NIST
PUBLICATIONS

NIST SPECIAL PUBLICATION 400-84

U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology

Semiconductor Measurement Technology:

A Software Program for Aiding the Analysis of Ellipsometric Measurements, Simple Spectroscopic Models

J. F. Marchiando

Semiconductor Measurement Technology:

A Software Program for Aiding the Analysis of Ellipsometric Measurements, Simple Spectroscopic Models

J. F. Marchiando

Semiconductor Electronics Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

April 1990

U.S. DEPARTMENT OF COMMERCE, Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, John W. Lyons, Director

National Institute of Standards and Technology Special Publication 400-84
Natl. Inst. Stand. Technol. Spec. Publ. 400-84, 362 pages (Apr. 1990)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1990

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402–9325

Table of Contents

Semiconductor Measurement Technology:

A Software Program for Aiding the Analysis of Ellipsometric Measurements, Simple Spectroscopic Models

J. F. Marchiando
Semiconductor Electronics Division
National Institute of Standards and Technology
Gaithersburg, Maryland 20899

MAIN2 is a software program for analyzing spectroscopic ellipsometric measurements. MAIN2 consists mainly of subroutines written in FORTRAN that are used to invert the standard reflection ellipsometry equations for simple systems. Here, a system is said to be simple if the solid material sample is characterized by models which assume at least the following: (1) materials are nonmagnetic; (2) samples exhibit depth-dependent optical properties, such as one with layered or laminar structure atop a substrate that behaves like a semi-infinite half-space; (3) layers are flat and of uniform thickness; and (4) the optical medium within each ambient/layer/substrate is isotropic, homogeneous, local, and linear. The ambient region refers to that region of space which lies external to the layer/substrate structure of the sample. Usually, the ambient region involves a medium of air or vacuum. Each layer is characterized by a thickness and a dielectric function. The dielectric function of a region, i.e., ambient, layer, or substrate, is represented by the Bruggeman effective medium approximation (EMA). Within the EMA, the effective medium of a region is characterized by an aggregate mixture of constituent media, and the dielectric function of each constituent medium is known a priori. The constituent dielectric functions are taken from the literature. The ellipsometric equations are formulated as a standard damped nonlinear least-squares problem and then solved by an iterative method when possible. The program is sufficiently modular to allow one to modify some of the models used in the calculations.

Key words: ellipsometry; EMA; FORTRAN; modeling; software; spectroscopic models.

1. Introduction

In general, when linearly polarized light is incident on a flat surface, it becomes elliptically polarized upon reflection. Ellipsometry involves measuring this induced change in polarization, as well as understanding the measured change in polarization in terms of the optical properties of the material medium. The optical properties of the medium are characterized by the dielectric function and its spatial distribution. MAIN2 is a program for analyzing ellipsometric measurements to find the dielectric function and its spatial distribution. MAIN2 consists mainly of FORTRAN* subroutines that are used to invert a standard set of reflection ellipsometry equations. The systems under consideration here involve only simple structures such as those containing flat, thin, solid films atop a substrate.

A system is said to be simple if the solid material sample may be characterized by models that include at least the following: (1) materials are nonmagnetic; (2) samples exhibit depth-dependent optical properties, e.g., one with a layered or laminar structure atop a substrate that behaves like a semi-infinite half-space; (3) layers are flat and of uniform thickness; and (4) the optical medium within each ambient/layer/substrate is isotropic, homogeneous, local, and linear. The ambient region refers to that region of space which lies external to the layer/substrate structure of the sample. Usually, the medium of the ambient region is air or vacuum. The use of these assumptions is well documented [1–9].

The models used here involve at least two kinds of parameters: each layer is characterized in part by a thickness, and the dielectric function of the medium is expressed in terms of the effective medium approximation (EMA) due to Bruggeman [10–13]. The optical behavior of the ambient, layer, or substrate region is characterized as a macroscopic or physical aggregate mixture of constituent media where the dielectric function of each constituent medium is assumed to be known a priori, and the physical size or cross section of any given aggregate is much smaller than the classical wavelength of the incident light.

* Disclaimer: Certain commercial equipment, instruments, materials, or software products are identified by name in this document in order to adequately specify the experimental procedure or software development. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials, equipment, or software products identified are necessarily the best available for the intended purpose.

Here, the dielectric functions of the constituent media serve as basis functions in representing the effective dielectric function of the spatial region. The relative amount that each constituent medium contributes in representing the effective medium is given by a scalar coefficient and is called a volume fraction. The effective dielectric function is expressed in terms of dielectric functions and volume fractions of the constituents.

A limited database of the characteristics of the constituent media is stored in files, see section 6.5 [4]. These files usually contain a profile of the dielectric function, e.g., a point function over a range of discrete photon energies. Each of these data files has its own distinct subroutine in the software program to interpret the format of stored data. The convention associated with naming these subroutines and data files allows one to easily increase the number and/or resolution of the databases. Also, some provision is made for passing parameters to these subroutines, so that one may alter or change the models used in the representation of the dielectric function.

With only three kinds of model parameters to characterize a region, a sample may be assembled layer by layer atop a substrate. Since the natural parameters of the formulation involve only thickness and dielectric function, it is straightforward to analyze several distinct samples collectively. Here, the program is said to have a multiple-sample capability.

Because of the method of indexing the parameters used in the program, it is possible for two distinct regions to have a model parameter in common, e.g., a common dielectric function but a distinct thickness. Also, it is possible for distinct samples to have parameters in common. Such cases may arise when analyzing samples which have undergone some type of limited processing, e.g., changes in thickness due to etching. Here, the program is said to allow coupling between samples and layers.

The phenomenological models and methods of solution are classical. The ellipsometric equations are formulated as a standard nonlinear, damped least-squares problem that is solved by an iterative method [14–17]. Following the convergence (or nonconvergence) of the iterations to a *good* solution, i.e., within the resolution of the measurement by the instrumentation, the program stops.

Regarding the uniqueness of the calculated solution that is found by minimizing a residual in the least-squares sense, it is important to realize that multiple local minima (pseudo

solutions) may exist, as well as correlation among the modeling parameters [18–21]. Although the program serves as a tool in finding solutions at local minima, the responsibility of assessing the consistency and appropriateness of the resulting solution remains entirely dependent on the user. Because it is possible for the number of model parameters to become large and the run times to become long, the program was designed to run in batch mode, as opposed to interactive mode. Regarding the calculation of the uncertainty of the model parameters, the program uses the LINPACK library [15], which is public domain software.

Although no graphic displays are provided while running the program, provision is made for selecting specific sets of data for output. These data are written to a file in a format amenable to the generation of graphics later. Thus, graphics capability is dependent upon the resource library available at the local computing site. Actually, any one of several software graphics packages would be more than adequate. But for convenience, as well as for completeness, this document refers to the NCAR software graphics package [22, 23]. This is mentioned briefly in section 4.1.3.

This report presents a brief overview of the theory and methods of solution and provides a brief manual of operations or guide for the user to the options and capabilities of the software program. Section 2 discusses the models of the dielectric functions. Section 3 discusses the methods of inverting the ellipsometric equations. Section 4 discusses the manual of operations for the software package. This includes an overview regarding some of the considerations that went into the design of the software package. This overview includes a short listing of parameters which are used in specifying the allocations of the various arrays. Presented next are the free-field formats which are used when entering the required sets of input data. These include the tabular lists of model parameters, the specification or characterization of the samples, and the collective sets of measurements of the ellipsometric angles. Finally, information is given regarding the command options that are available to the user for exercising operational control over the program, i.e., constraints of procedures during analyses. Section 5 discusses a few selected examples of actual runs of the software package. The input and output data files are listed. Section 6 contains a listing of the software source files. Lastly, the appendix contains a quick reference guide for using the program and building the necessary input data files.

The source code is written predominantly in standard FORTRAN-77. Minor compiler extensions involve the DO-ENDDO, the INCLUDE, and the OPEN and CLOSE statements; such are appropriate for a VAX computer. The system clock is also machine dependent, see section 6.3.12. The source code contains some in-line comments for the user's convenience, but some compilers may find this to be a nuisance. The more expert user may want to modify some of the routines while taking advantage of the interior data structure of the program, whereas the less expert user may simply want to operate the package as presented. A copy of the ASCII files of the source program and an installation guide are available from the author upon written request on letterhead stationery and receipt of a DOS-formatted floppy diskette or a small reel of nine-track magnetic tape.

2. The Dielectric Function

The theory describing the scattering of light for ellipsometry, as well as the numerical methods of calculation of the models, has been discussed in detail elsewhere [1–3]. The discussion presented in [3] involved the reflection and refraction of a time-harmonic monochromatic plane electromagnetic wave obliquely-incident on a flat plane surface of a stratified medium exhibiting depth-dependent optical properties. The optical properties of the material medium were assumed to be piecewise continuous, isotropic, local, linear, and not perturbed by the effects of free charges and surface currents.

The coordinate geometry used here is the same as that presented in [3]. The material medium is assumed to occupy the semi-infinite positive half-space ($z > 0$) with the flat surface being the plane ($z = 0$). For the isotropic homogeneous medium with depth-dependent (z) optical properties, there is translational invariance within the (x, y) plane. The z-axis is aligned in a direction normal to the surface and planes of stratification. A lossless isotropic homogeneous medium like air or vacuum is assumed to occupy the so-called ambient region ($z < 0$). A source time-harmonic monochromatic plane wave is assumed to be incident on the surface of the medium ($z = 0$), from the ambient region ($z < 0$), in a direction of angle ϕ with respect to the surface normal. The plane of incidence is taken to be the (z, x) plane, and the direction cosines of propagation are both positive.

The profile of the stratified structure of the dielectric function is approximated by a stepped profile; the dielectric function of a region, i.e., ambient, layer, or substrate, is assumed to depend only on the optical frequency of light. The dielectric function in [3] was expressed in terms of one index of refraction and one extinction coefficient per optical frequency. Although this may be adequate for analyzing measurements involving very few optical frequencies, spectroscopic ellipsometry usually involves analyzing measurements at very many frequencies. Here, dispersion is used to advantage in analysis; i.e., the dielectric function may change with frequency, whereas the layer thickness may not change. The dielectric function of the ambient, layer, or substrate region is assumed to be characterized by the optical behavior of a macroscopic or physical aggregate mixture of constituent media, where the dielectric function of each constituent medium is known a priori. Further, the cross-sectional diameter of any given aggregate is assumed to be small compared to the classical wavelength of the incident light, and there is negligible interaction between

neighboring aggregates. Such ideas in part form bases of several effective medium theories. The model assumed here is the effective medium approximation (EMA) associated with Bruggeman [10–13]. The relative amount that each constituent medium contributes to the effective medium of the region is given by a real-valued scalar number called a volume fraction. The following subsections present brief discussions regarding the calculations of the dielectric functions of the constituent media, the effective media within the EMA, and the partial derivatives of the dielectric functions with respect to the associated model parameters.

2.1 Constituent Media

Within any given region, i.e., ambient, layer, or substrate, the effective medium is assumed to behave like an aggregate mixture of distinct constituent media. Within the EMA, the dielectric function (ε) of the effective medium is expressed in terms of the dielectric functions (ε_j) of the constituent media, where j indexes the distinct constituent media. The (ε_j) serve as basis functions for (ε) and are assumed to be known functions of the optical frequency.

Here, a constituent medium is characterized by its optical properties, i.e., the dielectric function. A constituent medium is considered distinct insofar as its optical dielectric function is distinct. Hence, a constituent medium may refer to a distinct chemical entity and its associated optical properties, e.g., due to a phase state. For example, regarding a thermally grown oxide atop a silicon wafer, j may refer to the distinct chemical entities of SiO_2 or Si, as well as the associated phases that may be present, e.g., amorphous SiO_2, amorphous Si, or crystalline Si. Since these materials have distinct dielectric functions, they are considered to be three distinct constituent media.

To calculate the dielectric function of a constituent medium, a distinct subroutine is used for each distinct constituent medium. Each of these subroutines contain an argument list, so that it is possible to optimize on model parameters that are local to the individual constituent dielectric function. Since these dedicated subroutines have similar names and argument lists, one may easily modify the database of any individual constituent medium. The convention used here for naming subroutines that calculate constituent dielectric functions is of the form, DIELxx, where xx refers to a two-digit number between 01 and 99.

These subroutines calculate (ε_j) from those models that have been found or reported in the available open literature. Usually, this involves evaluating a Sellmeier equation or a spline fit of a point function defined on a discrete grid of energies, i.e., where the optical frequencies have been expressed in terms of energy. The point function is stored in and read from some data file, and the spline-fitting algorithms are from the GAMS software library [14]. The point functions are stored in files with suggestive filenames. For example, to calculate the dielectric function of gallium arsenide, the subroutine DIEL08 reads and stores the entire point function that is stored in file W.GA_AS during the first call, see sections 6.5.8 and 6.6.6, respectively. Then, DIEL08 needs to evaluate only two spline fits per call, i.e., the real and imaginary parts of the complex refractive index or dielectric function.

Note also that these subroutines, i.e., DIELxx, are all called from the same subroutine, DIELMN, see section 6.4.4. (LMN is meant to be a mnemonic for *elements*.) If it is ever decided (by the user) that the number of distinct constituent media in the database is to be enlarged by including additional subroutines, then appropriate calls to these additional subroutines must be included in the subroutine DIELMN. Further, it may be necessary to modify the parameter called `nlmnts` that is discussed in section 4.1.2 as well.

2.2 The Effective Medium Approximation

The effective medium approximation [10–13] for determining the dielectric function (ε) of a distinct spatial region, i.e., ambient, layer, or substrate, is given by the following equation

$$0 = \sum_{j=1}^{N} f_j \left(\frac{\varepsilon_j - \varepsilon}{\varepsilon_j + 2\varepsilon} \right) \tag{1}$$

subject to the constraint

$$\sum_{j=1}^{N} f_j = 1, \tag{2}$$

where N denotes the number of distinct constituent media in the mixture, f_j denotes the volume fraction of the j^{th} constituent, ε_j denotes the dielectric function of the j^{th} distinct constituent medium evaluated at the appropriate classical optical frequency of light, and ε is the dielectric function of the effective medium. It is assumed that f_j and ε_j are known on input, and that f_j is independent of the frequency and is a real valued scalar. Equation (1) must then be solved for ε. The method of solution used here is Newton iteration. This involves a three-step procedure.

The first step involves reformulating eq (1) in order to simplify the functional dependence on ε. Equation (1) may be written as

$$0 = \sum_{j=1}^{N} f_j \left\{ \frac{\varepsilon_j + 2\varepsilon - 3\varepsilon}{\varepsilon_j + 2\varepsilon} \right\} = \sum_{j=1}^{N} f_j \left\{ 1 - \frac{3\varepsilon}{\varepsilon_j + 2\varepsilon} \right\}.$$

Thus,

$$\frac{1}{3\varepsilon} = \sum_{j=1}^{N} \frac{f_j}{\varepsilon_j + 2\varepsilon}. \tag{3}$$

Equation (1) may also be written as

$$\sum_{j=1}^{N} f_j \frac{\varepsilon_j}{\varepsilon_j + 2\varepsilon} = \varepsilon \sum_{j=1}^{N} \frac{f_j}{\varepsilon_j + 2\varepsilon} = \varepsilon \left(\frac{1}{3\varepsilon} \right) = \frac{1}{3}, \tag{4}$$

where a substitution has been made from eq (3).

The second step involves applying the Newton algorithm to eq (4). Here, one seeks the root to an equation, say Q, given by

$$Q(\varepsilon) = \sum_{j=1}^{N} \frac{f_j \varepsilon_j}{(\varepsilon_j + 2\varepsilon)} - \frac{1}{3}. \tag{5}$$

The root minimizes $|Q|$. The variation of Q given by

$$\delta Q = -(\delta\varepsilon)\, 2 \sum_{j=1}^{N} \frac{f_j \varepsilon_j}{(\varepsilon_j + 2\varepsilon)^2} \tag{6}$$

forms the basis for the Newton step. The Newton step is related to $(Q/\delta Q)$.

The third step involves providing a reasonable initial solution. This is necessary for starting any iterative method of solution. For convenience, the initial solution is given by

$$\varepsilon = \frac{1}{3}(2\varepsilon_{\parallel} + \varepsilon_{\perp}),$$

where

$$\varepsilon_{\parallel} = \sum_{j=1}^{N} f_j \varepsilon_j \qquad \text{and} \qquad \frac{1}{\varepsilon_{\perp}} = \sum_{j=1}^{N} \frac{f_j}{\varepsilon_j},$$

which follows from considering two extreme cases when the electric field vector is oriented either parallel or perpendicular to the layers of the stratified media.

The criterion for convergence in finding the root is that the Newton step be at least six orders of magnitude smaller than the magnitude of the dielectric function of the effective medium. This algorithm is located in subroutine DIEEMA, see section 6.4.2.

To help understand eqs (1) and (5), consider the following example. Consider a sample of silicon with a thermally-grown oxide layer. Let the layered structure consist of two layers and a substrate. Here, the substrate is silicon, and the top layer is amorphous SiO_2. Let a transition region layer lie between the top layer and substrate, and let it be an aggregate mixture of half Si and half amorphous SiO_2. This structure involves three effective media, i.e., one to a region, but only two constituent media, i.e., crystalline Si and amorphous SiO_2. Accordingly, (ε_j) may refer to either (ε_{Si}) or (ε_{SiO_2}). Regarding eq (5), it follows that $(N = 1)$ and $(f_j = 1)$ for both the top layer and the substrate regions. Hence, $(\varepsilon = \varepsilon_{SiO_2})$ for the top layer, and $(\varepsilon = \varepsilon_{Si})$ for the substrate. Regarding the transition region, $(N = 2)$ and $(f_{Si} = f_{SiO_2} = 0.5)$. Letting the optical frequency be that corresponding to an energy of 2.0 eV, one may evaluate the complex dielectric functions for the constituent media. From sections 2.1, 6.6.1, and 6.6.3, it follows that

$$\varepsilon_{Si} \approx 15.256 + i\,0.172 \qquad \text{and} \qquad \varepsilon_{SiO_2} \approx 2.124 + i\,0.0,$$

where $i = \sqrt{-1}$. Then, regarding eq (5), the root is found to be

$$\varepsilon \approx 6.747 + i\,0.052,$$

i.e., the dielectric function of the effective medium of the transition region.

2.3 Partial Derivatives

The problem associated with inverting a standard set of reflection ellipsometry equations has been discussed in detail elsewhere [1–3]. The inverse problem is formulated as a least-squares problem. The method of solution used here involves an iterative procedure for improving the numerical values of the model parameters [3]. This requires some understanding of the functional dependence of the dielectric function on the necessary model parameters. Hence, one must calculate the partial derivatives of the dielectric function with respect to any model parameter that may be selected for optimization. Regarding the EMA, this involves at least two kinds of partial derivatives.

The first kind involves variations in model parameters that are local to the constituent dielectric function, but yet are independent of volume fraction and thickness. Letting the prime denote a partial derivative with respect to such a model parameter, and then operating on eq (4), it follows that

$$0 = \sum_{j=1}^{N} f_j \left\{ \frac{\varepsilon_j'}{(\varepsilon_j + 2\varepsilon)} - \frac{\varepsilon_j \varepsilon_j'}{(\varepsilon_j + 2\varepsilon)^2} - \frac{2\varepsilon_j \varepsilon'}{(\varepsilon_j + 2\varepsilon)^2} \right\},$$

which may be rearranged to yield

$$\varepsilon' = \varepsilon \frac{\sum_{j=1}^{N} \frac{f_j \varepsilon'_j}{(\varepsilon_j + 2\varepsilon)^2}}{\sum_{i=1}^{N} \frac{f_i \varepsilon_i}{(\varepsilon_i + 2\varepsilon)^2}}. \tag{7}$$

To evaluate ε' from eq (7), the program uses subroutine DIEMAD, see section 6.4.3.

The second kind involves variations in the volume fractions. Again, starting with eq (4), it follows that

$$0 = \sum_{j=1}^{N} \left(\frac{\varepsilon_j}{\varepsilon_j + 2\varepsilon} \right) (\delta f_j) \; - \; (\delta\varepsilon)\, 2 \sum_{j=1}^{N} \frac{f_j \varepsilon_j}{(\varepsilon_j + 2\varepsilon)^2},$$

which implies that

$$\left(\frac{\partial \varepsilon}{\partial f_j} \right) = \frac{\left(\frac{\varepsilon_j}{\varepsilon_j + 2\varepsilon} \right)}{2 \sum_{i=1}^{N} \frac{f_i \varepsilon_i}{(\varepsilon_i + 2\varepsilon)^2}} \tag{8}$$

when the f_j are mutually independent variables. Yet, from eq (2), it follows that

$$0 = \sum_{j=1}^{N} (\delta f_j). \tag{9}$$

Hence, the variations are linearly dependent. Since each independent constraint removes one degree of freedom, if some of the volume fractions of a layer are selected for optimization, then at least two or more of the f_j must be involved in the variation. Since,

$$\delta\varepsilon = \sum_{j=1}^{N} \left(\frac{\partial \varepsilon}{\partial f_j} \right) \delta f_j,$$

then one may evaluate the partial derivative of ε with respect to a set of unconstrained variables $(\hat{f}_j)$ by letting

$$\left(\frac{\partial \varepsilon}{\partial \hat{f}_j} \right) = \left(\frac{\partial \varepsilon}{\partial f_j} \right) - \left(\frac{\partial \varepsilon}{\partial f_k} \right), \tag{10}$$

where j and k involve the set of model parameters that are selected for variation but are coupled by the constraint, where k involves a fixed index selected arbitrarily by the user, and where $(j \neq k)$. For convenience, the program uses: $(k < j)$. To evaluate the $(\partial\varepsilon/\partial f_j)$ in eq (8), the software program uses subroutine DIEEMA, see section 6.4.2.

3. The Inverse Problem

3.1 Formulation of the Least-Squares Problem

In order to characterize the layered structure of the sample, it is necessary to invert the standard ellipsometric equations. These equations relate how the optical properties of the material sample induce a phase-shift in the reflected light that is measured by the ellipsometer. Because of the nonlinearity of the equations, it is usually not possible to find simple analytic expressions which will invert the equations. The common approach to performing such inversions is to formulate them as nonlinear least-squares problems [1–3]. Here, one considers a sequence of forward problems, where each increment of the sequence involves three distinct steps. The steps are: starting with a good estimate of values of the model parameters, determining the deviations between the experiment and the model, and then updating the model parameters with *better* values. This sequence is repeated until the magnitude of the corrections become sufficiently *small*.

The ellipsometric equations for the ellipsometric angles, Δ and ψ, are of the form:

$$\Delta = \Delta(\nu, \phi, \mathbf{b}) \qquad \text{and} \qquad \psi = \psi(\nu, \phi, \mathbf{b})$$

where ν is the frequency of the incident light; ϕ is the angle of incidence; and $\mathbf{b}$ is an array where the components specify the model parameters, e.g., layer thicknesses and volume fractions. The standard procedure for inverting the above equations involves minimizing some objective function or nonnegative scalar *error* expression containing the deviations between experiment and model of Δ and ψ in the least-squares sense, e.g.,

$$G(\mathbf{b}) = \frac{1}{2M} \sum_{i=1}^{M} \left[\left(\frac{\Delta_i^{\mathrm{e}} - \Delta_i^{\mathrm{m}}}{\delta \Delta_i^{\mathrm{e}}} \right)^2 + \left(\frac{\psi_i^{\mathrm{e}} - \psi_i^{\mathrm{m}}}{\delta \psi_i^{\mathrm{e}}} \right)^2 \right] \tag{11}$$

$$= \sum_{i=1}^{M} \left(g_{\Delta,i}^2 + g_{\psi,i}^2 \right) = \mathbf{g}^{\mathrm{T}} \mathbf{g} = |\mathbf{g}|^2$$

where superscripts (e, m) refer to experiment and model, respectively, M refers to the number of measurements of (Δ, ψ) from experiment, $\mathbf{g}$ is an array of the deviations, e.g., $(\Delta_i^{\mathrm{e}} - \Delta_i^{\mathrm{m}})$, that are scaled by the uncertainties in the measurement, i.e.,

$$g_{\Delta,i} = \frac{1}{\sqrt{2M}} \left(\frac{\Delta_i^{\mathrm{e}} - \Delta_i^{\mathrm{m}}}{\delta \Delta_i^{\mathrm{e}}} \right) = \mathbf{g}_{2i-1} \tag{12}$$

$$g_{\psi,i} = \frac{1}{\sqrt{2M}} \left(\frac{\psi_i^{\mathrm{e}} - \psi_i^{\mathrm{m}}}{\delta \psi_i^{\mathrm{e}}} \right) = \mathbf{g}_{2i} \tag{13}$$

where $(1 \leq i \leq M)$, the superscript T denotes transposition, and the uncertainties in the measurement of Δ_i^e and ψ_i^e are denoted by $\delta\Delta_i^e$ and $\delta\psi_i^e$, respectively.

The third step of the sequence is concerned with the procedure for obtaining *better* numerical values for the model parameters, i.e., the Newton step. An estimate of the necessary correction may be realized by linearizing the functional representation of the model and solving the resulting matrix equation, $\mathbf{g} = \mathbf{J}\mathbf{v}$, where $\mathbf{v}$ is a column array (Newton step) for improving the model parameters that were selected to undergo variation, i.e., $v_j \propto \delta b_j$, and where the Jacobian $\mathbf{J}$ is a sparse matrix, $J_{ij} \propto (\partial\Delta_i/\partial b_j)$.

Such matrix equations are common to optimization problems. It is well known that additional numerical stability may result if one requests that the norm of $\mathbf{v}$ be minimized as well. This involves modifying the *error* expression to

$$G = (\mathbf{g} - \mathbf{J}\mathbf{v})^{\mathrm{T}}(\mathbf{g} - \mathbf{J}\mathbf{v}) + \kappa\mathbf{v}^{\mathrm{T}}\mathbf{v}, \tag{14}$$

where κ is a positive scalar parameter subjectively chosen between 0.01 and 1.0 for our calculations. Of course, the final solution $\mathbf{v}$ ought to be independent of κ.

It is also known that the columnar scaling of $\mathbf{J}$ affects the accuracy of the solution, as well as the effectiveness of κ. A simple choice for the scaling can be found by considering the diagonal elements from $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ and then defining the diagonal matrix, $\mathbf{S}$, where

$$S_{jj} = \left(\left[\mathbf{J}^{\mathrm{T}}\mathbf{J}\right]_{jj}\right)^{1/2}. \tag{15}$$

Letting

$$\begin{aligned}
\tilde{\mathbf{J}} &= \mathbf{J}\mathbf{S}^{-1}, \\
\tilde{\mathbf{v}} &= \mathbf{S}\mathbf{v}, \\
\mathbf{r} &= \mathbf{g} - \mathbf{J}\mathbf{v} = \mathbf{g} - \tilde{\mathbf{J}}\tilde{\mathbf{v}},
\end{aligned}$$

a suitable *error* expression may be defined by

$$\begin{aligned}
G &= \left(\mathbf{g} - \tilde{\mathbf{J}}\tilde{\mathbf{v}}\right)^{\mathrm{T}}\left(\mathbf{g} - \tilde{\mathbf{J}}\tilde{\mathbf{v}}\right) + \kappa\tilde{\mathbf{v}}^{\mathrm{T}}\tilde{\mathbf{v}} \qquad (16) \\
&= \mathbf{r}^{\mathrm{T}}\mathbf{r} + \kappa\tilde{\mathbf{v}}^{\mathrm{T}}\tilde{\mathbf{v}}.
\end{aligned}$$

The criterion for critical points or relative minima is a vanishing variation, i.e., $\partial G/\partial \tilde{v}_j = 0$, which yields a set of equations that may be expressed as

$$\begin{bmatrix} \mathbf{1} & \tilde{\mathbf{J}} \\ \tilde{\mathbf{J}}^{\mathrm{T}} & -\kappa\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \tilde{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix},$$

and must then be solved for $\tilde{\mathbf{v}}$.

Because of the sparsity of the Jacobian, it is expedient to utilize an iterative method for solving the above matrix algebra problem. Algorithms [14, 15] exist that specifically address this type of problem, one [16, 17] of which utilizes a relatively stable Lanczos process (Krylov space decomposition) in formulating the method of steepest-descents. Essentially, the method requires that each updating vector be orthogonal to the previous update vectors. From this procedure, one finds $\tilde{\mathbf{v}}$, which leads to $\mathbf{v}$, which is then used to improve the estimate of the values for the model parameters, e.g., $b_j^{\mathrm{new}} = b_j^{\mathrm{old}} + v_j$.

Using this improved $\mathbf{b}$, the sequence is repeated again until either: $|\mathbf{g}|$ becomes sufficiently *small*, of the order of a few millidegrees, e.g., the resolution of the measurement, or until $|\tilde{\mathbf{v}}|$ become sufficiently *small* so that $|\mathbf{g}|$ suffers no further reduction regardless of magnitude. It is especially during this last case that it becomes necessary to scan a grid of model parameters. Multiple local-minima may be encountered, e.g., nonuniqueness. Often, this reveals either: (i) correlation which prevents model parameters from being resolved independently; i.e., the measurement data are not sufficiently functionally independent which thereby induces a functional dependence among the model parameters; or (ii) the inadequacy of the model in providing a sufficiently good physical description of the process, which is likely whenever $|\mathbf{g}|$ greatly exceeds the resolution of the measurements.

Finally, it is important to realize that in the above-outlined steps, the emphasis centers on searching for and ascribing *good* values to the parameters of a physical model which has already been specified. Only one specific model is assumed to have been applied to a given set of measurements. But often situations may occur, or questions may arise, where it is important to consider alternative models during the effort to further reduce the value of the *error* expression G, and these must be investigated as well. Also, it is possible that two or more distinct models may be found that reduce the deviations in $|\mathbf{g}|$ to comparable magnitudes. Then, the problem of characterizing the sample becomes one of comparing models which ought to lead toward a decision about selecting the *better* model. One heuristic approach has been to select that model which provides the smallest deviations,

i.e., $|\mathbf{g}|$, while utilizing the fewest modeling parameters and being consistent with physical reality. However, comparisons assume some criteria or ordering, and that requires a number. So the problem becomes one of reducing a model to a number. This reduction is certainly not simple and fully merits its own discussion, e.g., hypothesis testing and decision theory. But such a discussion is beyond the scope that is intended here, so that it is expedient to direct the reader to consider simple statistics and use the F–statistic in assessing the so-called *goodness-of-fit* test. For further discussion on topics such as parameter estimation, hypothesis testing, significance testing, and other formulations involving decision processes, the reader is advised to consider the statistics literature available.

3.2 Sensitivity Analysis of Model Parameters

In the above discussion of the least-squares problem, the emphasis was on obtaining accurate numerical values for the model parameters. Following this, it is natural to consider next some assessment of the uncertainties associated with those values. For a restricted class of linearizable problems and assumptions, one may ascribe estimates to these uncertainties by utilizing a formulism similar to that used in the least-squares problem [3].

Starting with the functional representation of the model, and then expanding about the critical or fixed point solution, the deviations may be expressed by assuming variations out to first order, i.e.,

$$\Delta_i^{\mathrm{e}} - \Delta_i^{\mathrm{m}} = \sum_{j=1}^{N} \left.\frac{\partial \Delta_i}{\partial b_j}\right|_o \delta b_j + \left.\frac{\partial \Delta_i}{\partial \phi}\right|_o \delta\phi + \tilde{e}_{\Delta,i} \tag{17}$$

and

$$\psi_i^{\mathrm{e}} - \psi_i^{\mathrm{m}} = \sum_{j=1}^{N} \left.\frac{\partial \psi_i}{\partial b_j}\right|_o \delta b_j + \left.\frac{\partial \psi_i}{\partial \phi}\right|_o \delta\phi + \tilde{e}_{\psi,i} \ , \tag{18}$$

where N refers to the number of distinct model parameters, and $\tilde{e}_i$ refers to the uncertainty associated with individual measurements of (Δ, ψ) performed by the instrument. Further, it may be the case that during the course of analysis of finding the critical point solution, some of the model parameters will have had values and uncertainties assigned to them from some earlier experiment or measurement that is external to and distinct from those being analyzed here, e.g., values taken from the literature. These values are assumed as given and remain unchanged during the calculations. Consequently, the set of model parameters (δb_j) may be partitioned into two disjoint sets, those that remain unchanged

(u_j) and those that are allowed to vary (v_j). This allows both of the above expansions to be combined and expressed as

$$g_i = J_{v,ij} v_j + J_{u,ij} u_j + J_{\phi,i} \tilde{\phi}_i + \tilde{e}_i, \tag{19}$$

where g_i refers to deviations in (Δ, ψ) between measurement and theory, e.g., $(\Delta_i^{\mathrm{c}} - \Delta_i^{\mathrm{m}})$ *without* scaling by the measurement uncertainty; J refers to the Jacobian or the array of partial derivatives that is appropriate for the partitioning and evaluated at the critical point; and $\tilde{\phi}_i$ involves the uncertainty in the angle of incidence *and* the assumption of stochastic independence between each measurement, and the implied summation affects only the j index. The sensitivity analysis of the model parameters centers on the procedure of assigning values to the uncertainty in (b_j) that are associated with (v_j). Since this has been discussed in [3], only terms associated with (v_j) are considered in what follows.

From eqs (11), (14), and (19), it follows that the minimum value of the *error* expression G is attributed to residuals that are due to measurement uncertainties $\tilde{e}_i$ that are assumed to behave as random variables in a statistical sense, i.e., being stochastically independent and identically distributed. So, when G is evaluated at the critical point solution, it is assumed that

$$G_o = |\mathbf{g}|^2 = |\mathbf{e}|^2, \tag{20}$$

and that an expansion of G about the critical point may be shifted to zero and written as

$$\tilde{G} = G - G_o = |\mathbf{g} - \mathbf{J}_v \mathbf{v}|^2 - |\mathbf{g}|^2. \tag{21}$$

A simple estimate of the variances of the model parameters may be found from

$$\langle v_j v_k \rangle = s_g^2 \left[\left(\mathbf{J}_v^{\mathrm{T}} \mathbf{J}_v \right)^{-1} \right]_{jk}, \tag{22}$$

where

$$s_g^2 = \frac{\mathbf{g}^{\mathrm{T}} \mathbf{g}}{2M - N} \tag{23}$$

estimates the variance of a χ^2 distribution with $2M-N$ degrees of freedom, M is the number of measurements of (Δ, ψ), and N is the number of model parameters (v_j) undergoing variation. In the absence of correlation, the magnitude of uncertainty $(\mathcal{U})$ assigned to model parameter (v_j) may be estimated by

$$\mathcal{U}(v_j) \leq \langle v_j v_j \rangle^{1/2}. \tag{24}$$

To calculate the necessary terms given in eq (22), the program uses subroutine CORLAT.

4. Manual of Operations

In order to use MAIN2, it is necessary to: (1) specify the model parameters that characterize the layered structure of the sample, (2) collect the necessary measurement data of ellipsometric angles and the associated energies and angles of the incident light, (3) assign size allocations of arrays which hold these data, (4) assign formats and filenames which are provided for entering the input data and receiving the output data from calculations, (5) understand the stored databases of constituent media, and (6) understand the limitations and capabilities of the software package and its utility for implementing strategies in analyzing the measurement data. The filename convention assumes a filename and an extension made of alphanumeric characters in the standard format given by: filename.extension. This section presents a brief overview of these considerations.

4.1 Software Development Considerations

The main program, MAIN, performs a small set of functions, see section 6.2.1. First, it calls a subroutine to open the necessary data files. Second, it requests a subroutine to read most of the input data file. Third, it allows the user to select an option from a menu or tabulated list of command options and then calls the appropriate subroutine. Following the return from the subroutine to the main program, the main program stops; no further command options are processed.

4.1.1 Input/Output Files

MAIN2 involves at least two kinds of data files. The first kind of file involves five input/output files for calculations. Two files serve for entering input data, and three files serve for collecting output data.

X.DAT is an input data file. It contains: the model parameters that characterize the sample, the configuration of the layered structure of the sample, and the ellipsometric angles of the measurement data. An example of an X.DAT file is shown in section 4.2.9.

X.INN is an input data file. It contains the sequential list of command options that provide control of the program. These are discussed further in section 4.3. Examples of X.INN files may be found in section 5.

X.OUT is an output data file. It contains a collective list of: all the entered input data and the general proceedings generated during the course of a calculation, as well as any informative error messages. Examples are shown in section 5.

X.SOUT is an output data file for intermediate solutions. It contains the breakpoint information that may be generated during any grid scan of the model parameters. This information is written to the file at periodic intervals of 15 CPU-minutes and overwrites the information from previous breakpoints, so that only the last or most recent breakpoint is retained. The procedure for restarting or resuming a previously interrupted calculation involves the simple task of appending the output of this file X.SOUT onto the end of file X.INN, i.e., without any intervening blank lines, and then re-executing the job. When the job is re-executed, the program will attempt to read only one set of breakpoint information in file X.INN before resuming calculations. Consequently, it is important to replace any earlier breakpoint data in X.INN, as appropriate.

X.PLOT is an output data file for graphics. The data are formatted in a manner that is intended to be amenable for later reading and plotting. Examples are shown in section 5.

These files are opened initially by subroutine FILEOP, see section 6.2.2. Since breakpointing involves opening/closing files, other open/close statements may be found in subroutines SCANO1 and SCANO2. The integer logical units that are associated with these files are assigned by the block data statement located in BLKDAT.FOR. These logical units are passed throughout the program by the named common block statement in file IOUNIT.

All output files are deleted at the start of the program, as may be seen from looking at files MAIN.FOR and FILEOP.FOR. Thus, it is important to append any breakpoint information contained in X.SOUT to X.INN, as appropriate, *before* the next execution of the program, lest that which was stored in X.SOUT be overwritten and lost.

The second kind of data file involves the database of constituent media. These data files serve as input to subroutines, i.e., DIELxx, which calculate the constituent dielectric functions. Each data file is associated with a distinct constituent medium. Usually, the data file contains a discrete profile of the complex dielectric function and/or the complex refractive index as a function of energy. Naturally, the filenames must match that assumed

by the subroutines that read them. Here, the filenames are rather suggestive and straightforward, e.g., W.SI for crystalline silicon, W.SIA for amorphous silicon, W.SI3_N4 for noncrystalline Si_3N_4, W.GA_AS for crystalline gallium arsenide, etc.

4.1.2 Allocation of Array or Work Space

In general, the allocation size of the arrays used in the program may be determined or estimated from a set of eight constants. These constants are assigned their numerical values prior to compilation by parameter statements located in file DEFNIT, see section 6.1.2. Consequently, if any changes are made to these assignments, it is usually necessary to recompile and relink most (if not all) of the subroutines in order to incorporate the said changes into the running program. The following list includes the eight basic constants.

`nsampl`, the maximum number of samples allowed for analysis, where the sample involves some finite number of films/layers atop a substrate.

`nfilms`, the maximum number of films/layers allowed atop the substrate of any given sample.

`nparms`, the maximum number of distinct parameters allowed per mixture or effective medium.

`nlmnts`, the maximum number of distinct constituent media (ε_j) allowed during analysis.

`nbient`, the maximum number of distinct ambients allowed atop the layers/substrate of any sample.

`nwaves`, the maximum number of distinct classical optical frequencies or wavelengths in vacuum of light incident on any sample.

`nanglx`, the maximum number of angles of incidence allowed per classical optical frequency of light incident on any sample.

`nrpeat`, the maximum number of repeats of an experiment run, where an experiment run involves taking measurements of (Δ, ψ) for a range of incident angles and energies on the same (ambient/sample) configuration.

In some cases, the array allocations are overestimated. The algorithms that estimate the array allocations from these eight constants assume a worst-case scenario involving the

largest size possible being calculated. This occurs especially when sizing arrays for the Jacobian. Here, it is convenient to assign some upper limit to these arrays. This is the purpose of the constant named `nnjaaa`. It is located in file ARRAYS, see section 6.1.7. A few internal check-tests are provided to help ensure that indices remain within bounds of the dimensions of these arrays.

4.1.3 Library Software

During the course of calculation of the least-squares fit, it is sometimes necessary to estimate the sensitivity of the model parameters. As discussed earlier in section 3.2, e.g., eq (22), it is necessary to calculate the inverse of a square matrix, i.e., the matrix formed by the product of the Jacobian and its transpose. This may be evaluated by using subroutines from the LINPACK library of mathematical software [15]. Reference calls to such subroutines may be found in subroutine CORLAT, see section 6.2.21.

4.2 Input Data Requirements

As mentioned previously in section 4.1.1, the input data file X.DAT contains: the model parameters, the characterization of the layered structure of the samples, and the measurement data of ellipsometric angles. Also, the input file contains information regarding the locations of the database files of point functions that are used in calculating the dielectric functions of the distinct constituent media. Here, all of the database files reside in the same directory.

The following subsections develop a line-by-line construction of file X.DAT. As mentioned above, the input data may be partitioned into subsets. Each subsection discusses one subset, and the subsets are presented in the order that they are expected to occur in the input file. Because each subset of data requires its own input format, each format is demonstrated with a worked example. For convenience, an example of a completed input file is presented in the last subsection, section 4.2.9.

The main program uses subroutine INPDAT to read all of the data in file X.DAT, see section 6.2.3. These data are stored in the arrays of named common areas, FILMMM and XPRMNT, see sections 6.1.3 and 6.1.4, respectively.

Note: The examples presented in this document are for the expressed single purpose of communicating the utility of the software package and are *not* to be construed as an endorsement of the *best* models or numerical values that may be assigned to the optical properties of the media at particular optical frequencies.

4.2.1 Database Information

Regarding the database of point functions that are used in evaluating dielectric functions of constituent media, as mentioned earlier in sections 2.1, 2.2, and 4.1.2, all these files are assumed to reside in the same directory. Further, since the complete specification of a file is of the form:

disk:[directory]filename.extension;version

when no defaults are assumed, it is convenient to let the database files have similar filenames, i.e., W, but have distinct extensions to identify distinct constituent media. Consequently, the first line of data in X.DAT is a character string which allows one to complete the specification of these files. Since the extension refers to a specific medium, the first line in X.DAT may contain everything *but* the extension. Incidentally, the file specification is completed by appending a character string of the appropriate extension that is stored in the subroutine that calculates the dielectric function of the constituent medium, i.e., DIEL*xx*, where *xx* is a two-digit integer. An example of the format may be something like the following:

```
drb1:[data_bases]w.
```

where *'drb1'* refers to the name of the disk or mass-storage device, *'data_bases'* refers to the name of the directory, and *'w.'* refers to the filename of the database files, i.e., including the final period and no extension.

4.2.2 Layer Thicknesses

The next set of data involves listing the distinct thicknesses of the layers that form the layered structure of the samples. The unit measure of thickness is nanometers.

As mentioned earlier in section 3, the numerical values assigned to these parameters may be selected to undergo variation (vary) or remain fixed (frozen) during the course of the

analysis or calculations. An integer switch (froz/vary) is provided for each model parameter, i.e., refractive index or extinction coefficient. If the switch is set equal to 0, the numerical value is frozen. If the switch is set equal to 1 upon input, the numerical value may undergo variation. The switches remain unchanged during the course of calculation.

An uncertainty value is also required for each model parameter. The magnitude of this uncertainty may serve either of two purposes. If the numerical value of the parameter undergoes variation, then the uncertainty value sets the maximum stepsize allowed for changing the numerical value of the parameter between consecutive iterations of the calculation. If the numerical value of the parameter is selected to remain frozen, then the uncertainty value is used during the sensitivity analysis calculation to estimate the uncertainty values of other 'vary' model parameters. This is discussed further in section 3.2.

To demonstrate the necessary format, it is convenient to consider an example of a sample that involves two layers atop a substrate of silicon, where one layer is atop the other layer. Let the layer adjacent to the ambient region be a thermally grown oxide. Let the layer adjacent to the substrate region be a thin transition region that involves some mixture of thermally grown oxide and crystalline silicon. The layered structure may then be expressed by the ordered form:

(ambient / oxide / oxide + silicon / silicon substrate).

Suppose further that the top layer of oxide is 100 nm thick, that the transition region is 2 nm thick, and that a reasonable initial estimate of the uncertainty is subjectively chosen to be 2 nm. Lastly, let only the transition region thickness be subject to optimization. This construct would require four lines of input data. An example of the format would be as the following:

```
------------------------------------------------
2                                ! mfilmz ' thicknesses /(i,z,zu,ivary)
  1    100.0    2.0    0        !    i,z,zu,ivary '    top layer, SiO2
  2      2.0    2.0    1        !    i,z,zu,ivary ' bottom layer, SiO2+Si
```

Note that the tabulation would include distinct thicknesses from all samples; thus any ordering among the thicknesses is not necessarily important here. Again, this format does not mention any sample or any ordering of layers; thus these thicknesses are not yet associated with any particular sample. The ordering of layers for a sample is presented later in section 4.2.7.

Also, note the initial line of connected hyphens; it is included. The intent here is to ease the readability of the input file. For convenience, a line of connected hyphens will be assumed to precede each format presented in the remaining subsections of 4.2.

Since the program uses READ statements to fetch its data from files, it is convenient to use a notation that is a suggestive adaptation of the argument list of a READ statement in FORTRAN. This notation was presented previously in [3]. Following this same convention for specifying input formats, the input format for thicknesses may be expressed by the form:

$$m_z \ /(i,\ z_i,\ \delta z_i,\ v_i)$$

where:

m_z is the number of distinct thicknesses that contribute to the layered structure of the samples;

i is an integer that indexes the line entries consecutively in unit increments, i.e., $(i = 1, 2, 3, \ldots, m_z)$;

z_i is the thickness of a layer measured in nanometers;

δz_i is the uncertainty that is assigned to the numerical value of z_i; and

v_i is the integer (froz/vary) switch with value 0 or 1, respectively.

The forward slash mark '/' delimits the first line of input data. The parentheses bound items that ought to appear on each subsequent line of data until the implied DO-loop has been satisfied. Such notation has been convenient in conveying the complicated formats that are required for the input entries as well as the associated data structures. This notation is assumed throughout the remainder of the document unless inferred otherwise from context.

Incidentally, if $(m_z = 0)$, this is the only line of information that ought to be entered for this set of data, i.e., apart from the short line of connected hyphens.

4.2.3 Supplementary Parameters (Integer)

Regarding the ability of the program to incorporate complicated models, it is sometimes important to be able to pass parameters to subroutines that calculate the dielectric functions of the constituents and mixtures. Such parameters may serve as integer switches for decision processes or serve as floating-point variables that may be subjected to optimization. The manner in which these parameters are discerned and passed among the subroutines may be readily seen by considering the subroutine named DIEFCN, see section 6.4.1. Although the structure for passing, referring, and interpreting the supplementary parameters is rather limited, it does provide some measure of convenience for the user in adapting or incorporating his/her own models. The input format for integer supplementary parameters is of the form:

$$m_I \ /(i,\ p_i)$$

where:

m_I is the number of distinct integer supplementary parameters;

i is an integer that indexes the line entries of data consecutively in unit increments, i.e., $(i = 1, 2, 3, \ldots, m_I)$; and

p_i is the integer supplementary parameter.

Since integer parameters are not subjected to optimization here, no uncertainties are included regarding their numerical value. This tabulation includes all integer supplementary parameters that may be associated with any of the samples or mixtures.

Since the integer supplementary parameters are used so infrequently, it is often the case that $(m_I = 0)$. Then, an example of the format would be of the form:

```
-------------------------------------------------
0                              ! mipars / (i,ip)
```

4.2.4 Supplementary Parameters (Floating-Point)

As mentioned above, the program provides the capability to input floating-point supplementary parameters to aid in modeling the optical behavior of the constituents and/or

mixtures. As may be seen from eq (7), these parameters may be subjected to optimization. The input format is of the form:

$$m_R \ /(i,\ p_i,\ \delta p_i,\ v_i)$$

where:

m_R is the number of distinct floating-point supplementary parameters;

i is an integer that indexes the line entries of data consecutively in unit increments, i.e., $(i = 1, 2, 3, \ldots, m_R)$;

p_i is the floating-point supplementary parameter;

δp_i is the uncertainty that is assigned to the numerical value of p_i; and

v_i is the integer (froz/vary) switch with value of either 0 or 1, respectively.

