt = y(ij)

      zt    = max(zt,1.05*zb)
      plmax = pimax
      if (a01 .le. pimax) go to 8
      t3a03 = zt - a03
      b3a03 = zb - a03
      if (a03.le.zb) go to 7
      if (a03.ge.zt) go to 6

c     observer btwn b,c

      t1a01= pimax-a01
      b1a01= t1a01
      go to 9

    6 t1a01=-(pimax+a01)
      b1a01= pimax-a01
      go to 9
```

```
    8 z0a03=z0-a03
      dobs=sqrt(a01**2+z0a03**2)
      go to 22

    7 t1a01= pimax-a01
      b1a01=-(pimax+a01)

    9 sqb   = sqrt(b1a01**2+b3a03**2)
      sqt   = sqrt(t1a01**2+t3a03**2)
      z0    = a03-a01* (sqb*t3a03+sqt*b3a03)/(sqb*t1a01+sqt*b1a01)
      z0a03 = z0 - a03
      dobs  = sqrt(a01**2+z0a03**2)

   22 dbot1  = zbot - a03
      dobsnv = 1./dobs
      dobssq = dobs*dobs
      omg01  =-a01*dobsnv
      omg03  = z0a03*dobsnv
      radsp  = pimax/float(nst-1)
      radspl = plmax/float(nst-1)
      dtop1  = ztop - a03

c        find freq independent geometry

      omgdk = z0a03*omg03 - a01*omg01
      nstp  = (nst-1)*nst

      do 60 i1=1,nlst
   60 xkpd2(i1) = xk(i1)**2 + dobssq

      kk=0
      do 64 j1=1,nlst
        yksq   = yk(j1)**2
        do 62 i1=nst,nlst
        kk = kk+1
 62     dkinv(kk) = 1.0/sqrt(xkpd2(i1) + yksq)
 64   continue

cemds  loop over the frequency bands of interest

      ib = 0
      do 350 mk=mkfst,mklst

        ib     = ib + 1
        hkomax = 0.0
```

```
        nko    = 0
        mk1    = mk
        kv     = 0
        iray   = 0

c       observer must be outside yaqui mesh.  omg is a unit vector
c       along ray.  omg1=omg*i, omg2=omg*j, omg3=omg*k
c       observer is in xz-plane such that his x-coord (a01) is ge 0.
c       right hand side of mesh is scanned so that omg2 ge 0.

      j1n=-nlst

      do 310 j1=1,nlst

        j1n    = j1n + nlst
        omg1kp = yk(j1)*omg03 - a01
        omg3kp = z0a03 - yk(j1)*omg01

      do 300 i1=nst,nlst

      j1ni1        = j1n+i1
      hkint(j1ni1,ib) = 2.e-30
      kv           = kv+1
      diinv        = dkinv(kv)
      omega(1)     = omg1kp*diinv
      omega(2)     = xk(i1)*diinv
      omg3         = omg3kp*diinv
      omega(3)     = omg3
      sintsq       = omega(1)**2 + omega(2)**2
      if (dtop1 .lt. 0.0) go to 170
      if (dbot1 .lt. 0.0) go to 90

c       observer below mesh, and rays could strike bottom.

      if (sintsq.eq.0.) go to 120
      a1omg3=a01*omg3+omega(1)*dbot1
      a2omg3=omega(2)*dbot1
      rhotst=a1omg3**2+a2omg3**2
      if (xmax.ge.a01) go to 130
      if (rhotst.lt.(xmaxsq*omg3**2)) go to 130

c       below ray strikes cylinder or could miss mesh entirely.

   90 disc=xmaxsq*sintsq-(a01*omega(2))**2
      if (disc.lt.0.) go to 300
      a0omg1=a01*omega(1)
```

```
      dcell=(-(a0omg1+sqrt(disc))/sintsq)*dist1
      a(3)=a03+dcell*omega(3)

      if (a(3).lt.zbot .or. a(3).ge.ztop) go to 300

      a(1)=dcell*omega(1)+a01
      a(2)=dcell*omega(2)
      a1a2sq=a(1)**2+a(2)**2
      rhop=sqrt(a1a2sq)

      if (rhop.ge.xmax) go to 300

      iold=nx
      do 100 j=3,nyp-1
      kk=j
      ij=(j-1)*nxp+nx
      if (a(3).lt.y(ij)) go to 110
  100 continue
  110 jold=kk-1
      go to 250
  120 dcell=dbot1
      a(1)=a01
      a(2)=0.0
      a(3)=zbot+distin
      rhop=a01
      a1a2sq=rhop*rhop
      go to 140
  130 if (omg3.eq.0.0) go to 300
      omg3nv=1.0/omg3

c       ray strikes the bottom,  omg3 is not 0.0.

      a(1)=a1omg3*omg3nv
      a(2)=a2omg3*omg3nv
      a(3)=zbot+distin
      a1a2sq=rhotst*omg3nv**2
      rhop=sqrt(a1a2sq)
      if (rhop.ge.xmax) go to 300
  140 continue
      do 150 i=3,nxp-1
      ii=i
      if (rhop.lt.x(nxp+i)) go to 160
  150 continue
  160 iold=ii-1
      jold=2
      go to 250
```

```
  170 continue

c      observer at alt. abv that of the yaqui mesh and
c            rays could strike mesh top

      if (sintsq.eq.0.0) go to 200

c        omg1= + or- =omg2=+ , omg3= - .if omg3=0. ray misses mesh

      a1omg3=a01*omega(3)+omega(1)*dtop1
      a2omg3=omega(2)*dtop1
      rhotst=a1omg3**2+a2omg3**2
      if (xmax.ge.a01) go to 210
      if (rhotst.lt.(xmaxsq*omg3**2)) go to 210

c        ray strikes cylinder or misses mesh entirely

      disc=xmaxsq*sintsq-(a01*omega(2))**2
      if (disc.lt.0.0) go to 300
      dcell=(-(a01*omega(1)+sqrt(disc))/sintsq)*dist1
      a(3)=a03+dcell*omega(3)
      if (a(3).lt.zbot.or.a(3).ge.ztop) go to 300
      a(1)=a01+dcell*omega(1)
      a(2)=dcell*omega(2)
      a1a2sq=a(1)**2+a(2)**2
      rhop=sqrt(a1a2sq)
      if (rhop.ge.xmax) go to 300
      iold=nx

      do 180 j=3,nyp-1
      kk=j
      ij=(j-1)*nxp+nx
      if (a(3).lt.y(ij)) go to 190
  180 continue

  190 jold=kk-1
      go to 250
  200 dcell=dtop1
      a(1)=a01
      a(2)=0.0
      a(3)=ztop-2.0e-06
      rhop=a01
      a1a2sq=rhop*rhop
      go to 220
  210 if (omg3.eq.0.0) go to 300
```

```
c        ray strikes top.  omg3 is not 0.0

      omg3nv=1.0/omg3
      a(1)=a1omg3*omg3nv
      a(2)=a2omg3*omg3nv
      a(3)=ztop-distin
      a1a2sq=rhotst*omg3nv**2
      rhop=sqrt(a1a2sq)
      if (rhop.ge.xmax) go to 300
  220 continue

      do 230 i=2,nxp-1
      ii=i
      ij=(ny-1)*nxp+i
      if (rhop.lt.x(ij)) go to 240
  230 continue

  240 iold=ii-1
      jold=ny
  250 g   = 1.0
      hki = 0.0

      do 270 l=1,1000

      ij=(jold-1)*nxp+iold
      ipj=ij+1
      ijp=ij+nxp
      ipjp=ijp+1

      xd(1)=x(ij)
      xd(2)=x(ipj)
      xd(3)=x(ipjp)
      xd(4)=x(ijp)
      xd(5)=xd(1)

      yd(1)=y(ij)
      yd(2)=y(ipj)
      yd(3)=y(ipjp)
      yd(4)=y(ijp)
      yd(5)=yd(1)

      call howfar(x,y,ms,iray,sintsq,a,omega,dcell,inew,jnew,
     &      iold,jold,xd,yd,iko,a1a2sq,rhop,nko)