Since the floating-point supplementary parameters are used so infrequently, it is often the case that $(m_R = 0)$. Then, an example of the format would be of the form:

```
------------------------------------------------
0                              ! mrpars / (i,rp,up,ivary)
```

4.2.5 Effective Media or Mixtures

For each effective medium that contributes toward the construction of the layered structure of the sample, as well as the ambient regions, one must specify the constituents and their volume fractions. If any constituent medium requires supplementary parameters, it is convenient to include this information as well. Further, it is convenient to group such information individually. Regarding the organization of these data for the effective media, the general format is given by the following form:

m_{media}

$$
\begin{array}{cccccc}
m_{f,1} & \cdots & & & & \\
\vdots & & & & & \\
m_{f,j-1} & \cdots & & & & \\
 & \vdots & & & & \\
m_{f,j} & m_{I,j} & m_{R,j} & & & \\
 & 1 & \tilde{\varepsilon}_1 & f_1 & \delta f_1 & v_1 \\
 & \vdots & \vdots & \vdots & \vdots & \vdots \\
 & i_f & \tilde{\varepsilon}_i & f_i & \delta f_i & v_i \\
 & \vdots & \vdots & \vdots & \vdots & \vdots \\
 & m_{f,j} & \tilde{\varepsilon}_{m_{f,j}} & f_{m_{f,j}} & \delta f_{m_{f,j}} & v_{m_{f,j}} \\
 & & & & & \\
 & 1 & \tilde{p}_{I,1} & & & \\
 & \vdots & \vdots & & & \\
 & i_I & \tilde{p}_{I,i} & & & \\
 & \vdots & \vdots & & & \\
 & m_{I,j} & \tilde{p}_{I,m_{I,j}} & & & \\
 & & & & & \\
 & 1 & \tilde{p}_{R,1} & & & \\
 & \vdots & \vdots & & & \\
 & i_R & \tilde{p}_{R,i} & & & \\
 & \vdots & \vdots & & & \\
 & m_{R,j} & \tilde{p}_{R,m_{R,j}} & & & \\
m_{f,j+1} & \cdots & & & & \\
\vdots & & & & & \\
m_{f,m_{\text{media}}} & \cdots & & & & \\
 & \vdots & & & &
\end{array}
$$

where

m_{media} is the number of distinct effective media. From sections 4.1.2 and 6.1.2, $(2 \leq m_{\text{media}} \leq \texttt{nsampl}*(\texttt{nfilms}+2))$.

j is an integer that indexes the distinct effective media, i.e., $(j = 1, 2, \ldots, m_{\text{media}})$.

$m_{f,j}$ is the number of distinct constituent media (or volume fractions) that are associated with the j^{th} effective medium. From section 4.1.2, $(1 \leq m_{f,j} \leq \texttt{nlmnts})$.

$m_{I,j}$ is the number of integer supplementary parameters that are associated with the j^{th} effective medium.

$m_{R,j}$ is the number of floating-point supplementary parameters that are associated with the j^{th} effective medium.

i_f is an integer that locally indexes the distinct constituent media (or volume fractions) of the j^{th} effective medium, i.e., $(i_f = 1, 2, \ldots, m_{f,j})$. For convenience and readability, the subscript j on i is suppressed here, as well as in the following set of entries.

$\tilde{\varepsilon}_i$ is an integer index label of the appropriate constituent medium that is associated with the $i_f{}^{\text{th}}$ constituent medium of the j^{th} effective medium. From section 4.1.2, $(1 \leq \tilde{\varepsilon}_i \leq \texttt{nlmnts})$.

f_i is the volume fraction of the $i_f{}^{\text{th}}$ constituent medium of the j^{th} effective medium.

δf_i is the uncertainty that is assigned to the numerical value of f_i.

v_i is the integer (froz/vary) switch for f_i with value 0 or 1, respectively.

i_I is an integer that locally indexes the integer supplementary parameters that are associated with the j^{th} effective medium, i.e., $(i_I = 1, 2, \ldots, m_{I,j})$.

$\tilde{p}_{I,i}$ is the integer index label of the appropriate integer supplementary parameter that is associated with the $i_I{}^{\text{th}}$ integer supplementary parameter of the j^{th} effective medium, i.e., $(1 \leq \tilde{p}_{I,i} \leq m_I)$, where m_I was presented earlier in section 4.2.3.

i_R is an integer that locally indexes the floating-point supplementary parameters that are associated with the j^{th} effective medium, i.e., $(i_R = 1, 2, \ldots, m_{R,j})$.

$\tilde{p}_{R,i}$ is the integer index label of the appropriate floating-point supplementary parameter that is associated with the $i_R{}^{\text{th}}$ floating-point supplementary parameter of the j^{th} effective medium, i.e., $(1 \leq \tilde{p}_{R,i} \leq m_R)$, where m_R was presented earlier in section 4.2.4.

Again, regarding the above entries that are capped with a tilde, it is important to note that these entries refer to index labels local to the appropriate subset of data, i.e., constituent media and supplementary parameters from sections 4.2.1, 4.2.3, and 4.2.4.

To demonstrate this format, consider again the example that was presented in section 4.2.2 regarding thicknesses. The layered structure involved the following ordered form:

(ambient / oxide / oxide + silicon / silicon substrate).

This structure involves four distinct effective media, i.e., one for each distinct spatial region. Letting the ambient be air, one need only consider three distinct constituent media, i.e., air, oxide, and silicon.

In section 6.5, one may locate the set of distinct constituent media that is of interest here, and then go on to identify the names of the subroutines which are used to calculate the dielectric functions, i.e., DIELxx, where xx refers to some two-digit number. Here, the index label that is associated with a distinct constituent medium is that integer which is derived from the last two characters of the subroutine name, i.e., xx. The constituent media index labels for air, silicon, and oxide are found to be 2, 3, and 5, respectively. Further, let the transition region be half oxide and half silicon. The format for this example could then be as the following:

```
-------------------------------------------------
4                                ! mixtures ~ number of effective media
1  0  0                          !    mlmnt,mipar,mrpar       #1
   1  2  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ~ air
1  0  0                          !    mlmnt,mipar,mrpar       #2
   1  3  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ~ Si
1  0  0                          !    mlmnt,mipar,mrpar       #3
   1  5  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ~ SiO2
2  0  0                          !    mlmnt,mipar,mrpar       #4
   1  3  0.5  0.02  1            !       j,lmnt,frac,ufrac,ivary  ~ Si
   2  5  0.5  0.02  1            !       j,lmnt,frac,ufrac,ivary  ~ SiO2
```

The uncertainties of the volume fractions of the transition region have been subjectively set at 0.02, and the volume fractions of the transition region have been selected to undergo optimization. Again, because the volume fractions must sum to one, the number of volume fractions that may be selected to undergo variation may be set equal to 0, 2, or more than 2. Although the line entries of the constituent media are not necessarily ordered, the constituent media must be distinct. This is check-tested; a violation stops the program. Regarding the comments that are appended onto each line of data, the word 'lmnt' is a mnemonic for *elements* or constituents. Lastly, since the format simply tabulates distinct effective media, ordering of the distinct effective media is not important here.

4.2.6 Ambients and External Parameters

The ambient region refers to that spatial region that lies external to the layered structure of the sample, i.e., the (layers/substrate) system. Sometimes, it is necessary to analyze ellipsometric measurements involving a number of distinct samples and ambients. In the Table of Contents, it may be seen that the database contains at least two distinct constituent media which are often used as ambients, i.e., vacuum and air.

Of course, the database may be enlarged to include nearly any number of lossless, isotropic, and homogeneous media. Enlarging the database is rather straightforward, because the user need only follow the conventions that are assumed by subroutine DIELMN, see section 6.4.4.

Because the analysis of a sample may involve more than one ambient, it is convenient to construct an indexed listing of effective media that are used as ambients. Further, in the event that a distinct (ambient/sample) system depends upon some independent or *external* parameter, e.g., an externally applied perturbation or stress field, this information may be included in the indexed listing as well. The indexed listing defines an ambient as being that composite entity which associates a distinct effective medium and two distinct sets of supplementary parameters that characterize the applied perturbation.

The indexed listing of the ambients simply provides another opportunity for assigning supplementary parameters to the effective or constituent media of the system configuration. The general format is given by the following form:

m_{ambients}

1	...		
⋮			
$(j-1)$	...		
	⋮		
j	$\tilde{\varepsilon}_j$	$m_{I,j}$	$m_{R,j}$
	1	$\tilde{p}_{I,1}$	
	⋮	⋮	
	i_I	$\tilde{p}_{I,i}$	
	⋮	⋮	
	$m_{I,j}$	$\tilde{p}_{I,m_{I,j}}$	
	1	$\tilde{p}_{R,1}$	
	⋮	⋮	
	i_R	$\tilde{p}_{R,i}$	
	⋮	⋮	
	$m_{R,j}$	$\tilde{p}_{R,m_{R,j}}$	
$(j+1)$	...		
⋮			
m_{ambients}	...		
	⋮		

where

m_{ambients} is the number of distinct ambients which involves associating distinct effective media with subsets of supplementary parameters for all configurations of the (ambient/sample) system. From sections 4.1.2 and 6.1.2, $(1 \leq m_{\text{ambients}} \leq \texttt{nbient})$.

j is an integer that indexes the distinct ambients, i.e., $(j = 1, 2, \ldots, m_{\text{ambients}})$.

$\tilde{\varepsilon}_j$ is an integer index label of the appropriate effective (*not* constituent) medium that is associated with the j^{th} ambient. From section 4.2.5, $(1 \leq \tilde{\varepsilon}_j \leq m_{\text{media}})$.

$m_{I,j}$ is the number of integer supplementary parameters that are associated with the j^{th} ambient.

$m_{R,j}$ is the number of floating-point supplementary parameters that are associated with the j^{th} ambient.

i_I is an integer that locally indexes the integer supplementary parameters that are associated with the j^{th} ambient, i.e., $(i_I = 1, 2, \ldots, m_{I,j})$. For convenience and readability, the subscript j on i is suppressed here, as well as in the following set of entries.

$\tilde{p}_{I,i}$ is the integer index label of the appropriate integer supplementary parameter that is associated with the ${i_I}^{\text{th}}$ integer supplementary parameter of the j^{th} ambient, i.e., $(1 \leq \tilde{p}_{I,i} \leq m_I)$, where m_I was presented earlier in section 4.2.3.

i_R is an integer that locally indexes the floating-point supplementary parameters that are associated with the j^{th} ambient, i.e., $(i_R = 1, 2, \ldots, m_{R,j})$.

$\tilde{p}_{R,i}$ is the integer index label of the appropriate floating-point supplementary parameter that is associated with the ${i_R}^{\text{th}}$ floating-point supplementary parameter of the j^{th} ambient, i.e., $(1 \leq \tilde{p}_{R,i} \leq m_R)$, where m_R was presented earlier in section 4.2.4.

Again, regarding the above entries that are capped with a tilde, it is important to note that these entries refer to index labels local to the appropriate subset of data, i.e., supplementary parameters and effective media presented in sections 4.2.3, 4.2.4, and 4.2.5.

To demonstrate this format, it is convenient to build upon the development of the previous example. The format for the ambient configurations could then be as the following:

```
------------------------------------------------
1                          ! mbient ' number of ambients
   1  1  0  0              !    j,imix,mipar,mrpar          ' air
```

because no supplementary parameters have been used thus far; the worked examples in sections 4.2.3 and 4.2.4 assumed that $(m_I = m_R = 0)$.

4.2.7 Sample Characterization or Construction

Now that the basic model parameters have been presented, it is possible to discuss the method of constructing a sample by characterizing each region of its layered structure in terms of the basic parameters presented heretofore. From section 4.1.2, the program is able to analyze a number of distinct samples collectively. Here, the samples become indexed according to the order in which they are entered for characterization in the input file. Furthermore, this ordering of samples will also govern how the measurement data are to be entered in the file; such is discussed later in the next subsection.

First, it is important to discuss the meaning of the word *sample*. A sample is defined as being that material structure of the (layers/substrate) system that is subjected to ellipsometric measurement. The business of characterizing a sample refers to that procedure of index-labeling which associates the distinct spatial regions of the (layers/substrate) system with the appropriate sets of model parameters, so that the direct or forward problem may be completely defined, i.e., apart from specifying the ambient. Since measurements on a sample may involve more than one ambient, the business of associating samples and ambients is deferred until the next subsection. Here, the samples are constructed or assembled in a layer-by-layer fashion. For each spatial region, one must specify an effective medium, and if the region is a layer, one must also specify a thickness.

Recall, the inverse problem is formulated as a least-squares problem. This involves a series of linear algebra problems. Consequently, the matrices associated with individual samples may be combined so that several samples may be analyzed together. Here, the program is said to have a multiple-sample capability. The maximum number of samples that may be analyzed together is set by the parameter `nsampl`, as mentioned in section 4.1.2.

Each previous subsection presented a format for inputting parameters. Each model parameter has a distinct index label; each line entry following the first line is usually indexed. Here, those index labels are used for specifying the configuration of the (layers/substrate) system. The sample is characterized by an ordered set of integers that point into those arrays which store the numerical values of the parameters.

The general format for specifying the layered structure of the samples is of the following form:

m_{samples}

$m_{z,1}$
$\vdots$
$m_{z,j}$

1	$\tilde{\varepsilon}_1$	$\tilde{z}_1$
$\vdots$	$\vdots$	$\vdots$
i	$\tilde{\varepsilon}_i$	$\tilde{z}_i$
$\vdots$	$\vdots$	$\vdots$
$m_{z,j}$	$\tilde{\varepsilon}_{m_{z,j}}$	$\tilde{z}_{m_{z,j}}$
$(m_{z,j}+1)$	$\tilde{\varepsilon}_{\text{substrate},j}$	

$m_{z,(j+1)}$
$\vdots$
$m_{z,m_{\text{samples}}}$

$\vdots$

where

m_{samples} is the number of distinct material samples that are subjected to ellipsometric measurement. From section 4.1.2, $(1 \leq m_{\text{samples}} \leq \texttt{nsampl})$.

j is an integer that indexes the distinct samples, i.e., $(j = 1, 2, \ldots, m_{\text{samples}})$.

$m_{z,j}$ is the number of layers that lie atop the substrate of the j^{th} sample. From section 4.1.2, $(0 \leq m_{z,j} \leq \texttt{nfilms})$.

i is an integer that locally indexes the distinct layers of the j^{th} sample, i.e., $(i = 1, 2, \ldots, m_{z,j})$. The layer adjacent to the ambient is indexed 1; the layer adjacent to the substrate is indexed $m_{z,j}$. For convenience and readability, the subscript j on i is suppressed here, as well as in the following set of entries.

$\tilde{\varepsilon}_i$ is the integer index label of the appropriate effective medium that is associated with the i^{th} layer of the j^{th} sample. From section 4.2.5, $(1 \leq \tilde{\varepsilon}_i \leq m_{\text{media}})$.

$\tilde{z}_i$ is the integer index label of the appropriate thickness that is associated with the i^{th} layer of the j^{th} sample. From section 4.2.2, $(1 \leq \tilde{z}_i \leq m_z)$.

Again, regarding the above entries that are capped with a tilde, it is important to note that these entries refer to index labels local to the appropriate subset of data, i.e., effective media and thicknesses presented in sections 4.2.5 and 4.2.2, respectively.

To demonstrate this format, it is convenient to build upon the development of the previous example. Again, the sample involved the form: (oxide / oxide + silicon / silicon substrate). The format for this sample configuration would then be as the following:

```
-------------------------------------------------
1                              ! msampl   ~ number of samples analyzed
2                              !    mfilm ~ number of layers on sample #1
   1  3  1                     !       j,imix,iz    ~ SiO2   , top layer
   2  4  2                     !       j,imix,iz    ~ SiO2+Si, transition region
   3  2                        !       j,imix       ~ Si     , substrate
```

No mention is made regarding the ambient; this is discussed in the next subsection.

4.2.8 Measurement Data (Δ, ψ)

The measurement data of ellipsometric angles are organized in the same fashion as was presented in the previous subsection, i.e., sample by sample. Regarding measurements, both the sample and ambient must be specified. To characterize the (ambient/sample) system, a similar scheme of index labels is assumed again.

For convenience, the integers are ordered in a format compatible with the collection algorithm of the laboratory instrumentation. The ellipsometric measurement data are ordered into a data structure that follows the FORTRAN indexing convention for multiply-indexed arrays. That data structure is of the following form:

(*angles & wavelengths, repeats, ambients, samples*)

where:

samples indexes the set of distinct samples that were subjected to ellipsometric measurement;

ambients indexes the set of distinct ambients used during measurement involving a given sample;

repeats indexes the sets of distinct repeats of multiple-angle of incidence and multiple-optical frequency measurements performed on a system involving a given ambient and sample; and

angles & wavelengths indexes the set of distinct angles of incidence and optical frequencies (or wavelengths) used during measurement on a system of given repeat index, ambient, and sample.

This form suggests that the measurement data (Δ, ψ) are grouped and ordered according to the following structure. The measurement data are grouped sample by sample. For each sample, they are grouped ambient by ambient. For each ambient, they are grouped by their repeat index label. For each repeat index, they are indexed by the source variables, i.e., the angle of incidence and the optical frequency (or wavelength in vacuum or corresponding photon energy) of the light. Ordering among the incident angles or among the optical frequencies is not important, i.e., apart from that associated with the input format. Hence, the forward problem becomes completely defined. The *repeats* simply partition the measurement data, e.g., distinguishing measurement data collected on different days.

To implement the above organization and thereby characterize the (ambient/sample) system, the program reads a tabulated set of integers line by line. Since the program assumes that data are entered sample by sample, one need only consider the data for one sample, e.g., sample s. The total set of input data of all samples is constructed simply by concatenating the individual data sets of each sample. Now, consider the data that are associated with sample s. The first line of data is the number of ambients, m_{as}.

Let a_s index the set of distinct ambients on sample s. Let $\tilde{a}_s$ be the index label of the appropriate ambient that is associated with the $a_s{}^{\text{th}}$ ambient on sample s. Let m_{ras} be the number of sets of repeat measurements of (Δ, ψ) involving the $a_s{}^{\text{th}}$ ambient on sample s. Let r_{as} index these sets of repeat measurements. Heading the set of data associated with the $a_s{}^{\text{th}}$ ambient would be a line that contains two integers, m_{ras} and $\tilde{a}_s$.

The ordering infers that r_{as} is sequenced through its range *before* the indexing of a_s is stepped; i.e., repeated sets of measurements on a sample are collected together before one considers changing the ambient.

Since the (*repeat, ambient, sample*) structure is an ordered form, the structure is completely specified by the aforementioned indices. Next, consider the data of the $r_{as}{}^{\text{th}}$ repeat set. Heading this set of data would be a line that contains one integer, the number of measurements of (Δ, ψ). It is denoted by $m_{\lambda\phi,ras}$. For convenience, it is not necessary to include the indexing label r_{as} on the same line.

This is followed by the measurement data of ellipsometric angles. They are entered into the data file line by line. Each set of measurements involves two lines of data, one being for the source variables (λ, ϕ) and measured ellipsometric angles (Δ, ψ), and the other being for the associated uncertainties. The format is of the following form:

$$m_{\lambda\phi,ras} \,/(i,\ \lambda_i,\ \phi_i,\ \Delta_i,\ \phi_i\ /\ \delta\lambda_i,\ \delta\phi_i,\ \delta\Delta_i,\ \delta\phi_i)_{ras}$$

where:

$m_{\lambda\phi,ras}$ is the total number of measurements of ellipsometric angles (Δ, ψ) involving multiple angles of incidence and multiple optical frequencies that involve the $r_{as}{}^{\text{th}}$ repeat set of measurements, i.e., $(r_{as} = 1, 2, \ldots, m_{ras})$, and the $a_s{}^{\text{th}}$ ambient of sample s, i.e., $(a_s = 1, 2, \ldots, m_{as})$. From sections 4.1.2, 6.1.2, and 6.2.3, $(1 \leq m_{\lambda\phi,ras} \leq \texttt{nwaves*nanglx} \leq \texttt{nexpts})$.

i is an integer that locally indexes the multiple-angle of incidence and multiple-optical frequency measurement data consecutively with unit increments, i.e., $(i = 1, 2, \ldots, m_{\lambda\phi,ras})$.

λ_i is a floating-point source parameter that may be of either sign depending upon how one chooses to characterize the optical frequency of light. When the value is negative, the unit of measure is nanometers. When the value is positive, the unit of measure is electron-volts. The program check-tests for either $(-1240 \leq \lambda_i \leq -200)$ or $(1.0 \leq \lambda_i \leq 6.0)$.

$\delta\lambda_i$ is the uncertainty that is assigned to the numerical value of λ_i.

ϕ_i is the angle of incidence measured in degrees, where a value of zero relates to that of normal incidence, i.e., $(0 \leq \phi_i \leq 90)$.

$\delta\phi_i$ is the uncertainty that is assigned to the numerical value of ϕ_i.

Δ_i is an ellipsometric angle measured in degrees, i.e., $(0 \leq \Delta_i < 360)$, assuming the Nebraska convention, i.e., $R_{p,H}$ from section 2.3 of [3].

$\delta\Delta_i$ is the uncertainty that is assigned to the numerical value of Δ_i.

ψ_i is an ellipsometric angle measured in degrees, i.e., $(0 \leq \psi_i \leq 90)$.

$\delta\psi_i$ is the uncertainty that is assigned to the numerical value of ψ_i.

Within any set of $m_{\lambda\phi,ras}$ measurement data, ordering of optical frequencies or angles of incidence is not important. Thus, regarding sample s, the general format is given by the following form:

$$
\begin{array}{cccccc}
m_{as} & & & & & \\
\vdots & & & & & \\
m_{ras} & \tilde{a}_s & & & & \\
 & \vdots & & & & \\
 & m_{\lambda\phi,ras} & & & & \\
 & & \vdots & & & \\
 & & i & \lambda_i & \phi_i & \Delta_i & \psi_i \\
 & & & \delta\lambda_i & \delta\phi_i & \delta\Delta_i & \delta\psi_i \\
 & & \vdots & & & &
\end{array}
$$

Again, to construct the final set of data for all of the samples, one simply concatenates the data for each individual sample, i.e., without any intervening lines of demarcation.

To demonstrate this format, let the measurements be those found by solving the forward problem for the (ambient/sample) configuration that was developed in the previous examples. Hence, these measurements would be *exact*. Let the measurements involve a grid of optical frequencies, e.g., where the associated photon energies are between 1.5 to 6.0 eV in steps of 0.5 eV, for one angle of incidence, e.g., 70 deg. Let $\delta\lambda$, $\delta\phi$, $\delta\Delta$, and $\delta\psi$ be given by 0.1, 0.01, 0.01, and 0.01 deg, respectively. Although these uncertainties depend on the instrumentation, the values are chosen subjectively for convenience. The format could then be as the following:

```
-------------------------------------------------
1                          ! mbien ' number of ambients on sample #1
1  1                       !    mrpeat,imbien
  10                       !       mexpt ' number of measurements
   1    1.50000E+00  7.00000E+01  8.04309E+01  3.08875E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   2    2.00000E+00  7.00000E+01  7.97129E+01  4.31275E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   3    2.50000E+00  7.00000E+01  1.02227E+02  6.93018E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   4    3.00000E+00  7.00000E+01  2.50415E+02  6.09989E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   5    3.50000E+00  7.00000E+01  2.62427E+02  4.00376E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   6    4.00000E+00  7.00C00E+01  2.47313E+02  3.31054E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   7    4.50000E+00  7.00000E+01  1.92800E+02  3.34557E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   8    5.00000E+00  7.00000E+01  1.24499E+02  3.21867E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   9    5.50000E+00  7.00000E+01  8.77637E+01  3.66717E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
  10    6.00000E+00  7.00000E+01  7.10687E+01  4.18238E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
```

Incidentally, these data are retained in arrays that are located in the named common area, **exprmt**, see section 6.1.4.

4.2.9 Combining

In general, the input data are composed of two kinds of information, those which define the system model and associated measurements, and those which define the options for processing these data. Thus far, only the first kind has been presented, and they are contained in the input file X.DAT. The input data file is constructed by simply combining the formats in the order that they were presented. In general, no intervening blank lines are allowed. Again, those lines of connected hyphens are merely conveniences; the intent is that of easing the readability of the input file.

Combining the examples of the formats presented thus far, and indexing the lines for convenience, i.e., the input data file would *not* actually contain such indexing, the input data file X.DAT would become:

```
drb1:[data_bases]w.
-------------------------------------------------
2                                ! mfilmz ' thicknesses /(i,z,zu,ivary)
  1    100.0    2.0    0        !    i,z,zu,ivary '    top layer, SiO2
  2      2.0    2.0    1        !    i,z,zu,ivary ' bottom layer, SiO2+Si
-------------------------------------------------
0                                ! mipars / (i,ip)
-------------------------------------------------
0                                ! mrpars / (i,rp,up,ivary)
-------------------------------------------------
4                                ! mixtures ' number of effective media
1  0  0                          !    mlmnt,mipar,mrpar       #1
   1  2  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ' air
1  0  0                          !    mlmnt,mipar,mrpar       #2
   1  3  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ' Si
1  0  0                          !    mlmnt,mipar,mrpar       #3
   1  5  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ' SiO2
2  0  0                          !    mlmnt,mipar,mrpar       #4
   1  3  0.5  0.02  1            !       j,lmnt,frac,ufrac,ivary  ' Si
   2  5  0.5  0.02  1            !       j,lmnt,frac,ufrac,ivary  ' SiO2
-------------------------------------------------
1                                ! mbient ' number of ambients
   1  1  0  0                    !    j,imix,mipar,mrpar          ' air
-------------------------------------------------
1                                ! msampl   ' number of samples analyzed
2                                !    mfilm ' number of layers on sample #1
   1  3  1                       !       j,imix,iz     ' SiO2   , top layer
   2  4  2                       !       j,imix,iz     ' SiO2+Si, transition region
   3  2                          !       j,imix        ' Si     , substrate
-------------------------------------------------
1                                ! mbien ' number of ambients on sample #1
1  1                             !    mrpeat,imbien
  10                             !       mexpt ' number of measurements
   1    1.50000E+00  7.00000E+01  8.04309E+01  3.08875E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   2    2.00000E+00  7.00000E+01  7.97129E+01  4.31275E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   3    2.50000E+00  7.00000E+01  1.02227E+02  6.93018E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   4    3.00000E+00  7.00000E+01  2.50415E+02  6.09989E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   5    3.50000E+00  7.00000E+01  2.62427E+02  4.00376E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   6    4.00000E+00  7.00000E+01  2.47313E+02  3.31054E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   7    4.50000E+00  7.00000E+01  1.92800E+02  3.34557E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   8    5.00000E+00  7.00000E+01  1.24499E+02  3.21867E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   9    5.50000E+00  7.00000E+01  8.77637E+01  3.66717E+01  (i,E,a,  d,p)
        1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
  10    6.00000E+00  7.00000E+01  7.10687E+01  4.18238E+01  (i,E,a,  d,p)
```

```
53          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
```

The example shows that three model parameters are selected for optimization; see lines 5, 19, and 20. These model parameters are associated with the transition region; see line 28. Hence, their numerical values will undergo variation if so requested from the command options in file X.INN. The command options are discussed in the next section.

Further, regarding model parameters selected for optimization, their uncertainty values were set to nonzero values. This condition is check-tested upon input; if violated, the user would be so notified of the zero values. Setting the uncertainty value to a nonzero value circumvents notification. The reason for the check-test is that the program uses the uncertainty value to limit the stepsize for updating the numerical value of the associated model parameter. With an allowed maximum stepsize of zero, the program would be unable to reduce the numerical value of the error expression, at least with respect to this model parameter. Being unable to justify further calculation, it would terminate. Such control is useful in other situations as well, e.g., limiting the calculation to a single evaluation of the residual or discerning relative contributions made to the uncertainty value of some 'vary' model parameter. This may be accomplished by assigning a sufficiently small number to the uncertainty value on input.

4.3 Command Options

As mentioned previously in section 4.1.1, the input data file X.INN contains the command options that are necessary for directing control of the software package. To direct the execution of the program, a menu-driven decision tree of command options is made available to the user. Incidentally, the program reads file X.DAT *before* reading file X.INN. Examples of X.INN files are shown in section 5.

The first level of the tree involves a menu of three options. One option is selected per execution of the program; the selection of that option is the first line of data in X.INN. After reading this line of input data, the program branches to the appropriate subroutine. After leaving this subroutine at level one, the main program stops. Hence, the main program does not loop back on itself to a condition where a request is made of the user to select another option at level one. This branching at level one is performed by the MAIN program, see section 6.2.1.

Before reading the first line of data in file X.INN, the MAIN program writes to the output file X.OUT the following menu of available options (at level one):

```
Enter:   option
           1,  forward problems, plots, ...
           2,  search      (vary)
           3,  search grid (vary)
```

Where the options at level one include:

1, requests the program to perform one of several simple tasks, such as providing a set of tabulated output that is amenable for plotting or initiating a series of calculations of the direct or forward problem. No iterative calculations are considered, i.e., no minimizations or inversions of the ellipsometric equations. This is discussed further in section 4.3.1.

2, requests the program to invert the ellipsometric equations by performing a series of unconstrained optimization calculations. The variation of numerical values of selected model parameters is generally unbounded. This is presented in section 4.3.2.

3, requests the program to invert the ellipsometric equations by performing a grid scan over a selected set of model parameters. A series of constrained optimization calculations is initiated at each point of the grid of model parameters. This is discussed further in section 4.3.3.

These options provide the current range of utility of the program. Regarding the general nature of the options, it may be expected that option one is more useful towards the beginning of analysis with plots, while options two and three form the primary tools during analysis. Since the grid scans are constructed from a simple nesting of DO-loops, there may be redundant calculations over parts of the parameter space whenever the 'vary' model parameters are not common to every sample. These options are discussed further in the following subsections. Here, it is convenient to simply follow the decision tree of options.

4.3.1 Forward Problems, Plots, ...

Following the selection of option '1' at level one, the program calls subroutine SCATOS, thus entering level two, see section 6.2.4. This subroutine presents the user with another

set of options. The options specify the sets of source variables (λ, ϕ) that may be used in the model calculations. The menu of options at level two is:

```
Enter:   choice of incident (energies, angles)
           1,   measurement data,     x.dat
           2,   grid scan.
```

where the options involve:

1, requests the program to use as source variables only those sets of ordered pairs (λ, ϕ) that are specified in the input file X.DAT for the measurement data of (Δ, ψ).

2, requests the program to use as source variables only those sets of ordered pairs (λ, ϕ) that lie on a two-dimensional grid that is specified later in a following prompt-request. The program prompts or requests information regarding the lower bound, the upper bound, and the stepsize for each dimension of the grid.

The above two options form a branch in the decision tree of options. The next step in the program involves specifying which field quantities are to be calculated and then written to the output file X.PLOT. Thus, another set of options is presented to the user, i.e., level three, but here the menu will depend upon which option is selected at level two. Hence, there are two alternative sets of options at level three. They are presented in succession.

After selecting option '1' at level two, the menu of options at level three is:

```
Enter:   choice of output suitable for:
           1,   input data,     x.dat,
           2,   plotting (Delta, psi),
           3,   plotting (Delta, psi) deviations,
                 deviation = measurement - model.
           4,   plotting |g| ^ rms deviation,  unscaled,
                 on a 1D or 2D grid of model parameters.
```

where

1, requests the program to calculate (Δ, ψ) using (λ, ϕ) as supplied by the file X.DAT. The output is written to the plot file X.PLOT. The format is suitable for use in file X.DAT.

2, requests the program to calculate (Δ, ψ) using (λ, ϕ) as supplied by the file X.DAT.

The output is written to the plot file X.PLOT. The format is suitable for later graphics.

3, requests the program to calculate and tabulate the deviations between the measurement data and that calculated by the model. The deviations are ordered similar to that of the measurement data. The output is written to the plot file X.PLOT.

4, requests the program to initiate a grid scan of model parameters and evaluate the error expression |g| for each point on the grid. For convenience, the grid may be either one or two dimensional; i.e., only one or two model parameters may undergo variation. It is then necessary to specify the domain of the grid. The format for specifying the grid of model parameters is presented later in section 4.3.3. But unlike that presented in section 4.3.3, there is no option regarding optimization here. The output is written to file X.PLOT.

After selecting option '2' at level two, the menu of options at level three is:

```
Enter:   choice of output suitable for:
           1,   dielectric function,   media,
           2,   (Delta, psi),          x.dat,
           3,   (Delta, psi),       plotting,
           4,   d/db,             b'(z,f,p),
           5,   d/da,     angle of incidence,
           6,   d/dE,                 energy.
```

where

1, requests the program to calculate the dielectric function for an effective medium that is specified in a later prompt-request from the program. The format is suitable for later graphics.

2, requests the program to calculate (Δ, ψ) for a grid of (λ, ϕ). The format is suitable for use in file X.DAT.

3, requests the program to calculate (Δ, ψ) for a grid of (λ, ϕ). The format is suitable for later graphics.

4, requests the program to calculate partial derivatives of (Δ, ψ) with respect to one of the model parameters, i.e., thickness, volume fraction, or supplementary parameter. Since volume fractions involve a linear constraint, the partial derivative with

respect to $\hat{f}_j$ is calculated according to eq (10), where $(k < j)$. Again, if the selection involves volume fractions, two volume fractions must be selected to undergo variation, that which is associated with $\hat{f}_j$, as given by eq (10), is written to the file X.PLOT. The model parameter is selected by setting the (froz/vary) switch to '1' in X.DAT. The unit of measure is degree per unit of the 'vary' model parameter.

5, request the program to calculate partial derivatives of (Δ, ψ) with respect to the angle of incidence, ϕ. The measure is unitless, i.e., degree per degree.

6, requests the program to calculate partial derivatives of (Δ, ψ) with respect to the photon energy of light. The unit of measure is degree per electron-volt.

Two things are yet unspecified; both are associated with the last set of options mentioned immediately above. These are presented in succession.

The first is associated with option '1.' One must select or specify one of the effective media for evaluation on the grid of photon energies. Accordingly, the program prompts the user with the following:

```
Enter:   kmix " effective medium of dielectric function
```

Here, one enters an integer for the index label of the appropriate effective medium, i.e, $(1 \leq \texttt{kmix} \leq m_{\text{media}})$ as discussed in section 4.2.5.

The second thing yet unspecified is the grid of source variables, the classical optical frequencies (or the wavelengths in vacuum or associated photon energies) and the angles of incidence, i.e., (λ, ϕ). Since the grid is constructed from a set of nested DO-loops, one specifies the grid by specifying three numerical values, i.e., the lower bound, the upper bound, and the stepsize, for each axis or dimension of the grid. The program prompts the user for each axis individually. These input values are check-tested, as appropriate. See section 6.2.5, subroutine SCATOI.

The first prompt-request is the following:

```
Enter:    range of incident energies (eV)
   or:                  - wavelengths (nm)
Enter:    ev1, ev2, ev3
```

where *ev1* is the lower bound, *ev2* is the upper bound, and *ev3* is the stepsize. Negative values are associated with wavelength in units of nanometers, and positive values are associated with energy in units of electron-volts.

Following an answer to the above prompt, the program issues the second prompt-request:

```
Enter:    range of incident angles (degrees)
Enter:    angle1, angle2, angle3
```

where *angle1* is the lower bound, *angle2* is the upper bound, and *angle3* is the stepsize.

Incidentally, the field quantities are written to the output file X.PLOT from either of two subroutines, i.e., SCATO1 or SCATO2, depending upon the set of ordered pairs (λ, ϕ) that is selected at level two, i.e., input from measurements in X.DAT or some discrete grid of values. Regarding these two subroutines, their use of suggestive names for variables, as well as from the lines of comments that are enclosed within the various subroutines, it ought to be rather straightforward for the user to add his own selection of field quantities to the program. This completes the list of options available to the user regarding calculations of the direct or forward problem and graphics.

4.3.2 Search (vary)

Following the selection of option '2' at level one, the main program calls the subroutine SEEKO1 to invert the ellipsometric equations, thus entering level two, see section 6.2.10. This subroutine initiates and maintains the search for a minimum to the *error* expression as an unconstrained optimization problem, as discussed in section 3.1. The (froz/vary) switch specifies the model parameters which have numerical values that are undergoing variation. At least one numerical value must undergo variation per measurement of (Δ, ψ), so that the associated rows in the Jacobian are not zero. Note that an iterative method requires an initial solution. Here, the initial value solution is given by the initial input of model parameters as presented in section 4.2. The problem is now completely defined; no further specifications are necessary; no further menus need to be issued to the user.

Since the optimization algorithm is unconstrained, unphysical fixed-point solutions are possible and likely during analyses. These pseudo solutions were discussed earlier in section 3.

The following presents a brief orientation regarding the internal organization of the subroutine SEEKO1. The organization is that for setting up an iterative loop. It calls subroutine ASMBL to construct the Jacobian matrix. It calls subroutine CGNL to solve for the Newton step. It updates the numerical value of the selected model parameters and tests the rate of reduction of the *error* expression. The progress regarding the rate of reduction is written to the output file, X.OUT. If the rate of reduction is sufficiently small to merit no further expenditure, it returns; otherwise, it continues iterating. Such calculations usually involve short durations of time, thus breakpointing is not necessary, and so it was not incorporated into the routine.

Upon completion of the above exercises, the program reports its *best* fixed-point solution. The deviations between the measurement data and that of the model are written to the plot file, X.PLOT, using subroutine SCATO1.

Statistics regarding the deviations are provided by subroutine STAT22. It reports the statistical means, standard deviations, and the correlation coefficient of the deviations, i.e.,

$$g_{\Delta,i} = \Delta_i^{\mathrm{e}} - \Delta_i^{\mathrm{m}} \qquad \text{and} \qquad g_{\psi,i} = \psi_i^{\mathrm{e}} - \psi_i^{\mathrm{m}},$$

which is distinct from eqs (12) and (13) from section 3.1. Here, the deviations involve no scaling by the measurement uncertainties. The estimated mean for Δ is defined by

$$\langle g_\Delta \rangle \equiv \frac{1}{M} \sum_{i=1}^{M} g_{\Delta,i},$$

the estimated variance is defined by

$$\langle (g_\Delta - \langle g_\Delta \rangle)^2 \rangle \equiv \frac{1}{M} \sum_{i=1}^{M} (g_{\Delta,i} - \langle g_\Delta \rangle)^2,$$

where the square root estimates the standard deviation, the covariance is defined by

$$\langle (g_\Delta - \langle g_\Delta \rangle)\,(g_\psi - \langle g_\psi \rangle) \rangle \equiv \frac{1}{M} \sum_{i=1}^{M} (g_{\Delta,i} - \langle g_\Delta \rangle)\,(g_{\psi,i} - \langle g_\psi \rangle),$$

and the correlation coefficient is defined by the ratio formed by the covariance divided by the product of standard deviations of $g_{\Delta,i}$ and $g_{\psi,i}$.

In regards to providing an analysis of sensitivities, here too, the Jacobian involves no row scaling by the measurement uncertainties. Similarly, a scaling matrix **S** is then defined. A

correlation matrix, defined by $\left[\tilde{\mathbf{J}}^{\mathrm{T}}\tilde{\mathbf{J}}\right]$, is calculated by subroutine CORLAT. Being a symmetric matrix, only the upper triangle is reported. The output file reports the condition number of this matrix. The diagonal elements of the scaling matrix are reported as well. Such reports help identify which model parameters may be correlated, and aid the decision process regarding the selection of frozen model parameters for a series of calculations.

Lastly, the output file reports the estimated uncertainties of the 'vary' model parameters as calculated from eq (22) in section 3.2. Following these exercises, the program stops.

4.3.3 Search Grid (vary)

After the selection of option '3' at level one, the main program calls subroutine SCANO2 to invert the ellipsometric equations, thus entering level two, see section 6.2.9. This subroutine seeks to find a minimum to the *error* expression, as a constrained optimization problem, by initiating a scan over a grid of numerical values associated with a selected set of model parameters. Here, the (froz/vary) switch specifies the selection; the model parameters contributing to the grid are those whose numerical values are selected to undergo variation. Since the grid is constructed from a single block of nested DO-loops, each selected model parameter contributes one independent axis or dimension to the grid, i.e., a hyper-cube, apart from the special consideration that is necessary for the volume fractions. Note that efficient use of the grid occurs only when the selected model parameters are common to all samples.

Each independent axis of the grid involves four items: the local index label for the model parameter, the lower bound, the upper bound, and the stepsize. To provide this information to the program, it is convenient to follow the convention of DO-loop specification and require that the input format be of the form:

$$i_p,\ p_1,\ p_2,\ p_3,$$

where i_p refers to the local index label of the model parameter p, and p_j refers to the numerical value of the model parameter associated with the initial value, final value, and stepsize, respectively.

To enter the DO-loop specification parameters, the parameters are entered in the same order that they occur in the file X.DAT. From section 4.2, the parameters are grouped and ordered as follows: thicknesses, supplementary parameters, and then volume fractions.

Within each group, the model parameters are indexed locally. By adhering to this group ordering and using only local indices, one is able to specify the necessary *vary* model parameters. Here, i_p refers to the local index label as used in file X.DAT.

Note that while the effective media are indexed as ordered in file X.DAT, the volume fractions are indexed locally from *within* the effective medium, i.e., starting with one. Hence, $(i_p = 2)$ could refer to the second volume fraction of any effective medium. This is check-tested upon input for consistency.

Regarding volume fractions and eq (10) from section 2.3, it is assumed that k refers to the smallest local index label among those selected to undergo variation for that particular effective medium. DO-loop specification parameters are *not* allowed for f_k, but are required for those f_j selected for optimization, i.e., $(k < j)$.

In regard to these considerations, the program issues forth the following lines:

```
Scan a grid of model parameters:  (z,p,f).
Grid info:   DO-loop parameters
Grid info:    i,        initial,      final,     increment
```

For each point on the multidimensional grid of model parameters, the program is designed to provide either one of two things. It may simply evaluate the residual at each grid point, and then compare values of the residual over the entire grid. Here, the *best* solution yields the smallest residual. The Jacobian is not calculated. Although small stepsizes are often requested, this may be a relatively fast calculation.

Alternatively, the calculation may initiate a series of unconstrained optimization problems using the Jacobian, except that the range of numerical values of the selected model parameters is restricted to remain within the bounds of the grid. At the conclusion of the grid scan, the program reports its *best* fixed-point solution. The program also reports any components of the solution that lie near the boundary of the selected grid. Solutions with components at grid boundaries are, of course, grid dependent, and as such, further calculation is usually necessary, i.e., after the grid is moved or redefined. Again, the calculations involve the Jacobian. Although large stepsizes are often requested, this may be a relatively slow calculation.

In regard to these considerations, the program presents the following menu of options:

```
Enter:   option  regarding the grid scan:
           0,      no optimization, |g| only,
           1,    full optimization, Jacobian.
```

The format convention for indicating the type of optimization is of the form:

$$i_o$$

where i_o is an integer with value 0 or 1, accordingly.

Since the grid specification includes the stepsize, the calculations may become very time-consuming. For this reason, the program writes breakpoint information to the output file X.SOUT at intervals of 15 cpu-minutes. This is discussed in section 4.1.1. The procedure for restarting a previously interrupted calculation is rather straightforward. One need merely append the contents of X.SOUT onto the end of the input data file X.INN, *without* any intervening blank lines. Upon starting any grid scan calculation, the program will attempt to read a set of breakpoint information. If restarting is not intended by the user, the input data file ought to be absent of excess lines. The program does initiate some measure of check-testing regarding the breakpoint information. Anything deemed irregular in the input file should inhibit the chances of erroneous restarts, but it is good practice to truncate the input data file with either an end-of-file condition or some other delimiter, e.g., a short line of connected equal signs.

Again, as presented in the previous subsection, i.e., section 4.3.2, the program reports a few basic statistics regarding deviations of the fit between the measurement data and that of the model, as well as the estimated uncertainties in the model parameters.

5. Worked Examples

The following subsections present worked examples using the program. Each subsection presents an example involving one selection from among the set of available top level options. A brief orientation is given regarding the purpose of each calculation. Since the program outputs a journal listing of the input data, only the output file needs to be presented in the following subsections. Again, an example of the input file X.DAT may be found in section 4.2.9. For convenience, the input file X.INN is included as well. The input/output files are shown as indexed for convenience.

5.1 Forward Problems, Plots, ...

This option is discussed in sections 4.3 and 4.3.1. Consider again the example that is presented in section 4.2.9. The input file X.DAT contains both the characterization of the sample and the measurement data of ellipsometric angles. For the sake of demonstration, suppose that only the characterization of the sample is known. Let the exercise here be that of generating a set of measurement data. A suitable input file X.DAT may be as follows:

```
 1  drb1:[data_bases]w.
 2  ------------------------------------------------
 3  2                              ! mfilmz ' thicknesses /(i,z,zu,ivary)
 4    1    100.0    2.0    0      !    i,z,zu,ivary '    top layer, SiO2
 5    2      2.0    2.0    0      !    i,z,zu,ivary ' bottom layer, SiO2+Si
 6  ------------------------------------------------
 7  0                              ! mipars / (i,ip)
 8  ------------------------------------------------
 9  0                              ! mrpars / (i,rp,up,ivary)
10  ------------------------------------------------
11  4                              ! mixtures ' number of effective media
12  1  0  0                       !    mlmnt,mipar,mrpar       #1
13     1  2  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' air
14  1  0  0                       !    mlmnt,mipar,mrpar       #2
15     1  3  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' Si
16  1  0  0                       !    mlmnt,mipar,mrpar       #3
17     1  5  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' SiO2
18  2  0  0                       !    mlmnt,mipar,mrpar       #4
19     1  3  0.5  0.02  1         !       j,lmnt,frac,ufrac,ivary  ' Si
20     2  5  0.5  0.02  1         !       j,lmnt,frac,ufrac,ivary  ' SiO2
21  ------------------------------------------------
22  1                              ! mbient ' number of ambients
23     1  1  0  0                 !    j,imix,mipar,mrpar          ' air
24  ------------------------------------------------
25  1                              ! msampl   ' number of samples analyzed
```

```
2                                   !    mfilm ' number of layers on sample #1
   1  3  1                         !       j,imix,iz     ' SiO2    , top layer
   2  4  2                         !       j,imix,iz     ' SiO2+Si, transition region
   3  2                            !       j,imix        ' Si      , substrate
-------------------------------------------------
1                                   ! mbien ' number of ambients on sample #1
1  1                                !    mrpeat,imbien
   1                                !       mexpt ' number of measurements
   1    1.5    70.0    0.0    0.0                      (i,E,a,  d,p)
        0.1    0.01    0.01   0.01                     (uncertainty)
```

Note, the above last two lines involve pseudo measurement data. Also, only two volume fractions are shown selected for variation, i.e., lines 19 and 20.. Due to the constraint among volume fractions for an effective medium, only one variable is subject to optimization, i.e., the oxide. This is mentioned in section 4.3.1 in option 4 near the bottom of page 43.

Suppose now that measurement data are requested for optical frequencies which correspond to energies between 1.5 and 6.0 eV with stepsize of 0.5 eV. Further, let this involve only one angle of incidence, e.g., 70 deg. A suitable input file X.INN may be as follows:

```
1                                   !   forward problems, plots, ...
2                                   !     grid scan
2                                   !        (Delta, psi),     x.dat
1.5    6.0    0.5                  !          grid energy     (eV)
70.    70.    0.0                  !          grid angle of incidence (degrees)
```

Given the above two input files, the program generates two output files. The output file X.OUT contains a journal listing of the program's activity. The output file X.OUT is given below.

```
Enter:   sub-directory     for constituent media
         sub-directory ' drb1:[data_bases]w.

-------------------------------------------------
Enter:     mfilmz ' number of distinct widths
   2       mfilmz ' number of distinct widths
Enter:                          i,z,zu,ivary
   1   100.000     2.000    0    i,z,zu,ivary
   2     2.000     2.000    0    i,z,zu,ivary

-------------------------------------------------
Enter:     mipars ' number of parameters (integer)
   0       mipars ' number of parameters (integer)

-------------------------------------------------
```

```
Enter:      mrpars ' number of parameters (floating-point)
   0        mrpars ' number of parameters (floating-point)

------------------------------------------------
Enter:      mmixtr ' number of distinct mixtures
   4        mmixtr ' number of distinct mixtures

Enter:          mlmnt, mipar, mrpar    (mix # 1)
   1   0   0    mlmnt, mipar, mrpar
Enter:                                  j, lmnt, frac, ufrac, ivary
   1    2   1.00000   0.00000    0      j, lmnt, frac, ufrac, ivary

Enter:          mlmnt, mipar, mrpar    (mix # 2)
   1   0   0    mlmnt, mipar, mrpar
Enter:                                  j, lmnt, frac, ufrac, ivary
   1    3   1.00000   0.00000    0      j, lmnt, frac, ufrac, ivary

Enter:          mlmnt, mipar, mrpar    (mix # 3)
   1   0   0    mlmnt, mipar, mrpar
Enter:                                  j, lmnt, frac, ufrac, ivary
   1    5   1.00000   0.00000    0      j, lmnt, frac, ufrac, ivary

Enter:          mlmnt, mipar, mrpar    (mix # 4)
   2   0   0    mlmnt, mipar, mrpar
Enter:                                  j, lmnt, frac, ufrac, ivary
   1    3   0.50000   0.02000    1      j, lmnt, frac, ufrac, ivary
   2    5   0.50000   0.02000    1      j, lmnt, frac, ufrac, ivary

------------------------------------------------
Enter:      mbient ' number of distinct ambients
   1        mbient ' number of distinct ambients
Enter:                 j, imix, mipar, mrpar
   1    1    0    0    j, imix, mipar, mrpar

------------------------------------------------
Enter:      msampl ' number of samples
   1        msampl ' number of samples

Enter:      mfilm ' number of films on sample # 1
   2        mfilm ' number of films on sample # 1
Enter:              j, imix, iwidth    (film/substrate)
   1    3    1      j, imix, iwidth
   2    4    2      j, imix, iwidth
   3    2           j, imix

------------------------------------------------
Enter:      mbien ' number of ambients on sample # 1
   1        mbien ' number of ambients on sample # 1
Enter:        mrpeat, imbien
   1    1     mrpeat, imbien
Enter:      mexpt ' number of measurement data
   1        mexpt ' number of measurement data
Enter:    j, wavln (nm), angli,delta,psi (degree)
```

```
69              wavlnu    , anglu,deltu,psiu
70
71     1     1.500    70.000     0.000     0.000     (j, wavln,angli, delta,psi )
72           0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
73
74
75  Enter:   option
76             1,  forward problems, plots, ...
77             2,  search      (vary)
78             3,  search grid (vary)
79  option =   1
80
81
82  Enter:   choice of incident (energies, angles)
83             1,   measurement data,     x.dat
84             2,   grid scan.
85  choice =   2
86
87  Enter:   choice of output suitable for:
88             1,   dielectric function,   media,
89             2,   (Delta, psi),          x.dat,
90             3,   (Delta, psi),       plotting,
91             4,   d/db,              b'(z,f,p),
92             5,   d/da,     angle of incidence,
93             6,   d/dE,                 energy.
94  choice =   2
95
96  Enter:    range of incident energies (eV)
97     or:                 - wavelengths (nm)
98  Enter:    ev1, ev2, ev3
99      1.5000    6.0000    0.5000              energy (eV)
100
101 Enter:    range of incident angles (degrees)
102 Enter:    angle1, angle2, angle3
103    70.0000   70.0000    0.0000          a1,a2,a3
104
105 lmnt '  2,  filename = air
106 lmnt '  5,  filename = Si_02g
107 lmnt '  3,  filename = Si
108
109 elapsed cpu-time =  37  centi-seconds
110                  +   1  seconds
```

Again, three constituent media are used in these calculations, and *'lmnt'* is a mnemonic for *elements* or constituents. Regarding the constituent media of air, crystalline silicon, and amorphous silicon dioxide, their index labels are given by 2, 3, and 5, respectively. This is shown above on lines 105 to 107.

The output file X.PLOT contains the requested set of measurement data of ellipsometric

angles, (Δ, ψ). The output file X.PLOT is given below.

```
 1      1                msampl
 2      1                mbien
 3      1    1           mrpeat, imbien
 4     10          mexpt ' measurements
 5      1    1.50000E+00  7.00000E+01  8.04309E+01  3.08875E+01  (i,E,a,  d,p)
 6           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
 7      2    2.00000E+00  7.00000E+01  7.97129E+01  4.31275E+01  (i,E,a,  d,p)
 8           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
 9      3    2.50000E+00  7.00000E+01  1.02227E+02  6.93018E+01  (i,E,a,  d,p)
10           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
11      4    3.00000E+00  7.00000E+01  2.50415E+02  6.09989E+01  (i,E,a,  d,p)
12           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
13      5    3.50000E+00  7.00000E+01  2.62427E+02  4.00376E+01  (i,E,a,  d,p)
14           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
15      6    4.00000E+00  7.00000E+01  2.47313E+02  3.31054E+01  (i,E,a,  d,p)
16           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
17      7    4.50000E+00  7.00000E+01  1.92800E+02  3.34557E+01  (i,E,a,  d,p)
18           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
19      8    5.00000E+00  7.00000E+01  1.24499E+02  3.21867E+01  (i,E,a,  d,p)
20           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
21      9    5.50000E+00  7.00000E+01  8.77637E+01  3.66717E+01  (i,E,a,  d,p)
22           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
23     10    6.00000E+00  7.00000E+01  7.10687E+01  4.18238E+01  (i,E,a,  d,p)
24           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
```

The data in file X.PLOT may then be inserted into file X.DAT, i.e., replacing the previous pseudo measurement data, see sections 4.2.8 and 4.2.9. This concludes one method or procedure that may be used for generating model measurement data.

5.2 Search (vary)

This option is discussed in sections 4.3 and 4.3.2 regarding the inverse problem. Suppose now that one is given the ellipsometric measurement data. For convenience, let the measurements be those presented in section 5.1. Since the inverse problem involves determining the model parameters from the measurements, let the initial solution be close to the true solution. Regarding the transition region of the sample, let the volume fractions be displaced from their correct values. Let the volume fraction for the silicon be judiciously set to 0.7, see line 19 below.

Further, let only one variable be subjected to optimization, and let it involve only the volume fractions of the transition region. The purpose here is to show the rate of convergence of the program. Attention is directed less toward optimality of iterations than that which is necessary to follow convergence. Again, the measurement data are *exact* for the correct model parameters, the system is nicely overdetermined, and the correct solution is found. The input file X.DAT is given below:

```
1  drb1:[data_bases]w.
2  ------------------------------------------------
3  2                              ! mfilmz ' thicknesses /(i,z,zu,ivary)
4    1    100.0    2.0    0      !    i,z,zu,ivary '    top layer, SiO2
5    2      2.0    2.0    0      !    i,z,zu,ivary ' bottom layer, SiO2+Si
6  ------------------------------------------------
7  0                              ! mipars / (i,ip)
8  ------------------------------------------------
9  0                              ! mrpars / (i,rp,up,ivary)
10 ------------------------------------------------
11 4                              ! mixtures ' number of effective media
12 1  0  0                        !    mlmnt,mipar,mrpar       #1
13    1  2  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' air
14 1  0  0                        !    mlmnt,mipar,mrpar       #2
15    1  3  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' Si
16 1  0  0                        !    mlmnt,mipar,mrpar       #3
17    1  5  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' SiO2
18 2  0  0                        !    mlmnt,mipar,mrpar       #4
19    1  3  0.7  0.02  1         !       j,lmnt,frac,ufrac,ivary  ' Si
20    2  5  0.3  0.02  1         !       j,lmnt,frac,ufrac,ivary  ' SiO2
21 ------------------------------------------------
22 1                              ! mbient ' number of ambients
23    1  1  0  0                 !    j,imix,mipar,mrpar          ' air
24 ------------------------------------------------
25 1                              ! msampl   ' number of samples analyzed
26 2                              !    mfilm ' number of layers on sample #1
27    1  3  1                    !       j,imix,iz    ' SiO2   , top layer
28    2  4  2                    !       j,imix,iz    ' SiO2+Si, transition region
```

```
 3  2                          !       j,imix       ' Si      , substrate
 ------------------------------------------------
 1                             ! mbien ' number of ambients on sample #1
 1  1                          !    mrpeat,imbien
   10                          !       mexpt ' number of measurements
    1    1.50000E+00  7.00000E+01  8.04309E+01  3.08875E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
    2    2.00000E+00  7.00000E+01  7.97129E+01  4.31275E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
    3    2.50000E+00  7.00000E+01  1.02227E+02  6.93018E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
    4    3.00000E+00  7.00000E+01  2.50415E+02  6.09989E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
    5    3.50000E+00  7.00000E+01  2.62427E+02  4.00376E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
    6    4.00000E+00  7.00000E+01  2.47313E+02  3.31054E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
    7    4.50000E+00  7.00000E+01  1.92800E+02  3.34557E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
    8    5.00000E+00  7.00000E+01  1.24499E+02  3.21867E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
    9    5.50000E+00  7.00000E+01  8.77637E+01  3.66717E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
   10    6.00000E+00  7.00000E+01  7.10687E+01  4.18238E+01  (i,E,a,  d,p)
         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
```

The option for unconstrained optimization is selected by setting the one and only line of data in the input file X.INN to the integer value 2, i.e.,

```
1  2                          ! search (vary)
```

The output file X.OUT contains a journal listing of the input data, progress reports regarding convergence, the final numerical values of the *vary* model parameters, and a selection of associated statistics, as well as the amount of cpu-time that is used during calculation. The output shows on line 132 that 27 iterations were used to converge to the correct final solution shown on line 135. Again, regarding line 135, the index label 5 refers to the volume fraction for amorphous silicon dioxide. This index label refers to the fifth successive volume fraction found by cumulative indexing the constituent media of the effective media, and *not* the constituent label *xx* that would be associated with DIELxx. The output file X.OUT is given below.

```
1  Enter:   sub-directory     for constituent media
2           sub-directory ' drb1:[data_bases]w.
3
```

```
4  ------------------------------------------------
5  Enter:     mfilmz ' number of distinct widths
6     2       mfilmz ' number of distinct widths
7  Enter:                 '          i,z,zu,ivary
8     1   100.000     2.000    0    i,z,zu,ivary
9     2     2.000     2.000    0    i,z,zu,ivary
10
11 ------------------------------------------------
12 Enter:     mipars ' number of parameters (integer)
13    0       mipars ' number of parameters (integer)
14
15 ------------------------------------------------
16 Enter:     mrpars ' number of parameters (floating-point)
17    0       mrpars ' number of parameters (floating-point)
18
19 ------------------------------------------------
20 Enter:     mmixtr ' number of distinct mixtures
21    4       mmixtr ' number of distinct mixtures
22
23 Enter:         mlmnt, mipar, mrpar    (mix # 1)
24    1   0   0   mlmnt, mipar, mrpar
25 Enter:                                  j, lmnt, frac, ufrac, ivary
26    1    2   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary
27
28 Enter:         mlmnt, mipar, mrpar    (mix # 2)
29    1   0   0   mlmnt, mipar, mrpar
30 Enter:                                  j, lmnt, frac, ufrac, ivary
31    1    3   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary
32
33 Enter:         mlmnt, mipar, mrpar    (mix # 3)
34    1   0   0   mlmnt, mipar, mrpar
35 Enter:                                  j, lmnt, frac, ufrac, ivary
36    1    5   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary
37
38 Enter:         mlmnt, mipar, mrpar    (mix # 4)
39    2   0   0   mlmnt, mipar, mrpar
40 Enter:                                  j, lmnt, frac, ufrac, ivary
41    1    3   0.70000   0.02000    1     j, lmnt, frac, ufrac, ivary
42    2    5   0.30000   0.02000    1     j, lmnt, frac, ufrac, ivary
43
44 ------------------------------------------------
45 Enter:     mbient ' number of distinct ambients
46    1       mbient ' number of distinct ambients
47 Enter:                j, imix, mipar, mrpar
48    1    1    0    0    j, imix, mipar, mrpar
49
50 ------------------------------------------------
51 Enter:     msampl ' number of samples
52    1       msampl ' number of samples
53
54 Enter:     mfilm ' number of films on sample # 1
55    2       mfilm ' number of films on sample # 1
56 Enter:              j, imix, iwidth    (film/substrate)
```

```
    1    3    1      j, imix, iwidth
    2    4    2      j, imix, iwidth
    3    2           j, imix

-------------------------------------------------
Enter:       mbien ' number of ambients on sample # 1
    1        mbien ' number of ambients on sample # 1
Enter:         mrpeat, imbien
    1    1     mrpeat, imbien
Enter:       mexpt ' number of measurement data
   10        mexpt ' number of measurement data
Enter:      j, wavln (nm), angli,delta,psi (degree)
              wavlnu     , anglu,deltu,psiu

    1     1.500    70.000    80.431    30.888     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
    2     2.000    70.000    79.713    43.127     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
    3     2.500    70.000   102.227    69.302     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
    4     3.000    70.000   250.415    60.999     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
    5     3.500    70.000   262.427    40.038     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
    6     4.000    70.000   247.313    33.105     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
    7     4.500    70.000   192.800    33.456     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
    8     5.000    70.000   124.499    32.187     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
    9     5.500    70.000    87.764    36.672     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   10     6.000    70.000    71.069    41.824     (j, wavln,angli, delta,psi )
          0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)


Enter:    option
            1,  forward problems, plots, ...
            2,  search      (vary)
            3,  search grid (vary)
option =    2

lmnt '  2,  filename = air
lmnt '  5,  filename = Si_02g
lmnt '  3,  filename = Si

seek:   loop,     ratio of reduction,      |g|
                 (rel)       (total)
            0                              1.268E+02
            1   8.800E-01   8.800E-01      1.116E+02
            2   8.668E-01   7.628E-01      9.670E+01
            3   8.508E-01   6.490E-01      8.227E+01
            4   8.308E-01   5.392E-01      6.836E+01
```

```
            5   8.050E-01   4.341E-01   5.503E+01
            6   7.702E-01   3.343E-01   4.238E+01
            7   7.200E-01   2.407E-01   3.051E+01
            8   6.388E-01   1.538E-01   1.949E+01
            9   6.143E-01   9.446E-02   1.197E+01
           10   6.093E-01   5.755E-02   7.295E+00
           11   6.056E-01   3.485E-02   4.418E+00
           12   6.039E-01   2.105E-02   2.668E+00
           13   6.020E-01   1.267E-02   1.606E+00
           14   6.012E-01   7.618E-03   9.657E-01
           15   6.019E-01   4.585E-03   5.813E-01
           16   6.003E-01   2.753E-03   3.489E-01
           17   6.037E-01   1.662E-03   2.107E-01
           18   6.001E-01   9.972E-04   1.264E-01
           19   6.082E-01   6.065E-04   7.688E-02
           20   6.108E-01   3.704E-04   4.696E-02
           21   6.391E-01   2.367E-04   3.001E-02
           22   6.788E-01   1.607E-04   2.037E-02
           23   7.420E-01   1.192E-04   1.512E-02
           24   8.563E-01   1.021E-04   1.294E-02
           25   9.337E-01   9.535E-05   1.209E-02
           26   9.660E-01   9.211E-05   1.168E-02
           27   9.638E-01   8.877E-05   1.125E-02

model parameter value along the minimum:
        1)         0.5000   0.00000,   for:    5 " (f,fu)

initial |g| =  1.26767E+02
  final |g| =  1.12530E-02

---------------------------------------------------------------
Statistics of deviations " experiment-model " g

    where:   g is a column vector of length 2M,
            () denotes either  (delta) or (psi)
       mean () =    m() = <g()> = (1/M) sum: g()
   variance () = < (g() -m() )**2 >
 covariance    = < (g(1)-m(1))*(g(2)-m(2)) >
       std dev = sqrt (variance)
 correlat coef = covariance / (sd(1) * sd(2))

                     mean,     std dev     (degrees)
      psi:          0.000       0.000
    delta:          0.000       0.000
                    0.172 " correlation coefficient " <psi|delta>

[J(T)*J],    renormalized for correlation.
   1)    1.00000

Normalization coefficients, sqrt:    [J(T)*J]_(i,i)
  7.80E-02

rcond =   1.00E+00,  condition number,  [J(T)*J]
```

```
163
164  [J(T)*J]**(-1):
165     Determinant:      10.0000 E  -1.0000
166         Inertia:    (   1   0   0),  number of (+,-,0) eigenvalues.
167         Inverse:     upper+diagonal  matrix
168     1)   1.00E+00
169
170  ------------------------------------------------------------
171  The standard deviation of the residuals.
172
173  s(g) = sqrt [<gg>/(2M-N)] =   1.15453E-04  (degrees)
174
175
176  The estimated uncertainty in the model parameters,
177  assuming no correlation, i.e., diagonal terms only.
178
179                   value,   uncertainty.
180      1)         0.5000        0.00003,   for:    5 ' (f,fu)
181
182  elapsed cpu-time =  10  centi-seconds
183                   +  10  seconds
```

Note that lines 137 and 138 refer to root-mean-square (rms) residuals, which involve deviations scaled by the measurement uncertainties, and hence, are unitless. The final rms residual is near 10^{-2}, and the measurement uncertainty is 10^{-2}. Hence, the final rms deviation is near 10^{-4}, as is shown on line 173.

The output file X.PLOT contains a listing of the deviations between the measurements of (Δ, ψ) and that calculated for the model. The output file X.PLOT is presented below.

```
1      1              msampl
2      1              mbien
3      1    1         mrpeat, imbien
4     10    2         mexpt, mu
5      1    1.50000E+00  7.00000E+01  0.00000E+00  6.83019E-06  (i,E,a,  g (d,p))
6      2    2.00000E+00  7.00000E+01  5.46415E-05 -4.09811E-05  (i,E,a,  g (d,p))
7      3    2.50000E+00  7.00000E+01  1.36604E-04 -1.02453E-04  (i,E,a,  g (d,p))
8      4    3.00000E+00  7.00000E+01 -4.78113E-05  2.04906E-05  (i,E,a,  g (d,p))
9      5    3.50000E+00  7.00000E+01 -2.59547E-04 -3.07358E-05  (i,E,a,  g (d,p))
10     6    4.00000E+00  7.00000E+01  3.21019E-04  5.12264E-05  (i,E,a,  g (d,p))
11     7    4.50000E+00  7.00000E+01 -5.46415E-05  1.36604E-05  (i,E,a,  g (d,p))
12     8    5.00000E+00  7.00000E+01  1.91245E-04  1.02453E-05  (i,E,a,  g (d,p))
13     9    5.50000E+00  7.00000E+01  0.00000E+00  3.41509E-05  (i,E,a,  g (d,p))
14    10    6.00000E+00  7.00000E+01  0.00000E+00 -4.09811E-05  (i,E,a,  g (d,p))
```

5.3 Search Grid (vary)

This option is discussed in sections 4.3 and 4.3.3. Suppose again that the exercise of the single search that was presented in the previous subsection yielded a residual much larger than the measurement uncertainties, and yet there is strong conviction that the model is *reasonable*. One may then want to find the *best* solution that is available for this given model of the sample. Hence, one may be led to consider searching a grid of model parameters.

Regarding the example presented in the previous section, suppose further that the thickness of the transition region is sought as well. For convenience, let the initial value solution be displaced from the correct value; let the thickness be set to 3 nm, see line 5. The input file X.DAT is given below.

```
1  drb1:[data_bases]w.
2  ------------------------------------------------
3  2                              ! mfilmz ' thicknesses /(i,z,zu,ivary)
4    1    100.0    2.0    0      !    i,z,zu,ivary '    top layer, SiO2
5    2      3.0    2.0    1      !    i,z,zu,ivary ' bottom layer, SiO2+Si
6  ------------------------------------------------
7  0                              ! mipars / (i,ip)
8  ------------------------------------------------
9  0                              ! mrpars / (i,rp,up,ivary)
10 ------------------------------------------------
11 4                              ! mixtures ' number of effective media
12 1  0  0                        !    mlmnt,mipar,mrpar       #1
13    1  2  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' air
14 1  0  0                        !    mlmnt,mipar,mrpar       #2
15    1  3  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' Si
16 1  0  0                        !    mlmnt,mipar,mrpar       #3
17    1  5  1.0  0.0   0         !       j,lmnt,frac,ufrac,ivary  ' SiO2
18 2  0  0                        !    mlmnt,mipar,mrpar       #4
19    1  3  0.7  0.02  1         !       j,lmnt,frac,ufrac,ivary  ' Si
20    2  5  0.3  0.02  1         !       j,lmnt,frac,ufrac,ivary  ' SiO2
21 ------------------------------------------------
22 1                              ! mbient ' number of ambients
23    1  1  0  0                 !    j,imix,mipar,mrpar          ' air
24 ------------------------------------------------
25 1                              ! msampl   ' number of samples analyzed
26 2                              !    mfilm ' number of layers on sample #1
27    1  3  1                    !       j,imix,iz    ' SiO2   , top layer
28    2  4  2                    !       j,imix,iz    ' SiO2+Si, transition region
29    3  2                       !       j,imix       ' Si     , substrate
30 ------------------------------------------------
31 1                              ! mbien ' number of ambients on sample #1
32 1  1                          !    mrpeat,imbien
33   10                          !       mexpt ' number of measurements
```

```
34     1    1.50000E+00  7.00000E+01  8.04309E+01  3.08875E+01  (i,E,a,  d,p)
35          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
36     2    2.00000E+00  7.00000E+01  7.97129E+01  4.31275E+01  (i,E,a,  d,p)
37          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
38     3    2.50000E+00  7.00000E+01  1.02227E+02  6.93018E+01  (i,E,a,  d,p)
39          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
40     4    3.00000E+00  7.00000E+01  2.50415E+02  6.09989E+01  (i,E,a,  d,p)
41          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
42     5    3.50000E+00  7.00000E+01  2.62427E+02  4.00376E+01  (i,E,a,  d,p)
43          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
44     6    4.00000E+00  7.00000E+01  2.47313E+02  3.31054E+01  (i,E,a,  d,p)
45          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
46     7    4.50000E+00  7.00000E+01  1.92800E+02  3.34557E+01  (i,E,a,  d,p)
47          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
48     8    5.00000E+00  7.00000E+01  1.24499E+02  3.21867E+01  (i,E,a,  d,p)
49          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
50     9    5.50000E+00  7.00000E+01  8.77637E+01  3.66717E+01  (i,E,a,  d,p)
51          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
52    10    6.00000E+00  7.00000E+01  7.10687E+01  4.18238E+01  (i,E,a,  d,p)
53          1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
```

Here, there are three model parameters selected for variation, but only two undergo optimization and contribute to the grid. Both model parameters involve the transition region, the thickness of the layer and the volume fraction for the amorphous silicon dioxide. Regarding the construction of the grid of model parameters: let the thickness range between 1.0 and 3.0 nm with stepsize of 2.0 nm, and let the volume fraction range between 0.3 and 0.7 with stepsize of 0.4, so that no grid point lies on the correct solution. Accordingly, the input file X.INN is given below.

```
1  3                                ! search grid
2  2    1.0    3.0    2.0          ! thickness,                transition region
3  2    0.3    0.7    0.4          ! volume fraction, oxide, transition region
4  1                                ! full optimization, use Jacobian
```

The output file X.OUT is given below.

```
1  Enter:   sub-directory     for constituent media
2           sub-directory ' drb1:[data_bases]w.
3
4  ------------------------------------------------
5  Enter:      mfilmz ' number of distinct widths
6     2        mfilmz ' number of distinct widths
7  Enter:                              i,z,zu,ivary
8     1   100.000      2.000     0     i,z,zu,ivary
9     2     3.000      2.000     1     i,z,zu,ivary
10
```

```
11 ------------------------------------------------
12 Enter:     mipars ' number of parameters (integer)
13    0       mipars ' number of parameters (integer)
14
15 ------------------------------------------------
16 Enter:     mrpars ' number of parameters (floating-point)
17    0       mrpars ' number of parameters (floating-point)
18
19 ------------------------------------------------
20 Enter:     mmixtr ' number of distinct mixtures
21    4       mmixtr ' number of distinct mixtures
22
23 Enter:         mlmnt, mipar, mrpar    (mix # 1)
24    1   0   0   mlmnt, mipar, mrpar
25 Enter:                                  j, lmnt, frac, ufrac, ivary
26    1    2   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary
27
28 Enter:         mlmnt, mipar, mrpar    (mix # 2)
29    1   0   0   mlmnt, mipar, mrpar
30 Enter:                                  j, lmnt, frac, ufrac, ivary
31    1    3   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary
32
33 Enter:         mlmnt, mipar, mrpar    (mix # 3)
34    1   0   0   mlmnt, mipar, mrpar
35 Enter:                                  j, lmnt, frac, ufrac, ivary
36    1    5   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary
37
38 Enter:         mlmnt, mipar, mrpar    (mix # 4)
39    2   0   0   mlmnt, mipar, mrpar
40 Enter:                                  j, lmnt, frac, ufrac, ivary
41    1    3   0.70000   0.02000    1     j, lmnt, frac, ufrac, ivary
42    2    5   0.30000   0.02000    1     j, lmnt, frac, ufrac, ivary
43
44 ------------------------------------------------
45 Enter:     mbient ' number of distinct ambients
46    1       mbient ' number of distinct ambients
47 Enter:              j, imix, mipar, mrpar
48    1    1    0    0    j, imix, mipar, mrpar
49
50 ------------------------------------------------
51 Enter:     msampl ' number of samples
52    1       msampl ' number of samples
53
54 Enter:     mfilm ' number of films on sample # 1
55    2       mfilm ' number of films on sample # 1
56 Enter:            j, imix, iwidth    (film/substrate)
57    1    3    1     j, imix, iwidth
58    2    4    2     j, imix, iwidth
59    3    2          j, imix
60
61 ------------------------------------------------
62 Enter:     mbien ' number of ambients on sample # 1
63    1       mbien ' number of ambients on sample # 1
```

```
Enter:        mrpeat, imbien
   1    1     mrpeat, imbien
Enter:      mexpt ' number of measurement data
  10          mexpt ' number of measurement data
Enter:     j, wavln (nm), angli,delta,psi (degree)
              wavlnu     , anglu,deltu,psiu

   1     1.500    70.000    80.431    30.888     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   2     2.000    70.000    79.713    43.127     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   3     2.500    70.000   102.227    69.302     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   4     3.000    70.000   250.415    60.999     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   5     3.500    70.000   262.427    40.038     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   6     4.000    70.000   247.313    33.105     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   7     4.500    70.000   192.800    33.456     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   8     5.000    70.000   124.499    32.187     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
   9     5.500    70.000    87.764    36.672     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)
  10     6.000    70.000    71.069    41.824     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)


Enter:   option
           1,  forward problems, plots, ...
           2,  search      (vary)
           3,  search grid (vary)
option =   3


Scan a grid of model parameters:  (z,p,f).
Grid info:   DO-loop parameters
Grid info:    i,        initial,      final,     increment

Enter:    i, z1, z2, z3             (widths)
     1)       2          1.0000       3.0000       2.0000     (  1)

Enter:    i, f1, f2, f3             (fraction)
     2)       2          0.3000       0.7000       0.4000     (  2)

Enter:   option  regarding the grid scan:
           0,     no optimization, |g| only,
           1,   full optimization, Jacobian.
option    ' 1

Restart by attempting to read breakpoint info
```

```
Note:    NO attempt was made to restart.

lmnt ' 2,  filename = air
lmnt ' 5,  filename = Si_O2g
lmnt ' 3,  filename = Si

number of grid points scanned, kts =         4
 population along the minimum, ktm =         1
         norm of the residual, |g| =  1.17853E-04  (degrees)

model parameter value along the minimum:
   1)    2.00014E+00,   for:    2,  (z)
   2)    4.99959E-01,   for:    5,  (f)

-------------------------------------------------------------
Statistics of deviations ' experiment-model ' g

    where:   g is a column vector of length 2M,
            () denotes either  (delta) or (psi)
       mean () =    m() = <g()> = (1/M) sum: g()
   variance () = < (g() -m() )**2 >
 covariance    = < (g(1)-m(1))*(g(2)-m(2)) >
       std dev = sqrt (variance)
 correlat coef = covariance / (sd(1) * sd(2))

                     mean,     std dev     (degrees)
      psi:          0.000       0.000
    delta:          0.000       0.000
                    0.422 ' correlation coefficient ' <psi|delta>

[J(T)*J],    renormalized for correlation.
   1)    1.00000
   2)    0.94674   1.00000

Normalization coefficients, sqrt:    [J(T)*J]_(i,i)
  2.90E-02  7.80E-02

rcond =   2.74E-02,  condition number,  [J(T)*J]

[J(T)*J]**(-1):
   Determinant:      1.0369 E  -1.0000
       Inertia:   (   2   0   0),  number of (+,-,0) eigenvalues.
       Inverse:    upper+diagonal  matrix
   1)   9.64E+00
   2)  -9.13E+00  9.64E+00

-------------------------------------------------------------
The standard deviation of the residuals.

s(g) = sqrt [<gg>/(2M-N)] =   1.24228E-04  (degrees)


The estimated uncertainty in the model parameters,
```

```
170  assuming no correlation, i.e., diagonal terms only.
171
172                    value,   uncertainty.
173       1)         2.0001         0.00023,   for:    2 * (z,zu)
174       2)         0.5000         0.00009,   for:    5 * (f,fu)
175
176  elapsed cpu-time =  52  centi-seconds
177                    +   0  seconds
178                    +   1  minutes
```

Regarding lines 173 and 174, one finds the values of the calculated uncertainty for the model parameters. Incidently, it is also possible to calculate the uncertainty by inserting information from lines 132 to 168, i.e., "Statistics of deviations," into eq (24) directly, but then one must respect the mixed units of measure of the appropriate quantities. Regarding line 166, s_g is expressed in degrees. The Jacobian is expressed in radians and is scaled as well. To help one understand how the output data of the "Statistics of deviations" may be inserted into eq (24), consider as an example the calculation that yields the value of uncertainty shown on line 173. From eq (24), it follows that

$$\begin{aligned}
\mathcal{U}(v_1) &= \left(\frac{\pi}{180}\right) \quad s_g \quad \left\{\left[\left(\mathbf{J}_v^{\mathrm{T}}\mathbf{J}_v\right)^{-1}\right]_{11}\right\}^{1/2} \\
&\approx \left(\frac{3.14159}{180.0}\right) \quad 1.24228 \times 10^{-4} \quad \frac{\sqrt{9.64}}{2.9 \times 10^{-2}} \\
&\approx 2.32 \times 10^{-4}
\end{aligned}$$

where information has been taken from lines 166, 160, and 152.

The output file X.PLOT is given below.

```
 1      1                 msampl
 2      1                 mbien
 3      1    1            mrpeat, imbien
 4     10    2            mexpt, mu
 5      1    1.50000E+00  7.00000E+01  0.00000E+00  1.70755E-05  (i,E,a,  g (d,p))
 6      2    2.00000E+00  7.00000E+01  2.73208E-05 -3.75660E-05  (i,E,a,  g (d,p))
 7      3    2.50000E+00  7.00000E+01  2.73208E-05 -7.51321E-05  (i,E,a,  g (d,p))
 8      4    3.00000E+00  7.00000E+01 -6.14717E-05 -2.04906E-05  (i,E,a,  g (d,p))
 9      5    3.50000E+00  7.00000E+01 -2.59547E-04 -5.12264E-05  (i,E,a,  g (d,p))
10      6    4.00000E+00  7.00000E+01  3.62000E-04  3.07358E-05  (i,E,a,  g (d,p))
11      7    4.50000E+00  7.00000E+01 -8.19623E-05 -1.16113E-04  (i,E,a,  g (d,p))
12      8    5.00000E+00  7.00000E+01  5.46415E-05 -1.70755E-05  (i,E,a,  g (d,p))
13      9    5.50000E+00  7.00000E+01 -1.36604E-04  1.70755E-05  (i,E,a,  g (d,p))
14     10    6.00000E+00  7.00000E+01 -1.36604E-04 -4.09811E-05  (i,E,a,  g (d,p))
```

5.4 Supplementary Parameters

As mentioned in the various subsections of section 4.2, supplementary parameters provide the user with additional degrees of freedom in characterizing the constituent and effective media. Currently, the program has incorporated only one constituent medium that requires a supplementary parameter, i.e., $Al_xGa_{1-x}As$, where x refers to the aluminum mole fraction and $(0 \leq x \leq 0.8)$. This may be seen by considering both the subroutine DIEL09 and its associated database file W.AL_GA_AS, see sections 6.5.9 and 6.6.7, respectively. The database contains profiles of the dielectric function as functions of energy for a set of discrete values (x_j) of the mole fraction of aluminum. To evaluate the dielectric function for intermediate values of x, subroutine DIEL09 linearly interpolates between two discrete profiles. Because of the rapid variations in the profiles and the coarseness of the (x_j) grid, a better approximation for $\varepsilon(x)$ than that used here is possible by fitting the profiles of the dielectric functions with a set of oscillators as functions of the mole fraction (x). To provide this approximation, one would need to modify the subroutine DIEL09 appropriately. Incidently, effects due to the internal electric field that is induced near the boundary between adjacent layers of GaAs and $Al_xGa_{1-x}As$ where $(x > 0)$ are *not* included here [6, 7].

The mechanism for making the proper correspondence between the model parameters and appropriate effective media and/or constituent media may be readily seen from the argument list of subroutines DIEFCN and DIELMN. The subroutine for the constituent medium receives supplementary parameters that are associated with the effective medium, as well as those associated with the ambient. Hence, it is incumbent on the subroutine of the constituent medium to interpret the information. Subroutine DIEL09 check-tests for the appropriate supplementary parameters, utilizes the parameters in calculating the constituent dielectric function, and calculates the appropriate partial derivative, which is required for any optimization variable. Regarding the supplementary parameters and their numerical values, the integer value labels the appropriate constituent medium, while the floating-point value provides the necessary relative concentration of aluminum.

For sake of demonstration, consider a room-temperature sample involving a substrate of GaAs and a top layer of $Al_{.3}Ga_{.7}As$. Let the top layer thickness be 50 nm. Consider generating a set of measurement data for this structure. The input file X.DAT may be as follows.

```
drb1:[data_bases]w.
-------------------------------------------------
1                                ! mfilmz / (j,z,zu,ivary)
   1   50.   1.0    0           !   thickness, top layer
-------------------------------------------------
1                                ! mipars / (i,ip)
   1    9                        !   lmnt " 9,   constituent " AlGaAs
-------------------------------------------------
1                                ! mrpars / (i,rp,up,ivary)
   1   0.3   0.01   1           !    stoichiometry " x,    Al(x)Ga(1-x)As
-------------------------------------------------
3                                ! mixtures " number of effective media
1  0  0                          !    mlmnt,mipar,mrpar       #1
   1  2  1.0  0.0   0           !       j,lmnt,frac,ufrac,ivary  " air
1  0  0                          !    mlmnt,mipar,mrpar       #2
   1  8  1.0  0.0   0           !       j,lmnt,frac,ufrac,ivary  " GaAs
1  1  1                          !    mlmnt,mipar,mrpar       #3
   1  9  1.0  0.0   0           !       j,lmnt,frac,ufrac,ivary  " AlGaAs
   1  1                          !       i, iip
   1  1                          !       i, irp
-------------------------------------------------
1                                ! mbient " number of distinct ambients
1   1   0   0                   !    j,imix,mipar,mrpar          " air
-------------------------------------------------
1                                ! msampl " number of samples
1                                !    mfilm " number of layers on sample #1
1   3   1                        !       j,imix,iz                " AlGaAs
2   2                            !       j,imix                   " GaAs substrate
-------------------------------------------------
1                                ! mbien " number of ambients for sample #1
1   1                            !    mrpeat,imbien
    1                            !       mexpt " number of measurements
    1    1.5      70.0     0.0     0.0                    (i,E,a,  d,p)
         0.1      0.01     0.01    0.01                   (uncertainty)
```

Again, the above last two lines involve pseudo measurement data.

Suppose now that measurement data are requested for energies that range between 1.5 and 4.0 eV with stepsize of 0.5 eV. Let the angles of incidence involve 65 and 70 deg. The input file X.INN may be as follows.

```
1                                ! forward problems, plots, ...
2                                !    grid scan
2                                !        (Delta, psi),    x.dat
1.5    4.0     0.5               !        incident energy (eV)
65.    70.     5.0               !        incident angles (degrees)
```

The output file X.OUT is given below.

```
Enter:   sub-directory     for constituent media
         sub-directory ' drb1:[data_bases]w.

-------------------------------------------------
Enter:     mfilmz ' number of distinct widths
   1       mfilmz ' number of distinct widths
Enter:                          i,z,zu,ivary
   1    50.000     1.000    0    i,z,zu,ivary

-------------------------------------------------
Enter:     mipars ' number of parameters (integer)
   1       mipars ' number of parameters (integer)
Enter:               j, iparm     (# items= 1)
   1             9    j, iparm

-------------------------------------------------
Enter:     mrpars ' number of parameters (floating-point)
   1       mrpars ' number of parameters (floating-point)
Enter:    j, rparm, uparm, ivary   (# items= 1)
   1     3.00000E-01  1.00000E-02    1    (j,rparm,uparm,ivary)

-------------------------------------------------
Enter:     mmixtr ' number of distinct mixtures
   3       mmixtr ' number of distinct mixtures

Enter:         mlmnt, mipar, mrpar    (mix # 1)
   1   0   0   mlmnt, mipar, mrpar
Enter:                                 j, lmnt, frac, ufrac, ivary
   1    2   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary

Enter:         mlmnt, mipar, mrpar    (mix # 2)
   1   0   0   mlmnt, mipar, mrpar
Enter:                                 j, lmnt, frac, ufrac, ivary
   1    8   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary

Enter:         mlmnt, mipar, mrpar    (mix # 3)
   1   1   1   mlmnt, mipar, mrpar
Enter:                                 j, lmnt, frac, ufrac, ivary
   1    9   1.00000   0.00000    0     j, lmnt, frac, ufrac, ivary
Enter:       j, iiparm    (# items = 1)
   1    1    j, iiparm
Enter:       j, irparm    (# items = 1)
   1    1    j, irparm

-------------------------------------------------
Enter:     mbient ' number of distinct ambients
   1       mbient ' number of distinct ambients
Enter:                 j, imix, mipar, mrpar
   1    1    0    0    j, imix, mipar, mrpar

-------------------------------------------------
Enter:     msampl ' number of samples
```

```
   1       msampl ' number of samples

Enter:     mfilm ' number of films on sample # 1
   1       mfilm ' number of films on sample # 1
Enter:             j, imix, iwidth    (film/substrate)
   1    3    1     j, imix, iwidth
   2    2          j, imix

------------------------------------------------
Enter:     mbien ' number of ambients on sample # 1
   1       mbien ' number of ambients on sample # 1
Enter:       mrpeat, imbien
   1    1    mrpeat, imbien
Enter:     mexpt ' number of measurement data
   1       mexpt ' number of measurement data
Enter:    j, wavln (nm), angli,delta,psi (degree)
             wavlnu    , anglu,deltu,psiu

   1     1.500    70.000     0.000     0.000     (j, wavln,angli, delta,psi )
         0.100     0.010     0.010     0.010     (   wavlu,anglu, deltu,psiu)


Enter:   option
           1,  forward problems, plots, ...
           2,  search      (vary)
           3,  search grid (vary)
option =   1


Enter:   choice of incident (energies, angles)
           1,   measurement data,     x.dat
           2,   grid scan.
choice =   2

Enter:   choice of output suitable for:
           1,   dielectric function,   media,
           2,   (Delta, psi),          x.dat,
           3,   (Delta, psi),       plotting,
           4,   d/db,             b'(z,f,p),
           5,   d/da,     angle of incidence,
           6,   d/dE,                 energy.
choice =   2

Enter:    range of incident energies (eV)
   or:                - wavelengths (nm)
Enter:    ev1, ev2, ev3
    1.5000    4.0000    0.5000             energy (eV)

Enter:    range of incident angles (degrees)
Enter:    angle1, angle2, angle3
   65.0000   70.0000    5.0000          a1,a2,a3

lmnt '  2,  filename = air
```

```
106  lmnt ´  9,  filename = Al_Ga_As
107  lmnt ´  8,  filename = Ga_As
108
109  elapsed cpu-time =  67  centi-seconds
110                   +   5  seconds
```

The requested measurement data of ellipsometric angles (Δ, ψ) are written to X.PLOT. The output file X.PLOT is given below.

```
 1      1                msampl
 2      1                mbien
 3      1    1           mrpeat, imbien
 4     12          mexpt ´ measurements
 5      1    1.50000E+00  6.50000E+01  1.76446E+02  1.24387E+01  (i,E,a,  d,p)
 6           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
 7      2    2.00000E+00  6.50000E+01  1.79401E+02  1.57825E+01  (i,E,a,  d,p)
 8           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
 9      3    2.50000E+00  6.50000E+01  1.77396E+02  1.95307E+01  (i,E,a,  d,p)
10           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
11      4    3.00000E+00  6.50000E+01  1.66208E+02  2.33677E+01  (i,E,a,  d,p)
12           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
13      5    3.50000E+00  6.50000E+01  1.49379E+02  2.43128E+01  (i,E,a,  d,p)
14           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
15      6    4.00000E+00  6.50000E+01  1.48705E+02  2.35367E+01  (i,E,a,  d,p)
16           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
17      7    1.50000E+00  7.00000E+01  1.69646E+02  4.65575E+00  (i,E,a,  d,p)
18           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
19      8    2.00000E+00  7.00000E+01  1.78701E+02  8.31343E+00  (i,E,a,  d,p)
20           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
21      9    2.50000E+00  7.00000E+01  1.75534E+02  1.26304E+01  (i,E,a,  d,p)
22           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
23     10    3.00000E+00  7.00000E+01  1.58723E+02  1.75083E+01  (i,E,a,  d,p)
24           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
25     11    3.50000E+00  7.00000E+01  1.35016E+02  1.96919E+01  (i,E,a,  d,p)
26           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
27     12    4.00000E+00  7.00000E+01  1.33538E+02  1.88574E+01  (i,E,a,  d,p)
28           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
```

6. Listing of Software Source Files and Routines

This section presents a listing of the source files for MAIN2. For convenience, the files are partitioned into six subsections. The first subsection involves files associated with storage arrays. The second subsection involves source programs that govern the direct or forward problem and the command options. The third subsection involves general utilities that provide basic tasks for the source programs. The fourth subsection involves source routines associated with evaluating the dielectric function for the effective media. The fifth subsection involves source routines associated with evaluating the dielectric function for the constituent media. The sixth subsection involves the databases that are associated with the constituent media.

For convenience, a brief orientation of the organization of the software development may be found in file A.HLP.

```
Notes:

1)   FORWRD solves the direct or forward problem.
     The model parameters of sample include:
               (z,e) ~ (thickness, dielectric function)
     The dielectric function involves variables:   (f,p)
     FORWRD calculates:   R,  dR/dz,  dR/de,  dR/da.
     Regarding (f,p),  one needs:     de/df,  de/dp
     so the Jacobian involves:      J(z,f,p)
     where:   z ~ film thicknesses,
              f ~ volume fractions of effective medium,
              p ~ parameters local to constituents,
              e ~ dielectric function,
              a ~ angle of incidence

2)   EMA:     1 = sum(j): f(j),      j indexes constituent media.

3)   Organization of software development:

     A)   Enter relevant information for processing.
          1)   DEFNIT ~ define parameters controlling array allocation.
          2)   MAIN   ~ main program ~ command processing of input data.
          3)   INPDAT ~ input data to describe:  models + measurements.

     B)   Command options for simulation, plotting, etc.
          1)   SCATOS ~ decision branch/tree
          2)   SCAT01 ~ scan grid, domain input measurement data
          3)   SCAT02 ~ scan grid, domain contiguous
          4)   SCAN01 ~ scan grid of model parameters, evaluate |g|

     C)   Minimize the residual in the least-squares sense.
```

```
32          1)   SEEK01 ' search:     single, unbounded
33          2)   SEEK02 ' used by:    SCAN02
34          3)   SCAN02 ' scan grid of (vary) model parameters
35
36      D)  Construct the necessary sparse matrices,    b=Ax
37          1)   ARRANG ' construct pointers for sparse matrix ' A
38          2)   ASMBL  ' assemble the sparse matrices,   A,b
39          3)   SCATTR ' (psi,delta) for a single experiment.
40          4)   FORWRD ' (Rs, Rp, dRs, dRp) ' direct or forward problem.
41          5)   ASMBL0 ' similar to the above, |g| only, no Jacobian.
42          6)   SCATT0 '
43          7)   FORWR0 '
44
45      E)  Calculate the complex dielectric function, effective media.
46          1)   DIEFCN ' pass parameters to constituent media
47          2)   DIEEMA ' effective medium approximation
48          3)   DIEMAD ' calculate local partial derivatives
49          4)   DIELMN ' discern constituent media elements
50
51      F)  Calculate the complex dielectric function, constituent media.
52          1)   DIEL01 ' vacuum
53          2)   DIEL02 ' air
54          3)   DIEL03 ' Silicon
55          4)   DIEL04 ' Silicon amorphous
56          5)   DIEL05 ' Silicon Dioxide (glass)
57          6)   DIEL06 ' Silicon Nitride (non-crystalline)
58          7)   DIEL07 ' Germanium
59          8)   DIEL08 ' Gallium Arsenide
60          9)   DIEL09 ' Aluminum Gallium Arsenide   (x ' stoichiometry)
61         10)   DIEL10 ' Gallium Arsenide Oxides
62         11)   DIEL11 ' Arsenic amorphous
63         12)   DIEL12 ' GaP
64         13)   DIEL13 ' GaSb
65         14)   DIEL14 ' InAs
66         15)   DIEL15 ' InP
67         16)   DIEL16 ' InSb
68         17)   DIEL17 ' AlSb
69         18)   DIEL18 '
70         19)   DIEL19 '
71         20)   DIEL20 '
```

6.1 Named COMMON and BLOCK DATA Statements

6.1.1 IOUNIT.

```
1       common / iounit / inn, iout, idat, isout, iplt, iscr
```

6.1.2 DEFNIT.

```
1  c     The configuration of each sample is:    (ambient/films/substrate).
2  c        each film    is characterized by:     widths, mixture.
3  c        each mixture is characterized by:
4  c                     elements + volume fractions, parameters.
5  c     The configuration of each experiment/measurement is:
6  c         (psi,delta| wavelength, incident angle,  ambient,sample).
7
8  c     Note:   the ambient mixture is NOT allowed to undergo variation.
9  c             This imposition is accounted within:   nrowsf.
10 c             Further, the constraint of:    1 = sum: f
11 c             has also been accounted within:    rowsf.
12
13 c     nsampl ' number of distinct sample configurations, films/substrate.
14 c     nfilms ' number of films layers allowed atop a substrate.
15 c     nparms ' number of distinct parameters per mixture/ambient.
16 c     nlmnts ' number of distinct elemental dielectric functions, e(w).
17 c     nbient ' number of distinct ambients allowed atop a sample.
18 c     nwaves ' number of wavelengths incident on sample.
19 c     nanglx ' number of incident angles per wavelength.
20 c     nrpeat ' number of repeats of an experiment.
21
22       parameter (nsampl=  6)
23       parameter (nfilms= 10)
24       parameter (nparms=  4)
25       parameter (nlmnts= 20)                    ! see:   DIELMN
26       parameter (nbient=  1)
27       parameter (nwaves=100)
28       parameter (nanglx= 10)
29       parameter (nrpeat=  1)
30
31       parameter (nrows = nfilms*2+1)            !     (z,e),(e)  'local
32                                                  ! (z),(f,p),(p)  'local
33       parameter (nrowsf= nfilms+(nfilms+1)*(nlmnts-1+nparms)+nparms)
34       parameter (nmixtr= nsampl*(nfilms+2))     !  mixtures  'global
35       parameter (nfilmz= nsampl* nfilms)        !  z         'global
36       parameter (nrowss= nsampl*nrowsf+(nbient-1)*nparms)  !  'global
37       parameter (nexpts= nsampl*nbient*nrpeat*nwaves*nanglx)
38       parameter (mrowss= nexpts*2)              ! (psi,delta)
39
40       complex   cmplx, conjg, sqrtt
```

6.1.3 FILMMM.

```
1       real    widths(nsampl*nfilms)    ! distinct widths of films
2       real    uwidth(nsampl*nfilms)    ! uncertainty in widths
3       integer lwidth(nsampl*nfilms)    ! variation   in widths
4
5       integer mmlmnt(nmixtr)           ! number of fractions per mixture
6       integer kklmnt(nmixtr)           ! offset location pointer
7       integer mvlmnt(nmixtr)           ! number of fractions varying in mix
8
9       integer iilmnt(nmixtr*nlmnts)    ! composition of mixture
10      real    fflmnt(nmixtr*nlmnts)    ! volume fraction
11      real    uflmnt(nmixtr*nlmnts)    ! uncertainty
12      integer lflmnt(nmixtr*nlmnts)    ! (vary/froz)
13
14      integer iiparm((nmixtr+nbient)*nparms)     ! parameters
15      integer miparm( nmixtr+nbient)             ! number of parameters
16      integer kiparm( nmixtr+nbient)             ! offset
17      integer jiparm((nmixtr+nbient)*nparms)     ! distribution
18
19      real    rrparm((nmixtr+nbient)*nparms)     ! parameters
20      real    urparm((nmixtr+nbient)*nparms)     ! uncertainty
21      integer lrparm((nmixtr+nbient)*nparms)     ! (vary/froz)
22      integer mrparm( nmixtr+nbient)             ! number of parameters
23      integer mvparm( nmixtr+nbient)             ! number of parameters varying
24      integer krparm( nmixtr+nbient)             ! offset
25      integer jrparm((nmixtr+nbient)*nparms)     ! distribution
26
27      integer mixmbn(nbient)                     ! ambient
28
29      integer mmfilm(nsampl)                     ! number of films on sample
30      integer kkfilm(nsampl)                     ! offset
31      integer iifilm(nsampl*nrows)               ! (z,mixture),(mixture)
32
33      common / filmmm / widths, uwidth, lwidth, mfilmz,
34     &                  mmlmnt, kklmnt, mvlmnt, mmixtr,
35     &                  iilmnt, fflmnt, uflmnt, lflmnt, mlmnts,
36     &                  iiparm, miparm,
37     &                  rrparm, urparm, lrparm, mrparm,
38     &                  miparm, kiparm, jiparm,
39     &                  mrparm, mvparm, krparm, jrparm,
40     &                  mixmbn, mbient,
41     &                  mmfilm, kkfilm, iifilm, msampl
```