      if (iko.le.0 ) go to 300
      ij=(jold-1)*nxp+iold
```

```
      rfr=rfrc(iold,jold)
      jro=jr(iold,jold)
      k=jt(iold,jold)
      tfr=tfrc(iold,jold)
      tfr1=1.-tfr
      if (rfr.ge.1.0) rfr=1.0
      if (rfr.lt.0.0) rfr=0.0
      rfr1=1.-rfr
      opac=(uk(k,jro,mk)*rfr1+uk(k,jro+1,mk)*rfr)*tfr1+(uk(k+1,jro,mk)*r
     1fr1+uk(k+1,jro+1,mk)*rfr)*tfr
      plnkb=plb(k,mk)*tfr1+plb(k+1,mk)*tfr
      ordc=opac*ro(ij)*dcell
      if (ordc.ge.1.e+02) go to 255
      eopd=exp(-ordc)
      go to 257
  255 eopd=0.0
  257 hki = hki +g*plnkb*(1.-eopd)
      g=g*eopd
      if (eopd.eq.0.0) go to 280
      dcell=dcell+1.e-06
      a(1)=a(1)+omega(1)*dcell
      a(2)=a(2)+omega(2)*dcell
      a(3)=a(3)+omg3*dcell
      a1a2sq=a(1)**2+a(2)**2
      rhop=sqrt(a1a2sq)
      kko=0
      if (dcell.lt.1.00001e-06) then
        call rtinc(kko,x,y,ms,a,omega,inew,jnew,iold,jold,
     &                  xd, yd, iko, a1a2sq, rhop)
      endif
      if (a(3).ge.ztop) go to 280
      if (a(3).le.zbot) go to 280
      if (rhop.ge.xmax) go to 280
      iold=inew
  270 jold=jnew

      l=1000

c     write(6,403) iray,iold,jold

  280 hkint(j1ni1,ib) = hki
      iray         = iray+1
      if(hki .ge. hkomax) hkomax=hki

  300 continue
  310 continue
```

```
cemds loop over xk,yk is now over

      hkomx = hkomax

c      write (6,402) iray, nko, hkomax

      hmxfr = hkomx*1.e-07

      if (hkomax.eq.0.0) go to 351

      hkomax = 1./hkomax

cemds if hkint = 0 or hkint = 2.e-30, then set to (1.e-7 * max value)

      j1n    = -nlst
      do 320 j1=1,nlst
        j1n  = j1n+nlst
        do i1= nst,nlst
          j1ni1 = j1n+i1
          if (hkint(j1ni1,ib) .eq. 2.e-30) hkint(j1ni1,ib) = hmxfr
          if (hkint(j1ni1,ib) .eq. 0.0)    hkint(j1ni1,ib) = hmxfr
          j1nn  = j1n+nlst-i1+1
          hkint(j1nn,ib) = hkint(j1ni1,ib)
        enddo
        j1n1 = j1n+nst
  320 continue

      d1irr = 0.0
      d2irr = 0.0
      j1n   = -nlst
      j11   = 0
      j1n2  = (nst-2)*nlst

      do 330 j1=1,nst
        j1n  = j1n  + nlst
        j1n2 = j1n2 + nlst
        do 330 i1=1,nst
          j11   = j11+1
          j12   = j11+nstp
          ik    = i1+nst-1
          j1ni1 = j1n+ik
          j1ni2 = j1n2+ik
          hwkij = hwk(j1)*hwk(i1)
          d1irr = d1irr + hwkij*hkint(j1ni1,ib)*(omgdk*dkinv(j11))**4
          d2irr = d2irr + hwkij*hkint(j1ni2,ib)*(omgdk*dkinv(j12))**4
```

```
 330  continue

      hkirad(mk) = 2.*radspl*radsp*(d1irr + d2irr)/dobssq

  350 continue

      siirad =  0.17*hkirad(7)  + 0.95*hkirad(8)  + 0.89*hkirad(9) +
     &          0.62*hkirad(10) + 0.37*hkirad(11) + 0.12*hkirad(12)
      siband =   hkirad(7)  + hkirad(8)  + hkirad(9)   +
     &           hkirad(10) + hkirad(11) + hkirad(12)

  351 continue
      return

  360 format (6h  a01=,1pe10.3,5h a03=,e10.3,6h dobs=,e10.3,7h  omg1=,e1
     10.3,7h  omg3=,e10.3)
  402 format (9h no rays=,i6,11h  wh calls=,i5,"   hkomax =",1pe11.4)
  403 format(6h iray=,i6,4h  i=,i5,4h  j=,i4)
  700 format (11(1x,e9.3))
  702 format(12h intensities)
      end
c**********************************************************************
      subroutine rdaesop

      common/rlc1/uk(100,7,51), plb (100,51), freq(52), freqd(51)
      common /state/ gt(100,7), fp(100,7)
      common /eqstk/ dk1,d2,d3,d4

      real*8 QBM(51), RHOTBL(8), DELTAQ(51), hnu(51)

C     Read in the opacities, Planck functions, and eq of state data.
C     50 frequency groups -- local Rosseland means.
C     Read in frequency info. and Planck functions
C     B=PI*Planck function. QBM, DELTAQ used for plots.
C     QBM are band misdpoint(Angstr.) in reverse order
C     QBM(1)=midpoint band 50 etc.
C     DELTAQ are band widths in Angstroms.