6.1.4 XPRMNT.

```
1       real  psiiis(nexpts), psiiiu(nexpts)
2       real  deltas(nexpts), deltau(nexpts)
3       real  angles(nexpts), angleu(nexpts)
4       real  wavlns(nexpts), wavlnu(nexpts)
5
6       integer  mmbent(nsampl)
7       integer  iibent(nsampl*nbient)
8       integer  mmpeat(nsampl*nbient)
9       integer  mmexpt(nsampl*nbient*nrpeat)
10
11      common / xprmnt / psiiis, psiiiu, deltas, deltau,
12     &                  angles, angleu, wavlns, wavlnu,
13     &                  mmbent, iibent, mmpeat, mmexpt, mexpts
```

6.1.5 FILMSS.

```
1  c     Each sample is characterized by the film/substrate:
2  c       geometry:    air / z(1) / ... / z(nfilms) / substrate
3  c     parameters:    three data (z,f,p) for each film,     may vary.
4  c                    two   data ( ,f,p) for the substrate, may vary.
5  c                    one   data ( ,f,p) for the air,   may not vary.
6
7  c     epsilon = electric permittivity ~ dielectric function ~ 1
8  c     sigma   = specific conductivity                        ~ 0
9  c     mu      = magnetic permeability                        ~ 1
10 c     omega   = angular frequency
11 c     die_r   = epsilon * mu
12 c     die_i   = 4*pi * sigma * mu / omega
13
14       complex  die(nfilms+1)
15       real     zzz(nfilms  ), air
16       integer      mfilm
17
18       common / filmss / die, zzz, air, mfilm
```

6.1.6 ARRAYD.

```
1       integer   mixflm(nfilms+2)
2       logical   firstx(nmixtr)
3
4       complex   dielec(nmixtr)          !  (e) ' dielectric function
5       complex   dielew(nmixtr)          ! d(e) /d (energy ' eV)
6       complex   dielff(nlmnts,nmixtr)   ! d(e) /d (volume fraction)
7       complex   dielpp(nparms,nmixtr)   ! d(e) /d (parameter mixture)
8       complex   dielpa(nparms,nmixtr)   ! d(e) /d (parameter ambient)
9
10      common / arrayd / dielec, dielew, dielff, dielpp, dielpa,
11     &                  mixflm, firstx
```

6.1.7 ARRAYS.

```
1  c*    parameter (nnjaaa = mrowss*nrowsf)          ! estimated need
2        parameter (nnjaaa = 2000000)                ! convenience
3
4        integer  ja(nnjaaa), ia(mrowss+1)
5        real     aa(nnjaaa), bb(mrowss  ), cc(mrowss)
6        real     xx(nrowss)
7
8        integer  iptu(nrowss), iptv(nrowss), iptw(nrowss), iptx(nrowss)
9        logical  llnorm
10
11       common / arrays / aa, bb, cc, xx, ja, ia,
12      &                  iptu, iptv, iptw, iptx,
13      &                  meqns, mvary, mfroz, llnorm
```

6.1.8 ABCDEF.

```
1       real      a(nrowsf*2), b(2), c(4)
2
3       common / abcdef / a,b,c
```

6.1.9 RSTACK.

```
1  c     Accessed or used by:    FORWRD, SCATTR
2
3  c     wavenumber ~ 2*pi/wavelength,          wavelength ~ nano-meters
4
5  c     d/dw (total   derivative wrt frequency ) =
6  c     d/dq (partial derivative wrt wavenumber) * (dq/dw) +
7  c     d/de (partial  wrt dielectric function ) * (de/dw)
8
9        complex   Rs,          Rp             ! reflection coeff in ambient
10       complex  dRs(nrows),  dRp(nrows)      ! Jacobians
11       complex  dRsa,        dRpa            ! d/d(angle of incidence)
12       complex  dRsq,        dRpq            ! d/d(wavenumber)
13
14       common / rstack / Rs,Rp,  dRs,dRp,  dRsa,dRpa,  dRsq,dRpq
```

6.1.10 WSTACK.

```
1       parameter (naat=(nrowss*(nrowss+1))/2)     ! upper triang + diag
2       real      aat (naat)                       ! A(T)*A
3       real      aats(nrowss)                     ! SCALJJ
4       integer   ipvt(nrowss)                     ! LINPACK workspace
5
6       real      p(mrowss), u(mrowss)             ! SCALII, CGN
7       real      v(nrowss), w(nrowss), xw(nrowss), se(nrowss)
8
9       common / wstack / aat,aats,ipvt,    p,u,  v,w,xw,se
```

6.1.11 ELMNTS.

```
1  c     Discern which among the files containing spectroscopic data
2  c     that describes the:   (dielectric functions, conductivities) (w),
3  c     have been accessed or elsewise opened.
4
5        character*40  subdir              ! sub-directory
6        logical       llmnts(nlmnts)
7
8        common / elmnts / llmnts, subdir
```

6.1.12 NESTO1.

```
1  c     Nest of do-loops associated with the multi-parameter grid scan.
2  c     Purpose:    SCANO2   passes arguments onto:   SEEKO2
3
4        integer   iiii(nrowss), iii2(nrowss)
5        real      pppp(nrowss), ppp1(nrowss), ppp2(nrowss), ppp3(nrowss)
6        real      psav(nrowss)
7
8        common / nesto1 / pppp, ppp1, ppp2, ppp3, psav,
9       &                  iiii,       iii2
```

6.1.13 HANDYY.

```
1  c     Convenient/handy set of data information.
2
3  c*    data      pi / 3.14159265E+00 /
4  c*    data    cccc / 2.99792458E+17 /  ! speed of light,  (nano-m/sec)
5  c*    data  wavlev / 1239.852056    /  ! hc, (nano-meters)-(electron-volts)
6
7        real              pi, cccc, wavlev
8        common / handyy / pi, cccc, wavlev
```

6.1.14 BLKDAT.FOR

```
1         block data  blkdat
2
3         include  'iounit.'
4         include  'handyy.'
5
6         data  inn,iout,idat,isout,iplt,iscr / 5,6,7,8,9,10 /
7
8         data      pi / 3.14159265E+00 /
9         data    cccc / 2.99792458E+17 /  ! speed of light,  (nano-m/sec)
10        data  wavlev / 1239.852056    /  ! (nano-meters)-(electron-volts)
11
12        end
13
```

6.2 Source Programs

6.2.1 MAIN.FOR

```
1       program   main
2       include  'iounit.'
3
4       call fileop                      ! open input data files
5       call inpdat                      ! read input data
6       call arrang                      ! set up pointers
7       it1 = istime (i)                 ! CPU time (milli-sec)
8
9       write (iout,101)
10      read  (inn,*,err=21,end=21)  ichoic
11      write (iout,102)             ichoic
12
13      goto (1,2,3), ichoic
14      goto 22
15
16     1 call scatos                     ! model ---> experiment
17          goto 11
18     2 call seeko1                     ! single search for minimum
19          goto 11
20     3 call scano2                     ! scan grid   (     vary)
21          goto 11
22
23    11 it2 = iftime (i)                ! CPU time (milli-sec)
24       it  = (it2-it1)/10              ! CPU time (centi-sec) elapsed
25       its = 100                       ! clock units / second
26       itm = its*60                    !             / minute
27       ith = itm*60                    !             / hour
28       itd = ith*24                    !             / day
29
30       id = it/itd                     ! days
31       it = it-id*itd
32       ih = it/ith                     ! hours
33       it = it-ih*ith
34       im = it/itm                     ! minutes
35       it = it-im*itm
36       is = it/its                     ! seconds
37       it = it-is*its                  ! remaining time in:   centi-seconds
38
39       if (id.ne.0) then
40           write (iout,111)  it, is, im, ih, id
41       else if (ih.ne.0) then
42           write (iout,111)  it, is, im, ih
43       else if (im.ne.0) then
44           write (iout,111)  it, is, im
45       else if (is.ne.0) then
46           write (iout,111)  it, is
47       else
48           write (iout,111)  it
```

```
49        end if
50
51        close (iout)
52        stop
53
54     21 write (iout,121)                ! error, opening file
55        stop
56     22 write (iout,122)                ! error, option choice
57        stop
58
59    101 format (/' Enter:   option                          '
60       &        /  12x, '1,  forward problems, plots, ... '
61       &        /  12x, '2,  search      (vary)            '
62       &        /  12x, '3,  search grid (vary)            ')
63    102 format ( ' option = ', i3 /)
64
65    111 format (/' elapsed cpu-time = ', i3, '  centi-seconds', :
66       &        /'                     + ', i3, '  seconds      ', :
67       &        /'                     + ', i3, '  minutes      ', :
68       &        /'                     + ', i3, '  hours        ', :
69       &        /'                     + ', i3, '  days         ')
70
71    121 format (/' oops,  unable to access input data file   ')
72    122 format (/' oops,  inconsistent with available options')
73
74        end
```

6.2.2 FILEOP.FOR

```
1         subroutine fileop
2         include 'iounit.'
3
4
5         open ( inn,file='x.inn',status='old',readonly,shared)          ! 5
6         open (idat,file='x.dat',status='old',readonly,shared)          ! 7
7
8       1 open (iout,file='x.out',status='old',disp='delete',err=2)      ! 6
9         close(iout)
10        goto 1
11      2 open (iout,file='x.out',status='new')
12
13      3 open (isout,file='x.sout',status='old',disp='delete',err=4)    ! 8
14        close(isout)
15        goto 3
16      4 continue      ! open (isout,file='x.sout',status='new')
17
18      5 open (iplt,file='x.plot',status='old',disp='delete',err=6)     ! 9
19        close(iplt)
20        goto 5
21      6 open (iplt,file='x.plot',status='new')
22
23  c*  7 open (iscr,file='x.scr',status='old',disp='delete',err=8)      ! 10
24  c*    close(iscr)
25  c*    goto 7
26  c*  8 continue      ! open (iscr,file='x.scr',status='new')
27
28        return
29        end
```

6.2.3 INPDAT.FOR

```
      subroutine inpdat
      include  'iounit.'
      include  'defnit.'
      include  'filmmm.'
      include  'xprmnt.'
      include  'handyy.'                ! pi,cccc,wavlev
      include  'elmnts.'                ! constituent media

      logical   lvary                   ! test whether something varies

      raddeg = pi/180.0                 ! radians/degree

c     ================================================================
c     Sub-directory associated with database of constituent media.

      write (iout,1001)
      read  (idat,1002)  subdir
      write (iout,1003)  subdir
      write (iout,1004)
      call ljchar (subdir)              ! left-justify character string

      do i=1,nlmnts
          llmnts(i) = .false.           ! initialize
      end do

c     ================================================================
c     The dielectric function of each distinct element is
c     already available and provided via the subroutines.

c     Distinct widths

      call hhline (idat,iout)           ! line of connected hyphens

      write (iout,1011)
      read  (idat,   *)  mfilmz
      write (iout,1012)  mfilmz
      if (mfilmz.lt.0 .or. mfilmz.gt.nfilmz) then
          write (iout,1013)
          stop
      end if
      if (mfilmz.ne.0) then
          write (iout,1014)
          do i=1,mfilmz                          ! distinct widths
              read  (idat,   *)  j, z,zu, ivary
              write (iout,1015)  j, z,zu, ivary
              if (j.ne.i     .or.
     &            z.lt.0.0   .or. zu.lt.0.0  .or.
     &            ivary.lt.0 .or. ivary.gt.2     ) then
                  write (iout,1016)
                  stop
```

```
51                end if
52                if (ivary.eq.1 .and. zu.eq.0.0) then
53                    write (iout,1017)
54                    stop
55                end if
56                widths(i) = z
57                uwidth(i) = zu
58                lwidth(i) = ivary
59            end do
60        end if
61        write (iout,1004)
62  c     ================================================================
63  c     Distinct supplementary parameters ' integer
64
65        call hhline (idat,iout)          ! line of connected hyphens
66
67        write (iout,1021)
68        read  (idat,   *)  mipars
69        write (iout,1022)  mipars
70        if (mipars .lt. 0    .or.
71       &    mipars .gt. (nmixtr+nbient)*nparms) then
72            write (iout,1023)
73            stop
74        end if
75        if (mipars.ne.0) then
76            write (iout,1024)  mipars
77            do i=1,mipars
78                read  (idat,   *)  j, iparm
79                write (iout,1025)  j, iparm
80                if (i.ne.j) then
81                    write (iout,1026)
82                    stop
83                end if
84                iiparm(i) = iparm
85            end do
86        end if       ! iiparm
87        write (iout,1004)
88  c     ================================================================
89  c     Distinct supplementary parameters ' floating-point
90
91        call hhline (idat,iout)          ! line of connected hyphens
92
93        write (iout,1031)
94        read  (idat,   *)  mrpars
95        write (iout,1032)  mrpars
96        if (mrpars .lt. 0    .or.
97       &    mrpars .gt. (nmixtr+nbient)*nparms) then
98            write (iout,1033)
99            stop
100       end if
101       if (mrpars.ne.0) then
102           write (iout,1034)  mrpars
103           do i=1,mrpars
```

```
104               read  (idat,   *)  j, rparm, uparm, ivary
105               write (iout,1035)  j, rparm, uparm, ivary
106               if (i.ne.j     .or. uparm.lt.0.0 .or.
107      &            ivary.lt.0 .or. ivary.gt.2       ) then
108                   write (iout,1036)
109                   stop
110               end if
111               if (ivary.eq.1 .and. uparm.eq.0.0) then
112                   write (iout,1037)
113                   stop
114               end if
115               rrparm(i) = rparm
116               urparm(i) = uparm
117               lrparm(i) = ivary
118           end do
119       end if       ! rrparm
120       write (iout,1004)
121 c     ================================================================
122 c     Distinct effective media ' mixtures
123
124       call hhline (idat,iout)          ! line of connected hyphens
125
126       write (iout,1111)
127       read  (idat,   *)  mmixtr        ! number of distinct mixtures
128       write (iout,1112)  mmixtr
129       write (iout,1004)
130       if (mmixtr.lt.1 .or. mmixtr.gt.nmixtr) then
131           write (iout,1113)
132           stop
133       end if
134       mlmnts = 0                       ! total number of elements
135       kk = 0                           ! index elements in the mixture
136       ki = 0
137       kr = 0
138       do m=1,mmixtr                    ! distinct mixtures
139           write (iout,1114)  m
140           read  (idat,   *)  mlmnt, mipar, mrpar
141           write (iout,1115)  mlmnt, mipar, mrpar
142           if (mlmnt.lt.0 .or. mlmnt.gt.nlmnts .or.
143      &        mipar.lt.0 .or. mipar.gt.nparms .or.
144      &        mrpar.lt.0 .or. mrpar.gt.nparms .or.
145      &        (mlmnt+mrpar .eq. 0)               ) then
146               write (iout,1116)
147               stop
148           end if
149 c         ------------------------------------------------
150           mv = 0                                          ! vary count
151           mmlmnt(m) = mlmnt          ! number of elements in mixture
152           kklmnt(m) = kk                                  ! offset
153           if (mlmnt.ne.0) then
154               mlmnts = mlmnts+mlmnt                       ! sum elements
155               sv = 0.0                                    !     vary
156               su = 0.0                                    !     froz
```

```
157                 write (iout,1121)
158                 do i=1,mlmnt                                    ! composition
159                     read  (idat,   *)  j, lmnt, frac, ufrac, ivary
160                     write (iout,1122)  j, lmnt, frac, ufrac, ivary
161                     if (    j.ne.i    .or.
162        &                 lmnt.lt.1    .or.  lmnt.gt.nlmnts .or.
163        &                 frac.lt.-0.5 .or.  frac.gt.1.5    .or.
164        &                ufrac.lt.0.0  .or. ufrac.gt.1.0    .or.
165        &                ivary.lt.0    .or. ivary.gt.2          ) then
166                         write (iout,1123)
167                         stop
168                     end if
169                     if (ivary.eq.1 .and. ufrac.eq.0.0) then
170                         write (iout,1124)
171                         stop
172                     end if
173                     if (i.ne.1) then                            ! redundancy
174                         do j=2,i
175                             if (lmnt .eq. iilmnt(kk+2-j)) then
176                                 write (iout,1131)
177                                 stop
178                             end if
179                         end do
180                     end if
181                     kk = kk+1                                   ! index
182                     iilmnt(kk) = lmnt                           ! distinct element
183                     fflmnt(kk) =  frac                          ! volume fraction
184                     uflmnt(kk) = ufrac                          ! uncertainty
185                     lflmnt(kk) = ivary                          ! vary
186                     if (lflmnt(kk).eq.1) then                   ! vary
187                         mv = mv+1
188                         sv = sv+frac
189                     else                                        ! froz
190                         su = su+frac
191                     end if
192                 end do          ! elements
193
194                 sum = sv+su                                     ! vary+froz
195                 if (sum.le.0.9 .or. sum.gt.1.1) then            ! consistency check
196                     write (iout,1132)
197                     stop
198                 end if
199
200                 if (mv.eq.0) then                               ! froz
201                     if (abs(1.0-su) .gt. 1.0E-6) then           ! renormalize
202                         write (iout,1133)
203                         j = kk-mlmnt
204                         do i=1,mlmnt
205                             j = j+1
206                             fflmnt(j) = fflmnt(j)/su            ! volume fraction
207                         end do
208                     end if
209                 else if (mv.eq.1) then                          ! inconsistency
```

```
                write (iout,1134)
                stop
            else                                        ! mv > 1
                j = kk-mlmnt                            ! convenience
                if (su .gt. 1.0) then                   ! consistency check
                    write (iout,1135)
                    stop
                end if
                if (sv .eq. 0.0) then                   ! renorm equally
                    write (iout,1136)
                    sv = (1.0-su)/float(mv)
                    do i=1,mlmnt
                        j = j+1
                        if (lflmnt(j).eq.1) then
                            fflmnt(j) = sv              ! volume fraction
                        end if
                    end do
                else if (abs(1.0-sum) .gt. 1.0E-6) then ! renorm linearly
                    write (iout,1137)
                    v = (1.0-su)/sv                     ! vary(new)
                    do i=1,mlmnt
                        j = j+1
                        if (lflmnt(j).eq.1) then
                            fflmnt(j) = v*fflmnt(j)     ! volume fraction
                        end if
                    end do
                end if
            end if          ! element test, vary+froz
        end if              ! composition of elements
        mvlmnt(m) = mv
c       -------------------------------------------------
        miparm(m) = mipar                               ! number
        kiparm(m) = ki                                  ! offset
        if (mipar.ne.0) then
            write (iout,1141)  mipar
            do i=1,mipar
                read  (idat,   *)  j,ip                 ! pointers
                write (iout,1142)  j,ip
                if (j.ne.i .or. ip.lt.1 .or. ip.gt.mipars) then
                    write (iout,1143)
                    stop
                end if
                if (i.gt.1) then                        ! redundancy
                    do im=2,i
                        if (ip .eq. jiparm(ki+2-im)) then
                            write (iout,1144)
                            stop
                        end if
                    end do
                end if
                ki = ki+1
                jiparm(ki) = ip                         ! distribution
            end do
```

```
263             end if
264   c         ------------------------------------------------
265             mv = 0
266             mrparm(m) = mrpar                                   ! number
267             krparm(m) = kr                                      ! offset
268             if (mrpar.ne.0) then
269                 write (iout,1151)  mrpar
270                 do i=1,mrpar
271                     read  (idat,   *)  j,ip                     ! pointer
272                     write (iout,1152)  j,ip
273                     if (j.ne.i .or. ip.lt.1 .or. ip.gt.mrpars) then
274                         write (iout,1153)
275                         stop
276                     end if
277                     if (i.gt.1) then                            ! redundancy
278                         do im=2,i
279                             if (ip .eq. jrparm(kr+2-im)) then
280                                 write (iout,1154)
281                                 stop
282                             end if
283                         end do
284                     end if
285                     kr = kr+1
286                     jrparm(kr) = ip                             ! distribution
287                     if (lrparm(ip).eq.1)  mv=mv+1
288                 end do
289             end if
290             mvparm(m) = mv
291   c         ------------------------------------------------
292             write (iout,1004)
293         end do       ! mixtures
294   c     ================================================================
295   c     Ambients & external parameters
296
297         call hhline (idat,iout)          ! line of connected hyphens
298
299         write (iout,1211)
300         read  (idat,   *)  mbient
301         write (iout,1212)  mbient
302         if (mbient.lt.1 .or. mbient.gt.nbient) then
303             write (iout,1213)
304             stop
305         end if
306         write (iout,1214)
307
308         do m=1,mbient
309             read  (idat,   *)  j, imix, mipar, mrpar            ! additional
310             write (iout,1215)  j, imix, mipar, mrpar            !   parameters
311             if (    j.ne.m  .or.
312        &        imix .lt.1  .or.   imix.gt.mmixtr .or.
313        &        mipar.lt.0  .or.  mipar.gt.nparms .or.
314        &        mrpar.lt.0  .or.  mrpar.gt.nparms     ) then
315                 write (iout,1216)
```

```
318             stop
317         end if
318
319         if (mvlmnt(imix).ne.0  .or.  mvparm(imix).ne.0) then
320             write (iout,1217)
321             stop
322         end if
323
324         mixmbn(m) = imix   ! specify the mixture composing the ambient
325         mm = mmixtr+m                          ! offset
326
327         miparm(mm) = mipar                     ! number
328         kiparm(mm) = ki                        ! offset
329         if (mipar.ne.0) then                   ! parameters
330             write (iout,1221)  mipar
331             do i=1,mipar
332                 read  (idat,   *)  j,ip        ! point to:   iiparm
333                 write (iout,1222)  j,ip
334                 if (j.ne.i .or. ip.lt.1 .or. ip.gt.mipars) then
335                     write (iout,1223)
336                     stop
337                 end if
338                 if (i.ne.1) then               ! redundancy
339                     do j=2,i
340                         if (ip .eq. jiparm(ki+2-j)) then
341                             write (iout,1224)
342                             stop
343                         end if
344                     end do
345                 end if
346                 ki = ki+1
347                 jiparm(ki) = ip                ! distribution
348             end do
349         end if
350
351         mv = 0
352         mrparm(mm) = mrpar                     ! number
353         krparm(mm) = kr                        ! offset
354         if (mrpar.ne.0) then                   ! parameter
355             write (iout,1231)  mrpar
356             do i=1,mrpar
357                 read  (idat,   *)  j,ip        ! point to:   rrparm
358                 write (iout,1232)  j,ip
359                 if (j.ne.i .or. ip.lt.1 .or. ip.gt.mrpars) then
360                     write (iout,1233)
361                     stop
362                 end if
363                 if (i.ne.1) then               ! redundancy
364                     do j=2,i
365                         if (ip .eq. jrparm(kr+2-j)) then
366                             write (iout,1234)
367                             stop
368                         end if
```

```
                    end do
                end if
                kr = kr+1
                jrparm(kr) = ip                 ! distribution
                if (lrparm(ip).eq.1)  mv=mv+1
            end do
        end if
        mvparm(mm) = mv

    end do       ! ambients + external parameters
    write (iout,1004)
c   ================================================================
c   Construction of sample configuration ' layered structure.

    call hhline (idat,iout)          ! line of connected hyphens

    write (iout,1311)
    read  (idat,   *)  msampl
    write (iout,1312)  msampl
    write (iout,1004)
    if (msampl.lt.1 .or. msampl.gt.nsampl) then
        write (iout,1313)
        stop
    end if

    k = 0
    do is=1,msampl                              ! films/substrate
        write (iout,1321)  is
        read  (idat,   *)  mfilm
        write (iout,1322)  mfilm,is
        if (mfilm.lt.0 .or. mfilm.gt.nfilms) then
            write (iout,1323)
            stop
        end if
        mmfilm(is) = mfilm
        kkfilm(is) = k                          ! offset
        mfilms = mfilm+1                        ! films,substrate
        lvary = .false.                         ! something should vary

        write (iout,1331)
        do m=1,mfilms
            if (m.eq.mfilms) then
                read  (idat,   *)  j, imixtr
                write (iout,1332)  j, imixtr
            else
                read  (idat,   *)  j, imixtr, iwidth
                write (iout,1333)  j, imixtr, iwidth
                if (iwidth.lt.1 .or. iwidth.gt.mfilmz) then
                    write (iout,1334)
                    stop
                end if
                k = k+1
                iifilm(k) = iwidth
```

```
422                 lvary = lvary .or. (lwidth(iwidth).eq.1) ! test vary
423             end if
424             if (imixtr.lt.1 .or. imixtr.gt.mmixtr) then
425                 write (iout,1335)
426                 stop
427             end if
428             if (j.ne.m) then                    ! sequential indexing
429                 write (iout,1336)
430                 stop
431             end if
432             if (m.ne.1) then                    ! adjacency of mixtures
433                 if (m.eq.mfilms) then
434                     kback = k                   ! previous mixture
435                 else
436                     kback = k-1                 ! previous mixture
437                 end if
438                 if (imixtr .eq. iifilm(kback)) then
439                     mv = mvlmnt(imixtr)+mvparm(imixtr)
440                     if (mv .ne. 0) then
441                         write (iout,1337)
442                         stop
443                     end if
444                 end if
445             end if
446             k = k+1
447             iifilm(k) = imixtr
448             lvary = lvary .or. (mvlmnt(imixtr).ne.0)
449      &                    .or. (mvparm(imixtr).ne.0)
450         end do   ! films
451
452 c*      if (.not.lvary) then          ! something should vary, however
453 c*          write (iout,1338)         !    in SCAT02   we allow (mv=0)
454 c*          stop                      !    so as to discern:
455 c*      end if                        !    d() /d(incident angle)
456
457         write (iout,1004)
458     end do       ! sample
459 c   ================================================================
460 c   Test compactness of domain variables.
461
462 c   ================================================================
463 c   Measurement data of ellipsometric angles by instrumentation.
464
465     call hhline (idat,iout)           ! line of connected hyphens
466
467     ias  = 0                                    !             ambient,sample
468     iras = 0                                    !      repeat,ambient,sample
469     i    = 0                                    ! expt,repeat,ambient,sample
470     ii   = 0                                    ! overflow
471     nexpt = nwaves*nanglx                       ! convenience
472     do is=1,msampl
473         write (iout,1411)  is
474         read  (idat,   *)  mbien                ! number of ambients
```

```
475             write (iout,1412)  mbien,is
476             if (mbien.lt.1 .or. mbien.gt.mbient) then
477                 write (iout,1413)
478                 stop
479             end if
480             mmbent(is) = mbien                      ! number of ambients
481             do mbn=1,mbien                          ! distinct ambients
482                 write (iout,1421)
483                 read  (idat,   *)  mrpeat, imbien
484                 write (iout,1422)  mrpeat, imbien
485                 if (mrpeat.lt.1 .or. mrpeat.gt.nrpeat .or.
486        &            imbien.lt.1 .or. imbien.gt.mbient     ) then
487                     write (iout,1423)
488                     stop
489                 end if
490                 if (mbn .ne. 1) then                ! redundancy
491                     do j=2,mbn
492                         if (imbien .eq. iibent(ias+2-j)) then
493                             write (iout,1424)
494                             stop
495                         end if
496                     end do
497                 end if
498
499                 ias = ias+1
500                 iibent(ias) = imbien                ! specify ambient
501                 mmpeat(ias) = mrpeat                ! number of repeats
502                 do irpeat=1,mrpeat
503                     write (iout,1431)
504                     read  (idat,   *)  mexpt
505                     write (iout,1432)  mexpt
506                     if (mexpt.lt.1 .or. mexpt.gt.nexpt) then
507                         write (iout,1433)  nexpt
508                         stop
509                     end if
510                     iras = iras+1
511                     mmexpt(iras) = mexpt        ! number of experiments
512                     write (iout,1441)
513                     do ixpt=1,mexpt             ! (wave, incident angle)
514
515                         read  (idat,   *)  j, wave,angl,  delta,psi
516                         write (iout,1442)  j, wave,angl,  delta,psi
517                         read  (idat,   *)     wavu,angu,  delu,psiu
518                         write (iout,1443)     wavu,angu,  delu,psiu
519
520                         if (delta .lt. 0.0) then    ! [0, 360)
521                             delta = delta + 360.0
522                         else if (delta .ge. 360.0) then
523                             delta = delta - 360.0
524                         end if
525 c       ---------------------------------------------------------------
526 c       The above measurement data is found from instrumentation.
527 c       Azzam and Horowitz assume:   n-ik ---> (Delta,psi)  (Nebraska)
```

```
c               Here, we assume:   n+ik ---> (Delta,psi)  (physics )
c     The convention affects only Delta.
c     The relationship is:   (physics) <--- conjg (Nebraska)

                      if (delta .ne. 0.0) then
                          delta = 360.0 - delta
                      end if
c     ------------------------------------------------------------------

                      if (wave .lt. 0.0) then              ! wavelength
                          if (wave .lt. -1240.0 .or.       ! nm
     &                        wave .gt.  -200.0 .or.
     &                        wavu .le.     0.0 .or.
     &                        wavu .gt.  1000.0     ) then
                              write (iout,1444)
                              stop
                          end if
c*                        wavu = -wavu/wave                ! > 0
c*                        wave = -wavlev/wave              ! eV
c*                        wavu =  wave*wavu                ! eV
                      else if (wave .lt. 1.0 .or.          ! energy
     &                         wave .gt. 6.0 .or.
     &                         wavu .lt. 0.0 .or.
     &                         wavu .gt. 6.0     ) then
                          write (iout,1444)
                          stop
                      end if

                      if (angl .lt.0.0  .or. angl .gt.  90.0 .or.
     &                    angu .lt.0.0  .or. angu .gt.   1.0 .or.
     &                    psi  .lt.0.0  .or. psi  .gt.  90.0 .or.
     &                    psiu .lt.0.0  .or. psiu .gt.1000.0 .or.
     &                    delta.lt.0.0  .or. delta.ge. 360.0 .or.
     &                    delu .lt.0.0  .or. delu .gt.1000.0 ) then
                          write (iout,1444)
                          stop
                      end if

                      angu = amax1 (angu, 0.01)  ! convenenience
                      delu = amax1 (delu, 0.01)
                      psiu = amax1 (psiu, 0.01)

                      angl  = angl *raddeg       ! radian <--- degree
                      angu  = angu *raddeg
                      psi   = psi  *raddeg
                      psiu  = psiu *raddeg
                      delta = delta*raddeg
                      delu  = delu *raddeg

                      if (i.lt.nexpts) then
                          i = i+1
                          wavlns(i) = wave       ! -nm, +eV
                          wavlnu(i) = wavu
```

```
581                             angles(i) = angl        ! radians
582                             angleu(i) = angu
583                             psiiis(i) = psi
584                             psiiiu(i) = psiu
585                             deltas(i) = delta
586                             deltau(i) = delu
587                         else                        ! overflow
588                             ii = ii+1
589                         end if
590                     end do     ! experiment, measurement
591                     write (iout,1004)               ! blank line
592                 end do         ! repeat
593 c               ------------------------------------------
594             end do   ! ambient
595         end do       ! sample
596
597         if (ii.ne.0) then
598             i = i+ii
599             write (iout,1445)  nexpts,i
600             stop
601         end if
602         mexpts = i              ! number of measured data
603
604         return
605
606   1001 format ( ' Enter:   sub-directory     for constituent media')
607   1002 format (a)
608   1003 format ( '          sub-directory ' ', a)
609   1004 format ( ' ')
610
611   1011 format ( ' Enter:     mfilmz ' number of distinct widths')
612   1012 format ( 1x, i4, 7x, 'mfilmz ' number of distinct widths')
613   1013 format (/' ... oops,  inconsistency')
614   1014 format ( ' Enter:   ',      24x, 'i,z,zu,ivary')
615   1015 format ( 1x, i4, 2f10.3, i5, 4x, 'i,z,zu,ivary')
616   1016 format (/' ... oops,  inconsistency')
617   1017 format (/' ... oops,  let:      zu > 0.0')
618
619   1021 format ( ' Enter:     mipars ' number of parameters (integer)')
620   1022 format ( 1x, i4, 7x, 'mipars ' number of parameters (integer)')
621   1023 format (/' ... oops,  inconsistency')
622   1024 format ( ' Enter:   ',   13x, 'j, iparm     (# items=',i2,')')
623   1025 format ( 1x, i4, 4x, i10, 4x, 'j, iparm')
624   1026 format (/' ... oops,  inconsistency')
625
626   1031 format ( ' Enter:     mrpars ' number of parameters ',
627      &                              '(floating-point)')
628   1032 format ( 1x, i4, 7x, 'mrpars ' number of parameters ',
629      &                              '(floating-point)')
630   1033 format (/' ... oops,  inconsistency')
631   1034 format ( ' Enter:    j, rparm, uparm, ivary   (# items=',i2,')')
632   1035 format ( 1x, i4, 3x, 1p2e13.5, 4x, i1, 4x,
633      &                     '(j,rparm,uparm,ivary)'   )
```

```
1036 format (/' ... oops,  inconsistency')
1037 format (/' ... oops,  let:       uparm =/= 0.0 ')


1111 format ( ' Enter:     mmixtr ' number of distinct mixtures')
1112 format ( 1x, i4, 7x, 'mmixtr ' number of distinct mixtures')
1113 format (/' ... oops,  inconsistency')
1114 format ( ' Enter:         mlmnt, mipar, mrpar    (mix #',i2,')')
1115 format ( 1x, 3i4, 3x,    'mlmnt, mipar, mrpar')
1116 format (/' ... oops,  inconsistency')

1121 format ( ' Enter:   ', 30x, 'j, lmnt, frac, ufrac, ivary')
1122 format ( 1x, i4, i5, 2f10.5, i5, 5x,
     &                          'j, lmnt, frac, ufrac, ivary')
1123 format (/' ... oops,  inconsistency')
1124 format (/' ... oops,  let:        ufrac > 0.0')

1131 format (/' ... oops,  redundancy among:  lmnt')
1132 format (/' ... oops,  sum of fractions not equal to unity')
1133 format (/' Note:      renormalizing frozen fractions,'
     &        /'           since the mixture in the layer is frozen')
1134 format (/' ... oops,  unable to vary one fraction, alone')
1135 format (/' ... oops,  constraint upon:   vary')
1136 format (/' Note:      renormalizing vary fractions, equally ')
1137 format (/' Note:      renormalizing vary fractions, linearly')

1141 format ( ' Enter:   ', 4x, 'j, iiparm    (# items =',i2,')')
1142 format ( 1x, i4, i5,   4x, 'j, iiparm')
1143 format (/' ... oops,  inconsistency')
1144 format (/' ... oops,  redundancy in:   iparm')

1151 format ( ' Enter:   ', 4x, 'j, irparm    (# items =',i2,')')
1152 format ( 1x, i4, i5,   4x, 'j, irparm')
1153 format (/' ... oops,  inconsistency')
1154 format (/' ... oops,  redundancy in:   irparm')


1211 format ( ' Enter:     mbient ' number of distinct ambients')
1212 format ( 1x, i4, 7x, 'mbient ' number of distinct ambients')
1213 format (/' ... oops,  inconsistency')
1214 format ( ' Enter:   ', 14x, 'j, imix, mipar, mrpar')
1215 format ( 1x, i4, 3i5,   4x, 'j, imix, mipar, mrpar')
1216 format (/' ... oops,  inconsistency')
1217 format (/' ... oops,  the ambient mixture should NOT vary,'
     &        /'           but the ambient parameters may vary')

1221 format ( ' Enter:   ', 4x, 'j, iiparm    (# items=', i2, ')')
1222 format ( 1x, i4, i5,   4x, 'j, iiparm')
1223 format (/' ... oops,  inconsistency')
1224 format (/' ... oops,  redundancy   ')

1231 format ( ' Enter:   ', 5x, 'j, irparm     (# items=', i2, ')')
1232 format ( 1x, i4, i5,   5x, 'j, irparm')
```

```
687   1233 format (/' ... oops,  inconsistency')
688   1234 format (/' ... oops,  redundancy   ')
689
690
691   1311 format ( ' Enter:     msampl ' number of samples')
692   1312 format ( 1x, i4, 7x, 'msampl ' number of samples')
693   1313 format (/' ... oops,  inconsistency')
694
695   1321 format ( ' Enter:     mfilm ' number of films on sample #', i2)
696   1322 format ( 1x, i4, 7x, 'mfilm ' number of films on sample #', i2)
697   1323 format (/' ... oops,  inconsistency')
698
699   1331 format ( ' Enter:   ', 10x, 'j, imix, iwidth', 4x,
700      &                          '(film/substrate)')
701   1332 format ( 1x, i4,  i5, 10x, 'j, imix        ')
702   1333 format ( 1x, i4, 2i5,  5x, 'j, imix, iwidth')
703   1334 format (/' ... oops,  inconsistency in:   iwidth')
704   1335 format (/' ... oops,  inconsistency in:   imixtr')
705   1336 format (/' ... oops,  inconsistency in:   j     ')
706   1337 format (/' ... oops,  two adjacent mixtures are identical,'
707      &        /'             yet undergoing variation.     Why ? ')
708   1338 format (/' ... oops,  inconsistency ' nothing is varying'
709      &        /'             neither width nor mixture parameter')
710
711
712   1411 format ( ' Enter:     mbien ' number of ambients on sample #',
713      &                          i2)
714   1412 format ( 1x, i4, 7x, 'mbien ' number of ambients on sample #',
715      &                          i2)
716   1413 format (/' ... oops,  inconsistency')
717
718   1421 format ( ' Enter:   ', 4x, 'mrpeat, imbien')
719   1422 format (  1x, i4,  i5, 4x, 'mrpeat, imbien')
720   1423 format (/' ... oops,  inconsistency')
721   1424 format (/' ... oops,  redundancy   ')
722
723   1431 format ( ' Enter:     mexpt ' number of measurement data')
724   1432 format ( 1x, i4, 7x, 'mexpt ' number of measurement data')
725   1433 format (/' ... oops,  inconsistency,   nwaves*nanglx=', i5)
726
727   1441 format ( ' Enter:   j, wavln (nm), angli,delta,psi (degree)'
728      &        /'              wavlnu    , anglu,deltu,psiu       '/)
729   1442 format ( 1x, i4,  f10.3, f10.3,  f10.3, f10.3,
730      &          5x, '(j, wavln,angli, delta,psi )')
731   1443 format ( 5x,      f10.3, f10.3,  f10.3, f10.3,
732      &          5x, '(   wavlu,anglu, deltu,psiu)')
733   1444 format (/' ... oops,  inconsistency')
734   1445 format (/' ... oops,  increase the value of:   nexpts'
735      &        /'             from:', i5, ',     to:', i5)
736
737        end
```

6.2.4 SCATOS.FOR

```
1       subroutine scatos                ! simulate measurement data
2       include   'iounit.'
3       integer    choice
4
5
6       write (iout,101)                 ! measurements  vs.  grid scan
7       read  ( inn,  *)  choice         !    involving source variables:
8       write (iout,102)  choice         !       photon energy, and
9       if (choice.lt.1 .or.             !       angle of incidence
10     &    choice.gt.2     ) then
11          write (iout,103)
12          stop
13      end if
14
15      if (choice.eq.1) then            ! measurements:   (E,a)
16          write (iout,111)
17          read  ( inn,  *)  choice
18          write (iout,112)  choice
19          if (choice.ge.1 .and.
20     &        choice.le.3      ) then
21              call scato1 (choice)
22          else if (choice.eq.4) then
23              call scano1
24          else
25              write (iout,113)
26              stop
27          end if
28
29      else if (choice.eq.2) then       ! grid scan:   (E,a)
30          write (iout,121)
31          read  ( inn,  *)  choice
32          write (iout,122)  choice
33          if (choice.lt.1 .or.
34     &        choice.gt.6     ) then
35              write (iout,123)
36              stop
37          end if
38          call scato2 (choice)
39      end if
40
41      return
42
43  101 format (/' Enter:   choice of incident (energies, angles)'
44     &        /  12x, '1,   measurement data,     x.dat'
45     &        /  12x, '2,   grid scan.                 ')
46  102 format ( ' choice = ', i3)
47  103 format (/' scatos,  ... oops,  inconsistent choice')
48
49  111 format (/' Enter:   choice of output suitable for:     '
50     &        /  12x, '1,   input data,     x.dat,          '
```

```
51      &        /  12x, '2,   plotting (Delta, psi),          '
52      &        /  12x, '3,   plotting (Delta, psi) deviations,'
53      &        /  12x, '      deviation = measurement - model,'
54      &        /  12x, '4,   plotting |g| ~ rms deviation,  unscaled,'
55      &        /  12x, '      on a 1D or 2D grid of model parameters.')
56    112 format ( ' choice = ', i3)
57    113 format (/' scatos,  ... oops,  inconsistent choice')
58
59    121 format (/' Enter:   choice of output suitable for: '
60      &        /  12x, '1,   dielectric function,   media,'
61      &        /  12x, '2,   (Delta, psi),          x.dat,'
62      &        /  12x, '3,   (Delta, psi),       plotting,'
63      &        /  12x, '4,   d/db,              b~(z,f,p),'
64      &        /  12x, '5,   d/da,     angle of incidence,'
65      &        /  12x, '6,   d/dE,                 energy.')
66    122 format ( ' choice = ', i3)
67    123 format (/' scatos,  ... oops,  inconsistent choice')
68
69        end
```

6.2.5 SCATOI.FOR

```
1  c     ------------------------------------------------------------------
2  c     Input the range parameters for the grid scan, (energy, angle)
3  c     ------------------------------------------------------------------
4
5        subroutine scatoi (mevs,  ev1,  ev2,  ev3,
6       &                   mang, ang1, ang2, ang3 )
7        include   'iounit.'
8        include   'handyy.'                          ! pi,cccc,wavlev
9
10 c     ------------------------------------------------------------------
11 c     Optical frequency, wavelength, or associated photon energy.
12 c     ------------------------------------------------------------------
13
14       write (iout,111)                             ! incident energy
15       read  ( inn,  *)  ev1, ev2, ev3              !    -nm, +eV
16
17       if (ev1.lt.0.0 .and. ev2.gt.0.0 .or.         ! no mixing
18      &    ev1.gt.0.0 .and. ev2.lt.0.0 .or.
19      &    ev1.gt.ev2                      ) then
20           write (iout,114)
21           stop
22       end if
23
24       if (ev1 .lt. 0.0) then
25           write (iout,112)  ev1, ev2, ev3          !    -nm, wavelength
26           if (ev1 .eq. ev2) then
27               m = 1                                !     number of steps
28           else if (ev3 .eq. 0.0) then
29               m = 2
30           else
31               m = max (1, nint (abs (  (ev2-ev1)/ev3  )))
32           end if
33           ev1 =  -wavlev/ev1                       !    +eV, energy
34           ev2 =  -wavlev/ev2
35           ev3 =  (ev2-ev1)/float (max (1, m-1))
36       end if
37       write (iout,113)  ev1, ev2, ev3
38
39       if (ev1.lt.1.0 .or. ev2.gt.6.0 .or.          !    Check-test
40      &    ev3.lt.0.0 .or. ev2.lt.ev1     ) then
41           write (iout,114)
42           stop
43       end if
44
45       if (ev1.eq.ev2) then
46           mevs = 1
47           ev3 = 0.0
48       else if (ev3.eq.0.0) then
49           mevs = 2
50           ev3 = ev2-ev1
```

```
51        else
52            ev4 = ev2-ev1
53            mevs = 1 + nint (ev4/ev3)
54            if (mevs.eq.1) then
55                ev3 = ev4
56            else
57                ev3 = ev4/float (mevs-1)
58            end if
59        end if
60
61  c     ------------------------------------------------------------
62  c     Angle of incidence
63  c     ------------------------------------------------------------
64
65        write (iout,121)                          ! incident angles
66        read  ( inn,  *)  ang1, ang2, ang3        ! degrees
67        write (iout,122)  ang1, ang2, ang3
68        if (ang1.lt.0.0 .or. ang2.gt.90.0 .or.
69       &    ang3.lt.0.0 .or. ang2.lt.ang1     ) then
70            write (iout,123)
71            stop
72        end if
73
74        if (ang1.eq.ang2) then
75            mang = 1
76            ang3 = 0.0
77        else if (ang3.eq.0.0) then
78            mang = 2
79            ang3 = ang2-ang1
80        else
81            ang4 = ang2-ang1
82            mang = 1 + nint (ang4/ang3)
83            if (mang.eq.1) then
84                ang3 = ang4
85            else
86                ang3 = ang4/float (mang-1)
87            end if
88        end if
89  c     ------------------------------------------------------
90        write (iout,131)
91
92        return
93
94    111 format (/' Enter:    range of incident energies (eV)'
95       &        /'    or:                  - wavelengths (nm)'
96       &        /' Enter:    ev1, ev2, ev3        ')
97    112 format ( 1x, 3f10.2, 10x, 'wavelength (nm)')
98    113 format ( 1x, 3f10.4, 10x, '    energy (eV)')
99    114 format ( ' ... oops, inconsistent')
100
101   121 format (/' Enter:    range of incident angles (degrees) '
102      &        /' Enter:    angle1, angle2, angle3             ')
103   122 format ( 1x, 3f10.4, 10x, 'a1,a2,a3')
```

```
104     123 format ( ' ... oops, inconsistent')
105
106     131 format ( ' ')
107
108         end
```

6.2.6 SCATO1.FOR

```
      subroutine scato1 (icase)

c     ------------------------------------------------------------------
c     Retain and use information specifying the configuration of the:
c     1)    source probe    " (angle of incidence, wavelength)
c     2)    sample geometry " (z,f,p)

c     Calculate the forward scattering problem " (Delta, psi).
c     Simulate ellipsometric measurements of     (Delta, psi).
c     Format the output scattering data          (Delta, psi):
c     1)   suitable for INPDAT
c     2)   suitable for plotting
c     3)   suitable for plotting differences in the fit.
c          difference = experiment - model

c     The template for this routine is from:  ASMBL
c     ------------------------------------------------------------------

      include  'iounit.'
      include  'defnit.'
      include  'filmmm.'
      include  'xprmnt.'
      include  'arrayd.'
      include  'arrays.'
      include  'filmss.'
      include  'abcdef.'
      include  'handyy.'                ! pi

      logical   first, firstv


      raddeg = 180.0/pi

      mmm = mfilmz+mlmnts+mrpars
      do m=1,mmm                        ! local,  column in a row of:  aa
          if (iptw(m).ne.-1) then       ! utilized
              iptw(m) = 0               ! global ---> local     into:  aa
          end if
          iptx(m) = 0                   !  local ---> global    into:  iptw
      end do

      izs = 0                           !              (z,e) ,sample
      ias = 0                           !             ambient,sample
      iras = 0                          !      repeat,ambient,sample
      ixpts = 0                         ! expt,repeat,ambient,sample

      write (iplt,101) msampl

      do is=1,msampl
          mfilm = mmfilm(is)                    ! FILMSS
```

```
51          mfilms = mfilm+1                        !         films/substrate
52          mfilma = mfilm+2                        ! ambient/films/substrate
53          mzs = mfilm*2+1                         ! (z,e),(e)
54          mv = 0                                  ! local, unique, vary
55
56          do m=1,mfilms
57              if (m.ne.mfilms) then               ! films
58                  izs = izs+1                     ! widths
59                  iwidth = iifilm(izs)
60                  zzz(m) = widths(iwidth)         ! FILMSS
61                  if (lwidth(iwidth).eq.1) then   ! vary
62                      j = iwidth
63                      if (iptw(j).eq.0) then      !     unique, compress
64                          mv = mv+1               !     local
65                          iptw(j ) = mv
66                          iptx(mv) = j
67                      end if
68                  end if
69              end if
70              izs = izs+1                         ! films/substrate
71              imixtr = iifilm(izs)                ! mixture
72
73              mixflm(m) = imixtr                  ! initialize
74
75              mvlmn = mvlmnt(imixtr)              ! vary
76              if (mvlmn.ne.0) then                !     fraction
77                  first = .true.                  !     constraint, 1=u+v
78                  mmlmn = mmlmnt(imixtr)          !     quantity
79                  kk    = kklmnt(imixtr)          !     offset
80                  do lmn=1,mmlmn
81                      kk = kk+1                   !     monotonic
82                      if (lflmnt(kk).eq.1) then   !     vary
83                          j = mfilmz+kk           !     global
84                          if (first) then         !     constraint on:  dv(1)
85                              first = .false.
86                          else if (iptw(j).eq.0) then
87                              mv = mv+1           !     local unique
88                              iptw(j ) = mv
89                              iptx(mv) = j
90                          end if
91                      end if      ! vary
92                  end do         ! fraction
93              end if            ! something varies
94
95              mvpar = mvparm(imixtr)              ! vary
96              if (mvpar.ne.0) then                !     parameter
97                  mrpar = mrparm(imixtr)          !     quantity
98                  kr    = krparm(imixtr)          !     offset
99                  do irp=1,mrpar
100                     kr = kr+1                   !     offset
101                     ip = jrparm(kr)             !     specify index
102                     if (lrparm(ip).eq.1) then   !     vary
103                         j = mfilmz+mlmnts+ip    !     global, unique
```

```
                    if (iptw(j).eq.0) then !     unique
                        mv = mv+1          !     local, compress, aa
                        iptw(j ) = mv
                        iptx(mv) = j
                    end if
                end if      ! vary
            end do          ! parameters
        end if              ! something varies
    end do                  ! mfilms

    mvsav = mv                              ! convenience
    mbien = mmbent(is)                      ! quantity

    write (iplt,102)  mbien

    do mbn=1,mbien                          ! ambients
        ias = ias+1
        mrpeat = mmpeat(ias)
        imbien = iibent(ias)                ! specify ambient
        imixtr = mixmbn(imbien)             ! mixture used as ambient

        mixflm(mfilms+1) = imixtr
        mv = mvsav

        write (iplt,103)  mrpeat, imbien

        mvlmn = mvlmnt(imixtr)              ! vary
        if (mvlmn.ne.0) then                !     fractions
            first = .true.
            mmlmn = mmlmnt(imixtr)
            kk    = kklmnt(imixtr)          !     offset
            do lmn=1,mmlmn                  !     fractions
                kk = kk+1
                if (lflmnt(kk).eq.1) then   !     vary
                    j = mfilmz+kk
                    if (first) then         !     constraint
                        first = .false.
                    else if (iptw(j).eq.0) then
                        mv = mv+1           !     local, compress
                        iptw(j ) = mv
                        iptx(mv) = j
                    end if
                end if      ! vary
            end do          ! fractions
        end if              ! something varies

        mvpar = mvparm(imixtr)              ! vary
        if (mvpar.ne.0) then                !     parameters
            mrpar = mrparm(imixtr)
            kr    = krparm(imixtr)          !     offset
            do irp=1,mrpar
                kr = kr+1
                ip = jrparm(kr)             !     specify index
```

```
157                     if (lrparm(ip).eq.1) then
158                         j = mfilmz+mlmnts+ip
159                         if (iptw(j).eq.0) then !    unique
160                             mv = mv+1
161                             iptw(j ) = mv
162                             iptx(mv) = j
163                         end if
164                     end if
165                 end do
166             end if
167
168             mm = mmixtr+imbien                  ! offset,  ambient
169             mvpar = mvparm(mm)                  ! vary
170             if (mvpar.ne.0) then                !     parameter
171                 mrpar = mrparm(mm)              !     quantity
172                 kr    = krparm(mm)              !     offset
173                 do irp=1,mrpar
174                     kr = kr+1                   !     offset
175                     ip = jrparm(kr)             !     specify index
176                     if (lrparm(ip).eq.1) then   !     vary
177                         j = mfilmz+mlmnts+ip    !     global
178                         if (iptw(j).eq.0) then  !     unique
179                             mv = mv+1           !     local, compress, aa
180                             iptw(j ) = mv
181                             iptx(mv) = j
182                         end if
183                     end if
184                 end do
185             end if          ! something varies
186
187             mvari = mv                          ! local quantity
188             firstv = .true.                     ! yet need row:  delta
189
190             do irpeat=1,mrpeat                  ! repeats
191                 iras = iras+1
192                 mexpt = mmexpt(iras)
193                 m = 2
194                 write (iplt,104)  mexpt, m      ! nx,nu /(i,x,u)
195
196                 do ixpt=1,mexpt                 ! measurements
197                     ixpts = ixpts+1
198
199                     wavln1 = wavlns(ixpts)      ! -nm, +eV
200                     wavln2 = wavlnu(ixpts)
201                     angle1 = angles(ixpts)      ! radians
202                     angle2 = angleu(ixpts)
203                     psi1   = psiiis(ixpts)      ! radians
204                     psi2   = psiiiu(ixpts)
205                     delta1 = deltas(ixpts)      ! radians
206                     delta2 = deltau(ixpts)
207
208                     do m=1,mfilma               ! initialize, indicate
209                         i = mixflm(m)
```

```
210                         firstx(i) = .true.      !    need of evaluation
211                       end do
212
213   c                   Discern:    dielectric functions (z)
214
215                       imixtr = mixmbn(imbien)              ! ambient
216                       firstx(imixtr) = .false.
217                       call diefcn (imbien, imixtr, angle1, wavln1,
218         &                   dielec(  imixtr), dielew(  imixtr),
219         &                   dielff(1,imixtr),
220         &                   dielpp(1,imixtr), dielpa(1,imixtr))
221
222                       izs = izs-mzs                        ! reset pointer
223                       do m=1,mfilms
224                           if (m.ne.mfilms) then            ! skip widths
225                               izs = izs+1
226                           end if
227                           izs = izs+1
228                           imixtr = iifilm(izs)
229                           if (firstx(imixtr)) then         ! first
230                               firstx(imixtr) = .false.
231                               call diefcn (imbien, imixtr,
232         &                         angle1, wavln1,
233         &                         dielec(  imixtr), dielew(  imixtr),
234         &                         dielff(1,imixtr),
235         &                         dielpp(1,imixtr), dielpa(1,imixtr))
236                           end if
237                           die(m) = dielec(imixtr)
238                       end do           ! films
239
240                       isampl = is
241                       izsmpl = izs-mzs                     ! reset index
242                       call scattr (wavln1, angle1,
243         &                          isampl, izsmpl, imbien, mvari)
244
245                       angl  = angle1*raddeg                ! degrees
246                       psi   = b(1)  *raddeg
247                       delta = b(2)  *raddeg
248                       au    = angle2*raddeg
249                       pu    = psi2  *raddeg
250                       du    = delta2*raddeg
251
252                       if (delta.lt.0.0) then               ! [0,360)
253                           delta = delta + 360.0
254                       end if
255                       if (delta.ne.0.0) then               ! phase shift
256                           delta = 360.0 - delta
257                       end if
258
259   c                   ---------------------------------------------
260   c                   Now, consider the distinct formats for output
261
262                     if (icase .eq. 1) then               ! for INPDAT
```

```
263                     write (iplt,105)  ixpt, wavln1,angl,delta,psi
264                     write (iplt,106)        wavln2,  au,   du, pu
265
266                 else if (icase .eq. 2) then         ! for plotting
267                     write (iplt,105)  ixpt, wavln1,angl,delta,psi
268
269                 else if (icase .eq. 3) then         ! diff = expt-model
270                     dpsi = (  psi1-b(1)) *raddeg    ! degrees
271 c*                  ddel = (delta1-b(2)) *raddeg
272                     call differ (delta1, b(2), diff)! deviation, mod fcn
273                     ddel = diff *raddeg
274                     write (iplt,107)  ixpt, wavln1,angl,ddel,dpsi
275                 else                                ! oops
276                     call exit (2)
277                 end if
278 c                   ------------------------------------------------
279                 end do      ! measurements
280             end do          ! repeats
281
282             if (mv.gt.mvsav) then    ! reset unique-ness
283                 mv1 = mvsav+1
284                 do i=mv1,mv
285                     j = iptx(i)      ! point to:    iptw
286                     iptw(j) = 0      ! unique-ness
287                     iptx(i) = 0
288                 end do
289             end if
290         end do        ! ambients
291
292         if (mvsav.ne.0) then         ! reset unique-ness
293             do i=1,mvsav
294                 j = iptx(i)          ! point to:   iptw
295                 iptw(j) = 0          ! unique-ness
296                 iptx(i) = 0
297             end do
298         end if
299     end do             ! sample
300
301     return
302
303   101 format (1x,  i5,         15x,      'msampl')
304   102 format (1x,  i5,         15x,      'mbien ')
305   103 format (1x, 2i5,         10x,      'mrpeat, imbien')
306   104 format (1x, 2i5,         10x,      'mexpt, mu')
307   105 format (1x,  i5, 2x, 1p4e13.5, 2x, '(i,E,a,  d,p) ')
308   106 format (         8x, 1p4e13.5, 2x, '(uncertainty)')
309   107 format (1x,  i5, 2x, 1p4e13.5, 2x, '(i,E,a,  g (d,p))')
310
311     end
```

6.2.7 SCATO2.FOR

```
      subroutine scato2 (choice)                ! grid scan, contiguous

c     ------------------------------------------------------------
c     Perform a grid scan:    (wavelength-energy, incident angle)
c     Ouput:      function:    dielectric function (energy), media
c                 function:     (Delta, psi), uncertainties, x.dat
c                 function:                     (Delta, psi)
c                 Jacobian:    (d/dz, d/df, d/dp)_(Delta, psi)
c                 Jacobian:    (d/da            )_(Delta, psi)
c                 Jacobian:    (d/dE            )_(Delta, psi)
c     The template for this routine is from:   ASMBL
c     ------------------------------------------------------------

      include  'iounit.'
      include  'defnit.'
      include  'filmmm.'
      include  'xprmnt.'
      include  'arrayd.'
      include  'arrays.'
      include  'filmss.'
      include  'abcdef.'
      include  'handyy.'                        ! pi,cccc,wavlev

      logical   first, firstv
      integer   choice


      if (choice.eq.1) then                     ! dielectric function
          write (iout,141)                      !     specify mixture
          read  ( inn,  *)  kmix                !     for plotting
          write (iout,142)  kmix
          if (kmix.lt.1 .or. kmix.gt.mmixtr) then
              write (iout,143)
              stop
          end if
      end if

c     Discern the bounds or domain of the grid.

      call scatoi (mevs,  ev1,  ev2,  ev3,      ! wavelength-energy
     &             mang, ang1, ang2, ang3 )     ! angles of incidence

c     ------------------------------------------
      mmm = mfilmz+mlmnts+mrpars
      do m=1,mmm                          ! local,  column in a row of:  aa
          if (iptw(m).ne.-1) then         ! utilized
              iptw(m) = 0                 ! global ---> local     into:  aa
          end if
          iptx(m) = 0                     !  local ---> global    into:  iptw
      end do
```

```
51
52      raddeg = 180.0/pi
53      izs = 0                          !              (z,e) ,sample
54      ias = 0                          !             ambient,sample
55      iras = 0                         !       repeat,ambient,sample
56      ixpts = 0                        ! expt,repeat,ambient,sample
57
58      write (iplt,211)  msampl
59
60      do is=1,msampl
61          mfilm = mmfilm(is)                    ! FILMSS
62          mfilms = mfilm+1                      !         films/substrate
63          mfilma = mfilm+2                      ! ambient/films/substrate
64          mzs = mfilm*2+1                       ! (z,e),(e)
65          mv = 0                                ! local, unique, vary
66
67          if (choice.eq.1) then                 ! initialize
68              kmix = iabs (kmix)                !     referencing
69          end if
70
71          do m=1,mfilms
72              if (m.ne.mfilms) then             ! films
73                  izs = izs+1                   ! widths
74                  iwidth = iifilm(izs)
75                  zzz(m) = widths(iwidth)       ! FILMSS
76                  if (lwidth(iwidth).eq.1) then ! vary
77                      j = iwidth
78                      if (iptw(j).eq.0) then    !     unique, compress
79                          mv = mv+1             !     local
80                          iptw(j ) = mv
81                          iptx(mv) = j
82                      end if
83                  end if
84              end if
85              izs = izs+1                       ! films/substrate
86              imixtr = iifilm(izs)              ! mixture
87              mixflm(m) = imixtr                ! initialize
88
89              if (choice.eq.1) then
90                  if (kmix.eq.imixtr) then      ! referenced
91                      kmix = -kmix              !     negative
92                  end if
93              end if
94
95              mvlmn = mvlmnt(imixtr)            ! vary
96              if (mvlmn.ne.0) then              !     fraction
97                  first = .true.                !     constraint, 1=u+v
98                  mmlmn = mmlmnt(imixtr)        !     quantity
99                  kk    = kklmnt(imixtr)        !     offset
100                 do lmn=1,mmlmn
101                     kk = kk+1                 !     monotonic
102                     if (lflmnt(kk).eq.1) then !     vary
103                         j = mfilmz+kk         !     global
```

```
104                         if (first) then         !     constraint, dv(1)
105                             first = .false.
106                         else if (iptw(j).eq.0) then
107                             mv = mv+1           !     local unique
108                             iptw(j ) = mv
109                             iptx(mv) = j
110                         end if
111                     end if      ! vary
112                 end do          ! fraction
113             end if              ! something varies
114
115             mvpar = mvparm(imixtr)              ! vary
116             if (mvpar.ne.0) then                !     parameter
117                 mrpar = mrparm(imixtr)          !     quantity
118                 kr    = krparm(imixtr)          !     offset
119                 do irp=1,mrpar
120                     kr = kr+1                   !     offset
121                     ip = jrparm(kr)             !     specify index
122                     if (lrparm(ip).eq.1) then   !     vary
123                         j = mfilmz+mlmnts+ip    !     global, unique
124                         if (iptw(j).eq.0) then  !     unique
125                             mv = mv+1           !     local, compress, aa
126                             iptw(j ) = mv
127                             iptx(mv) = j
128                         end if
129                     end if      ! vary
130                 end do          ! parameters
131             end if              ! something varies
132         end do                  ! mfilms
133
134         mvsav = mv                              ! convenience
135         mbien = mmbent(is)                      ! quantity
136         write (iplt,212)  mbien
137
138         do mbn=1,mbien                          ! ambients
139             ias = ias+1
140             mrpeat = mmpeat(ias)
141             imbien = iibent(ias)                ! specify ambient
142             imixtr = mixmbn(imbien)             ! mixture used as ambient
143             mixflm(mfilms+1) = imixtr
144             mv = mvsav
145
146             if (choice.eq.1) then
147                 if (kmix.eq.imixtr) then        ! referenced
148                     kmix = -kmix                !     negative
149                 end if
150             end if
151
152             write (iplt,213)  mrpeat, imbien
153
154             mvlmn = mvlmnt(imixtr)              ! vary
155             if (mvlmn.ne.0) then                !     fractions
156                 first = .true.
```

```
157                 mmlmn = mmlmnt(imixtr)
158                 kk    = kklmnt(imixtr)          !     offset
159                 do lmn=1,mmlmn                  !     fractions
160                     kk = kk+1
161                     if (lflmnt(kk).eq.1) then   !     vary
162                         j = mfilmz+kk
163                         if (first) then         !     constraint, dv(1)
164                             first = .false.
165                         else if (iptw(j).eq.0) then
166                             mv = mv+1           !     local, compress
167                             iptw(j ) = mv
168                             iptx(mv) = j
169                         end if
170                     end if      ! vary
171                 end do          ! fractions
172             end if              ! something varies
173
174             mvpar = mvparm(imixtr)              ! vary
175             if (mvpar.ne.0) then                !     parameters
176                 mrpar = mrparm(imixtr)
177                 kr    = krparm(imixtr)          !     offset
178                 do irp=1,mrpar
179                     kr = kr+1
180                     ip = jrparm(kr)             !     specify index
181                     if (lrparm(ip).eq.1) then
182                         j = mfilmz+mlmnts+ip
183                         if (iptw(j).eq.0) then  !    unique
184                             mv = mv+1
185                             iptw(j ) = mv
186                             iptx(mv) = j
187                         end if
188                     end if
189                 end do
190             end if
191
192             mm = mmixtr+imbien                  ! offset,  ambient
193             mvpar = mvparm(mm)                  ! vary
194             if (mvpar.ne.0) then                !     parameter
195                 mrpar = mrparm(mm)              !     quantity
196                 kr    = krparm(mm)              !     offset
197                 do irp=1,mrpar
198                     kr = kr+1                   !     offset
199                     ip = jrparm(kr)             !     specify index
200                     if (lrparm(ip).eq.1) then   !     vary
201                         j = mfilmz+mlmnts+ip    !     global
202                         if (iptw(j).eq.0) then  !     unique
203                             mv = mv+1           !     local, compress, aa
204                             iptw(j ) = mv
205                             iptx(mv) = j
206                         end if
207                     end if
208                 end do
209             end if          ! something varies
```

```
210
211               mvari = mv                          ! local quantity
212               firstv = .true.                     ! yet need row:  delta
213
214               if (mv.gt.1) then                   ! vary only (0,1) things
215                   write (iout,214)  mv, mbn, is
216                   stop
217               end if
218               if (mrpeat.ne.1) then               ! convenience
219                   write (iout,215)  mrpeat, mbn, is
220               end if
221
222  c*           do irpeat=1,mrpeat                  ! repeats
223  c*               iras = iras+1
224  c*               mexpt = mmexpt(iras)
225                   mexpt = mang*mevs               ! measurements
226                   ixpts = 0                       ! reset
227
228                   mu = 2                          ! columns of output
229                   if (choice.eq.1) then           !    dielectric fcn
230                       write (iplt,223)  mevs, mang, mu, mexpt
231                   else if (choice.eq.2) then      !    x.dat
232                       write (iplt,222)  mexpt
233                   else                            !    values, Jacobians
234                       write (iplt,223)  mevs, mang, mu, mexpt
235                   end if
236
237  c*               do ixpt=1,mexpt                 ! measurements
238  c*                   ixpts  = ixpts+1
239  c*                   wavln1 = wavlns(ixpts)      !    -nm, +eV
240  c*                   wavln2 = wavlnu(ixpts)
241  c*                   angle1 = angles(ixpts)      !    radians
242  c*                   angle2 = angleu(ixpts)
243  c*                   psi1   = psiiis(ixpts)      !    radians
244  c*                   psi2   = psiiiu(ixpts)
245  c*                   delta1 = deltas(ixpts)      !    radians
246  c*                   delta2 = deltau(ixpts)
247
248  c                ---------------------------------------------! grid scan
249                   do iang=1,mang                                ! incident angles
250                       if (iang.eq.mang .and. mang.ne.1) then
251                           angl = ang2
252                       else
253                           angl = ang1 + ang3*float (iang-1)     !     degrees
254                       end if
255                       angle1 = angl*pi/180.0                    !     radians
256                       angle2 = 0.01                             !     degrees
257                       do ievs=1,mevs                            ! photon energy
258                           if (ievs.eq.mevs .and. mevs.ne.1) then
259                               energy = ev2
260                           else
261                               energy = ev1 + ev3*float (ievs-1)!     eV
262                           end if
```

```
263                     wavln1 = -wavlev/energy                 !     -nm, +eV
264                     wavln2 =  0.1                           !     convenience
265   c                 ---------------------------------------
266                     ixpts = ixpts+1                         !     update
267
268                     do m=1,mfilma               ! initialize, indicate
269                         i = mixflm(m)
270                         firstx(i) = .true.      !    need of evaluation
271                     end do
272
273   c                 Discern:    dielectric functions (z)
274
275                     imixtr = mixmbn(imbien)                 ! ambient
276                     firstx(imixtr) = .false.
277                     call diefcn (imbien, imixtr, angle1, wavln1,
278         &                  dielec(  imixtr), dielew(  imixtr),
279         &                  dielff(1,imixtr),
280         &                  dielpp(1,imixtr), dielpa(1,imixtr))
281
282                     izs = izs-mzs                           ! reset pointer
283                     do m=1,mfilms
284                         if (m.ne.mfilms) then               ! skip widths
285                             izs = izs+1
286                         end if
287                         izs = izs+1
288                         imixtr = iifilm(izs)
289                         if (firstx(imixtr)) then            ! first
290                             firstx(imixtr) = .false.
291                             call diefcn (imbien, imixtr,
292         &                       angle1, wavln1,
293         &                       dielec(  imixtr), dielew(  imixtr),
294         &                       dielff(1,imixtr),
295         &                       dielpp(1,imixtr), dielpa(1,imixtr))
296                         end if
297                         die(m) = dielec(imixtr)
298                     end do            ! films
299
300                     isampl = is
301                     izsmpl = izs-mzs                        ! reset index
302                     call scattr (wavln1, angle1,
303         &                        isampl, izsmpl, imbien, mvari)
304
305   c                 ---------------------------------------
306                     if (choice.eq.1) then                   ! dielectric
307                         if (kmix.lt.0) then                 !    referenced
308                             delta = real (dielec (-kmix))
309                             psi  = aimag (dielec (-kmix))
310                             write (iplt,311)
311         &                       ixpts, energy, angl, delta, psi
312                         else if (ixpts.eq.1) then           !    notify
313                             write (iout,144)  is, mbn, kmix
314                         end if
315   c                     ---------------------------------
```

```
316                     else if (choice .eq.2) then          ! (Delta, psi)
317                         psi   = b(1)*raddeg              !    degrees
318                         delta = b(2)*raddeg
319
320                         if (delta.lt.0.0) then           !    [0, 360)
321                             delta = delta + 360.0
322                         end if
323                         if (delta.gt.360.0) then         !    [0, 360)
324                             delta = delta - 360.0
325                         end if
326                         if (delta.ne.0.0) then           !    Nebraska
327                             delta = 360.0 - delta        !    convention
328                         end if
329
330                         write (iplt,321)
331       &                     ixpts, energy, angl, delta, psi
332   c*  &                     ixpts, wavln1, angl, delta, psi
333   c                                                      !    convenience
334                         du = 0.01                        !    degrees
335                         pu = 0.01                        !    degrees
336                         write (iplt,322)  wavln2, angle2, du, pu
337   c                     ---------------------------------
338                     else if (choice .eq.3) then          ! (Delta, psi)
339                         psi   = b(1)*raddeg              !    degrees
340                         delta = b(2)*raddeg
341
342                         if (delta.lt.0.0) then           !    [0, 360)
343                             delta = delta + 360.0
344                         end if
345                         if (delta.gt.360.0) then         !    [0, 360)
346                             delta = delta - 360.0
347                         end if
348                         if (delta.ne.0.0) then           !    Nebraska
349                             delta = 360.0 - delta        !    convention
350                         end if
351
352                         write (iplt,331)
353       &                     ixpts, energy, angl, delta, psi
354   c                     ---------------------------------
355                     else if (choice.eq.4) then           ! d/d(z,f,p)
356                         if (mv.eq.1) then
357                             psi   = a(1)*raddeg          !    degrees
358                             delta = a(2)*raddeg
359                         else
360                             stop
361                         end if
362
363                         write (iplt,341)
364       &                     ixpts, energy, angl, delta, psi
365   c                     ---------------------------------
366                     else if (choice.eq.5) then           ! d/da
367                         psi   = c(1)*raddeg              !    degrees
368                         delta = c(2)*raddeg
```

```
369
370                             write (iplt,361)
371       &                         ixpts, energy, angl, delta, psi
372   c                           --------------------------------
373                           else if (choice.eq.6) then           ! d/dE
374                               psi   = c(3)*raddeg              !    degrees
375                               delta = c(4)*raddeg
376
377                               write (iplt,361)
378       &                         ixpts, energy, angl, delta, psi
379   c                           --------------------------------
380                           end if
381
382                       end do      ! energy
383                   end do         ! angles of incidence
384   c*                  end do    ! measurements
385   c*              end do       ! repeats
386
387                   if (mv.gt.mvsav) then    ! reset unique-ness
388                       mv1 = mvsav+1
389                       do i=mv1,mv
390                           j = iptx(i)      ! point to:    iptw
391                           iptw(j) = 0      ! unique-ness
392                           iptx(i) = 0
393                       end do
394                   end if
395               end do        ! ambients
396
397               if (mvsav.ne.0) then         ! reset unique-ness
398                   do i=1,mvsav
399                       j = iptx(i)          ! point to:   iptw
400                       iptw(j) = 0          ! unique-ness
401                       iptx(i) = 0
402                   end do
403               end if
404           end do            ! sample
405
406           return
407
408       141 format (/' Enter:   kmix ' effective medium of',
409         &                          ' dielectric function')
410       142 format ( '          kmix ', i3 /)
411       143 format (/' ... oops,  inconsistent')
412       144 format (/' scato2,  ... oops,         sample ', i3
413         &        /'                            ambient ', i3
414         &        /'          does not contain mixture ', i3)
415
416       211 format (1x,  i5, 16x, 'msampl')
417       212 format (1x,  i5, 16x, 'mbien ')
418       213 format (1x, 2i5, 10x, 'mrpeat, imbien')
419       214 format (' ... oops, vary only zero or one thing at a time'
420         &       /'           you are assuming:     mvari ', i4
421         &       /'           for the case  of:   ambient ', i4
```

```
422       &       /'                                  sample ', i4)
423     215 format (' ... oops, only one repeat of experiment is necessary'
424       &       /'           you are assuming:    mrpeat ', i4
425       &       /'           for the case  of:   ambient ', i4
426       &       /'                                sample ', i4 /)
427
428     221 format (1x, 2i5, 10x, 'nx,nu /(i,x,u)')
429     222 format (1x,  i5, 10x, 'mexpt ' measurements')
430     223 format (1x, 4i5, 10x, 'nx,ny,nu, (nx*ny) /(i,x,y,u)')
431
432     311 format (1x, i5, 2x, 1p4e13.5, 2x, '(i,E,a, er,ei)')
433     321 format (1x, i5, 2x, 1p4e13.5, 2x, '(i,E,a,  d,p)')
434     322 format (        8x, 1p4e13.5, 2x, '(uncertainty)')
435     331 format (1x, i5, 2x, 1p4e13.5, 2x, '(i,E,a,  d,p)')
436     341 format (1x, i5, 2x, 1p4e13.5, 2x, '(i,E,a, d/db (d,p)')
437     351 format (1x, i5, 2x, 1p4e13.5, 2x, '(i,E,a, d/da (d,p)')
438     361 format (1x, i5, 2x, 1p4e13.5, 2x, '(i,E,a, d/dE (d,p)')
439
440         end
```

6.2.8 SCANO1.FOR

```
      subroutine scano1                ! adapted from:   SCANO2
c                                      !    called by:   SCATOS
c     ------------------------------------------------------------
c     Scan the domain grid of model parameters (z,f,p)
c     to the set of ellipsometric equations governing experiment.
c     Here, we set up the structure of do-loops necessary for the scan.
c     Inside the do-nest, we solve the direct or forward problem.
c     ------------------------------------------------------------

      include   'iounit.'
      include   'defnit.'
      include   'filmmm.'
      include   'arrays.'
      include   'nesto1.'              ! do-nest, SEEK02
      include   'handyy.'              ! pi

      logical    lbdry, local, first
      character  bufft*8, buffd*9, buffc(9)*1, buff*1, blank*1
      equivalence  (buffd,buffc(1))

      data       blank /' '/


      if (mvary.eq.0 .or. mvary.gt.2) then       ! some model parameter
          write (iout,101)                       !        ought to vary
          stop
      end if

c     Input grid specifications for model parameters (z,p,f).

      write (iout,102)
      i = 0                    ! index of vary parameters
      k = 0                    ! index for convenience
c     ------------------------------------------------------
      if (mfilmz .ne. 0) then                                 ! widths
          local = .true.                                      ! first
          do m=1,mfilmz
              if (lwidth(m).eq.1 .and. iptw(m).ne.-1) then ! vary + utilized
                  if (local) then                             ! first
                      local = .false.
                      write (iout,111)                        ! Enter:  widths
                  end if
                  i = i+1                                     ! vary
                  k = k+1                                     ! convenience

                  read  ( inn,  *)     j, z1, z2, z3
                  write (iout,112)  k, j, z1, z2, z3, i

                  if ( j.ne.m   .or. z2.le.z1  .or.
     &                 z1.lt.0.0 .or. z2.lt.0.0 .or. z3.lt.0.0) then
```

```
51                      write  (iout,113)
52                      stop
53                  end if
54
55                  ppp1(i) = z1
56                  ppp2(i) = z2
57
58                  if (z3 .eq. 0.0) then
59                      iii2(i) = 2
60                      ppp3(i) = z2-z1
61                  else
62                      iii2(i) = 1 + nint ((z2-z1)/z3)
63                      ppp3(i) = (z2-z1) /float (max (1, iii2(i)-1))
64                  end if
65
66                  psav(i) = widths(m)
67              end if    ! utilized + vary
68          end do       ! mfilmz
69      end if
70  c   ------------------------------------------------------
71  c   Since ordering in PPP assumes (z,f,p) rather than (z,p,f),
72  c   and if we impose the same ordering of groups as in X.DAT,
73  c   then we must account for this interchange of group ordering.
74  c   Method:    pre-index the 'vary' volume fractions.
75
76      ipsav1 = i                          ! retain last value of index
77      do m=1,mmixtr                       ! effective media
78          mmlmn = mmlmnt(m)
79          mvlmn = mvlmnt(m)
80          kk    = kklmnt(m)
81          if (mvlmn .ne. 0) then                              ! vary
82              j = mfilmz+kk+1
83              if (iptw(j).ne.-1) then                         ! utilized
84                  first = .true.
85                  do lmn=1,mmlmn
86                      kk = kk+1
87                      if (lflmnt(kk).eq.1) then               ! vary
88                          if (first) then                     !    skip
89                              first = .false.
90                          else
91                              i = i+1                         !    hop
92                          end if      ! first
93                      end if         ! vary
94                  end do            ! fractions
95              end if               ! utilized
96          end if                  ! something varies
97      end do                     ! mixtures
98  c   ------------------------------------------------------
99      if (mrpars .ne. 0) then                                 ! parameters
100         local = .true.                                      ! first
101         do m=1,mrpars
102             j = mfilmz+mlmnts+m
103             if (lrparm(m).eq.1 .and. iptw(j).ne.-1) then
```

```
104                     if (local) then
105                         local = .false.
106                         write (iout,131)
107                     end if
108                     i = i+1                                     ! vary
109                     k = k+1                                     ! convenience
110
111                     read  ( inn,  *)     j, p1, p2, p3
112                     write (iout,132)  k, j, p1, p2, p3, i
113
114                     if (j.ne.m .or. p2.le.p1 .or. p3.lt.0.0) then
115                         write (iout,133)
116                         stop
117                     end if
118
119                     ppp1(i) = p1
120                     ppp2(i) = p2
121
122                     if (p3 .eq. 0.0) then
123                         iii2(i) = 2
124                         ppp3(i) = p2-p1
125                     else
126                         iii2(i) = 1 + nint ((p2-p1)/p3)
127                         ppp3(i) = (p2-p1) /float (max (1, iii2(i)-1))
128                     end if
129
130                     psav(i) = rrparm(m)
131                 end if
132             end do
133         end if
134         ipsav2 = i                          ! retain last value of index
135 c       ------------------------------------------------------
136         i = ipsav1                          ! recover initial indexing
137         local = .true.                                          ! first
138         do m=1,mmixtr                                           ! fractions
139             mmlmn = mmlmnt(m)
140             mvlmn = mvlmnt(m)
141             kk    = kklmnt(m)
142             if (mvlmn .ne. 0) then                              ! vary
143                 j = mfilmz+kk+1
144                 if (iptw(j).ne.-1) then                         ! utilized
145                     if (local) then                             ! first
146                         local = .false.
147                         write (iout,121)
148                     end if
149                     first = .true.
150                     do lmn=1,mmlmn
151                         kk = kk+1
152                         if (lflmnt(kk).eq.1) then               ! vary
153                             if (first) then                     !    skip
154                                 first = .false.
155                             else
156                                 i = i+1                         ! vary
```

```
157                                 k = k+1                          ! convenience
158
159                                 read  ( inn,  *)     j, f1, f2, f3
160                                 write (iout,122)  k, j, f1, f2, f3, i
161
162                                 if ( j.ne.lmn .or. f2.le.f1  .or.
163       &                             f1.lt.0.0 .or. f2.lt.0.0 .or.
164       &                             f3.lt.0.0 .or. f2.gt.1.0     ) then
165                                     write (iout,123)
166                                     stop
167                                 end if
168
169                                 ppp1(i) = f1
170                                 ppp2(i) = f2
171
172                                 if (f3 .eq. 0.0) then
173                                     iii2(i) = 2
174                                     ppp3(i) = f2-f1
175                                 else
176                                     iii2(i) = 1 + nint ((f2-f1)/f3)
177                                     ppp3(i) = (f2-f1) /
178       &                                       float (max (1, iii2(i)-1))
179                                 end if
180
181                                 psav(i) = fflmnt(kk)
182                             end if      ! first
183                         end if          ! vary
184                     end do              ! fractions
185                 end if                  ! utilized
186             end if                      ! something varies
187         end do                        ! mixtures
188   c     ------------------------------------------------------
189         i = ipsav2                          ! recover last value of indexing
190         if (mvary .ne. i) then              ! depth of do-loop nest,  ARRANG
191             write (iout,135)
192             stop
193         end if
194
195   c     ------------------------------------------------------------
196   c*    write (iout,141)                    ! optimization strategies
197   c*    read  ( inn,  *)  ichoic            ! 0,1
198   c*    write (iout,142)  ichoic
199   c*    if (ichoic.lt.0 .or. ichoic.gt.1) then
200   c*        write (iout,143)
201   c*        stop
202   c*    end if                              ! NO optimization
203
204   c     ------------------------------------------------------------
205   c     Discern breakpoints ... from previous attempts
206
207         write (iout,151)
208         read  ( inn,*,err=11,end=11)  mvarii, kts, ktm, old
209         write (iout,152)              mvarii, kts, ktm, old
```

```
210
211       if (mvarii.ne.mvary .or. old.le.0.0 .or.
212      &    kts.lt.1        .or. ktm.lt.1       ) goto 11
213
214       write (iout,153)
215       do i=1,mvary
216           read  ( inn,*,err=11,end=11)  j, iiii(i), pppp(i), psav(i)
217           write (iout,154)              j, iiii(i), pppp(i), psav(i)
218
219           if (iiii(i) .lt. 1     .or.   j.ne.i   .or.
220      &        iiii(i) .gt. iii2(i)                .or.
221      &        pppp(i) .lt. ppp1(i)-ppp3(i)*0.001 .or.
222      &        pppp(i) .gt. ppp2(i)+ppp3(i)*0.001 .or.
223      &        psav(i) .lt. ppp1(i)-ppp3(i)*0.001 .or.
224      &        psav(i) .gt. ppp2(i)+ppp3(i)*0.001     ) then
225               write (iout,155)
226               stop
227           end if
228
229           if (iiii(i).lt.iii2(i)  .or.  iii2(i).eq.1) then
230               ppppi = ppp1(i) + ppp3(i)*float(iiii(i) -1)       ! [)
231           else
232               ppppi = ppp2(i)                                   !  ]
233           end if
234
235           if (abs(pppp(i)-ppppi) .gt. ppp3(i)*0.01) then
236               write (iout,156)
237               stop
238           end if
239       end do
240
241       write (iout,161)
242       iiistp = 1                                  ! do-increment
243       iii = mvary                                 ! depth of do-nest
244       it1 = iftime(i)                             ! start clock
245       goto 2
246    11 continue
247       write (iout,162)
248       it1 = iftime(i)                             ! start clock
249
250 c     ---------------------------------------------------------------
251 c     Formulate header card for the plot utility.
252 c     Plot format:     nx,ny,mu /(i,x,y,u)   ====> index backwards
253
254       m = 1
255       kt = mvary+1
256       if (mvary .eq. 1) then
257           write (iplt,212)  (iii2(kt-j), j=1,mvary), m     ! header card
258       else
259           write (iplt,213)  (iii2(kt-j), j=1,mvary), m     ! header card
260       end if
261
262 c     ---------------------------------------------------------------
```

```
c     Emulate, initialize, activate:   the nest of do-loops

      do i=1,mvary
          iiii(i) = 0                                ! i1-i3
      end do

      kts = 0                                        ! counter
      iiistp = 1                                     ! do-increment
      iii = 0                                        ! index do-levels
    1 iii = iii+1                                    ! index of:   iii-th  level
      if (iii .gt. mvary) goto 3
    2 iiii(iii) = iiii(iii) + iiistp                 ! update do parameter
      if (iiii(iii) .lt. iii2(iii)) then             ! test upper limit
          pppp(iii) = ppp1(iii) + ppp3(iii)*float (iiii(iii) -1)
          goto 1
      else if (iiii(iii) .eq. iii2(iii)) then
          if (iii2(iii) .eq. 1) then
              pppp(iii) = ppp1(iii)
          else
              pppp(iii) = ppp2(iii)
          end if
          goto 1
      end if
      iiii(iii) = 0                                  ! i1-i3,  reset inner do
      iii = iii-1                                    ! backup one do-level
      if (iii.eq.0) goto 4                           ! escape do-nest
      goto 2
    3 iii = iii-1                                    ! level of inner-most do.
c     ------------------------------------------
      kts = kts+1                                    ! simple counter

c     Reset the model parameters to their appropriate values.

      do i=1,mvary
          j = iptu(i)
          if (j .le. mfilmz) then
              widths(j) = pppp(i)
          else if (j .le. mfilmz+mlmnts) then
              j = j-mfilmz
              fflmnt(j) = pppp(i)
          else
              j = j-mfilmz-mlmnts
              rrparm(j) = pppp(i)
          end if
      end do

      do m=1,mmixtr                                          ! mixtures
          mvlmn = mvlmnt(m)
          if (mvlmn.ne.0) then                               ! vary
              mmlmn = mmlmnt(m)
              kk    = kklmnt(m)                              ! offset
              j = mfilmz+kk+1
              if (iptw(j).ne.-1) then                        ! utilized
```

```
                    fu = 0.0
                    fv = 0.0
                    first = .true.
                    do lmn=1,mmlmn                          ! fractions
                        kk = kk+1
                        if (lflmnt(kk).eq.1) then           ! vary
                            if (first) then
                                first = .false.
                                kk1 = kk
                            else
                                fv = fv+fflmnt(kk)
                            end if
                        else                                ! froz
                            fu = fu+fflmnt(kk)
                        end if
                    end do        ! fractions
                    fflmnt(kk1) = 1.0-fu-fv                 ! constraint
                end if            ! utilized
            end if               ! vary
        end do                  ! mixture
c       ----------------------------------------
c*      if (ichoic.eq.1) then
c*          call seeko2                 ! constrained optimization
c*      end if

        llnorm = .false.
        call asmbl0                     ! residual only, |g|.
        call norm (meqns, bb, bnorm, 1) ! retain norm of residual
        bnorm = bnorm *180.0/pi         ! degrees <--- radians

c       --------------------------------
        m = mvary+1                     ! backwards
        write (iplt,214)  kts, (pppp(m-j), j=1,mvary), bnorm

c       --------------------------------
        if (kts .eq. 1) then            ! first
            ktm = 1                     ! density of states along minimum
            old = bnorm
            do i=1,mvary
                j = iptu(i)
                if (j .le. mfilmz) then
                    psav(i) = widths(j)
                else if (j .le. mfilmz+mlmnts) then
                    j = j-mfilmz
                    psav(i) = fflmnt(j)
                else
                    j = j-mfilmz-mlmnts
                    psav(i) = rrparm(j)
                end if
            end do
        else if (bnorm .lt. old) then
            ktm = 1                     ! density of states along minimum
            old = bnorm
```

```
369             do i=1,mvary
370                 j = iptu(i)
371                 if (j .le. mfilmz) then
372                     psav(i) = widths(j)
373                 else if (j .le. mfilmz+mlmnts) then
374                     j = j-mfilmz
375                     psav(i) = fflmnt(j)
376                 else
377                     j = j-mfilmz-mlmnts
378                     psav(i) = rrparm(j)
379                 end if
380             end do
381         else if (bnorm .eq. old) then
382             ktm = ktm+1                  ! density of states along minimum
383         end if
384
385         it2 = iftime (i)                             ! CPU clock
386         tim = float (it2-it1) /6.0E4                 ! CPU minutes elapsed
387
388         if (tim .gt. 15.0) then                      ! breakpoint
389            it1 = it2
390             open (unit=isout, file='x.sout', status='unknown')
391             write (isout,171)  mvary, kts, ktm, old
392             do i=1,mvary
393                 write (isout,172)  i, iiii(i), pppp(i), psav(i)
394             end do
395             call time (bufft)                        ! hh:mm:ss
396             call date (buffd)                        ! dd-mmm-yy
397
398             buff = buffc(1)                          ! interchange (dd,yy)
399             buffc(1) = buffc(8)
400             buffc(8) = buff
401             buff = buffc(2)
402             buffc(2) = buffc(9)
403             buffc(9) = buff
404
405             write (isout,173)  bufft, buffd          ! convenience
406             close (unit=isout)
407         end if
408 c       ------------------------------------------! finish processing body
409         goto 2                                       ! nested do-loop:   iiii
410       4 continue                                     ! last line of do-nest, iiii
411 c       ================================================
412
413         lbdry = .false.
414
415         write (iout,181)  kts, ktm, old               ! DOS along minimum
416         write (iout,182)
417
418         do i=1,mvary
419             local = (psav(i) .le. ppp1(i)+ppp3(i)*0.001) .or.
420      &              (psav(i) .ge. ppp2(i)-ppp3(i)*0.001)
421 c*          local = local .and. (iii2(i).ne.1)
```

```
422           lbdry = lbdry .or. local                ! retain hitting boundary
423           j = iptu(i)
424
425           if (j .le. mfilmz) then
426               widths(j) = psav(i)
427               if (local) then
428                   write (iout,183)  i, psav(i), j, blank
429               else
430                   write (iout,183)  i, psav(i), j
431               end if
432           else if (j .le. mfilmz+mlmnts) then
433               j = j-mfilmz
434               fflmnt(j) = psav(i)
435               if (local) then
436                   write (iout,184)  i, psav(i), j, blank
437               else
438                   write (iout,184)  i, psav(i), j
439               end if
440           else
441               j = j-mfilmz-mlmnts
442               rrparm(j) = psav(i)
443               if (local) then
444                   write (iout,185)  i, psav(i), j, blank
445               else
446                   write (iout,185)  i, psav(i), j
447               end if
448           end if
449       end do                  ! vary
450
451       do m=1,mmixtr                                   ! mixtures
452           mvlmn = mvlmnt(m)
453           if (mvlmn.ne.0) then                        ! vary
454               mmlmn = mmlmnt(m)
455               kk    = kklmnt(m)
456               j = mfilmz+kk+1
457               if (iptw(j).ne.-1) then                 ! utilized
458                   fu = 0.0
459                   fv = 0.0
460                   first = .true.
461                   do lmn=1,mmlmn                      ! fractions
462                       kk = kk+1
463                       if (lflmnt(kk).eq.1) then       ! vary
464                           if (first) then
465                               first = .false.
466                               kk1 = kk
467                           else
468                               fv = fv+fflmnt(kk)
469                           end if
470                       else                            ! froz
471                           fu = fu+fflmnt(kk)
472                       end if
473                   end do        ! fractions
474                   fflmnt(kk1) = 1.0-fu-fv             ! constraint
```

```
475                end if            ! utilized
476             end if              ! vary
477          end do                 ! mixture
478
479          if (lbdry)  write (iout,186)
480    c     ---------------------------------------------------------------
481
482          call corlat
483          call scato1 (3)
484
485          return
486
487       101 format (/' oops, all model parameters are frozen, or   '
488         &         /'       too many model parameters are varying.'
489         &         /'       For convenience, only 1 or 2 may vary.')
490       102 format (/' Scan a grid of model parameters:  (z,p,f). '
491         &         /' Grid info:   DO-loop parameters            '
492         &         /' Grid info:', 4x, 'i,',     8x,
493         &             'initial,', 6x, 'final,', 5x, 'increment' )
494
495       111 format (/' Enter:    i, z1, z2, z3             (widths) ')
496       112 format (3x,i4, ')', 3x, i5, 3x, 3f13.4, 5x, '(', i3, ')')
497       113 format (/' oops, inconsistent data input ')
498
499       121 format (/' Enter:    i, f1, f2, f3             (fraction) ')
500       122 format (3x,i4, ')', 3x, i5, 3x, 3f13.4, 5x, '(', i3, ')')
501       123 format (/' oops, inconsistent data input ')
502
503       131 format (/' Enter:    i, p1, p2, p3             (parameters) ')
504       132 format (3x,i4, ')', 3x, i5, 3x, 1p3e13.4, 5x, '(', i3, ')')
505       133 format (/' oops, inconsistent data input ')
506
507       135 format (/' oops, inconsistent info:   mvary')
508
509       141 format (/' Enter:   option  regarding the grid scan:'
510         &         /'              0,     no optimization, |g| only,'
511         &         /'              1,   full optimization, Jacobian.')
512       142 format ( ' option    ', i1)
513       143 format (/' oops, inconsistent data input ')
514
515       151 format (/' Restart by attempting to read breakpoint info')
516       152 format (   1x, 3i10, 1pe15.6, 5x, '(mvary, kts, ktm, residual)')
517       153 format (/' Enter:    i, ip, pp, psav        (breakpoint info) ')
518       154 format (   1x, 2(i4,1x), 1p2e15.6 )
519       155 format (/' oops, your attempt at restarting has failed.')
520       156 format (/' oops, your attempt at restarting has failed.')
521
522       161 format ( ' Good, your attempt at restarting was successful.')
523       162 format (/' Note:    NO attempt was made to restart.'/)
524
525       171 format (1x, i4,1x, 2i10, 1p1e15.6, 5x, ' mvary,kts,ktm,residual')
526       172 format (1x, 2(i4,1x),    1p2e15.6, 5x, ' i, ip, p, p(min)     ')
527       173 format (' wall clock:    time = ', a8, '   hh:mm:ss'
```

```
528       &       /'                date = ', a9, ' ' yy-mmm-dd' )
529
530     181 format (/' number of grid points scanned, kts =', i10
531       &        /'  population along the minimum, ktm =', i10
532       &        /'          norm of the residual, |g| =', 1pe13.5,
533       &                  '  (degrees)' )
534     182 format (/' model parameter value along the minimum:')
535     183 format (1x, i4, ')', 1pe15.5, ',   for:', i5, ',  (z)',
536       &                          a1, '   ' boundary')
537     184 format (1x, i4, ')', 1pe15.5, ',   for:', i5, ',  (f)',
538       &                          a1, '   ' boundary')
539     185 format (1x, i4, ')', 1pe15.5, ',   for:', i5, ',  (p)',
540       &                          a1, '   ' boundary')
541     186 format (/' Note:    minimum point is near a boundary.')
542
543     212 format ( 1x, 2i5, ' ' mx, mu /(i, x, |g|) ')
544     213 format ( 1x, 3i5, ' ' mx(2), mx(1), mu /(i, x(2), x(1), |g|)')
545     214 format ( 1x,  i6, 1p3e15.6)
546         end
```

6.2.9 SCANO2.FOR

```
      subroutine scano2

c     ---------------------------------------------------------------
c     Scan the domain grid of model parameters (z,f,p)
c     to the set of ellipsometric equations governing experiment.
c     Minimize the sum of error residuals in the least-squares sense.
c     Method of solution:    damped least-squares analysis.
c     Here, we set up the structure of do-loops necessary for the scan.
c     Inside the do-nest, we solve the direct or forward problem.
c     ---------------------------------------------------------------

      include   'iounit.'
      include   'defnit.'
      include   'filmmm.'
      include   'arrays.'
      include   'nesto1.'              ! do-nest, SEEK02
      include   'handyy.'              ! pi

      logical    lbdry, local, first
      character  bufft*8, buffd*9, buffc(9)*1, buff*1, blank*1
      equivalence  (buffd,buffc(1))

      data       blank /' '/


      if (mvary .eq. 0) then           ! something needs to vary
          write (iout,101)
          stop
      end if

c     Input grid specifications for model parameters (z,p,f).

      write (iout,102)
      i = 0                  ! index of vary parameters
      k = 0                  ! index for convenience
c     ------------------------------------------------
      if (mfilmz .ne. 0) then                              ! widths
          local = .true.                                   ! first
          do m=1,mfilmz
              if (lwidth(m).eq.1 .and. iptw(m).ne.-1) then ! vary + utilized
                  if (local) then                          ! first
                      local = .false.
                      write (iout,111)                     ! Enter:  widths
                  end if
                  i = i+1                                  ! vary
                  k = k+1                                  ! convenience

                  read  ( inn,  *)     j, z1, z2, z3
                  write (iout,112)  k, j, z1, z2, z3, i