      MMAX    =  51
      ITBLMAX = 100

      open(file='aesop51',unit=3,status='old')

      READ(3, *) ((plb(I,J), I=1,100), J=1,51), (QBM(K), K=1,51),
     8            (DELTAQ(L), L=1,51), MMAX
```

```
C     GT, FP are tables for interpolating T(eV) and P(dynes/cm**2)

      DO 40 J = 1,7
        READ(3, *) RHOTBL(J), (GT(I,J), I=1,ITBLMAX),
     8             (FP(I,J), I=1,ITBLMAX)
        DO 30 K=1,100
          READ(3, *) (uk(K,J,M), M=1,MMAX)
            DO M = 1,MMAX
              IF (uk(K,J,M) .LT. 0.D0) THEN
                WRITE(7,'('' A uk is < 0.'',3I4,1PE10.3)')
     8             K, J, M, uk(K,J,M)
                STOP
              ENDIF
            ENDDO
  30      CONTINUE
  40    CONTINUE

      close(3)

C     Calc constants for E.O.S. interpolation.  These depend
C     on range of densities and energies in E.O.S. data
C     FI=D3* (LN(EOFX) -D2)    I=FI   EFR= FI-DBLE(I)
C     ALFA=EXP(LN(E(90)/E(1))/89.)  D2=LN(E1/ALFA)  D3=1/LN(ALFA)
C     FJ=  D4*LN(RHOFX) +D1
C     D4= 1./LN(10)    D1= -LN(RHO0)/LN(10)= -D4*LN(RHO0)

      D4    = 1./LOG(10.D0)
      DK1    = -D4*LOG(0.1D0*RHOTBL(1))
      ALFA  = EXP(LOG(1.D16/2.D9)/89.D0)
      D3    = 1.D0/LOG(ALFA)
      D2    = LOG(2.D9/ALFA)

C     Set up the frequency group array.
C     Use HNUR for setting up band edges in eV. 1st bndry=.3185eV,
C     Last one used to be 40393.07 eV.  Now it's 99.4 keV.  Note that
C     we are generating the HNUR(M) here because they are not supplied on
C     the aesop file.  But these hnur are supposed to be the same as the
C     ones used in generating the aesop (aesop51) file.
CPLD  HNUR(39) and above match the bands that Steve White uses for wpn outputs.

cemds units are in eV

      IF (RHOTBL(1) .LT. 1.E-07) THEN
        freq(1) = 0.1
      ELSE
        freq(1) = 0.3185
```

```
      END IF
      DO 60 M=2,43
        IF (M.LE.21) THEN
          freq(M) = .3185*1.2142**(M-1)
        ELSE
          freq(M) = 15.44948501*1.43**(M-21)
        END IF
  60  CONTINUE
      freq(39) = 9.290E+3
      DO 65 M = 40,52
        freq(M) = 1.200*freq(M-1)
  65  CONTINUE

C     Calculate HNU, band mid-points in eV.

      DO M=1,MMAX
        freqd(M) = freq(M+1) - freq(M)
        hnu(M)   = 0.5*(freq(M+1) + freq(M))
      enddo

      return
      end
c**********************************************************************
      subroutine rtinc(kko,x,y,ms,a,omega,inew,jnew,iold,jold,
     &                 xd, yd, iko, a1a2sq, rhop)

      dimension x(ms),y(ms), a(3), omega(3), xd(5), yd(5)

      dx=xd(iko+1)-xd(iko)
      dy=yd(iko+1)-yd(iko)
      if (a(1).eq.0.0.and.a(2).eq.0.0) a(1)=1.e-06

c       we increm. perp. to sfc on whc. pt. lies

      csddr=1.e-06/sqrt(dx**2+dy**2)
      if ((dy*(a(1)*omega(1)+a(2)*omega(2))-rhop*omega(3)*dx).lt.0.) go
     1 to 134
      a(1)=a(1)*(1.+dy*csddr)
      a(2)=a(2)*(1.+dy*csddr)
      a(3)=a(3)-dx*csddr*rhop
      go to 136
  134 a(1)=a(1)*(1.-dy*csddr)
      a(2)=a(2)*(1.-dy*csddr)
      a(3)=a(3)+dx*csddr*rhop
  136  a1a2sq=a(1)**2+a(2)**2
      rhop=sqrt(a1a2sq)
```