```

```
51                  if ( j.ne.m   .or. z2.le.z1  .or.
52      &               z1.lt.0.0 .or. z2.lt.0.0 .or. z3.lt.0.0) then
53                      write  (iout,113)
54                      stop
55                  end if
56
57                  ppp1(i) = z1
58                  ppp2(i) = z2
59
60                  if (z3 .eq. 0.0) then
61                      iii2(i) = 2
62                      ppp3(i) = z2-z1
63                  else
64                      iii2(i) = 1 + nint ((z2-z1)/z3)
65                      ppp3(i) = (z2-z1) /float (max (1, iii2(i)-1))
66                  end if
67
68                  psav(i) = widths(m)
69              end if    ! utilized + vary
70          end do       ! mfilmz
71      end if
72  c   ----------------------------------------------------
73  c   Since ordering in PPP assumes (z,f,p) rather than (z,p,f),
74  c   and if we impose the same ordering of groups as in X.DAT,
75  c   then we must account for this interchange of group ordering.
76  c   Method:    pre-index the 'vary' volume fractions.
77
78      ipsav1 = i                      ! retain last value of index
79      do m=1,mmixtr                   ! effective media
80          mmlmn = mmlmnt(m)
81          mvlmn = mvlmnt(m)
82          kk    = kklmnt(m)
83          if (mvlmn .ne. 0) then                          ! vary
84              j = mfilmz+kk+1
85              if (iptw(j).ne.-1) then                     ! utilized
86                  first = .true.
87                  do lmn=1,mmlmn
88                      kk = kk+1
89                      if (lflmnt(kk).eq.1) then           ! vary
90                          if (first) then                 !    skip
91                              first = .false.
92                          else
93                              i = i+1                     !    hop
94                          end if      ! first
95                      end if         ! vary
96                  end do            ! fractions
97              end if               ! utilized
98          end if                  ! something varies
99      end do                     ! mixtures
100 c   ----------------------------------------------------
101     if (mrpars .ne. 0) then                             ! parameters
102         local = .true.                                  ! first
103         do m=1,mrpars
```

```
104                 j = mfilmz+mlmnts+m
105                 if (lrparm(m).eq.1 .and. iptw(j).ne.-1) then
106                     if (local) then
107                         local = .false.
108                         write (iout,131)
109                     end if
110                     i = i+1                                    ! vary
111                     k = k+1                                    ! convenience
112
113                     read  ( inn,  *)     j, p1, p2, p3
114                     write (iout,132)  k, j, p1, p2, p3, i
115
116                     if (j.ne.m .or. p2.le.p1 .or. p3.lt.0.0) then
117                         write (iout,133)
118                         stop
119                     end if
120
121                     ppp1(i) = p1
122                     ppp2(i) = p2
123
124                     if (p3 .eq. 0.0) then
125                         iii2(i) = 2
126                         ppp3(i) = p2-p1
127                     else
128                         iii2(i) = 1 + nint ((p2-p1)/p3)
129                         ppp3(i) = (p2-p1) /float (max (1, iii2(i)-1))
130                     end if
131
132                     psav(i) = rrparm(m)
133                 end if
134             end do
135         end if
136         ipsav2 = i                          ! retain last value of index
137 c       -------------------------------------------------------
138         i = ipsav1                          ! recover initial indexing
139         local = .true.                                         ! first
140         do m=1,mmixtr                                          ! fractions
141           mmlmn = mmlmnt(m)
142           mvlmn = mvlmnt(m)
143           kk    = kklmnt(m)
144           if (mvlmn .ne. 0) then                               ! vary
145               j = mfilmz+kk+1
146               if (iptw(j).ne.-1) then                          ! utilized
147                   if (local) then                              ! first
148                       local = .false.
149                       write (iout,121)
150                   end if
151                   first = .true.
152                   do lmn=1,mmlmn
153                       kk = kk+1
154                       if (lflmnt(kk).eq.1) then                ! vary
155                           if (first) then                      !    skip
156                               first = .false.
```

```
157                             else
158                                 i = i+1                         ! vary
159                                 k = k+1                         ! convenience
160
161                                 read  ( inn,  *)     j, f1, f2, f3
162                                 write (iout,122)  k, j, f1, f2, f3, i
163
164                                 if ( j.ne.lmn  .or. f2.le.f1  .or.
165       &                             f1.lt.-0.5 .or. f2.gt.1.5 .or.
166       &                             f3.lt. 0.0                    ) then
167                                     write (iout,123)
168                                     stop
169                                 end if
170
171                                 ppp1(i) = f1
172                                 ppp2(i) = f2
173
174                                 if (f3 .eq. 0.0) then
175                                     iii2(i) = 2
176                                     ppp3(i) = f2-f1
177                                 else
178                                     iii2(i) = 1 + nint ((f2-f1)/f3)
179                                     ppp3(i) = (f2-f1) /
180       &                                       float (max (1, iii2(i)-1))
181                                 end if
182
183                                 psav(i) = fflmnt(kk)
184                             end if      ! first
185                         end if          ! vary
186                     end do              ! fractions
187                 end if                  ! utilized
188             end if                      ! something varies
189         end do                          ! mixtures
190   c     -------------------------------------------------------
191         i = ipsav2                          ! recover last value of indexing
192         if (mvary .ne. i) then              ! depth of do-loop nest,  ARRANG
193             write (iout,135)
194             stop
195         end if
196
197   c     ------------------------------------------------------------------
198         write (iout,141)                    ! optimization strategies
199         read  ( inn,  *)  ichoic            ! 0,1
200         write (iout,142)  ichoic
201         if (ichoic.lt.0 .or. ichoic.gt.1) then
202             write (iout,143)
203             stop
204         end if
205
206   c     ------------------------------------------------------------------
207   c     Discern breakpoints ... from previous attempts
208
209         write (iout,151)
```

```
      read  ( inn,*,err=11,end=11)  mvarii, kts, ktm, old
      write (iout,152)              mvarii, kts, ktm, old

      if (mvarii.ne.mvary .or. old.le.0.0 .or.
     &    kts.lt.1        .or. ktm.lt.1       ) goto 11

      write (iout,153)
      do i=1,mvary
          read  ( inn,*,err=11,end=11)  j, iiii(i), pppp(i), psav(i)
          write (iout,154)              j, iiii(i), pppp(i), psav(i)

          if (iiii(i) .lt. 1     .or.   j.ne.i   .or.
     &        iiii(i) .gt. iii2(i)               .or.
     &        pppp(i) .lt. ppp1(i)-ppp3(i)*0.001 .or.
     &        pppp(i) .gt. ppp2(i)+ppp3(i)*0.001 .or.
     &        psav(i) .lt. ppp1(i)-ppp3(i)*0.001 .or.
     &        psav(i) .gt. ppp2(i)+ppp3(i)*0.001     ) then
              write (iout,155)
              stop
          end if

          if (iiii(i).lt.iii2(i)  .or.  iii2(i).eq.1) then
              ppppi = ppp1(i) + ppp3(i)*float(iiii(i) -1)       ! [)
          else
              ppppi = ppp2(i)                                   !  ]
          end if

          if (abs(pppp(i)-ppppi) .gt. ppp3(i)*0.01) then
              write (iout,156)
              stop
          end if
      end do

      write (iout,161)
      iiistp = 1                                    ! do-increment
      iii = mvary                                   ! depth of do-nest
      it1 = iftime(i)                               ! start clock
      goto 2
   11 continue
      write (iout,162)
      it1 = iftime(i)                               ! start clock

c     ---------------------------------------------------------------
c     Emulate, initialize, activate:   the nest of do-loops

      do i=1,mvary
          iiii(i) = 0                               ! i1-i3
      end do

      kts = 0                                       ! counter
      iiistp = 1                                    ! do-increment
      iii = 0                                       ! index do-levels
    1 iii = iii+1                                   ! index of:   iii-th  level
```

```
263       if (iii .gt. mvary) goto 3
264     2 iiii(iii) = iiii(iii) + iiistp              ! update do parameter
265       if (iiii(iii) .lt. iii2(iii)) then          ! test upper limit
266           pppp(iii) = ppp1(iii) + ppp3(iii)*float (iiii(iii) -1)
267           goto 1
268       else if (iiii(iii) .eq. iii2(iii)) then
269           if (iii2(iii) .eq. 1) then
270               pppp(iii) = ppp1(iii)
271           else
272               pppp(iii) = ppp2(iii)
273           end if
274           goto 1
275       end if
276       iiii(iii) = 0                               ! i1-i3,  reset inner do
277       iii = iii-1                                 ! backup one do-level
278       if (iii.eq.0) goto 4                        ! escape do-nest
279       goto 2
280     3 iii = iii-1                                 ! level of inner-most do.
281 c     -----------------------------------------
282       kts = kts+1                                 ! simple counter
283
284 c     Reset the model parameters to their appropriate values.
285
286       do i=1,mvary
287           j = iptu(i)
288           if (j .le. mfilmz) then
289               widths(j) = pppp(i)
290           else if (j .le. mfilmz+mlmnts) then
291               j = j-mfilmz
292               fflmnt(j) = pppp(i)
293           else
294               j = j-mfilmz-mlmnts
295               rrparm(j) = pppp(i)
296           end if
297       end do
298
299       do m=1,mmixtr                                          ! mixtures
300           mvlmn = mvlmnt(m)
301           if (mvlmn.ne.0) then                               ! vary
302               mmlmn = mmlmnt(m)
303               kk    = kklmnt(m)                              ! offset
304               j = mfilmz+kk+1
305               if (iptw(j).ne.-1) then                        ! utilized
306                   fu = 0.0
307                   fv = 0.0
308                   first = .true.
309                   do lmn=1,mmlmn                             ! fractions
310                       kk = kk+1
311                       if (lflmnt(kk).eq.1) then              ! vary
312                           if (first) then
313                               first = .false.
314                               kk1 = kk
315                           else
```

```
316                                 fv = fv+fflmnt(kk)
317                             end if
318                         else                                        ! froz
319                             fu = fu+fflmnt(kk)
320                         end if
321                     end do          ! fractions
322                     fflmnt(kk1) = 1.0-fu-fv                         ! constraint
323                 end if              ! utilized
324             end if                  ! vary
325         end do                      ! mixture
326 c       -----------------------------------------
327         if (ichoic.eq.1) then
328             call seeko2                     ! constrained optimization
329         end if
330
331         llnorm = .false.
332         call asmbl0                         ! residual only, |g|.
333         call norm (meqns, bb, bnorm, 1)     ! retain norm of residual
334         bnorm = bnorm *180.0/pi             ! degrees <--- radians
335
336         if (kts .eq. 1) then                ! first
337             ktm = 1                         ! density of states along minimum
338             old = bnorm
339             do i=1,mvary
340                 j = iptu(i)
341                 if (j .le. mfilmz) then
342                     psav(i) = widths(j)
343                 else if (j .le. mfilmz+mlmnts) then
344                     j = j-mfilmz
345                     psav(i) = fflmnt(j)
346                 else
347                     j = j-mfilmz-mlmnts
348                     psav(i) = rrparm(j)
349                 end if
350             end do
351         else if (bnorm .lt. old) then
352             ktm = 1                         ! density of states along minimum
353             old = bnorm
354             do i=1,mvary
355                 j = iptu(i)
356                 if (j .le. mfilmz) then
357                     psav(i) = widths(j)
358                 else if (j .le. mfilmz+mlmnts) then
359                     j = j-mfilmz
360                     psav(i) = fflmnt(j)
361                 else
362                     j = j-mfilmz-mlmnts
363                     psav(i) = rrparm(j)
364                 end if
365             end do
366         else if (bnorm .eq. old) then
367             ktm = ktm+1                     ! density of states along minimum
368         end if
```

```
369
370         it2 = iftime (i)                            ! CPU clock
371         tim = float (it2-it1) /6.0E4                ! CPU minutes elapsed
372
373         if (tim .gt. 15.0) then                     ! breakpoint
374             it1 = it2
375             open (unit=isout, file='x.sout', status='unknown')
376             write (isout,171)  mvary, kts, ktm, old
377             do i=1,mvary
378                 write (isout,172)  i, iiii(i), pppp(i), psav(i)
379             end do
380             call time (bufft)                       ! hh:mm:ss
381             call date (buffd)                       ! dd-mmm-yy
382
383             buff = buffc(1)                         ! interchange (dd,yy)
384             buffc(1) = buffc(8)
385             buffc(8) = buff
386             buff = buffc(2)
387             buffc(2) = buffc(9)
388             buffc(9) = buff
389
390             write (isout,173)  bufft, buffd         ! convenience
391             close (unit=isout)
392         end if
393 c       ----------------------------------------------! finish processing body
394         goto 2                                      ! nested do-loop:   iiii
395       4 continue                                    ! last line of do-nest, iiii
396 c       ====================================================
397
398         lbdry = .false.
399
400         write (iout,181)  kts, ktm, old              ! DOS along minimum
401         write (iout,182)
402
403         do i=1,mvary
404             local = (psav(i) .le. ppp1(i)+ppp3(i)*0.001) .or.
405      &              (psav(i) .ge. ppp2(i)-ppp3(i)*0.001)
406 c*          local = local .and. (iii2(i).ne.1)
407             lbdry = lbdry .or. local                ! retain hitting boundary
408             j = iptu(i)
409
410             if (j .le. mfilmz) then
411                 widths(j) = psav(i)
412                 if (local) then
413                     write (iout,183)  i, psav(i), j, blank
414                 else
415                     write (iout,183)  i, psav(i), j
416                 end if
417             else if (j .le. mfilmz+mlmnts) then
418                 j = j-mfilmz
419                 fflmnt(j) = psav(i)
420                 if (local) then
421                     write (iout,184)  i, psav(i), j, blank
```

```
422                 else
423                     write (iout,184)  i, psav(i), j
424                 end if
425             else
426                 j = j-mfilmz-mlmnts
427                 rrparm(j) = psav(i)
428                 if (local) then
429                     write (iout,185)  i, psav(i), j, blank
430                 else
431                     write (iout,185)  i, psav(i), j
432                 end if
433             end if
434         end do                  ! vary
435
436         do m=1,mmixtr                                          ! mixtures
437             mvlmn = mvlmnt(m)
438             if (mvlmn.ne.0) then                               ! vary
439                 mmlmn = mmlmnt(m)
440                 kk    = kklmnt(m)
441                 j = mfilmz+kk+1
442                 if (iptw(j).ne.-1) then                        ! utilized
443                     fu = 0.0
444                     fv = 0.0
445                     first = .true.
446                     do lmn=1,mmlmn                             ! fractions
447                         kk = kk+1
448                         if (lflmnt(kk).eq.1) then              ! vary
449                             if (first) then
450                                 first = .false.
451                                 kk1 = kk
452                             else
453                                 fv = fv+fflmnt(kk)
454                             end if
455                         else                                   ! froz
456                             fu = fu+fflmnt(kk)
457                         end if
458                     end do        ! fractions
459                     fflmnt(kk1) = 1.0-fu-fv                    ! constraint
460                 end if            ! utilized
461             end if                ! vary
462         end do                   ! mixture
463
464         if (lbdry)  write (iout,186)
465 c       ------------------------------------------------------------------
466
467         call corlat
468         call scato1 (3)
469
470         return
471
472   101 format (/' oops, all model parameters are frozen, at least'
473        &        /'       one of them must be allowed to vary.     ')
474   102 format (/' Scan a grid of model parameters:  (z,p,f). '
```

```
475       &        /' Grid info:   DO-loop parameters
476       &        /' Grid info:',  4x,  'i,',     8x,
477       &            'initial,',  6x,  'final,', 5x, 'increment' )
478
479     111 format (/' Enter:    i, z1, z2, z3            (widths) ')
480     112 format (3x,i4, ')', 3x, i5, 3x, 3f13.4, 5x, '(', i3, ')')
481     113 format (/' oops, inconsistent data input ')
482
483     121 format (/' Enter:    i, f1, f2, f3            (fraction) ')
484     122 format (3x,i4, ')', 3x, i5, 3x, 3f13.4, 5x, '(', i3, ')')
485     123 format (/' oops, inconsistent data input ')
486
487     131 format (/' Enter:    i, p1, p2, p3            (parameters) ')
488     132 format (3x,i4, ')', 3x, i5, 3x, 1p3e13.4, 5x, '(', i3, ')')
489     133 format (/' oops, inconsistent data input ')
490
491     135 format (/' oops, inconsistent info:   mvary')
492
493     141 format (/' Enter:   option  regarding the grid scan:'
494       &        /'              0,     no optimization, |g| only,'
495       &        /'              1,   full optimization, Jacobian.')
496     142 format ( ' option    ' ', i1)
497     143 format (/' oops, inconsistent data input ')
498
499     151 format (/' Restart by attempting to read breakpoint info')
500     152 format (   1x, 3i10, 1pe15.6, 5x, '(mvary, kts, ktm, residual)')
501     153 format (/' Enter:    i, ip, pp, psav        (breakpoint info) ')
502     154 format (   1x, 2(i4,1x), 1p2e15.6 )
503     155 format (/' oops, your attempt at restarting has failed.')
504     156 format (/' oops, your attempt at restarting has failed.')
505
506     161 format ( ' Good, your attempt at restarting was successful.')
507     162 format (/' Note:    NO attempt was made to restart.'/)
508
509     171 format (1x, i4,1x, 2i10, 1p1e15.6, 5x, ' mvary,kts,ktm,residual')
510     172 format (1x, 2(i4,1x),    1p2e15.6, 5x, ' i, ip, p, p(min)     ')
511     173 format (' wall clock:    time = ', a8, '  ' hh:mm:ss'
512       &       /'                date = ', a9, ' ' yy-mmm-dd' )
513
514     181 format (/' number of grid points scanned, kts =', i10
515       &        /'  population along the minimum, ktm =', i10
516       &        /'          norm of the residual, |g| =', 1pe13.5,
517       &                ' (degrees)' )
518     182 format (/' model parameter value along the minimum:')
519     183 format (1x, i4, ')', 1pe15.5, ',   for:', i5, ',  (z)',
520       &                          a1, '   ' boundary')
521     184 format (1x, i4, ')', 1pe15.5, ',   for:', i5, ',  (f)',
522       &                          a1, '   ' boundary')
523     185 format (1x, i4, ')', 1pe15.5, ',   for:', i5, ',  (p)',
524       &                          a1, '   ' boundary')
525     186 format (/' Note:    minimum point is near a boundary.')
526
527         end
```

6.2.10 SEEKO1.FOR

```
      subroutine seeko1
c     ------------------------------------------------------------------
c     Find a single set of model parameters (z,f,p)
c     which minimizes the sum of least-squares error residuals
c     to the set of ellipsometric equations governing experiment.
c     Method of solution:   damped least-squares analysis.
c     ------------------------------------------------------------------
      include   'iounit.'
      include   'defnit.'
      include   'filmmm.'
      include   'xprmnt.'
      include   'arrays.'                          ! aa,bb,xx
      include   'wstack.'                          ! p,u,v,w,xw,se, aat,aats
      include   'handyy.'                          ! pi

      logical    first, jump

      raddeg = 180.0 /pi                           ! degrees/radian
      llnorm = .true.                              ! ASMBL, renormalize g
      call asmbl                                   ! ia,ja,aa,bb
      call norm (meqns, bb, bn, 1)                 ! residual

      if (bn .eq. 0.0) return
      bn1 = bn                                     ! initial
      bn2 = bn                                     ! last
      loop = 0
      niter = mvary*4                              ! number of iterations
      write (iout,111)
      write (iout,112)  loop, bn

    1 itry = 0                                     ! stepsize reductions
      loop = loop+1                                ! iterations
      do i=1,mvary                                 ! initialize Newton step
          xx(i) = 0.0
      end do
      call scaljj (meqns,mvary,ia,ja,aa,xx, aats,w,2) ! scale columns, A
      call cgnl   (meqns,mvary,ia,ja,aa,bb,xx,
     &             niter,       u,v,w,  xw,se)
      do i=1,mvary                                 ! account for scaling
          xx(i) = xx(i)/aats(i)                    !   columns in SCALJJ
          se(i) = se(i)/aats(i)
      end do

    2 do i=1,mvary                                 ! update
          j = iptu(i)
          h = xx(i)*0.4                            ! step
          s = se(i)*0.4                ! estimated standard deviation

          if (j .le. mfilmz) then                  ! z, thicknesses
              f = widths(j)                        !      value
```

```
51                  g = uwidth(j)                                   !     uncertainty
52  c*              write (iout,131)  i,f,h,j
53                  ha = amax1 (s, f*0.2, 0.05)                     !     upper bound
54                  ha = amin1 (g, abs (h), ha)                     !     lower bound
55                  if (h.lt.0.0)  ha=-ha
56                  aat(i) = f                                      !     retain
57                  widths(j) = amax1 (0.0, f+ha)                   !     constraint
58              .else if (j .le. mfilmz+mlmnts) then                ! volume fractions
59                  j = j-mfilmz
60                  f = fflmnt(j)                                   !     value
61                  g = uflmnt(j)                                   !     uncertainty
62  c*              write (iout,132)  i,f,h,j
63                  ha = amax1 (s, abs (f*0.2), 0.01)               !     upper bound
64                  ha = amin1 (g, abs (h), ha)                     !     lower bound
65                  if (h.lt.0.0)  ha=-ha
66                  aat(i) = f                                      !     retain
67                  fflmnt(j) = f+ha
68              else                                                ! parameters
69                  j = j-mfilmz-mlmnts
70                  f = rrparm(j)                                   !     value
71                  g = urparm(j)                                   !     uncertainty
72  c*              write (iout,133)  i,f,h,j
73                  ha = amax1 (s, abs (f*0.2))                     !     upper bound
74                  ha = amin1 (g, abs (h), ha)                     !     lower bound
75                  if (h.lt.0.0)  ha=-ha
76                  aat(i) = f                                      !     retain
77                  rrparm(j) = f+ha
78              end if
79          end do
80
81          jump = .false.                              ! escape switch
82
83      3 do m=1,mmixtr                                 ! algebraic constraint
84              mvlmn = mvlmnt(m)
85              if (mvlmn.ne.0) then                    ! something varies
86                  mmlmn = mmlmnt(m)
87                  kk    = kklmnt(m)
88                  j = mfilmz+kk+1
89                  if (iptw(j).ne.-1) then             ! utilized
90                      fu = 0.0
91                      fv = 0.0
92                      first = .true.
93                      do lmn=1,mmlmn
94                          kk = kk+1
95                          if (lflmnt(kk).eq.1) then   ! vary
96                              if (first) then
97                                  first = .false.
98                                  kk1 = kk
99                              else
100                                 fv = fv+fflmnt(kk)
101                             end if
102                         else                        ! froz
103                             fu = fu+fflmnt(kk)
```

```
                    end if
                end do    ! volume fractions
                fflmnt(kk1) = 1.0-fu-fv        ! constraint
            end if    ! utilized
        end if       ! vary
    end do          ! mixture

    if (jump) goto 6                           ! escape

    call asmbl                                 ! ia,ja,aa,bb
    call norm (meqns, bb, bn, 1)               ! residual
    total = bn /bn1                            ! total    reduction
    relat = bn /bn2                            ! relative reduction

    if (total .le. 1.E-5) then                 ! convergence
        bn2 = bn                               !     retain
        goto 5                                 !     escape
    end if

    if (relat .le. 0.999) then                 ! converging
        bn2 = bn                               !     retain
        goto 4                                 !     iterate
    end if

    do i=1,mvary                               ! reset
        j = iptu(i)
        if (j .le. mfilmz) then
            widths(j) = aat(i)
        else if (j .le. mfilmz+mlmnts) then
            j = j-mfilmz
            fflmnt(j) = aat(i)
        else
            j = j-mfilmz-mlmnts
            rrparm(j) = aat(i)
        end if
    end do

    if (itry.lt.3) then                        ! try again
        itry = itry+1
        do i=1,mvary
            xx(i) = xx(i)*0.5                  ! reduce stepsize
        end do
        goto 2
    end if

    if (loop .le. 3) then                      ! convenience
        write (iout,114)
        if (relat .le. 1.0) then               ! marginal
            write (iout,115)
        else                                   ! divergence
            write (iout,116)
        end if
    end if
```

```
157         jump = .true.                                   ! escape
158         goto 3                                          ! constraint
159
160       4 write (iout,113)  loop, relat, total, bn
161         goto 1
162       5 write (iout,113)  loop, relat, total, bn
163       6 continue
164
165   c     ------------------------------------------------------------
166   c     Output results of iterations involving least-squares.
167
168         write (iout,121)                                ! results
169         do i=1,mvary
170             j = iptu(i)
171             if (j .le. mfilmz) then                     ! widths
172                 write (iout,122)  i, widths(j), se(i), j
173             else if (j .le. mfilmz+mlmnts) then         ! fractions
174                 j = j-mfilmz
175                 write (iout,123)  i, fflmnt(j), se(i), j
176             else                                        ! parameters
177                 j = j-mfilmz-mlmnts
178                 write (iout,124)  i, rrparm(j), se(i), j
179             end if
180         end do
181         write (iout,125)  bn1, bn2                      ! compare
182
183   c     ------------------------------------------------------------
184         call corlat                                     ! correlation matrix
185         call scato1 (3)                                 ! plot deviations of fit
186         return
187
188     111 format (/' seek:   loop,    ratio of reduction,     |g|'
189        &        /'                  (rel)       (total)         ')
190     112 format (   8x, i5, 24x, 1p1e12.3)
191     113 format (   8x, i5,      1p3e12.3)
192     114 format (   8x, 'stepsize reduction attempted.')
193     115 format (   8x, '... slow convergence.         ')
194     116 format (   8x, '... divergence.               ')
195
196     121 format (/' model parameter value along the minimum:')
197     122 format (5x, i5, ')', f15.4, f10.5, ',   for:', i5, ' " (z,zu)')
198     123 format (5x, i5, ')', f15.4, f10.5, ',   for:', i5, ' " (f,fu)')
199     124 format (5x, i5, ')', f15.4, f10.5, ',   for:', i5, ' " (p,pu)')
200     125 format (/' initial |g| =', 1p1e13.5,
201        &        /'   final |g| =',    e13.5)
202
203     131 format (20x, i5, ')', 1p2e15.5, ',   for:', i5, ' " (z,dz)')
204     132 format (20x, i5, ')', 1p2e15.5, ',   for:', i5, ' " (f,df)')
205     133 format (20x, i5, ')', 1p2e15.5, ',   for:', i5, ' " (p,dp)')
206
207         end
```

6.2.11 SEEKO2.FOR

```
      subroutine seeko2
c     ------------------------------------------------------------------
c     Find a single set of model parameters (z,f,p)
c     which minimizes the sum of least-squares error residuals
c     to the set of ellipsometric equations governing experiment.
c     Method of solution:   damped least-squares analysis.
c     ------------------------------------------------------------------
      include   'iounit.'
      include   'defnit.'
      include   'filmmm.'
      include   'xprmnt.'
      include   'arrays.'                          ! aa,bb,xx
      include   'wstack.'                          ! p,u,v,w,xw,se, aat,aats
      include   'nesto1.'                          ! ppp1, ppp2, ppp3
      include   'handyy.'                          ! pi

      logical   first, jump

      raddeg = 180.0 /pi
      llnorm = .true.                              ! ASMBL, renormalize g
      call asmbl                                   ! ia,ja,aa,bb
      call norm (meqns, bb, bn, 1)                 ! residual

      if (bn .eq. 0.0) return
      bn1 = bn                                     ! initial
      bn2 = bn                                     ! last
      loop = 0
      niter = mvary*4                              ! number of iterations
c     write (iout,111)
c     write (iout,112)  loop, bn

    1 itry = 0                                     ! stepsize reductions
      loop = loop+1                                ! iterations
      do i=1,mvary                                 ! initialize Newton step
          xx(i) = 0.0
      end do
      call scaljj (meqns,mvary,ia,ja,aa,xx, aats,w,2) ! scale columns, A
      call cgnl   (meqns,mvary,ia,ja,aa,bb,xx,
     &             niter,       u,v,w,  xw,se)
      do i=1,mvary                                 ! account for scaling
          xx(i) = xx(i)/aats(i)                    !   columns in SCALJJ
          se(i) = se(i)/aats(i)
      end do

    2 do i=1,mvary                                 ! update
          j = iptu(i)
          h = xx(i)*0.4                            ! step
          s = se(i)*0.4                ! estimated standard deviation

          if (j .le. mfilmz) then                  ! z, thicknesses
```

```
51              f = widths(j)                                   !     value
52              g = uwidth(j)                                   !     uncertainty
53  c*          write (iout,131)  i,f,h,j
54              ha = amax1 (s, f*0.2, 0.05)                     !     upper bound
55              ha = amin1 (g, abs (h), ha, ppp3(i))            !     lower bound
56              if (h.lt.0.0)  ha=-ha
57              aat(i) = f                                      !     retain
58              widths(j) = amax1 (0.0, f+ha)                   !     constraint
59
60              if (widths(j) .lt. ppp1(i)) then                !     bounds
61                  widths(j) = ppp1(i)
62              else if (widths(j) .gt. ppp2(i)) then
63                  widths(j) = ppp2(i)
64              end if
65
66          else if (j .le. mfilmz+mlmnts) then                 ! volume fractions
67              j = j-mfilmz
68              f = fflmnt(j)                                   !     value
69              g = uflmnt(j)                                   !     uncertainty
70  c*          write (iout,132)  i,f,h,j
71              ha = amax1 (s, abs (f*0.2), 0.01)               !     upper bound
72              ha = amin1 (g, abs (h), ha, ppp3(i))            !     lower bound
73              if (h.lt.0.0)  ha=-ha
74              aat(i) = f                                      !     retain
75              fflmnt(j) = f+ha
76
77              if (fflmnt(j) .lt. ppp1(i)) then                !     bounds
78                  fflmnt(j) = ppp1(i)
79              else if (fflmnt(j) .gt. ppp2(i)) then
80                  fflmnt(j) = ppp2(i)
81              end if
82
83          else                                                ! parameters
84              j = j-mfilmz-mlmnts
85              f = rrparm(j)                                   !     value
86              g = urparm(j)                                   !     uncertainty
87  c*          write (iout,133)  i,f,h,j
88              ha = amax1 (s, abs (f*0.2))                     !     upper bound
89              ha = amin1 (g, abs (h), ha, ppp3(i))            !     lower bound
90              if (h.lt.0.0)  ha=-ha
91              aat(i) = f                                      !     retain
92              rrparm(j) = f+ha
93
94              if (rrparm(j) .lt. ppp1(i)) then                !     bounds
95                  rrparm(j) = ppp1(i)
96              else if (rrparm(j) .gt. ppp2(i)) then
97                  rrparm(j) = ppp2(i)
98              end if
99
100         end if          ! z,f,p
101     end do              ! vary
102
103     jump = .false.                              ! escape switch
```

```
104
105     3 do m=1,mmixtr                                 ! algebraic constraint
106           mvlmn = mvlmnt(m)
107           if (mvlmn.ne.0) then                      ! something varies
108               mmlmn = mmlmnt(m)
109               kk    = kklmnt(m)
110               j = mfilmz+kk+1
111               if (iptw(j).ne.-1) then               ! utilized
112                   fu = 0.0
113                   fv = 0.0
114                   first = .true.
115                   do lmn=1,mmlmn
116                       kk = kk+1
117                       if (lflmnt(kk).eq.1) then     ! vary
118                           if (first) then
119                               first = .false.
120                               kk1 = kk
121                           else
122                               fv = fv+fflmnt(kk)
123                           end if
124                       else                          ! froz
125                           fu = fu+fflmnt(kk)
126                       end if
127                   end do    ! volume fractions
128                   fflmnt(kk1) = 1.0-fu-fv           ! constraint
129               end if    ! utilized
130           end if       ! vary
131       end do           ! mixture
132
133       if (jump) goto 6                              ! escape
134
135       call asmbl                                    ! ia,ja,aa,bb
136       call norm (meqns, bb, bn, 1)                  ! residual
137       total = bn /bn1                               ! total    reduction
138       relat = bn /bn2                               ! relative reduction
139
140       if (total .le. 1.E-5) then                    ! convergence
141           bn2 = bn                                  !     retain
142           goto 5                                    !     escape
143       end if
144
145       if (relat .le. 0.999) then                    ! converging
146           bn2 = bn                                  !     retain
147           goto 4                                    !     iterate
148       end if
149
150       do i=1,mvary                                  ! reset
151           j = iptu(i)
152           if (j .le. mfilmz) then
153               widths(j) = aat(i)
154           else if (j .le. mfilmz+mlmnts) then
155               j = j-mfilmz
156               fflmnt(j) = aat(i)
```

```
157             else
158                 j = j-mfilmz-mlmnts
159                 rrparm(j) = aat(i)
160             end if
161         end do
162
163         if (itry.lt.3) then                          ! try again
164             itry = itry+1
165             do i=1,mvary
166                 xx(i) = xx(i)*0.5                    ! reduce stepsize
167             end do
168             goto 2
169         end if
170
171 c       if (loop .le. 3) then                        ! convenience
172 c           write (iout,114)
173 c           if (relat .le. 1.0) then                 ! marginal
174 c               write (iout,115)
175 c           else                                     ! divergence
176 c               write (iout,116)
177 c           end if
178 c       end if
179         jump = .true.                                ! escape
180         goto 3                                       ! constraint
181
182       4 continue
183 c       write (iout,113)  loop, relat, total, bn
184         goto 1
185       5 continue
186 c       write (iout,113)  loop, relat, total, bn
187       6 continue
188
189 c       ---------------------------------------------------------------
190 c       Output results of iterations involving least-squares.
191
192 c       write (iout,121)                             ! results
193 c       do i=1,mvary
194 c           j = iptu(i)
195 c           if (j .le. mfilmz) then                  ! widths
196 c               write (iout,122)  i, widths(j), se(i), j
197 c           else if (j .le. mfilmz+mlmnts) then      ! fractions
198 c               j = j-mfilmz
199 c               write (iout,123)  i, fflmnt(j), se(i), j
200 c           else                                     ! parameters
201 c               j = j-mfilmz-mlmnts
202 c               write (iout,124)  i, rrparm(j), se(i), j
203 c           end if
204 c       end do
205 c       write (iout,125)  bn1, bn2                   ! compare
206
207 c       ---------------------------------------------------------------
208 c       call corlat                                  ! correlation matrix
209 c       call scato1 (3)                              ! plot deviations of fit
```

```
210
211         return
212
213     111 format (/' seek:   loop,    ratio of reduction,      |g|'
214        &        /'                  (rel)       (total)          ')
215     112 format (   8x, i5, 24x, 1p1e12.3)
216     113 format (   8x, i5,      1p3e12.3)
217     114 format (   8x, 'stepsize reduction attempted.')
218     115 format (   8x, '... slow convergence.         ')
219     116 format (   8x, '... divergence.               ')
220
221     121 format (/' model parameter value along the minimum:')
222     122 format (5x, i5, ')', f15.4, f10.5, ',   for:', i5, ' ~ (z,zu)')
223     123 format (5x, i5, ')', f15.4, f10.5, ',   for:', i5, ' ~ (f,fu)')
224     124 format (5x, i5, ')', f15.4, f10.5, ',   for:', i5, ' ~ (p,pu)')
225     125 format (/' initial |g| =', 1p1e13.5,
226        &        /'   final |g| =',    e13.5)
227
228     131 format (20x, i5, ')', 1p2e15.5, ',   for:', i5, ' ~ (z,dz)')
229     132 format (20x, i5, ')', 1p2e15.5, ',   for:', i5, ' ~ (f,df)')
230     133 format (20x, i5, ')', 1p2e15.5, ',   for:', i5, ' ~ (p,dp)')
231
232         end
```

6.2.12 ASMBL.FOR

```
      subroutine  asmbl
      include    'iounit.'
      include    'defnit.'
      include    'filmmm.'
      include    'xprmnt.'
      include    'arrayd.'
      include    'arrays.'
      include    'filmss.'
      include    'abcdef.'
      include    'handyy.'              ! pi

      logical     first, firstv


      mmm = mfilmz+mlmnts+mrpars
      do m=1,mmm                        ! local,  column in a row of:  aa
          if (iptw(m).ne.-1)  then      ! utilized, retain info from ARRANG
              iptw(m) = 0               ! global ---> local     into:  aa
          end if
          iptx(m) = 0                   !  local ---> global    into:  iptw
      end do

      ia(1) = 1
      ii = 1                            ! IA
      jj = 0                            ! JA
      izs = 0                           !              (z,e) ,sample
      ias = 0                           !             ambient,sample
      iras = 0                          !      repeat,ambient,sample
      ixpts = 0                         ! expt,repeat,ambient,sample

      do is=1,msampl
          mfilm = mmfilm(is)                      ! FILMSS
          mfilms = mfilm+1                        !         films/substrate
          mfilma = mfilm+2                        ! ambient/films/substrate
          mzs = mfilm*2+1                         ! (z,e),(e)
          mv = 0                                  ! local, unique, vary

          do m=1,mfilms
              if (m.ne.mfilms) then               ! films
                  izs = izs+1                     ! widths
                  iwidth = iifilm(izs)
                  zzz(m) = widths(iwidth)         ! FILMSS
                  if (lwidth(iwidth).eq.1) then   ! vary
                      j = iwidth
                      if (iptw(j).eq.0) then      !   . unique, compress
                          mv = mv+1               !     local
                          iptw(j ) = mv
                          iptx(mv) = j
                          jj = jj+1               !     within row A
                          ja(jj) = iptv(j)        !     column, global
```

```
                    end if
                end if
            end if
            izs = izs+1                         ! films/substrate
            imixtr = iifilm(izs)                ! mixture

            mixflm(m) = imixtr                  ! initialize

            mvlmn = mvlmnt(imixtr)              ! vary
            if (mvlmn.ne.0) then                !     fraction
                first = .true.                  !     constraint, 1=u+v
                mmlmn = mmlmnt(imixtr)          !     quantity
                kk    = kklmnt(imixtr)          !     offset
                do lmn=1,mmlmn
                    kk = kk+1                   !     monotonic
                    if (lflmnt(kk).eq.1) then   !     vary
                        j = mfilmz+kk           !     global
                        if (first) then         !     constraint on:  dv(1)
                            first = .false.
                        else if (iptw(j).eq.0) then
                            mv = mv+1           !     local unique
                            iptw(j ) = mv
                            iptx(mv) = j
                            jj = jj+1           !     along a row within A
                            ja(jj) = iptv(j)    !     column
                        end if
                    end if      ! vary
                end do          ! fraction
            end if              ! something varies

            mvpar = mvparm(imixtr)              ! vary
            if (mvpar.ne.0) then                !     parameter
                mrpar = mrparm(imixtr)          !     quantity
                kr    = krparm(imixtr)          !     offset
                do irp=1,mrpar
                    kr = kr+1                   !     offset
                    ip = jrparm(kr)             !     specify index
                    if (lrparm(ip).eq.1) then   !     vary
                        j = mfilmz+mlmnts+ip    !     global, unique
                        if (iptw(j).eq.0) then  !     unique
                            mv = mv+1           !     local, compress, aa
                            iptw(j ) = mv
                            iptx(mv) = j
                            jj = jj+1
                            ja(jj) = iptv(j)    !     column
                        end if
                    end if      ! vary
                end do          ! parameters
            end if              ! something varies
        end do                  ! mfilms

        mvsav = mv                              ! convenience
        jjsav = jj
```

```
104
105             mbien = mmbent(is)                      ! quantity
106             do mbn=1,mbien                          ! ambients
107                 ias = ias+1
108                 mrpeat = mmpeat(ias)
109                 imbien = iibent(ias)                ! specify ambient
110                 imixtr = mixmbn(imbien)             ! mixture used as ambient
111
112                 mixflm(mfilms+1) = imixtr           ! mixture used as ambient
113                 mv = mvsav
114                 jj = jjsav
115
116                 mvlmn = mvlmnt(imixtr)              ! vary
117                 if (mvlmn.ne.0) then                !     fractions
118                     first = .true.
119                     mmlmn = mmlmnt(imixtr)
120                     kk    = kklmnt(imixtr)          !     offset
121                     do lmn=1,mmlmn                  !     fractions
122                         kk = kk+1
123                         if (lflmnt(kk).eq.1) then   !     vary
124                             j = mfilmz+kk
125                             if (first) then         !     constraint
126                                 first = .false.
127                             else if (iptw(j).eq.0) then
128                                 mv = mv+1           !     local, compress
129                                 iptw(j ) = mv
130                                 iptx(mv) = j
131                                 jj = jj+1           !     along a row in A
132                                 ja(jj) = iptv(j)    !     column
133                             end if
134                         end if      ! vary
135                     end do         ! fractions
136                 end if            ! something varies
137
138                 mvpar = mvparm(imixtr)              ! vary
139                 if (mvpar.ne.0) then                !     parameters
140                     mrpar = mrparm(imixtr)
141                     kr    = krparm(imixtr)          !     offset
142                     do irp=1,mrpar
143                         kr = kr+1
144                         ip = jrparm(kr)             !     specify index
145                         if (lrparm(ip).eq.1) then
146                             j = mfilmz+mlmnts+ip
147                             if (iptw(j).eq.0) then !    unique
148                                 mv = mv+1
149                                 iptw(j ) = mv
150                                 iptx(mv) = j
151                                 jj = jj+1
152                                 ja(jj) = iptv(j)
153                             end if
154                         end if
155                     end do
156                 end if
```

```
157
158                 mm = mmixtr+imbien                  ! offset,  ambient
159                 mvpar = mvparm(mm)                  ! vary
160                 if (mvpar.ne.0) then                !     parameter
161                     mrpar = mrparm(mm)              !     quantity
162                     kr    = krparm(mm)              !     offset
163                     do irp=1,mrpar
164                         kr = kr+1                   !     offset
165                         ip = jrparm(kr)             !     specify index
166                         if (lrparm(ip).eq.1) then   !     vary
167                             j = mfilmz+mlmnts+ip    !     global
168                             if (iptw(j).eq.0) then  !     unique
169                                 mv = mv+1           !     local, compress, aa
170                                 iptw(j ) = mv
171                                 iptx(mv) = j
172                                 jj = jj+1
173                                 ja(jj) = iptv(j)
174                             end if
175                         end if
176                     end do          .
177                 end if          ! something varies
178
179                 if (mv.eq.0) then                   ! something should vary
180                     write (iout,102)  is,mbn
181                     stop
182                 end if
183
184                 mvari = mv                          ! local quantity
185                 firstv = .true.                     ! yet need row:  delta
186
187                 do irpeat=1,mrpeat                  ! repeats
188                     iras = iras+1
189                     mexpt = mmexpt(iras)
190
191                     needs = jj + mv + mv*(mexpt-1)*2
192                     if (nnjaaa .lt. needs) then
193                         write (iout,103)  needs
194                         stop
195                     end if
196
197                     do ixpt=1,mexpt                 ! measurements
198                         ixpts = ixpts+1
199
200                         wavln1 = wavlns(ixpts)      ! nano-meters
201                         wavln2 = wavlnu(ixpts)
202                         angle1 = angles(ixpts)      ! radians
203                         angle2 = angleu(ixpts)
204                         psi1   = psiiis(ixpts)      ! radians
205                         psi2   = psiiiu(ixpts)
206                         delta1 = deltas(ixpts)      ! radians
207                         delta2 = deltau(ixpts)
208
209                         do m=1,mfilma               ! initialize, indicate
```

```
                    i = mixflm(m)
                    firstx(i) = .true.      !    need of evaluation
                end do

c               Discern:    dielectric functions (z)

                imixtr = mixmbn(imbien)                 ! ambient
                firstx(imixtr) = .false.
                call diefcn (imbien, imixtr, angle1, wavln1,
     &               dielec(  imixtr), dielew(  imixtr),
     &                dielff(1,imixtr),
     &                dielpp(1,imixtr), dielpa(1,imixtr))

                izs = izs-mzs                           ! reset pointer
                do m=1,mfilms
                    if (m.ne.mfilms) then               ! skip widths
                        izs = izs+1
                    end if
                    izs = izs+1
                    imixtr = iifilm(izs)
                    if (firstx(imixtr)) then            ! first
                        firstx(imixtr) = .false.
                        call diefcn (imbien, imixtr,
     &                       angle1, wavln1,
     &                       dielec(  imixtr), dielew(  imixtr),
     &                       dielff(1,imixtr),
     &                       dielpp(1,imixtr), dielpa(1,imixtr))
                    end if
                    die(m) = dielec(imixtr)
                end do          ! films

                isampl = is
                izsmpl = izs-mzs                        ! reset index
                call scattr (wavln1, angle1,
     &                       isampl, izsmpl, imbien, mvari)

                bb(ii  ) = psi1   - b(1)                !   psi
c*              bb(ii+1) = delta1 - b(2)                ! delta
                call differ (delta1, b(2), diff)
                bb(ii+1) = diff

                ia(ii+1) = ia(ii  )+mv                  !   psi
                ia(ii+2) = ia(ii+1)+mv                  ! delta

                if (firstv) then      ! psi (row A) already completed
                    firstv = .false.
                    do j=1,mv                           ! unique
                        jj = jj+1
                        ja(jj) = ja(jj-mv)              ! delta
                    end do
                else
                    do j=1,mv
                        jj = jj+1
```

```
263                             ja(jj   ) = ja(jj-mv)        !   psi
264                             ja(jj+mv) = ja(jj   )        ! delta
265                         end do
266                         jj = jj+mv                       ! end of second row
267                     end if
268
269                     jv = 0
270                     j1 = ia(ii  )
271                     j2 = ia(ii+1)-1
272                     do j=j1,j2                           ! initialize
273                         aa(j   ) = a(jv+1)               !   psi
274                         aa(j+mv) = a(jv+2)               ! delta
275                         jv = jv+2
276                     end do
277
278 c                   Renormalize the rows in the Jacobian matrix
279
280                     if (llnorm) then
281                         bb(ii  ) = bb(ii  )/psi2         !   psi
282                         bb(ii+1) = bb(ii+1)/delta2       ! delta
283                         do j=j1,j2
284                             aa(j   ) = aa(j   )/psi2     !   psi
285                             aa(j+mv) = aa(j+mv)/delta2   ! delta
286                         end do
287                     end if
288
289                     ii = ii+2
290                 end do      ! measurements
291             end do         ! repeats
292
293             if (mv.gt.mvsav) then    ! reset unique-ness
294                 mv1 = mvsav+1
295                 do i=mv1,mv
296                     j = iptx(i)      ! point to:    iptw
297                     iptw(j) = 0      ! unique-ness
298                     iptx(i) = 0
299                 end do
300             end if
301         end do        ! ambients
302
303         if (mvsav.ne.0) then         ! reset unique-ness
304             do i=1,mvsav
305                 j = iptx(i)          ! point to:   iptw
306                 iptw(j) = 0          ! unique-ness
307                 iptx(i) = 0
308             end do
309         end if
310     end do            ! sample
311     meqns = ii-1
312
313     if (jj .ne. ia(ii)-1) then
314         write (iout,104)  ii, jj, ia(ii)
315         stop
```

```
316         end if
317
318         return
319
320     102 format (' asmbl,   something should vary in the Jacobian,'
321        &        /'          sample=', i2, ',    ambient=', i2)
322     103 format (' asmbl,   insufficient array allocation for:   aa,ja'
323        &        /'          see named common:    arrays. '
324        &        /'          update to atleast:  ', i10   )
325     104 format (' asmbl,   inconsistent formating of sparse matrix,'
326        &        /'          ii = ', i10
327        &        /'          jj = ', i10, ' =/=', i10, ' = ia(ii)-1')
328
329         end
```

6.2.13 ASMBL0.FOR

```
      subroutine  asmbl0                ! residual only
      include    'iounit.'
      include    'defnit.'
      include    'filmmm.'
      include    'xprmnt.'
      include    'arrayd.'
      include    'arrays.'
      include    'filmss.'
      include    'abcdef.'
      include    'handyy.'              ! pi

      logical     first, firstv

      mmm = mfilmz+mlmnts+mrpars
      do m=1,mmm                        ! local,  column in a row of:  aa
          if (iptw(m).ne.-1)  then      ! utilized, retain info from ARRANG
              iptw(m) = 0               ! global ---> local     into:  aa
          end if
          iptx(m) = 0                   !  local ---> global    into:  iptw
      end do

      ia(1) = 1
      ii = 1                            ! IA
      jj = 0                            ! JA
      izs = 0                           !              (z,e) ,sample
      ias = 0                           !             ambient,sample
      iras = 0                          !      repeat,ambient,sample
      ixpts = 0                         ! expt,repeat,ambient,sample

      do is=1,msampl
          mfilm = mmfilm(is)                    ! FILMSS
          mfilms = mfilm+1                      !         films/substrate
          mfilma = mfilm+2                      ! ambient/films/substrate
          mzs = mfilm*2+1                       ! (z,e),(e)
          mv = 0                                ! local, unique, vary

          do m=1,mfilms
              if (m.ne.mfilms) then             ! films
                  izs = izs+1                   ! widths
                  iwidth = iifilm(izs)
                  zzz(m) = widths(iwidth)       ! FILMSS
                  if (lwidth(iwidth).eq.1) then ! vary
                      j = iwidth
                      if (iptw(j).eq.0) then    !     unique, compress
                          mv = mv+1             !     local
                          iptw(j ) = mv
                          iptx(mv) = j
                          jj = jj+1             !     within row A
                          ja(jj) = iptv(j)      !     column, global
                      end if
```

```
51              end if
52          end if
53          izs = izs+1                         ! films/substrate
54          imixtr = iifilm(izs)                ! mixture
55
56          mixflm(m) = imixtr                  ! initialize
57
58          mvlmn = mvlmnt(imixtr)              ! vary
59          if (mvlmn.ne.0) then                !     fraction
60              first = .true.                  !     constraint, 1=u+v
61              mmlmn = mmlmnt(imixtr)          !     quantity
62              kk    = kklmnt(imixtr)          !     offset
63              do lmn=1,mmlmn
64                  kk = kk+1                   !     monotonic
65                  if (lflmnt(kk).eq.1) then   !     vary
66                      j = mfilmz+kk           !     global
67                      if (first) then         !     constraint on:  dv(1)
68                          first = .false.
69                      else if (iptw(j).eq.0) then
70                          mv = mv+1           !     local unique
71                          iptw(j ) = mv
72                          iptx(mv) = j
73                          jj = jj+1           !     along a row within A
74                          ja(jj) = iptv(j)    !     column
75                      end if
76                  end if      ! vary
77              end do          ! fraction
78          end if              ! something varies
79
80          mvpar = mvparm(imixtr)              ! vary
81          if (mvpar.ne.0) then                !     parameter
82              mrpar = mrparm(imixtr)          !     quantity
83              kr    = krparm(imixtr)          !     offset
84              do irp=1,mrpar
85                  kr = kr+1                   !     offset
86                  ip = jrparm(kr)             !     specify index
87                  if (lrparm(ip).eq.1) then   !     vary
88                      j = mfilmz+mlmnts+ip    !     global, unique
89                      if (iptw(j).eq.0) then  !     unique
90                          mv = mv+1           !     local, compress, aa
91                          iptw(j ) = mv
92                          iptx(mv) = j
93                          jj = jj+1
94                          ja(jj) = iptv(j)    !     column
95                      end if
96                  end if      ! vary
97              end do          ! parameters
98          end if              ! something varies
99      end do                  ! mfilms
100
101     mvsav = mv                              ! convenience
102     jjsav = jj
103
```

```
104             mbien = mmbent(is)                      ! quantity
105             do mbn=1,mbien                          ! ambients
106                 ias = ias+1
107                 mrpeat = mmpeat(ias)
108                 imbien = iibent(ias)                ! specify ambient
109                 imixtr = mixmbn(imbien)             ! mixture used as ambient
110
111                 mixflm(mfilms+1) = imixtr           ! mixture used as ambient
112                 mv = mvsav
113                 jj = jjsav
114
115                 mvlmn = mvlmnt(imixtr)              ! vary
116                 if (mvlmn.ne.0) then                !     fractions
117                     first = .true.
118                     mmlmn = mmlmnt(imixtr)
119                     kk    = kklmnt(imixtr)          !     offset
120                     do lmn=1,mmlmn                  !     fractions
121                         kk = kk+1
122                         if (lflmnt(kk).eq.1) then   !     vary
123                             j = mfilmz+kk
124                             if (first) then         !     constraint
125                                 first = .false.
126                             else if (iptw(j).eq.0) then
127                                 mv = mv+1           !     local, compress
128                                 iptw(j ) = mv
129                                 iptx(mv) = j
130                                 jj = jj+1           !     along a row in A
131                                 ja(jj) = iptv(j)    !     column
132                             end if
133                         end if      ! vary
134                     end do          ! fractions
135                 end if              ! something varies
136
137                 mvpar = mvparm(imixtr)              ! vary
138                 if (mvpar.ne.0) then                !     parameters
139                     mrpar = mrparm(imixtr)
140                     kr    = krparm(imixtr)          !     offset
141                     do irp=1,mrpar
142                         kr = kr+1
143                         ip = jrparm(kr)             !     specify index
144                         if (lrparm(ip).eq.1) then
145                             j = mfilmz+mlmnts+ip
146                             if (iptw(j).eq.0) then  !    unique
147                                 mv = mv+1
148                                 iptw(j ) = mv
149                                 iptx(mv) = j
150                                 jj = jj+1
151                                 ja(jj) = iptv(j)
152                             end if
153                         end if
154                     end do
155                 end if
156
```

```
157                 mm = mmixtr+imbien                  ! offset,  ambient
158                 mvpar = mvparm(mm)                  ! vary
159                 if (mvpar.ne.0) then                !     parameter
160                     mrpar = mrparm(mm)              !     quantity
161                     kr    = krparm(mm)              !     offset
162                     do irp=1,mrpar
163                         kr = kr+1                   !     offset
164                         ip = jrparm(kr)             !     specify index
165                         if (lrparm(ip).eq.1) then   !     vary
166                             j = mfilmz+mlmnts+ip    !     global
167                             if (iptw(j).eq.0) then  !     unique
168                                 mv = mv+1           !     local, compress, aa
169                                 iptw(j ) = mv
170                                 iptx(mv) = j
171                                 jj = jj+1
172                                 ja(jj) = iptv(j)
173                             end if
174                         end if
175                     end do
176                 end if          ! something varies
177
178 c*              if (mv.eq.0) then                   ! something should vary
179 c*                  write (iout,102)  is,mbn
180 c*                  stop
181 c*              end if
182
183                 mvari = mv                          ! local quantity
184                 firstv = .true.                     ! yet need row:  delta
185
186                 do irpeat=1,mrpeat                  ! repeats
187                     iras = iras+1
188                     mexpt = mmexpt(iras)
189
190 c*                  needs = jj + mv + mv*(mexpt-1)*2
191 c*                  if (nnjaaa .lt. needs) then
192 c*                      write (iout,103)  needs
193 c*                      stop
194 c*                  end if
195
196                     do ixpt=1,mexpt                 ! measurements
197                         ixpts = ixpts+1
198
199                         wavln1 = wavlns(ixpts)      ! nano-meters
200                         wavln2 = wavlnu(ixpts)
201                         angle1 = angles(ixpts)      ! radians
202                         angle2 = angleu(ixpts)
203                         psi1   = psiiis(ixpts)      ! radians
204                         psi2   = psiiiu(ixpts)
205                         delta1 = deltas(ixpts)      ! radians
206                         delta2 = deltau(ixpts)
207
208                         do m=1,mfilma               ! initialize, indicate
209                             i = mixflm(m)
```

```
                        firstx(i) = .true.      !    need of evaluation
                      end do

c                     Discern:    dielectric functions (z)

                      imixtr = mixmbn(imbien)               ! ambient
                      firstx(imixtr) = .false.
                      call diefcn (imbien, imixtr, angle1, wavln1,
     &                     dielec(  imixtr), dielew(  imixtr),
     &                     dielff(1,imixtr),
     &                     dielpp(1,imixtr), dielpa(1,imixtr))

                      izs = izs-mzs                         ! reset pointer
                      do m=1,mfilms
                          if (m.ne.mfilms) then             ! skip widths
                              izs = izs+1
                          end if
                          izs = izs+1
                          imixtr = iifilm(izs)
                          if (firstx(imixtr)) then          ! first
                              firstx(imixtr) = .false.
                              call diefcn (imbien, imixtr,
     &                             angle1, wavln1,
     &                             dielec(  imixtr), dielew(  imixtr),
     &                             dielff(1,imixtr),
     &                             dielpp(1,imixtr), dielpa(1,imixtr))
                          end if
                          die(m) = dielec(imixtr)
                      end do            ! films

                      isampl = is
                      izsmpl = izs-mzs                      ! reset index
                      call scatt0 (wavln1, angle1,
     &                             isampl, izsmpl, imbien, mvari)

                      bb(ii  ) = psi1   - b(1)              !   psi
c*                    bb(ii+1) = delta1 - b(2)              ! delta
                      call differ (delta1, b(2), diff)
                      bb(ii+1) = diff

c*                    ia(ii+1) = ia(ii  )+mv                !   psi
c*                    ia(ii+2) = ia(ii+1)+mv                ! delta

c*                    if (firstv) then      ! psi (row A) already completed
c*                        firstv = .false.
c*                        do j=1,mv                         ! unique
c*                            jj = jj+1
c*                            ja(jj) = ja(jj-mv)            ! delta
c*                        end do
c*                    else
c*                        do j=1,mv
c*                            jj = jj+1
c*                            ja(jj   ) = ja(jj-mv)         !   psi
```

```
263 c*                          ja(jj+mv) = ja(jj   )        ! delta
264 c*                      end do
265 c*                      jj = jj+mv                       ! end of second row
266 c*                  end if
267
268 c*                  jv = 0
269 c*                  j1 = ia(ii  )
270 c*                  j2 = ia(ii+1)-1
271 c*                  do j=j1,j2                           ! initialize
272 c*                      aa(j   ) = a(jv+1)               !   psi
273 c*                      aa(j+mv) = a(jv+2)               ! delta
274 c*                      jv = jv+2
275 c*                  end do
276
277 c                   Renormalize the rows in the Jacobian matrix
278
279                     if (llnorm) then
280                         bb(ii  ) = bb(ii  )/psi2         !   psi
281                         bb(ii+1) = bb(ii+1)/delta2       ! delta
282 c*                      do j=j1,j2
283 c*                          aa(j   ) = aa(j   )/psi2     !   psi
284 c*                          aa(j+mv) = aa(j+mv)/delta2   ! delta
285 c*                      end do
286                     end if
287
288                     ii = ii+2
289                 end do      ! measurements
290             end do         ! repeats
291
292             if (mv.gt.mvsav) then    ! reset unique-ness
293                 mv1 = mvsav+1
294                 do i=mv1,mv
295                     j = iptx(i)      ! point to:    iptw
296                     iptw(j) = 0      ! unique-ness
297                     iptx(i) = 0
298                 end do
299             end if
300         end do        ! ambients
301
302         if (mvsav.ne.0) then         ! reset unique-ness
303             do i=1,mvsav
304                 j = iptx(i)          ! point to:   iptw
305                 iptw(j) = 0          ! unique-ness
306                 iptx(i) = 0
307             end do
308         end if
309     end do            ! sample
310     meqns = ii-1
311
312 c*  if (jj .ne. ia(ii)-1) then
313 c*      write (iout,104)  ii, jj, ia(ii)
314 c*      stop
315 c*  end if
```

```
316
317         return
318
319     102 format (' asmbl0,  something should vary in the Jacobian,'
320        &       /'          sample=', i2, ',     ambient=', i2)
321     103 format (' asmbl0,  insufficient array allocation for:   aa,ja'
322        &       /'          see named common:    arrays. '
323        &       /'          update to atleast:  ', i10   )
324     104 format (' asmbl0,  inconsistent formating of sparse matrix,'
325        &       /'          ii = ', i10
326        &       /'          jj = ', i10, ' =/=', i10, ' = ia(ii)-1')
327
328         end
```

6.2.14 ARRANG.FOR

```
      subroutine  arrang                ! discern global pointers
      include  'iounit.'
      include  'defnit.'
      include  'filmmm.'
      include  'xprmnt.'
      include  'arrays.'

      logical   first

c     The sparse matrix format of model parameters involve:
c         [z(1), z(2), ..., z(mfilmz)],
c         [      f(2), ..., f(mlmnts)]_(mixture first), ...
c         [      f(2), ...           ]_(mixture last ),
c         [p(1), p(2), ...           ]_(mixture first), ...
c         [p(1), ...                 ]_(mixture last ),
c         [p(1), ...                 ]_(ambient first), ...
c         [p(1), ...                 ]_(ambient last ).

c     where entities under-going variation are:
c         z ' widths,
c         f ' volume fractions,
c         p ' parameters associated with distinct mixtures,
c         p ' parameters associated with distinct ambient/modulation.

c     The constraint on volume fractions within each mixture is:
c         1=v+u    ====>    dv(1) = -dv(2)-dv(3)- ...


c     Initialize distinct quantities which may undergo variation.

      mmm = mfilmz+mlmnts+mrpars
      do m=1,mmm
          iptw(m) = -1                          ! initialize
      end do

c     Discern contiguous-ness of model parameters.

      k = 0                                     ! (z,e),(e)
      ias = 0                                   ! ambient,sample
      do is=1,msampl
          mfilm = mmfilm(is)
          mfilms = mfilm+1                      ! films,substrate
          do m=1,mfilms
              if (m.ne.mfilms) then             ! widths
                  k = k+1
                  iwidth = iifilm(k)
                  if (lwidth(iwidth).eq.1) then ! vary
                      iptw(iwidth) = 1
                  else                          ! froz
                      iptw(iwidth) = 0
```

```
51                  end if
52              end if
53              k = k+1                             ! dielectric function
54              imixtr = iifilm(k)                  ! mixture
55
56              mmlmn = mmlmnt(imixtr)              ! volume fractions
57              if (mmlmn.ne.0) then                ! existence
58                  kk = kklmnt(imixtr)             ! offset
59                  do lmn=1,mmlmn
60                      kk = kk+1
61                      j = mfilmz+kk
62                      if (lflmnt(kk).eq.1) then   ! vary
63                          iptw(j) = 1
64                      else                        ! froz
65                          iptw(j) = 0
66                      end if
67                  end do
68              end if          ! fractions
69
70              mrpar = mrparm(imixtr)              ! parameters
71              if (mrpar.ne.0) then                ! existence
72                  kr = krparm(imixtr)             ! offset
73                  do irp=1,mrpar
74                      kr = kr+1
75                      ip = jrparm(kr)
76                      j = mfilmz+mlmnts+ip
77                      if (lrparm(ip).eq.1) then   ! vary
78                          iptw(j) = 1
79                      else                        ! froz
80                          iptw(j) = 0
81                      end if
82                  end do
83              end if          ! parameters
84          end do             ! films
85
86          mbien = mmbent(is)                      ! quantity
87          do mbn=1,mbien                          ! ambients + modulations
88              ias = ias+1
89              imbien = iibent(ias)                ! specify ambient
90              imixtr = mixmbn(imbien)             ! mixture used as ambient
91
92              mmlmn = mmlmnt(imixtr)              ! volume fractions
93              if (mmlmn.ne.0) then                ! existence
94                  kk = kklmnt(imixtr)             ! offset
95                  do lmn=1,mmlmn
96                      kk = kk+1
97                      j = mfilmz+kk
98                      if (lflmnt(kk).eq.1) then   ! vary
99                          iptw(j) = 1
100                     else                        ! froz
101                         iptw(j) = 0
102                     end if
103                 end do
```

```
104                 end if          ! fractions
105
106                 mrpar = mrparm(imixtr)              ! parameters
107                 if (mrpar.ne.0) then                ! existence
108                     kr = krparm(imixtr)             ! offset
109                     do i=1,mrpar
110                         kr = kr+1
111                         ip = jrparm(kr)
112                         j = mfilmz+mlmnts+ip
113                         if (lrparm(ip).eq.1) then   ! vary
114                             iptw(j) = 1
115                         else                        ! froz
116                             iptw(j) = 0
117                         end if
118                     end do
119                 end if          ! parameters
120
121                 mm = mmixtr+imbien
122                 mrpar = mrparm(mm)                  ! ambient modulation
123                 if (mrpar.ne.0) then                ! existence
124                     kr = krparm(mm)                 ! offset
125                     do i=1,mrpar
126                         kr = kr+1
127                         ip = jrparm(kr)
128                         j = mfilmz+mlmnts+ip
129                         if (lrparm(ip).eq.1) then   ! vary
130                             iptw(j) = 1
131                         else                        ! froz
132                             iptw(j) = 0
133                         end if
134                     end do
135                 end if          ! modulation parameters
136             end do              ! ambients
137         end do                  ! sample
138 c       ========================================================
139 c       Induce ordering on model parameters, (z,f,p).
140
141         jj = 0                                      ! index extraneous baggage
142         i = 0                                       ! vary " compress
143         k = 0                                       ! froz " compress
144         if (mfilmz .ne. 0) then
145             do m=1,mfilmz                           ! film widths
146                 if (iptw(m).eq.1) then              ! vary + utilized
147                     i = i+1                         !     compress
148                     iptu(i) = m                     !     full (vary)
149                     iptv(m) = i                     !     vary (full)
150                 else if (iptw(m).eq.0) then         ! froz + utilized
151                     k = k+1                         !     compress
152                     km = nrowss+1-k                 !     backwards
153                     iptu(km) = m                    !     full (froz)
154                     iptv( m) = km                   !     froz (full)
155                 else                                ! extraneous baggage
156                     jj = jj+1
```

```
157             write (iout,1011)  m
158         end if
159     end do
160 end if
161
162 do m=1,mmixtr                           ! volume fractions
163     mmlmn = mmlmnt(m)                   ! quantity
164     if (mmlmn.ne.0) then                ! existence
165         kk = kklmnt(m)                  ! offset
166         j = mfilmz+kk+1                 ! test extraneous mixture
167
168         if (iptw(j).eq.-1) then         ! extraneous baggage
169             jj = jj+1
170             write (iout,1012) m
171         else                            ! utilized
172             first = .true.              ! constraint:   1 = v+u
173             do lmn=1,mmlmn
174                 kk = kk+1               ! offset, monotonic
175                 j = mfilmz+kk           ! distinct, unique
176                 if (lflmnt(kk).eq.1) then ! vary volume fraction
177                     if (first) then     !     constraint:
178                         first = .false. !     dv(1)=-dv(2)-dv(3)-...
179                     else
180                         i = i+1         !     compress
181                         iptu(i) = j     !     full (vary)
182                         iptv(j) = i     !     vary (full)
183                     end if
184                 else                    ! froz + utilized
185                     k = k+1             !     compress
186                     km = nrowss+1-k     !     backwards
187                     iptu(km) = j        !     full (froz)
188                     iptv( j) = km       !     froz (full)
189                 end if
190             end do     ! fractions
191         end if        ! utilized
192     end if           ! existence
193 end do               ! mixture
194
195 if (mrpars.ne.0) then                   ! parameters
196     j = mfilmz+mlmnts                   ! offset
197     do m=1,mrpars
198         j = j+1
199         if (iptw(j).eq.1) then          ! vary, utilized
200             i = i+1                     !     compress
201             iptu(i) = j                 !     full (vary)
202             iptv(j) = i                 !     vary (full)
203         else if (iptw(j).eq.0) then     ! froz, utilized
204             k = k+1                     !     compress
205             km = nrowss+1-k             !     backwards
206             iptu(km) = j                !     full (froz)
207             iptv( j) = km               !     froz (full)
208         else                            ! extraneous
209             jj = jj+1
```

```
210                     if (lrparm(m).eq.1) then        !     vary
211                         write (iout,1013)  m
212                     else                            !     froz
213                         write (iout,1014)  m
214                     end if
215                 end if
216             end do        ! parameters
217         end if           ! existence
218
219 c       ------------------------------------------
220 c       This is not necessary, because we test:    iptw(j)=/= -1,
221 c       are set to zero from within:
222 c           ASMBL, SCAT01, SCAT02.
223
224         if (jj.ne.0) stop
225 c       ------------------------------------------
226
227         mvary = i                                   ! compress
228         mfroz = k                                   ! compress
229
230         return
231    1011 format (' note:   extraneous     width ', i4)
232    1012 format (' note:   extraneous   mixture ', i4, ', (fraction)')
233    1013 format (' note:   extraneous parameter ', i4, ', (vary)'    )
234    1014 format (' note:   extraneous parameter ', i4, ', (froz)'    )
235         end
```

6.2.15 SCATTR.FOR

```
1       subroutine scattr (wavlen, anglei, isampl, izsmpl, imbien, mvari)
2
3       include  'iounit.'
4       include  'defnit.'
5       include  'filmmm.'
6       include  'arrayd.'
7       include  'arrays.'
8       include  'filmss.'              ! z,e
9       include  'rstack.'              ! R, dR
10      include  'abcdef.'              ! a,b,c   ----> output results
11      include  'handyy.'              ! pi,cccc,wavlev
12
13      complex   dRssum(nrowsf), dRpsum(nrowsf),  dRsw,dRpw
14      complex   dedf, dedff, dedp
15      logical   first
16
17
18      imixtr = mixmbn(imbien)                     ! mixture used as ambient
19      air = sqrt (real (dielec (imixtr)))         ! refractive index, FILMSS
20
21      if (wavlen .lt. 0.0) then                   ! wavelength, nm
22          qqq = -(pi+pi)/wavlen                   !     inverse nm
23      else                                        ! energy, eV
24          qqq =  (pi+pi)*wavlen/wavlev            !     inverse nm
25      end if
26
27      call forwrd (qqq, anglei)
28
29      sx =  real (Rs)
30      sy = aimag (Rs)
31      px =  real (Rp)
32      py = aimag (Rp)
33      call polar (sx,sy,sr,sa,1)
34      call polar (px,py,pr,pa,1)
35
36      if (sr.eq.0.0) then
37          if (pr.eq.0.0) then
38              psi = 0.0
39          else
40              psi = 0.5*pi
41          end if
42      else if (pr.eq.0.0) then          ! sr > 0.0
43          psi = 0.0
44      else if (pr.le.sr) then
45          psi = atan (pr/sr)
46      else
47          psi = 0.5*pi - atan(sr/pr)
48      end if
49
50      delta = pa-sa
```

```
51      1 if (delta.lt.0.0) then
52            delta = delta+2.0
53            goto 1
54          else if (delta.ge.2.0) then
55            delta = delta-2.0
56            goto 1
57          end if
58          delta = delta*pi
59
60          b(1) = psi                          ! radians
61          b(2) = delta                        ! radians
62
63    c     ---------------------------------
64    c     Note:    q ~ radian/nanometer,      dR/dq ~ nanometer/radian
65    c              q = w/c = (2*pi/hc) * [energy (eV)]
66    c              dq/d(energy (eV)) ~ radian/[nanometer-(eV)]
67
68          hcp = (pi+pi)/wavlev                ! dq/d(energy) ~ (2*pi/hc)
69          dRsw = dRsq*cmplx (hcp, 0.0)        ! inverse (eV)
70          dRpw = dRpq*cmplx (hcp, 0.0)
71    c     ========================================================
72          if (mvari.eq.0) goto 2
73
74          do i=1,mvari                        ! remove degeneracy, compress
75              dRssum(i) = cmplx (0.0, 0.0)
76              dRpsum(i) = cmplx (0.0, 0.0)
77          end do
78
79          mm = mmixtr+imbien                              ! offset, ambient
80          iv = 0                                          ! (z,e), (e)
81          izs = izsmpl
82          mzs = mfilm*2+1                                 ! (z,e), (e)
83          mfilms = mfilm+1
84          do m=1,mfilms                                   ! films/substrate
85              if (m.ne.mfilms) then
86                  iv = iv+1
87                  izs = izs+1
88                  iwidth = iifilm(izs)
89                  if (lwidth(iwidth).eq.1) then
90                      j = iwidth
91                      mv = iptw(j)
92                      dRssum(mv) = dRssum(mv) + dRs(iv)
93                      dRpsum(mv) = dRpsum(mv) + dRp(iv)
94                  end if
95              end if
96              iv = iv+1
97              izs = izs+1
98              imixtr = iifilm(izs)
99
100             mmlmn = mmlmnt(imixtr)                      ! fractions
101             mvlmn = mvlmnt(imixtr)
102             if (mvlmn.ne.0) then                        ! vary
103                 kk = kklmnt(imixtr)                     ! offset
```

```
104                 first = .true.
105                 do lmn=1,mmlmn
106                     kk = kk+1
107                     if (lflmnt(kk).eq.1) then       ! vary
108                         j = mfilmz+kk
109                         if (first) then             ! constraint
110                             first = .false.
111                             dedff = dielff(lmn,imixtr)
112                         else
113                             mv = iptw(j)
114                             dedf = dielff(lmn,imixtr) - dedff
115                             dRssum(mv) = dRssum(mv) + dRs(iv)*dedf
116                             dRpsum(mv) = dRpsum(mv) + dRp(iv)*dedf
117                         end if
118                     end if      ! vary
119                 end do         ! fraction
120             end if             ! existence
121
122             if (mmlmn.ne.0) then                    ! dR/dw = (dR/de)*(de/dw)
123                 dRsw = dRsw + dRs(iv)*dielew(imixtr)
124                 dRpw = dRpw + dRp(iv)*dielew(imixtr)
125             end if
126
127             mrpar = mrparm(imixtr)                  ! mixture parameters
128             mvpar = mvparm(imixtr)
129             if (mvpar.ne.0) then                    ! vary
130                 kr = krparm(imixtr)                 ! offset
131                 do irp=1,mrpar
132                     kr = kr+1
133                     ip = jrparm(kr)
134                     if (lrparm(ip).eq.1) then
135                         j = mfilmz+mlmnts+ip
136                         mv = iptw(j)
137                         dedp = dielpp(irp,imixtr)
138                         dRssum(mv) = dRssum(mv) + dRs(iv)*dedp
139                         dRpsum(mv) = dRpsum(mv) + dRp(iv)*dedp
140                     end if      ! vary
141                 end do         ! parameters
142             end if             ! existence
143
144 c           Here, we assume that ambient mixture does NOT vary, but
145 c           parameters associated with an external modulation MAY vary,
146 c           i.e., whenever they modify the dielectric properties of a
147 c           layer within the sample.
148
149             mrpar = mrparm(mm)                      ! ambient parameters
150             mvpar = mvparm(mm)
151             if (mvpar.ne.0) then                    ! vary
152                 kr = krparm(mm)                     ! offset
153                 do irp=1,mrpar
154                     kr = kr+1
155                     ip = jrparm(kr)
156                     if (lrparm(ip).eq.1) then
```

```
157                       j = mfilmz+mlmnts+ip
158                       mv = iptw(j)
159                       dedp = dielpa(irp,imixtr)
160                       dRssum(mv) = dRssum(mv) + dRs(iv)*dedp
161                       dRpsum(mv) = dRpsum(mv) + dRp(iv)*dedp
162                   end if      ! vary
163               end do         ! parameters
164           end if             ! existence
165
166       end do                 ! films
167 c     ================================================
168
169       k = 0
170       do i=1,mvari                                          ! Jacobian
171           sxd =  real (dRssum(i))
172           syd = aimag (dRssum(i))
173           pxd =  real (dRpsum(i))
174           pyd = aimag (dRpsum(i))
175           ssrd = (sx*sxd+sy*syd)/sr                         !  |Rs|'
176           pprd = (px*pxd+py*pyd)/pr                         !  |Rp|'
177           sdel = (sx*syd-sy*sxd)/(sr*sr)                    !  delta(s)'
178           pdel = (px*pyd-py*pxd)/(pr*pr)                    !  delta(p)'
179
180           ddel = pdel-sdel                                  !  delta'
181           dpsi = (sr*pprd-pr*ssrd)/(sr*sr+pr*pr)            !  psi'
182
183           a(k+1) = dpsi                                     ! d(psi  ) /d(e)
184           a(k+2) = ddel                                     ! d(delta) /d(e)
185           k = k+2
186       end do
187
188     2 continue
189
190       sxd =  real (dRsa)                                    ! incident angle
191       syd = aimag (dRsa)
192       pxd =  real (dRpa)
193       pyd = aimag (dRpa)
194       ssrd = (sx*sxd+sy*syd)/sr                             !  |Rs|'
195       pprd = (px*pxd+py*pyd)/pr                             !  |Rp|'
196       sdel = (sx*syd-sy*sxd)/(sr*sr)                        !  delta(s)'
197       pdel = (px*pyd-py*pxd)/(pr*pr)                        !  delta(p)'
198
199       ddel = pdel-sdel                                      !  delta'
200       dpsi = (sr*pprd-pr*ssrd)/(sr*sr+pr*pr)                !  psi'
201
202       c(1) = dpsi            ! d(psi  ) /d(angle),   Jacobian
203       c(2) = ddel            ! d(delta) /d(angle),   Jacobian
204 c     ------------------------------------------------
205       sxd =  real (dRsw)                                    ! frequency
206       syd = aimag (dRsw)
207       pxd =  real (dRpw)
208       pyd = aimag (dRpw)
209       ssrd = (sx*sxd+sy*syd)/sr                             !  |Rs|'
```

```
210         pprd = (px*pxd+py*pyd)/pr                          !  |Rp|'
211         sdel = (sx*syd-sy*sxd)/(sr*sr)                     !  delta(s)'
212         pdel = (px*pyd-py*pxd)/(pr*pr)                     !  delta(p)'
213
214         ddel = pdel-sdel                                   !  delta'
215         dpsi = (sr*pprd-pr*ssrd)/(sr*sr+pr*pr)             !  psi'
216
217         c(3) = dpsi                                        ! d(psi  )/dw
218         c(4) = ddel                                        ! d(delta)/dw
219
220         return
221         end
```

6.2.16 SCATT0.FOR

```
      subroutine scatt0 (wavlen, anglei, isampl, izsmpl, imbien, mvari)

      include   'iounit.'
      include   'defnit.'
      include   'filmmm.'
      include   'arrayd.'
      include   'arrays.'
      include   'filmss.'              ! z,e
      include   'rstack.'              ! R, dR
      include   'abcdef.'              ! a,b,c   ----> output results
      include   'handyy.'              ! pi,cccc,wavlev


      imixtr = mixmbn(imbien)                     ! mixture used as ambient
      air = sqrt (real (dielec (imixtr)))         ! refractive index, FILMSS

      if (wavlen .lt. 0.0) then                   ! wavelength, nm
          qqq = -(pi+pi)/wavlen                   !     inverse nm
      else                                        ! energy, eV
          qqq =  (pi+pi)*wavlen/wavlev            !     inverse nm
      end if

      call forwr0 (qqq, anglei)                   ! R only

      sx =  real (Rs)
      sy = aimag (Rs)
      px =  real (Rp)
      py = aimag (Rp)
      call polar (sx,sy,sr,sa,1)
      call polar (px,py,pr,pa,1)

      if (sr.eq.0.0) then
          if (pr.eq.0.0) then
              psi = 0.0
          else
              psi = 0.5*pi
          end if
      else if (pr.eq.0.0) then         ! sr > 0.0
          psi = 0.0
      else if (pr.le.sr) then
          psi = atan (pr/sr)
      else
          psi = 0.5*pi - atan(sr/pr)
      end if

      delta = pa-sa
    1 if (delta.lt.0.0) then
          delta = delta+2.0
          goto 1
      else if (delta.ge.2.0) then
```

```
51          delta = delta-2.0
52          goto 1
53        end if
54        delta = delta*pi
55
56        b(1) = psi                          ! radians
57        b(2) = delta                        ! radians
58
59        return
60        end
```

6.2.17 FORWRD.FOR

```
      subroutine forwrd (qqq, angl)
      real     qqq, angl

      include  'defnit.'              ! nfilms, nrows
      include  'filmss.'              ! z, e
      include  'rstack.'              ! R, dR    -----> output results

      complex  ys(nfilms+1), eta(nfilms+1)
      complex  yp(nfilms+1), cta(nfilms+1)
      complex  ee(nfilms), em(nfilms), ep(nfilms), zq(nfilms)

      complex  half,one,two,four,  eta0,eta0a,  cta0,cta0a
      complex  top,bot,ss,pp,  ssz,ppz,sse,ppe,ssa,ppa,  pp1,pp2,pp3

c     Solve:   the direct or forward problem.
c   Discern:   the reflection coefficient and Jacobian.
c     Given:   dielectric function of films (isotropic,homogeneous,uniform)
c              incident angle:    angl ' radians
c                  wavenumber:    qqq  ' 2*pi/wavelength
c                  wavelength:    nano-meters

c      Find:     Rs, Rp.      ------> (psi, delta)
c               dRs,dRp.      <------ partial wrt:   (z, dielectric function)
c              dRsa,dRpa      <------ partial wrt:  -[n(air)*sin(angle)]**2
c                             <------ partial wrt:   incident angle in radians
c              dRsq,dRpq      <------ partial wrt:   q ' wavenumber

      half = cmplx (0.5, 0.0)
      one  = cmplx (1.0, 0.0)
      two  = cmplx (2.0, 0.0)
      four = cmplx (4.0, 0.0)

      sa = sin (angl)
      ca = cos (angl)
      as = air*sa
      ac = air*ca
      as2 = as*as                                  ! air

      eta0  = cmplx ( ac, 0.0)                     ! air   TE
      eta0a = cmplx (-as, 0.0)                     ! air   d(eta)/d(angl)
      cta0  = cmplx ( ca/air, 0.0)                 ! air   TM
      cta0a = cmplx (-sa/air, 0.0)                 ! air   d(cta)/d(angl)

      mfilms = mfilm+1                             ! films, substrate

      do i=1,mfilms                                ! films, substrate
          eta(i) = sqrtt (die(i)-cmplx(as2,0.0)) ! TE
          cta(i) = eta(i)/die(i)                   ! TM
      end do

```

```
c     Determine the reflection coefficients in air.
c     Method for TE:   admittance = Y = - Hx/Ey,       Rs uses E field.
c                 TM:    impedance = Z =   Ex/Hy,       Rp uses H field.

      ys(mfilms) = eta(mfilms)                                  ! substrate
      yp(mfilms) = cta(mfilms)
      if (mfilm.ne.0) then
          do i=mfilm,1,-1                                       ! backwards
              x = zzz(i)*qqq
              zq(i) = cmplx (0.0, x)                            ! izq
              ss =    cmplx (0.0, x+x) * eta(i)                 ! izqn2
              x  = exp (real (ss))
              y  =     aimag (ss)
              ss = cmplx (x*cos(y), x*sin(y))                   ! exp (izqn2)
              ee(i) = ss
              em(i) = one-ss
              ep(i) = one+ss
              ys(i) = eta(i)* ((em(i)*eta(i)+ep(i)*ys(i+1)) /
     &                         (ep(i)*eta(i)+em(i)*ys(i+1)) )
              yp(i) = cta(i)* ((em(i)*cta(i)+ep(i)*yp(i+1)) /
     &                         (ep(i)*cta(i)+em(i)*yp(i+1)) )
          end do
      end if
      Rs = (eta0-ys(1)) /(eta0+ys(1))                           ! air
      Rp = (cta0-yp(1)) /(cta0+yp(1))                           ! air

c     Jacobian

      dRsq = cmplx (0.0,0.0)                                    ! d/dq
      dRpq = cmplx (0.0,0.0)

      do i=mfilms,1,-1                                          ! backwards
          if (i.eq.mfilms) then                                 ! source substrate
              sse = half /eta(i)                                ! d(eta)/d(e)
              ssa = sse                                         ! d(eta)/d(-as**2)
              ssz = cmplx (0.0,0.0)                             ! convenience

              ppe = (cmplx (2.0*as2, 0.0) - die(i))
     &              * half / (eta(i)* die(i)*die(i))            ! d(cta)/d(e)
              ppa = half / (eta(i)*die(i))                      ! d(cta)/d(-as**2)
              ppz = cmplx (0.0,0.0)                             ! convenience
          else                                                  ! source film
              bot = ep(i)*eta(i) + em(i)*ys(i+1)                ! denominator,  TE
              top = em(i)*eta(i) + ep(i)*ys(i+1)                ! numerator
              ss  = em(i) + zq(i)*ee(i)*(ys(i+1)-eta(i))
     &                    +  ep(i)*half*(ys(i+1)/eta(i))
              pp  = ep(i)*half - zq(i)*ee(i)*(ys(i+1)-eta(i))

              sse = (ss/bot) - (pp*top)/(bot*bot)
              ssa = (ss + ep(i)*eta(i)*ssa) /bot -
     &              (pp + em(i)*eta(i)*ssa)*top /(bot*bot)
              ssz = two*ee(i)* (ys(i+1)-eta(i)) * (ys(i+1)+eta(i))
     &                       * (eta(i)/bot)**2                  ! d/d (izq2)
```

```
104
105                 bot = ep(i)*eta(i) + em(i)*yp(i+1)*die(i)    ! denominator,  TM
106                 top = em(i)*eta(i) + ep(i)*yp(i+1)*die(i)    ! numerator
107                 pp  = (cmplx (2.0*as2, 0.0) - die(i))
108       &               * half / (eta(i)* die(i)*die(i))       ! d(cta)/d(e)
109
110                 pp1 = pp*top/bot
111                 pp2 = em(i)*half/die(i) + ep(i)*cta(i)*yp(i+1) +
112       &               zq(i)*ee(i)*(yp(i+1)-cta(i))
113                 pp3 = ep(i)*half/die(i) + em(i)*cta(i)*yp(i+1) -
114       &               zq(i)*ee(i)*(yp(i+1)-cta(i))
115                 ppe = pp1 + pp2/bot - pp3*top/(bot*bot)
116
117                 pp1 = half*top /(bot*eta(i)*die(i))
118                 pp2 = em(i)*half/die(i) + ep(i)*eta(i)*ppa +
119       &               zq(i)*ee(i)*(yp(i+1)-cta(i))
120                 pp3 = ep(i)*half/die(i) + em(i)*eta(i)*ppa -
121       &               zq(i)*ee(i)*(yp(i+1)-cta(i))
122                 ppa = pp1 + pp2/bot - pp3*top/(bot*bot)
123
124                 pp  = (eta(i)-die(i)*yp(i+1)) * (eta(i)+die(i)*yp(i+1))
125                 ppz = -two*ee(i)*eta(i)*cta(i)* (pp/(bot*bot))
126             end if
127
128             if (i.ne.1) then                                 ! transport
129                 k = i-1
130                 do j=k,1,-1                                  ! backwards
131                    ss = eta(j) / (ep(j)*eta(j)+em(j)*ys(j+1))
132                    pp = cta(j) / (ep(j)*cta(j)+em(j)*yp(j+1))
133                    ss = four*ee(j)*ss*ss                     ! coefficient
134                    pp = four*ee(j)*pp*pp
135                    sse = sse*ss                              ! Y'(e)
136                    ppe = ppe*pp
137                    ssz = ssz*ss                              ! Y'(izq2)
138                    ppz = ppz*pp
139                 end do
140             end if
141             ss = -two*eta0 /((eta0+ys(1))**2)                ! coefficient
142             pp = -two*cta0 /((cta0+yp(1))**2)
143             sse = sse*ss                                     ! R'(e)
144             ppe = ppe*pp
145             ssz = ssz*ss                                     ! R'(izq2)
146             ppz = ppz*pp
147
148   c         The format of the Jacobian vector has the form:
149   c          air / film#1 / film#2 / ... / mfilm / substrate
150   c               [z,e,     z,e,     ...   z,e,        e]
151
152             k = 2*(i-1)                                      ! position Jacobian
153             if (i.eq.mfilms) then                            ! substrate, ( ,e)
154                 k = k-1
155             else                                             ! film,      (z,e)
156                 ss = cmplx (0.0, qqq*2.0)                    ! d(izq2) /dz
```

```
157                 dRs(k+1) = ssz*ss                          ! R'(z)
158                 dRp(k+1) = ppz*ss                          ! R'(z)
159
160                 ss = cmplx (0.0, zzz(i)*2.0)               ! d(izq2) /dq
161                 dRsq = dRsq+ssz*ss                         ! R'(q)
162                 dRpq = dRpq+ppz*ss                         ! R'(q)
163             end if
164             dRs(k+2) = sse                                 ! R'(e)
165             dRp(k+2) = ppe                                 ! R'(e)
166
167         end do        ! mfilms
168
169
170 c       Since the angular partials  involved:      dR/d (-(air*sin)**2)
171 c       and we want partials wrt angle, i.e.:      dR/d (angl)     angl"radians
172
173         ss = cmplx (-as*ac*2.0, 0.0)               ! d (-(air*sin)**2) /d(angl)
174         ssa = ssa*ss                               ! d(Ys) /d(angl)
175         ppa = ppa*ss                               ! d(Yp) /d(angl)
176
177         ssa = two* (eta0a*ys(1)-eta0*ssa) /((eta0+ys(1))**2)      ! R'(a)
178         ppa = two* (cta0a*yp(1)-cta0*ppa) /((cta0+yp(1))**2)
179         dRsa = ssa                                 ! R'(a)
180         dRpa = ppa
181
182 c       Note one distinction regarding conventions in defintions of fields.
183 c       The TM calculation utilizes H fields, Rp " H(  reflected)/H(  incident),
184 c       while some conventions use E fields,  Rp " E(x,reflected)/E(x,incident).
185 c       Consequently, in order to change to this other convention, one should
186 c       induce a minus sign (-) onto the terms of Rp.
187
188 c*      k = 2*mfilm+1
189 c*      do i=1,k
190 c*          dRp(i) = -dRp(i)
191 c*      end do
192 c*      dRpa = -dRpa
193 c*      dRpq = -dRpq
194 c*      Rp = -Rp
195
196         return
197         end
```

6.2.18 FORWR0.FOR

```
      subroutine forwr0 (qqq, angl)
      real              qqq, angl

      include   'defnit.'              ! nfilms, nrows
      include   'filmss.'              ! z, e
      include   'rstack.'              ! R, dR     -----> output results

      complex  ys(nfilms+1), eta(nfilms+1)
      complex  yp(nfilms+1), cta(nfilms+1)
      complex  ee(nfilms), em(nfilms), ep(nfilms), zq(nfilms)

      complex  half,one,  eta0,cta0
      complex  top,bot,ss,pp,  ssz,ppz,sse,ppe,ssa,ppa,  pp1,pp2,pp3

c     Solve:   the direct or forward problem.
c   Discern:   the reflection coefficient and Jacobian.
c     Given:   dielectric function of films (isotropic,homogeneous,uniform)
c              incident angle:    angl ~ radians
c                  wavenumber:    qqq  ~ 2*pi/wavelength
c                  wavelength:    nano-meters

c      Find:     Rs, Rp.      ------> (psi, delta)
c               dRs,dRp.      <------ partial wrt:   (z, dielectric function)
c              dRsa,dRpa      <------ partial wrt:  -[n(air)*sin(angle)]**2
c                             <------ partial wrt:   incident angle in radians
c              dRsq,dRpq      <------ partial wrt:   q ~ wavenumber

      half = cmplx (0.5, 0.0)
      one  = cmplx (1.0, 0.0)

      sa = sin (angl)
      ca = cos (angl)
      as = air*sa
      ac = air*ca
      as2 = as*as                                  ! air

      eta0  = cmplx ( ac, 0.0)                     ! air   TE
      cta0  = cmplx ( ca/air, 0.0)                 ! air   TM

      mfilms = mfilm+1                             ! films, substrate

      do i=1,mfilms                                ! films, substrate
          eta(i) = sqrtt (die(i)-cmplx(as2,0.0)) ! TE
          cta(i) = eta(i)/die(i)                   ! TM
      end do

c     Determine the reflection coefficients in air.
c     Method for TE:   admittance = Y = - Hx/Ey,       Rs uses E field.
c                 TM:    impedance = Z =   Ex/Hy,       Rp uses H field.
```

```
51        ys(mfilms) = eta(mfilms)                                  ! substrate
52        yp(mfilms) = cta(mfilms)
53        if (mfilm.ne.0) then
54            do i=mfilm,1,-1                                       ! backwards
55                x = zzz(i)*qqq
56                zq(i) = cmplx (0.0, x)                            ! izq
57                ss =    cmplx (0.0, x+x) * eta(i)                 ! izqn2
58                x  = exp (real (ss))
59                y  =     aimag (ss)
60                ss = cmplx (x*cos(y), x*sin(y))                   ! exp (izqn2)
61                ee(i) = ss
62                em(i) = one-ss
63                ep(i) = one+ss
64                ys(i) = eta(i)* ((em(i)*eta(i)+ep(i)*ys(i+1)) /
65       &                         (ep(i)*eta(i)+em(i)*ys(i+1)) )
66                yp(i) = cta(i)* ((em(i)*cta(i)+ep(i)*yp(i+1)) /
67       &                         (ep(i)*cta(i)+em(i)*yp(i+1)) )
68            end do
69        end if
70        Rs = (eta0-ys(1)) /(eta0+ys(1))                           ! air
71        Rp = (cta0-yp(1)) /(cta0+yp(1))                           ! air
72
73        return
74        end
```

6.2.19 DIFFER.FOR

```
1       subroutine differ (exprmt, theory, diff)            ! radians
2       real   exprmt, theory, diff
3       include  'handyy.'              ! pi
4
5
6       x1 = exprmt - theory            ! 0 .le. theory .lt. 2*pi
7       x2 = x1 - (pi+pi)               ! mod  2*pi
8       x3 = x1 + (pi+pi)
9
10      a1 = abs (x1)
11      a2 = abs (x2)
12      a3 = abs (x3)
13
14      if (a1.le.a2) then              !  a1 < a2
15          diff = x1
16          if (a3.lt.a1) then
17              diff = x3
18          end if
19      else                            !  a2 < a1
20          diff = x2
21          if (a3.lt.a2) then
22              diff = x3
23          end if
24      end if
25
26      return
27      end
```

6.2.20 STAT22.FOR

```
1       subroutine stat22 (n,b)
2       real       b(n)
3       include   'iounit.'
4       include   'handyy.'                    ! pi
5       logical    llzero
6
7   c   Perform:    simple statistics of deviations from the model.
8   c   Discern:    (mean, standard deviation)     of deviations.
9   c   Assume:      b(1) ~ d(psi  ) ~ deviation ~ experiment-model
10  c                b(2) ~ d(delta) ~ deviation ~ experiment-model
11
12      raddeg = 180.0 / pi                    ! degrees <--- radians
13      nh = n/2                               ! psi, delta
14      h = float (nh)
15
16      write (iout,111)
17      if (n.le.1 .or. n.ne.nh+nh) then
18          write (iout,112)  n
19          stop
20      end if
21
22      a1 = 0.0
23      a2 = 0.0
24      do i=1,n,2                             ! mean deviation
25          a1 = a1 + b(i  )                   !     psi
26          a2 = a2 + b(i+1)                   !     delta
27      end do
28      a1 = a1 /h                             !     psi
29      a2 = a2 /h                             !     delta
30
31      llzero = .false.                       ! zero variance
32      s1 = 0.0
33      s2 = 0.0
34      s3 = 0.0
35      do i=1,n,2                             ! variance of deviations
36          s1 = s1 + (b(i  )-a1)**2           !     psi
37          s2 = s2 + (b(i+1)-a2)**2           !     delta
38          s3 = s3 + (b(i+1)-a2)*(b(i)-a1)    !     correlation
39      end do
40      s1 = sqrt (s1 /h)                      ! standard deviation
41      s2 = sqrt (s2 /h)
42      s3 =       s3 /h                       ! covariance
43
44      if (s1.eq.0.0 .or. s2.eq.0.0) then     ! exact fit, no scatter
45          llzero = .true.
46      else
47          s3 = s3 /(s1*s2)                   ! correlation coefficient
48      end if
49
50      a1 = a1 *raddeg                        ! degrees
```

```
      a2 = a2 *raddeg
      s1 = s1 *raddeg
      s2 = s2 *raddeg

      write (iout,113)  a1,s1,  a2,s2,  s3
      if (llzero)  write (iout,114)

      return

  111 format (/  1x, 15('----')
     &        /' Statistics of deviations ' experiment-model ' g'
     &       //'     where:   g is a column vector of length 2M,'
     &        /'              () denotes either  (delta) or (psi)'
     &        /'        mean () =    m() = <g()> = (1/M) sum: g()'
     &        /'    variance () = < (g() -m() )**2 >            '
     &        /'  covariance    = < (g(1)-m(1))*(g(2)-m(2)) >   '
     &        /'        std dev = sqrt (variance)               '
     &        /'  correlat coef = covariance / (sd(1) * sd(2))  ' )

  112 format (/' stat22,   ... oops ' inconsistency, n=', i5)

  113 format (/  20x, ' mean,',  4x, ' std dev     (degrees)'
     &        /   5x, '  psi:',  4x, f10.3, 3x, f10.3
     &        /   5x, 'delta:',  4x, f10.3, 3x, f10.3
     &        /   5x,     6x,    4x, f10.3, 1x,
     &              '' correlation coefficient ' <psi|delta>    ')
  114 format (   26x, '' UNnormalized, because atleast one of the '
     &        /  26x, '                standard deviations vanish.'/)

      end
```