```
      call whrko(kko,x,y,ms,rhop,a(3),inew,jnew,iold,jold,
     &   xd(1),xd(2),xd(3),xd(4),xd(5), yd(1),yd(2),yd(3),yd(4),yd(5))

      return
      end
c**********************************************************************
      subroutine whrko(kko,x,y,ms,xp,yp,inew,jnew,iold,jold,
     &                 x4,x1,x2,x3,x5, y4,y1,y2,y3,y5)

      common /mesh/ nxp,nyp,nx,ny

      dimension x(ms),y(ms)

      inew=iold
      jnew=jold
      kko=1
      go to 20
   10 kko=kko+1
      if (kko.ge.200) go to 100
      ij=(jnew-1)*nxp+inew
      ijp=ij+nxp
      ipj=ij+1
      ipjp=ijp+1
      x4=x(ij)
      x1=x(ipj)
      x2=x(ipjp)
      x3=x(ijp)
      y4=y(ij)
      y1=y(ipj)
      y2=y(ipjp)
      y3=y(ijp)
   20 xpx3=xp-x3
      ypy3=yp-y3
      y13=y1-y3
      x13=x1-x3

c       vectors are-
c         d13 =(r1-r3)x (rp-r3)=(y13*xpx3-x13*ypy3)
c         d34 =(rp-r3)x (r4-r3)= ypy3*x43-xpx3*y43
c         d14 =(rp-r4) x(r1-r4) =ypy4*x14-xpx4*y14
c            triangle 123
c         d13= -d13 of triangle 134
c         d23 =(r2-r3) x(rp-r3) =y23*xpx3-x23*ypy3
c         d12 =(rp-r1) x(r2-r1) =ypy1*x21-xpx1*y21
```

```
      d13=y13*xpx3-x13*ypy3
      if (d13.ge.0.0) go to 40

c       if abv fails, p is to left of vector r1-r3

      if (((y2-y3)*xpx3-(x2-x3)*ypy3).lt.0.0) go to 80
      if (((yp-y1)*(x2-x1)-(xp-x1)*(y2-y1)).lt.0.0) go to 30
      go to 50
   30 inew=inew+1
      if (inew.ge.nxp-1) go to 90
      go to 10
   40 if (((x4-x3)*ypy3-(y4-y3)*xpx3).lt.0.0) go to 60

c       p to right of r1-r3.  if abv fails, p in or below triang 134

      if (((yp-y4)*(x1-x4)-(xp-x4)*(y1-y4)).lt.0.0) go to 70

c          if abv fails, d13,d34, and d14 all + and p in tri. 134

   50 iold=inew
      jold=jnew
      x5=x4
      y5=y4
      return
   60 if (inew.eq.2) go to 100
      inew=inew-1
      go to 10
   70 jnew=jnew-1
      if (jnew.lt.2) go to 90
      go to 10
   80 jnew=jnew+1
      if (jnew.ge.nyp-1) go to 90
      go to 10
   90 kko=0

c     means cell out of range

      go to 50
  100 kko=-1

c       means cell not found

      write (6,110) kko,inew,jnew,xp,yp,x4,x1,x2,x3
      write (6,120) y4,y1,y2,y3
      jnew=nyp
      go to 50
```

```
  110 format (2x,3i5,6(1x,e12.6))
  120 format (42x,4(1x,e12.6))
      end
c =====================================================================
      subroutine owtvar1(xk, yk, z, siirad, hkirad, a01, a03,
     &     tsn, ncyc, nlst, mkfst, mklst, nbands)

      dimension xk(nlst), yk(nlst), z(nlst,nlst,nbands)
      dimension hkirad(51)

      write(10,'(2(1pe11.4,1x),"   x, z in cm")') a01, a03
      write(10,'(i6,e13.5)')ncyc,tsn
      write(10,'(i5,i5)')   nlst, nlst
      write(10,'(4e13.5)') (xk(ii),ii=1,nlst)
      write(10,'(4e13.5)') (yk(ii),ii=1,nlst)

      do nb=1,nbands
      do j=1,nlst
       write(10,'(5e14.6)') (z(i,j,nb),i=1,nlst)
      enddo
      enddo

      write(10,'(5e13.6)')(hkirad(kk),kk=mkfst,mklst)
      write(10,'(e13.6)') siirad

      return
      end
c =====================================================================
      subroutine owtvar2(siband, siirad, a01, a03, tsn, ncyc, nang)
      dimension siband(nang), siirad(nang)

c     write(14,'(2(1pe11.4,1x),"   x, z in cm")') a01, a03
c     write(14,'("     t        si band      si bhangmeter  ")')
c     write(14,'(3(1x,1pe11.4))') tsn, siband, siirad

      return
      end
c =====================================================================
      subroutine hwkset(hwk,nlstk,nst)
      dimension hwk(nlstk)

cemds  hwk() = 1/3, 4/3, 2/3, 4/3, 2/3, ..., 1/3 .

      hwk(1)   = 1./3.
      if (nst.eq.3) go to 4
```

```
      do 2 k=2,nst-1,2
        hwk(k)   = 4./3.
        hwk(k+1) = 0.5*hwk(k)
    2 continue
      go to 17

    4 hwk(2)   = 4./3.
   17 continue

      hwk(nst) = 1./3.

      return
      end
c =====================================================================
      subroutine setxkyk(sie,x,y,xk,yk,ikpt,jkpt,ms,nlst,nst,nx,nxp)
      dimension sie(ms), x(ms), y(ms)
      dimension xk(nlst), yk(nlst)

      ijkp   = (jkpt-1)*nxp + ikpt
      sietst = min(1.0e+09,0.5*sie(ijkp))

      do 30 i=ikpt,nx
        ijkpt = (jkpt-1)*nxp+i
        pimax = x(ijkpt)
        if(sie(ijkpt) .lt. sietst) go to 40
   30 continue
   40 continue

      radsp  = pimax/float(nst-1)
      radspl = pimax/float(nst-1)

c        find freq independent geometry

      do 60 i1=1,nlst
        xk(i1)    = float(i1-nst)*radsp
   60 continue

      do 64 j1=1,nlst
        yk(j1) = float(j1-nst)*radspl
 64   continue

      return
      end
c =====================================================================
```

10.3 OpenMP make file on a MAC G5

Next we give a listing of the make file used on a four processor Mac.

```
#! /bin/csh -f
#
FC = gfortran
FFLAGS = -c -O3 -s -fopenmp -fdefault-real-8 -fdefault-integer-8 -fdefault-double-8
FFLAGS1 =  -O3 -s -fopenmp -fdefault-real-8 -fdefault-integer-8 -fdefault-double-8
#
SRCS = isopost.f
SRC_OBJS = ${SRCS:.f=.o}
EXEC = runisop
#
%.o: %.f
${FC} $(FFLAGS) $<
#
${EXEC}: $(SRC_OBJS)
${FC} ${FFLAGS1} -o ${EXEC} $(SRC_OBJS)
#
# additional dependencies
#
isopost.o: comdeck comdeck1
clean:
rm *.o ${EXEC}
```

10.4 Make file for serial processor such as Sun

```
#! /bin/csh -f
# run by executing make -f makiso
# where makeiso is this file
# #
# this is for one program as it stands.
# doing a make debug -f makeiso will compile
# with -g option for debug
# isopost.f is the OpenMP version
#
FC = f90
FFLAGS = -O3 -c
FOPTS = -xtypemap=real:64,integer:64
#
SRC = isopostp
SRCS = isopost.f
SRC_OBJS = ${SRCS:.f=.o}
EXEC = ${SRC:=x}
#OBJ = ${SRC:=.o}
```

```
#
%.o: %.f
${FC} $(FFLAGS) $(FOPTS) $<
#
${EXEC}: $(SRC_OBJS)
${FC} -o $@ $(SRC_OBJS)
#
debug:
${MAKE} "FFLAGS = -g -c" "EXEC = ${EXEC:x=gx}" -f makeiso

#${OBJ}: $(SRCS) comda comdb
#${FC} $(FFLAGS) $(SRCS)
#
#
clean:
rm *.o ${EXEC}
```


Los Alamos
NATIONAL LABORATORY
EST. 1943