6.2.21 CORLAT.FOR

```
1       subroutine corlat
2
3   c   Construct the correlation matrix for model parameters (z,f,p)
4   c   for the given sampling of measurementd data from experiment.
5   c   Correlation matrix ~ normalized:   [J(Transpose)*J].
6
7       include  'iounit.'
8       include  'defnit.'
9       include  'filmmm.'
10      include  'arrays.'                          ! aa
11      include  'wstack.'                          ! aat
12      include  'handyy.'                          ! pi
13
14      real     det(2)                             ! LINPACK
15      integer  inert(3)                           ! LINPACK
16
17
18      llnorm = .false.                            ! UNscaled by experiment
19      call asmbl                                  ! ia,ja,aa,bb
20      call stat22 (meqns,bb)                      ! mean, std dev
21      call norm   (meqns,bb,bbnorm,0)             ! |<gg>|
22      bbnorm = bbnorm/sqrt (float (meqns-mvary)) ! variance, <gg>/(2M-N)
23      raddeg = 180.0/pi                           ! degree/radian
24
25      kv = 0
26      do jv=1,mvary                               ! A(T)*A,        parameters
27          do iv=1,jv                              ! upper triangle + diagonal
28              sum = 0.0
29              do i=1,meqns,2
30                  j1 = ia(i  )
31                  j2 = ia(i+1)-1
32                  mv = j2-j1+1
33                  s1 = 0.0
34                  s2 = 0.0
35                  s3 = 0.0
36                  s4 = 0.0
37                  do j=j1,j2                      ! within a row of A
38                      jaj = ja(j)                 ! column
39                      if (jaj.eq.iv) then
40                          s1 = aa(j   )
41                          s3 = aa(j+mv)
42                      end if
43                      if (jaj.eq.jv) then
44                          s2 = aa(j   )
45                          s4 = aa(j+mv)
46                      end if
47                  end do
48                  sum = sum + s1*s2 + s3*s4       ! dot product
49              end do
50              sum = sum/float (meqns)             ! convenience
```

```
51                kv = kv+1                              ! packed format ' storage
52                aat (kv) = sum                         ! upper triangle + diagonal
53                if (jv.eq.iv) then                     ! diagonal
54                    xx(jv) = sqrt (sum)
55                end if
56            end do
57        end do
58
59        kv = 0
60        do jv=1,mvary                                  ! correlation matrix
61            do iv=1,jv                                 ! renormalize
62                kv = kv+1
63                aat(kv) = aat(kv)/(xx(jv)*xx(iv))
64            end do
65        end do
66
67        write (iout,211)
68        k2 = 0
69        do i=1,mvary
70           k1 = k2+1
71           k2 = k2+i
72           write (iout,212)  i, (aat(k), k=k1,k2)
73        end do
74        write (iout,214)
75        write (iout,215)  (xx(i), i=1,mvary)           ! scaling coefficients
76
77  c     ------------------------------------------------------
78  c     LINPACK, Chapter 5, Solving symmetric indefinite matrices.
79
80        call sspco (aat, mvary, ipvt, rcond, w)        ! UD*T(U)  decomposition
81        write (iout,216)  rcond                        ! inverse condition number
82
83  c*    if (1.0 .eq. 1.0+rcond     ) then             !     singular matrix
84        if (1.0 .ne. 1.0+rcond*0.01) then             ! NOT ill-conditioned
85            call sspdi (aat, mvary, ipvt, det, inert, w, 111)
86            write (iout,217)  det, inert
87            k2 = 0
88            do i=1,mvary
89               k1 = k2+1
90               k2 = k2+i
91               write (iout,213)  i, (aat(k), k=k1,k2)
92            end do
93        else                                           ! indeterminate
94            return                                     ! no reason to continue
95        end if
96
97  c     ------------------------------------------------------
98  c     Estimate the uncertainty in the model parameters.
99  c     Ignore correlation among model parameters,
100 c     i.e., consider only the diagonal contributions.
101
102       k2 = 0
103       do i=1,mvary
```

```
104           k1 = k2+1
105           k2 = k2+i                                          ! diagonal
106           xx(i) = bbnorm*sqrt (aat(k2)) /xx(i)
107        end do
108        bbnorm = bbnorm*raddeg                                ! degrees
109
110        write (iout,121)  bbnorm
111        do i=1,mvary
112            j = iptu(i)
113            if (j .le. mfilmz) then                           ! widths
114                write (iout,122)  i, widths(j), xx(i), j
115            else if (j .le. mfilmz+mlmnts) then               ! fractions
116                j = j-mfilmz
117                write (iout,123)  i, fflmnt(j), xx(i), j
118            else                                              ! parameters
119                j = j-mfilmz-mlmnts
120                write (iout,124)  i, rrparm(j), xx(i), j
121            end if
122        end do
123
124        return
125
126     121 format (/  1x, 15('----')
127        &        /' The standard deviation of the residuals.'
128        &       //' s(g) = sqrt [<gg>/(2M-N)] =', 1p1e14.5,'  (degrees)'
129        &      ///' The estimated uncertainty in the model parameters, '
130        &        /' assuming no correlation, i.e., diagonal terms only.'
131        &       //  17x, 'value,'       3x, 'uncertainty.'            )
132     122 format (1x, i5, ')', f15.4, f15.5, ',   for:', i5, ' ~ (z,zu)')
133     123 format (1x, i5, ')', f15.4, f15.5, ',   for:', i5, ' ~ (f,fu)')
134     124 format (1x, i5, ')', f15.4, f15.5, ',   for:', i5, ' ~ (p,pu)')
135
136     211 format (/' [J(T)*J],    renormalized for correlation.')
137     212 format (1x, i4, ')', 1x,   10f10.5, : /(7x, 10f10.5))
138     213 format (1x, i4, ')', 1x, 1p10e10.2, : /(7x, 10e10.2))
139     214 format (/' Normalization coefficients, sqrt:    [J(T)*J]_(i,i)')
140     215 format ( 1x, 1p10e10.2)
141
142     216 format (/' rcond = ', 1pe10.2, ',  condition number,  [J(T)*J]')
143     217 format (/' [J(T)*J]**(-1):    '
144        &        /'    Determinant:    ', f8.4, ' E ', f8.4
145        &        /'        Inertia:   (', 3i4,  '), ',
146        &                             ' number of (+,-,0) eigenvalues.'
147        &        /'        Inverse:    upper+diagonal  matrix')
148
149        end
```

6.3 General Utilities

6.3.1 DOT.FOR

```
1       subroutine dot (n,x,y,  xy,k)
2       real       x(1), y(1)
3
4       xy = 0.0
5       xn = 0.0
6       yn = 0.0
7       do i=1,n                ! find maximum
8           xn = amax1 (xn, abs (x(i)))
9           yn = amax1 (yn, abs (y(i)))
10      end do
11      if (xn.eq.0.0 .or. yn.eq.0.0)  return
12
13      xs = 0.0
14      ys = 0.0
15      do i=1,n
16          xx = x(i)/xn        ! scaled vector component
17          yy = y(i)/yn
18          xs = xs + xx*xx     ! dot product = |x*x|
19          ys = ys + yy*yy     ! dot product = |y*y|
20          xy = xy + xx*yy     ! dot product = |x*y|
21      end do
22
23      h  = float (n)
24      xs = xs/h               ! mean square value
25      ys = ys/h
26      xy = xy/h
27      xs = sqrt (xs)          ! root mean square value
28      ys = sqrt (ys)
29
30      if (k.eq.0) then        ! usual dot product
31          xy = xy*xn*yn*h
32      else if (k.eq.1) then   ! un-normalized
33          xy = xy*xn*yn
34      else                    !    normalized
35          xy = xy/(xs*ys)
36      end if
37
38      return
39      end
```

6.3.2 NORM.FOR

```
1       subroutine norm (n,x, xn,k)
2       real  x(1)
3
4       xn = 0.0
5       do i=1,n                          !  find maximum
6          xn = amax1 (xn, abs (x(i)))
7       end do
8       if (xn .eq. 0.0) return
9
10      xx = 0.0
11      do i=1,n                          !  dot product
12          xx = xx + (x(i)/xn)**2        !  |x*x|
13      end do
14
15      if (k.eq.1) then                  ! mean squared value
16          xx = xx/float (n)
17      end if
18      xn = xn*sqrt (xx)                 !  Euclidean norm
19
20      return
21      end
```

6.3.3 APROD.FOR

```
      subroutine aprod (mode,m,n,x,y,  ia,ja,aa)
      integer    mode, m, n, ia(1), ja(1)
      real       x(n), y(m), aa(1)
      data       iout / 6 /

c     ------------------------------------------------------------------
c     A ~ A(m,n),                Transpose operator ~ (')
c     Operation:     mode= 1,    set:   y  = y  + A *x
c                    mode= 2,    set:   x  = x  + A'*y
c                                       x' = x' + y'*A
c     ------------------------------------------------------------------

      if (mode.eq.1) then              ! y = y + Ax
          do i=1,m                     ! scan rows of matrix A
              mj = ia(i+1)-ia(i)       ! number of columns in row
              if (mj.ne.0) then
                  ss = 0.0             ! sum
                  jj = ia(i)-1         ! indexing
                  do j=1,mj            ! scan columns of row
                      jj = jj+1
                      kk = ja(jj)      ! column
                      ss = ss + aa(jj)*x(kk)
                  end do
                  y(i) = y(i)+ss       ! y = y+Ax
              end if
          end do

      else if (mode.eq.2) then         ! x' = x' + y'*A
          do i=1,m                     ! scan rows of matrix A
              mj = ia(i+1)-ia(i)       ! number of columns in row
              if (mj.ne.0) then
                  yy = y(i)
                  jj = ia(i)-1         ! indexing
                  do j=1,mj            ! scan columns of row
                      jj = jj+1
                      kk = ja(jj)      ! column
                      x(kk) = x(kk) + yy*aa(jj)
                  end do
              end if
          end do

      else
          write (iout,10)  mode
          stop
      end if

      return

   10 format (/' aprod,  ... error,  mode= ', i2)
      end
```

6.3.4 SCALII.FOR

```
      subroutine scalii (m,n,  ia,ja,a,b,  p,k)
      integer    ia(1), ja(1), m, n, k
      real       a(1), b(1), p(1)

c     ---------------------------------------------------------------
c     Scale the rows in matrix,       A(m,n), b(m).
c     Retain scaling coefficients in  P(m).
c     Evaluate  P  only when    k  = 0,1,2,3.
c     Re-scale  A  only when   |k| =     2,3.
c     Re-scale  B  only when   |k| =   1,  3.
c     Matrix A is stored (row-wise) in the Yale Sparse Matrix Format.
c     ---------------------------------------------------------------

      ka = iabs (k)                                 ! convenience
      if (ka.gt.3) then                             ! out of range
          do i=1,m                                  ! scan rows
              p(i) = 1.0                            ! default
          end do
          return
      end if

      if (k.ge.0) then                              ! determine scaling
          do i=1,m                                  ! scan rows
              mj = ia(i+1)-ia(i)                    ! number of columns in row
              if (mj.ne.0) then
                  big = 0.0                         ! initialize
                  jj = ia(i)-1                      ! indexing
                  do j=1,mj                         ! scan columns in row
                      jj = jj+1
                      big = amax1 (big, abs (a(jj)))
                  end do
                  if (big .eq. 0.0) then            ! trivial case
                      p(i) = 1.0                    ! default
                  else
                      ss = 0.0                      ! initialize
                      jj = jj-mj                    ! reset indexing
                      do j=1,mj                     ! scan columns in row
                          jj = jj+1
                          ss = ss + (a(jj)/big)**2
                      end do
                      p(i) = big*sqrt (ss)          ! scale factor
                  end if
              end if
          end do
      end if

      if (ka.eq.2 .or. ka.eq.3) then                ! rescale A
          do i=1,m                                  ! scan rows, A
              mj = ia(i+1)-ia(i)                    ! number of columns in row
              if (mj.ne.0) then
```

```
51                  pp = p(i)                       ! scale factor of row
52                  jj = ia(i)-1                    ! indexing
53                  do j=1,mj                       ! scan columns in row
54                      jj = jj+1
55                      a(jj) = a(jj)/pp            ! rescale the row
56                  end do
57              end if
58          end do
59      end if
60
61      if (ka.eq.1 .or. ka.eq.3) then              ! rescale b
62          do i=1,m                                ! scan rows
63              b(i) = b(i)/p(i)
64          end do
65      end if
66
67      return
68      end
```

6.3.5 SCALJJ.FOR

```
      subroutine  scaljj (m,n,  ia,ja,a,b,  p,w,k)
      integer     ia(1), ja(1), k, m, n
      real        a(1), b(1), p(1), w(1)

c     ------------------------------------------------------------------
c     Scale matrix A so the diagonal of:      [Transpose(A)*A] = 1.
c     Matrix A is stored row-wise in the Yale Sparse Matrix Format.
c     Scale the columns of matrices,  A(m,n), b(n).
c     Retain the scaling coefficients in      P(n).
c     Use as a dummy work storage array,      W(n).
c     Evaluate  P  only when     k  = 0,1,2,3.
c     Re-scale  A  only when    |k| =     2,3.
c     Re-scale  B  only when    |k| =   1,  3.
c     ------------------------------------------------------------------

      ka = iabs (k)                                 ! convenience
      if (ka.gt.3) then                             ! out of range
          do j=1,n                                  ! scan columns
              p(j) = 1.0                            !      default
          end do
          return
      end if

      if (k.ge.0) then                              ! determine scaling
          do j=1,n                                  ! scan columns
              p(j) = 0.0                            ! initialize
              w(j) = 0.0                            ! sums
          end do

          do i=1,m                                  ! scan rows, A
              mj = ia(i+1)-ia(i)                    ! number of columns in row
              if (mj.ne.0) then
                  jj = ia(i)-1                      ! indexing
                  do j=1,mj                         ! scan columns of row
                      jj = jj+1
                      kk = ja(jj)                   ! column
                      p(kk) = amax1 (p(kk), abs (a(jj)))   ! max |A(,j)|
                  end do
              end if
          end do

          do i=1,m                                  ! scan rows, A
              mj = ia(i+1)-ia(i)                    ! number of columns
              if (mj.ne.0) then
                  jj = ia(i)-1                      ! indexing
                  do j=1,mj                         ! scan columns in row
                      jj = jj+1
                      kk = ja(jj)                               ! column
                      w(kk) = w(kk) + (a(jj)/p(kk))**2          ! sums
                  end do
```

```
51              end if
52          end do
53
54          do j=1,n                                ! scan columns
55              p(j) = p(j)*sqrt (w(j))             ! retain scale factor
56          end do
57      end if
58
59      if (ka.eq.2 .or. ka.eq.3) then              ! rescale A
60          do i=1,m                                ! scan rows, A
61              mj = ia(i+1)-ia(i)                  ! number of columns
62              if (mj.ne.0) then
63                  jj = ia(i)-1                    ! indexing
64                  do j=1,mj                       ! scan columns in row
65                      jj = jj+1
66                      kk = ja(jj)                 ! column
67                      a(jj) = a(jj)/p(kk)         ! rescale column
68                  end do
69              end if
70          end do
71      end if
72
73      if (ka.eq.1 .or. ka.eq.3) then              ! rescale b
74          do j=1,n                                ! scan columns
75              b(j) = b(j)/p(j)                    ! rescale column
76          end do
77      end if
78
79      return
80      end
```

6.3.6 CGNL.FOR

```
      subroutine cgnl (ma,na,ia,ja,aa,  b,x,
     &                 itmax,   u,v,w,  xx,se)

      integer    ia(1), ja(1)
      real       aa(1), b(1), x(1)
      real       u(1), v(1), w(1), xx(1), se(1)
      external   aprod
      data       iout /6/

c     ------------------------------------------------------------------
c     real       b(ma), u(ma),              aa(ma,na)
c     real       x(na), v(na), w(na), xx(na), se(na)

c     Solve the linear or matrix algebra problem, Ax=b.
c     Matrix A is stored row-wise in the Yale Sparse Matrix Format.
c     Reference:     C.C.Paige and M.A.Saunders,
c        ''LSQR:     An Algorithm for Sparse Linear Equations
c                    and Sparse Least Squares'',
c              Association for Computing Machinery,
c              Transactions on Mathematical Software,
c              Volume 8, Number 1, March 1982, pp.  43-71, (Note pp. 50-51).
c     ibid.,   Volume 8, Number 2,  June 1982, pp. 195-209.
c     ------------------------------------------------------------------

      loop = 0                                        ! initialize
    1 loop = loop+1                                   ! update counter
      rr = 0.0                                        ! norm of residual
      do i=1,ma                                       ! scan rows,  r=b-Ax
          mj = ia(i+1)-ia(i)                          ! number of columns in row
          if (mj .ne. 0) then
              ss = 0.0                                ! initialize sum
              jj = ia(i)-1                            ! indexing
              do j=1,mj                               ! scan columns of row
                  jj = jj+1
                  kk = ja(jj)                         ! column
                  ss = ss + aa(jj)*x(kk)              ! Ax
             end do
             ss = b(i)-ss                             !  r = b-Ax = residual
             u(i) = ss                                !  r        = residual
             rr = rr+ss*ss                            ! |r|**2
          end if
      end do
      if (rr .eq. 0.0) return

      if (loop .eq. 1) then                           ! retain first norm
          rrr = rr
      else if (rr .ge. rrlast*0.98) then              ! rate of convergence
c*        ratio = sqrt (rr/rrr)
c*        write (iout,103)  loop,  ratio
c*        write (iout,104)  istop, anorm, acond, rnorm, arnorm, xnorm
```

```
51            return
52         end if
53         rrlast = rr                                  ! update
54
55   c------------------------------------------------ ! Set-up for LSQR
56         relpr =  1.0E-06  ! relative precision of floating point arithmetic
57         damp =   1.0E+00  !
58         atol =   1.0E-06  ! relative error of data in A
59         btol =   1.0E-06  ! relative error of data in B * rhs
60         conlim = 1.0E+04  ! apparent condition number of matrix A-bar
61                           ! (upper limit)
62         itnlim = itmax    ! upper limit on number of iterations
63         nout   = -iout    ! output index to printer
64
65         call lsqr (ma, na, aprod, damp,
66        &           ia, ja, aa,
67        &           u, v, w, xx, se,
68        &           atol, btol, conlim, itnlim, nout,
69        &           istop, anorm, acond, rnorm, arnorm, xnorm)
70
71         do i=1,na
72             x(i) = x(i) + xx(i)          ! update solution
73             xx(i) = 0.0                  ! reset
74         end do
75         goto 1                           ! loop back
76
77     101 format (' cgnl, singular    row= ', i10)
78     102 format (' cgnl, singular column= ', i10)
79     103 format (' cgnl,  ', i5, 1p1e11.3, ' * loop, ratio '
80        &                                , '(residual reduction)')
81     104 format (' cgnl,  ', i5, 1p5e11.3)
82         end
```

6.3.7 LSQR.FOR

The following subroutine is located in the software library,

> *Guide to Available Mathematical Software*, by R. F. Boisvert, S. E. Howe, and D. K. Kahaner, Center for Applied Mathematics, National Institute of Standards and Technology (formerly National Bureau of Standards), U.S. Department of Commerce, 1985.

The source code is copyrighted by the Association for Computing Machinery, Inc. The references for the following source code include:

1) C. C. Paige and M. A. Saunders, "LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares," *ACM Trans. Math. Softw.* 8, 43–71 (1982).

2) C. C. Paige and M. A. Saunders, "Algorithm 583, LSQR: Sparse Linear Equations and Least Squares Problems," *ACM Trans. Math. Softw.* 8, 195–209 (1982).

```
1       SUBROUTINE LSQR (M,N,APROD,DAMP,
2      1                IA,JA,AA,
3      2                U,V,W,X,SE,
4      3                ATOL,BTOL,CONLIM,ITNLIM,NOUT,
5      4                ISTOP,ANORM,ACOND,RNORM,ARNORM,XNORM )
6   C
7       EXTERNAL   APROD
8       INTEGER    M,N,ITNLIM,NOUT,ISTOP
9       INTEGER    IA(M),JA(M)
10      REAL       AA(M),                    U(M),V(N),W(N),X(N),SE(N),
11     1           ATOL,BTOL,CONLIM,DAMP,ANORM,ACOND,RNORM,ARNORM,XNORM
12  C    ------------------------------------------------------------------
13  C
14  C    LSQR  FINDS A SOLUTION  X  TO THE FOLLOWING PROBLEMS...
15  C
16  C    1. UNSYMMETRIC EQUATIONS --    SOLVE  A*X = B
17  C
18  C    2. LINEAR LEAST SQUARES  --    SOLVE  A*X = B
19  C                                   IN THE LEAST-SQUARES SENSE
20  C
21  C    3. DAMPED LEAST SQUARES  --    SOLVE  (   A    )*X = ( B )
22  C                                          ( DAMP*I )     ( 0 )
23  C                                   IN THE LEAST-SQUARES SENSE
24  C
25  C    WHERE  A  IS A MATRIX WITH  M  ROWS AND  N  COLUMNS,
26  C           B  IS AN M-VECTOR, AND
27  C       DAMP  IS A SCALAR (ALL QUANTITIES REAL).
28  C    THE MATRIX  A  IS INTENDED TO BE LARGE AND SPARSE.
29  C    IT IS ACCESSED BY MEANS OF SUBROUTINE CALLS OF THE FORM
```

```
C
C                CALL APROD ( MODE,M,N,X,Y,  IA,JA,AA )
C
C     WHICH MUST PERFORM THE FOLLOWING FUNCTIONS...
C
C                IF MODE = 1, COMPUTE  Y = Y + A*X.
C                IF MODE = 2, COMPUTE  X = X + A(TRANSPOSE)*Y.
C
C     THE VECTORS X AND Y ARE INPUT PARAMETERS IN BOTH CASES.
C     IF MODE = 1, Y SHOULD BE ALTERED WITHOUT CHANGING X.
C     IF MODE = 2, X SHOULD BE ALTERED WITHOUT CHANGING Y.
C     THE PARAMETERS:   IA, JA, AA.
C     MAY BE USED FOR WORKSPACE AS DESCRIBED BELOW.
C
C     THE RHS VECTOR  B  IS INPUT VIA  U,  AND SUBSEQUENTLY OVERWRITTEN.
C
C
C     NOTE.  LSQR USES AN ITERATIVE METHOD TO APPROXIMATE THE SOLUTION.
C     THE NUMBER OF ITERATIONS REQUIRED TO REACH A CERTAIN ACCURACY
C     DEPENDS STRONGLY ON THE SCALING OF THE PROBLEM.  POOR SCALING OF
C     THE ROWS OR COLUMNS OF  A  SHOULD THEREFORE BE AVOIDED WHERE
C     POSSIBLE.
C
C     FOR EXAMPLE, IN PROBLEM 1 THE SOLUTION IS UNALTERED BY
C     ROW-SCALING.  IF A ROW OF  A  IS VERY SMALL OR LARGE COMPARED TO
C     THE OTHER ROWS OF  A,  THE CORRESPONDING ROW OF  ( A  B )  SHOULD
C     BE SCALED UP OR DOWN.
C
C     IN PROBLEMS 1 AND 2, THE SOLUTION  X  IS EASILY RECOVERED
C     FOLLOWING COLUMN-SCALING.  IN THE ABSENCE OF BETTER INFORMATION,
C     THE NONZERO COLUMNS OF  A  SHOULD BE SCALED SO THAT THEY ALL HAVE
C     THE SAME EUCLIDEAN NORM (E.G.  1.0).
C
C     IN PROBLEM 3, THERE IS NO FREEDOM TO RE-SCALE IF  DAMP  IS
C     NONZERO.  HOWEVER, THE VALUE OF  DAMP  SHOULD BE ASSIGNED ONLY
C     AFTER ATTENTION HAS BEEN PAID TO THE SCALING OF  A.
C
C     THE PARAMETER  DAMP  IS INTENDED TO HELP REGULARIZE
C     ILL-CONDITIONED SYSTEMS, BY PREVENTING THE TRUE SOLUTION FROM
C     BEING VERY LARGE.  ANOTHER AID TO REGULARIZATION IS PROVIDED BY
C     THE PARAMETER  ACOND,  WHICH MAY BE USED TO TERMINATE ITERATIONS
C     BEFORE THE COMPUTED SOLUTION BECOMES VERY LARGE.
C
C
C     NOTATION
C     --------
C
C     THE FOLLOWING QUANTITIES ARE USED IN DISCUSSING THE SUBROUTINE
C     PARAMETERS...
C
C     ABAR   =  (   A    ),          BBAR  =  ( B )
C               ( DAMP*I )                   ( 0 )
C
```

```
C     R      = B  -  A*X,           RBAR  =  BBAR  -  ABAR*X
C
C     RNORM  =  SQRT( NORM(R)**2  +  DAMP**2 * NORM(X)**2 )
C            =  NORM( RBAR )
C
C     RELPR  =  THE RELATIVE PRECISION OF FLOATING-POINT ARITHMETIC
C               ON THE MACHINE BEING USED.  FOR EXAMPLE, ON THE IBM 370,
C               RELPR  IS ABOUT 1.0E-6 AND 1.0D-16 IN SINGLE AND DOUBLE
C               PRECISION RESPECTIVELY.
C
C     LSQR  MINIMIZES THE FUNCTION  RNORM  WITH RESPECT TO  X.
C
C
C     PARAMETERS
C     ----------
C
C     M        INPUT      THE NUMBER OF ROWS IN  A.
C
C     N        INPUT      THE NUMBER OF COLUMNS IN  A.
C
C     APROD    EXTERNAL   SEE ABOVE.
C
C     DAMP     INPUT      THE DAMPING PARAMETER FOR PROBLEM 3 ABOVE.
C                         (DAMP  SHOULD BE 0.0 FOR PROBLEMS 1 AND 2.)
C                         IF THE SYSTEM  A*X = B  IS INCOMPATIBLE, VALUES
C                         OF  DAMP  IN THE RANGE 0 TO SQRT(RELPR)*NORM(A)
C                         WILL PROBABLY HAVE A NEGLIGIBLE EFFECT.
C                         LARGER VALUES OF  DAMP  WILL TEND TO DECREASE
C                         THE NORM OF  X  AND TO REDUCE THE NUMBER OF
C                         ITERATIONS REQUIRED BY LSQR.
C
C                         THE WORK PER ITERATION AND THE STORAGE NEEDED
C                         BY LSQR ARE THE SAME FOR ALL VALUES OF  DAMP.
C
C     IA       INPUT      CONTAINS ROW    INFORMATION OF       ARRAY   A.
C     JA       INPUT      CONTAINS COLUMN INFORMATION OF A ROW WITHIN   A.
C     AA       INPUT      THE  A  ARRAY.
C
C              NOTE.  LSQR DOES NOT EXPLICITLY USE THE PREVIOUS FOUR
C              PARAMETERS, BUT PASSES THEM TO SUBROUTINE APROD FOR
C              POSSIBLE USE AS WORKSPACE.  IF APROD DOES NOT NEED
C              IW  OR  RW,  THE VALUES  LENIW = 1  OR  LENRW = 1  SHOULD
C              BE USED, AND THE ACTUAL PARAMETERS CORRESPONDING TO
C              IW  OR  RW  MAY BE ANY CONVENIENT ARRAY OF SUITABLE TYPE.
C
C     U(M)     INPUT      THE RHS VECTOR  B.  BEWARE THAT  U  IS
C                         OVER-WRITTEN BY LSQR.
C
C     V(N)     WORKSPACE
C     W(N)     WORKSPACE
C
C     X(N)     OUTPUT     RETURNS THE COMPUTED SOLUTION  X.
C
```

```
C     SE(N)   OUTPUT     RETURNS STANDARD ERROR ESTIMATES FOR THE
C                        COMPONENTS OF  X.  FOR EACH  I,  SE(I)  IS SET
C                        TO THE VALUE  RNORM * SQRT( SIGMA(I,I) / T ),
C                        WHERE  SIGMA(I,I)  IS AN ESTIMATE OF THE I-TH
C                        DIAGONAL OF THE INVERSE OF ABAR(TRANSPOSE)*ABAR
C                        AND  T = 1      IF  M .LE. N,
C                             T = M - N  IF  M .GT. N  AND  DAMP = 0,
C                             T = M      IF  DAMP .NE. 0.
C
C     ATOL    INPUT      AN ESTIMATE OF THE RELATIVE ERROR IN THE DATA
C                        DEFINING THE MATRIX  A.  FOR EXAMPLE,
C                        IF  A  IS ACCURATE TO ABOUT 6 DIGITS, SET
C                        ATOL = 1.0E-6 .
C
C     BTOL    INPUT      AN ESTIMATE OF THE RELATIVE ERROR IN THE DATA
C                        DEFINING THE RHS VECTOR  B.  FOR EXAMPLE,
C                        IF  B  IS ACCURATE TO ABOUT 6 DIGITS, SET
C                        BTOL = 1.0E-6 .
C
C     CONLIM  INPUT      AN UPPER LIMIT ON  COND(ABAR),  THE APPARENT
C                        CONDITION NUMBER OF THE MATRIX  ABAR.
C                        ITERATIONS WILL BE TERMINATED IF A COMPUTED
C                        ESTIMATE OF  COND(ABAR)  EXCEEDS  CONLIM.
C                        THIS IS INTENDED TO PREVENT CERTAIN SMALL OR
C                        ZERO SINGULAR VALUES OF  A  OR  ABAR  FROM
C                        COMING INTO EFFECT AND CAUSING UNWANTED GROWTH
C                        IN THE COMPUTED SOLUTION.
C
C                        CONLIM  AND  DAMP  MAY BE USED SEPARATELY OR
C                        TOGETHER TO REGULARIZE ILL-CONDITIONED SYSTEMS.
C
C                        NORMALLY,  CONLIM  SHOULD BE IN THE RANGE
C                        1000  TO  1/RELPR.
C                        SUGGESTED VALUE --
C                        CONLIM = 1/(100*RELPR)  FOR COMPATIBLE SYSTEMS,
C                        CONLIM = 1/(10*SQRT(RELPR))  FOR LEAST SQUARES.
C
C              NOTE.  IF THE USER IS NOT CONCERNED ABOUT THE PARAMETERS
C              ATOL, BTOL  AND  CONLIM,  ANY OR ALL OF THEM MAY BE SET
C              TO ZERO.  THE EFFECT WILL BE THE SAME AS THE VALUES
C              RELPR, RELPR  AND  1/RELPR  RESPECTIVELY.
C
C     ITNLIM  INPUT      AN UPPER LIMIT ON THE NUMBER OF ITERATIONS.
C                        SUGGESTED VALUE --
C                        ITNLIM = N/2     FOR WELL CONDITIONED SYSTEMS,
C                        ITNLIM = 4*N     OTHERWISE.
C
C     NOUT    INPUT      FILE NUMBER FOR PRINTER.  IF POSITIVE,
C                        A SUMMARY WILL BE PRINTED ON FILE  NOUT.
C
C     ISTOP   OUTPUT     AN INTEGER GIVING THE REASON FOR TERMINATION...
C
C                0       X = 0  IS THE EXACT SOLUTION.
```

```
C                       NO ITERATIONS WERE PERFORMED.
C
C                  1    THE EQUATIONS  A*X = B  ARE PROBABLY
C                       COMPATIBLE.  NORM(A*X - B)  IS SUFFICIENTLY
C                       SMALL, GIVEN THE VALUES OF  ATOL  AND  BTOL.
C
C                  2    THE SYSTEM  A*X = B  IS PROBABLY NOT
C                       COMPATIBLE.  A LEAST-SQUARES SOLUTION HAS
C                       BEEN OBTAINED WHICH IS SUFFICIENTLY ACCURATE,
C                       GIVEN THE VALUE OF  ATOL.
C
C                  3    AN ESTIMATE OF  COND(ABAR)  HAS EXCEEDED
C                       CONLIM.  THE SYSTEM  A*X = B  APPEARS TO BE
C                       ILL-CONDITIONED.  OTHERWISE, THERE COULD BE AN
C                       AN ERROR IN SUBROUTINE APROD.
C
C                  4    THE EQUATIONS  A*X = B  ARE PROBABLY
C                       COMPATIBLE.  NORM(A*X - B)  IS AS SMALL AS
C                       SEEMS REASONABLE ON THIS MACHINE.
C
C                  5    THE SYSTEM  A*X = B  IS PROBABLY NOT
C                       COMPATIBLE.  A LEAST-SQUARES SOLUTION HAS
C                       BEEN OBTAINED WHICH IS AS ACCURATE AS SEEMS
C                       REASONABLE ON THIS MACHINE.
C
C                  6    COND(ABAR)  SEEMS TO BE SO LARGE THAT THERE IS
C                       NOT MUCH POINT IN DOING FURTHER ITERATIONS,
C                       GIVEN THE PRECISION OF THIS MACHINE.
C                       THERE COULD BE AN ERROR IN SUBROUTINE APROD.
C
C                  7    THE ITERATION LIMIT  ITNLIM  WAS REACHED.
C
C     ANORM   OUTPUT    AN ESTIMATE OF THE FROBENIUS NORM OF  ABAR.
C                       THIS IS THE SQUARE-ROOT OF THE SUM OF SQUARES
C                       OF THE ELEMENTS OF  ABAR.
C                       IF  DAMP  IS SMALL AND IF THE COLUMNS OF  A
C                       HAVE ALL BEEN SCALED TO HAVE LENGTH  1.0,
C                       ANORM  SHOULD INCREASE TO ROUGHLY  SQRT(N).
C                       A RADICALLY DIFFERENT VALUE FOR  ANORM  MAY
C                       INDICATE AN ERROR IN SUBROUTINE APROD (THERE
C                       MAY BE AN INCONSISTENCY BETWEEN MODES 1 AND 2).
C
C     ACOND   OUTPUT    AN ESTIMATE OF  COND(ABAR),  THE CONDITION
C                       NUMBER OF  ABAR.  A VERY HIGH VALUE OF  ACOND
C                       MAY AGAIN INDICATE AN ERROR IN APROD.
C
C     RNORM   OUTPUT    AN ESTIMATE OF THE FINAL VALUE OF NORM(RBAR),
C                       THE FUNCTION BEING MINIMIZED (SEE NOTATION
C                       ABOVE).  THIS WILL BE SMALL IF  A*X = B  HAS
C                       A SOLUTION.
C
C     ARNORM  OUTPUT    AN ESTIMATE OF THE FINAL VALUE OF
C                       NORM( ABAR(TRANSPOSE)*RBAR ), THE NORM OF
```

```
C                       THE RESIDUAL FOR THE USUAL NORMAL EQUATIONS.
C                       THIS SHOULD BE SMALL IN ALL CASES.  (ARNORM
C                       WILL OFTEN BE SMALLER THAN THE TRUE VALUE
C                       COMPUTED FROM THE OUTPUT VECTOR  X.)
C
C     XNORM   OUTPUT    AN ESTIMATE OF THE NORM OF THE FINAL
C                       SOLUTION VECTOR  X.
C
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C
C     USER        APROD
C     LSQR        NORMLZ
C     BLAS        SCOPY,SNRM2,SSCAL  (SEE LAWSON ET AL. BELOW)
C                 (SNRM2 IS USED ONLY IN NORMLZ)
C     FORTRAN     ABS,MOD,SQRT
C
C
C     PRECISION
C     ---------
C
C     THE NUMBER OF ITERATIONS REQUIRED BY LSQR WILL USUALLY DECREASE
C     IF THE COMPUTATION IS PERFORMED IN HIGHER PRECISION.  TO CONVERT
C     LSQR  AND  NORMLZ  BETWEEN SINGLE- AND DOUBLE-PRECISION, CHANGE
C     THE WORDS
C                 SCOPY, SNRM2, SSCAL
C                 ABS, REAL, SQRT
C     TO THE APPROPRIATE BLAS AND FORTRAN EQUIVALENTS.
C
C
C     REFERENCES
C     ----------
C
C     PAIGE, C.C. AND SAUNDERS, M.A.  LSQR@D AN ALGORITHM FOR SPARSE
C        LINEAR EQUATIONS AND SPARSE LEAST SQUARES.
C        ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE 8, 1 (MARCH 1982).
C
C     LAWSON, C.L., HANSON, R.J., KINCAID, D.R. AND KROGH, F.T.
C        BASIC LINEAR ALGEBRA SUBPROGRAMS FOR FORTRAN USAGE.
C        ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE 5, 3 (SEPT 1979),
C        308-323 AND 324-325.
C
C
C     LSQR.       THIS VERSION DATED 22 FEBRUARY 1982.
C     ------------------------------------------------------------------
C
C     FUNCTIONS AND LOCAL VARIABLES
C
      INTEGER     I,ITN,MOD,NCONV,NSTOP
      REAL        ABS,SQRT
      REAL        ALFA,BBNORM,BETA,BNORM,
     1            CS,CS1,CS2,CTOL,DAMPSQ,DDNORM,DELTA,
```

```
     2            GAMMA,GAMBAR,ONE,PHI,PHIBAR,PSI,
     3            RES1,RES2,RHO,RHOBAR,RHBAR1,RHBAR2,RHS,RTOL,
     4            SN,SN1,SN2,T,TAU,TEST1,TEST2,TEST3,
     5            THETA,T1,T2,T3,XXNORM,Z,ZBAR,ZERO
C
C
C     INITIALIZE.
C
      IF (NOUT .GT. 0)
     1   WRITE(NOUT, 1000) M,N,DAMP,ATOL,CONLIM,BTOL,ITNLIM
      ZERO   = 0.0
      ONE    = 1.0
      CTOL   = ZERO
      IF (CONLIM .GT. ZERO) CTOL = ONE/CONLIM
      DAMPSQ = DAMP**2
      ANORM  = ZERO
      ACOND  = ZERO
      BBNORM = ZERO
      DDNORM = ZERO
      RES2   = ZERO
      XNORM  = ZERO
      XXNORM = ZERO
      CS2    = -ONE
      SN2    = ZERO
      Z      = ZERO
      ITN    = 0
      ISTOP  = 0
      NSTOP  = 0
C
      DO 10 I = 1, N
         V(I) = ZERO
         X(I) = ZERO
        SE(I) = ZERO
   10 CONTINUE
C
C     SET UP THE FIRST VECTORS FOR THE BIDIAGONALIZATION.
C     THESE SATISFY   BETA*U = B,   ALFA*V = A(TRANSPOSE)*U.
C
      CALL NORMLZ( M,U,BETA )
      CALL APROD ( 2,M,N,V,U,  IA,JA,AA )
      CALL NORMLZ( N,V,ALFA )
      CALL SCOPY ( N,V,1,W,1 )
C
      RHOBAR = ALFA
      PHIBAR = BETA
      BNORM  = BETA
      RNORM  = BETA
      ARNORM = ALFA*BETA
      IF (ARNORM .LE. ZERO) GO TO 800
      IF (NOUT   .LE.  0  ) GO TO 100
      IF (DAMPSQ .LE. ZERO) WRITE(NOUT, 1200)
      IF (DAMPSQ .GT. ZERO) WRITE(NOUT, 1300)
      TEST1  = ONE
```

```
      TEST2  = ALFA/BETA
      WRITE(NOUT, 1500) ITN,X(1),RNORM,TEST1,TEST2
      WRITE(NOUT, 1600)
C
C     ------------------------------------------------------------------
C     MAIN ITERATION LOOP.
C     ------------------------------------------------------------------
  100 ITN = ITN + 1
C
C     PERFORM THE NEXT STEP OF THE BIDIAGONALIZATION TO OBTAIN THE
C     NEXT  BETA, U, ALFA, V.  THESE SATISFY THE RELATIONS
C               BETA*U  =  A*V  -  ALFA*U,
C               ALFA*V  =  A(TRANSPOSE)*U  -  BETA*V.
C
      CALL SSCAL ( M,(-ALFA),U,1 )
      CALL APROD ( 1,M,N,V,U,  IA,JA,AA )
      CALL NORMLZ( M,U,BETA )
      BBNORM = BBNORM + ALFA**2 + BETA**2 + DAMPSQ
      CALL SSCAL ( N,(-BETA),V,1 )
      CALL APROD ( 2,M,N,V,U,  IA,JA,AA )
      CALL NORMLZ( N,V,ALFA )
C
C
C     USE A PLANE ROTATION TO ELIMINATE THE DAMPING PARAMETER.
C     THIS ALTERS THE DIAGONAL (RHOBAR) OF THE LOWER-BIDIAGONAL MATRIX.
C
      RHBAR2 = RHOBAR**2 + DAMPSQ
      RHBAR1 = SQRT(RHBAR2)
      CS1    = RHOBAR/RHBAR1
      SN1    = DAMP/RHBAR1
      PSI    = SN1*PHIBAR
      PHIBAR = CS1*PHIBAR
C
C
C     USE A PLANE ROTATION TO ELIMINATE THE SUBDIAGONAL ELEMENT (BETA)
C     OF THE LOWER-BIDIAGONAL MATRIX, GIVING AN UPPER-BIDIAGONAL MATRIX.
C
      RHO    = SQRT(RHBAR2 + BETA**2)
      CS     = RHBAR1/RHO
      SN     = BETA/RHO
      THETA  =  SN*ALFA
      RHOBAR = -CS*ALFA
      PHI    =  CS*PHIBAR
      PHIBAR =  SN*PHIBAR
      TAU    =  SN*PHI
C
C
C     UPDATE  X, W  AND THE STANDARD ERROR ESTIMATES.
C
      T1 =    PHI/RHO
      T2 = -THETA/RHO
      T3 =    ONE/RHO
C
```

```
401         DO 200 I = 1, N
402            T     = W(I)
403            X(I)  = T1*T + X(I)
404            W(I)  = T2*T + V(I)
405            T     =(T3*T)**2
406            SE(I) = T + SE(I)
407            DDNORM= T + DDNORM
408       200 CONTINUE
409   C
410   C
411   C     USE A PLANE ROTATION ON THE RIGHT TO ELIMINATE THE
412   C     SUPER-DIAGONAL ELEMENT (THETA) OF THE UPPER-BIDIAGONAL MATRIX.
413   C     THEN USE THE RESULT TO ESTIMATE  NORM(X).
414   C
415         DELTA  =  SN2*RHO
416         GAMBAR = -CS2*RHO
417         RHS    = PHI - DELTA*Z
418         ZBAR   = RHS/GAMBAR
419         XNORM  = SQRT(XXNORM + ZBAR**2)
420         GAMMA  = SQRT(GAMBAR**2 + THETA**2)
421         CS2    = GAMBAR/GAMMA
422         SN2    = THETA/GAMMA
423         Z      = RHS/GAMMA
424         XXNORM = XXNORM + Z**2
425   C
426   C
427   C     TEST FOR CONVERGENCE.
428   C     FIRST, ESTIMATE THE NORM AND CONDITION OF THE MATRIX  ABAR,
429   C     AND THE NORMS OF  RBAR  AND  ABAR(TRANSPOSE)*RBAR.
430   C
431         ANORM  = SQRT(BBNORM)
432         ACOND  = ANORM*SQRT(DDNORM)
433         RES1   = PHIBAR**2
434         RES2   = RES2 + PSI**2
435         RNORM  = SQRT(RES1 + RES2)
436         ARNORM = ALFA*ABS(TAU)
437   C
438   C     NOW USE THESE NORMS TO ESTIMATE CERTAIN OTHER QUANTITIES,
439   C     SOME OF WHICH WILL BE SMALL NEAR A SOLUTION.
440   C
441         TEST1  = RNORM/BNORM
442         TEST2  = ARNORM/(ANORM*RNORM)
443         TEST3  = ONE/ACOND
444         T1     = TEST1/(ONE + ANORM*XNORM/BNORM)
445         RTOL   = BTOL +  ATOL*ANORM*XNORM/BNORM
446   C
447   C     THE FOLLOWING TESTS GUARD AGAINST EXTREMELY SMALL VALUES OF
448   C     ATOL, BTOL  OR  CTOL.  (THE USER MAY HAVE SET ANY OR ALL OF
449   C     THE PARAMETERS  ATOL, BTOL, CONLIM  TO ZERO.)
450   C     THE EFFECT IS EQUIVALENT TO THE NORMAL TESTS USING
451   C     ATOL = RELPR,  BTOL = RELPR,  CONLIM = 1/RELPR.
452   C
453         T3 = ONE + TEST3
```

```
454         T2 = ONE + TEST2
455         T1 = ONE + T1
456         IF (ITN .GE. ITNLIM) ISTOP = 7
457         IF (T3  .LE. ONE   ) ISTOP = 6
458         IF (T2  .LE. ONE   ) ISTOP = 5
459         IF (T1  .LE. ONE   ) ISTOP = 4
460   C
461   C     ALLOW FOR TOLERANCES SET BY THE USER.
462   C
463         IF (TEST3 .LE. CTOL) ISTOP = 3
464         IF (TEST2 .LE. ATOL) ISTOP = 2
465         IF (TEST1 .LE. RTOL) ISTOP = 1
466   C     ==================================================================
467   C
468   C     SEE IF IT IS TIME TO PRINT SOMETHING.
469   C
470         IF (NOUT .LE. 0) GO TO 600
471         IF (M.LE.40 .OR. N.LE.40) GO TO 400
472         IF (ITN .LE. 10)          GO TO 400
473         IF (ITN .GE. ITNLIM-10)   GO TO 400
474         IF (MOD(ITN,10) .EQ. 0)   GO TO 400
475         IF (TEST3 .LE.  2.0*CTOL) GO TO 400
476         IF (TEST2 .LE. 10.0*ATOL) GO TO 400
477         IF (TEST1 .LE. 10.0*RTOL) GO TO 400
478         GO TO 600
479   C
480   C     PRINT A LINE FOR THIS ITERATION.
481   C
482     400 WRITE(NOUT, 1500) ITN,X(1),RNORM,TEST1,TEST2,ANORM,ACOND
483         IF (MOD(ITN,10) .EQ. 0) WRITE(NOUT, 1600)
484   C     ==================================================================
485   C
486   C     STOP IF APPROPRIATE.
487   C     THE CONVERGENCE CRITERIA ARE REQUIRED TO BE MET ON  NCONV
488   C     CONSECUTIVE ITERATIONS, WHERE  NCONV  IS SET BELOW.
489   C     SUGGESTED VALUE --   NCONV = 1, 2  OR  3.
490   C
491     600 IF (ISTOP .EQ. 0) NSTOP = 0
492         IF (ISTOP .EQ. 0) GO TO 100
493         NCONV = 1
494         NSTOP = NSTOP + 1
495         IF (NSTOP .LT. NCONV  .AND.  ITN .LT. ITNLIM) ISTOP = 0
496         IF (ISTOP .EQ. 0) GO TO 100
497   C     ------------------------------------------------------------------
498   C     END OF ITERATION LOOP.
499   C     ------------------------------------------------------------------
500   C
501   C
502   C     FINISH OFF THE STANDARD ERROR ESTIMATES.
503   C
504         T = ONE
505         IF (M .GT. N) T = M - N
506         IF (DAMPSQ .GT. ZERO) T = M
```

```
507         T = RNORM/SQRT(T)
508   C
509         DO 700 I = 1, N
510            SE(I) = T*SQRT(SE(I))
511     700 CONTINUE
512   C
513   C     PRINT THE STOPPING CONDITION.
514   C
515     800 IF (NOUT .LE. 0) GO TO 900
516         WRITE(NOUT, 1900) ITN,ISTOP
517         IF (ISTOP .EQ. 0) WRITE(NOUT, 2000)
518         IF (ISTOP .EQ. 1) WRITE(NOUT, 2100)
519         IF (ISTOP .EQ. 2) WRITE(NOUT, 2200)
520         IF (ISTOP .EQ. 3) WRITE(NOUT, 2300)
521         IF (ISTOP .EQ. 4) WRITE(NOUT, 2400)
522         IF (ISTOP .EQ. 5) WRITE(NOUT, 2500)
523         IF (ISTOP .EQ. 6) WRITE(NOUT, 2600)
524         IF (ISTOP .EQ. 7) WRITE(NOUT, 2700)
525     900 RETURN
526   C     ------------------------------------------------------------------
527   C
528    1000 FORMAT(
529        1   // 25X, 46HLSQR   --   LEAST-SQUARES SOLUTION OF  A*X = B
530        2   // 25X, 18HTHE MATRIX  A  HAS, I6, 11H ROWS   AND, I6, 5H COLS
531        3    / 25X, 36HTHE DAMPING PARAMETER IS    DAMP   =, 1PE10.2
532        4   // 25X,  8HATOL   =, 1PE10.2, 10X, 8HCONLIM =, 1PE10.2
533        5    / 25X,  8HBTOL   =, 1PE10.2, 10X, 8HITNLIM =, I10)
534    1200 FORMAT(// 3X, 3HITN, 9X, 4HX(1), 14X, 8HFUNCTION, 7X,
535        1   45HCOMPATIBLE INCOMPATIBLE    NORM(A)    COND(A) /)
536    1300 FORMAT(// 3X, 3HITN, 9X, 4HX(1), 14X, 8HFUNCTION, 7X,
537        1   45HCOMPATIBLE INCOMPATIBLE NORM(ABAR) COND(ABAR) /)
538    1500 FORMAT(I6, 1PE20.10, 1PE19.10, 1P2E13.3, 1P2E11.2)
539    1600 FORMAT(1X)
540    1900 FORMAT(/ 20H NO. OF ITERATIONS =, I6,
541        1   8X, 21H STOPPING CONDITION =, I3)
542    2000 FORMAT(/ 52H THE EXACT SOLUTION IS  X = 0.                      )
543    2100 FORMAT(/ 52H A*X - B  IS SMALL ENOUGH, GIVEN  ATOL, BTOL        )
544    2200 FORMAT(/ 52H THE LEAST-SQRS SOLN IS GOOD ENOUGH, GIVEN  ATOL    )
545    2300 FORMAT(/ 52H THE ESTIMATE OF  COND(ABAR)  HAS EXCEEDED  CONLIM  )
546    2400 FORMAT(/ 52H A*X - B   IS SMALL ENOUGH FOR THIS MACHINE         )
547    2500 FORMAT(/ 52H THE LEAST-SQRS SOLN IS GOOD ENOUGH FOR THIS MACHINE)
548    2600 FORMAT(/ 52H COND(ABAR)   SEEMS TO BE TOO LARGE FOR THIS MACHINE)
549    2700 FORMAT(/ 52H THE ITERATION LIMIT HAS BEEN REACHED               )
550   C     END OF LSQR
551         END
552   C-----------------------------------------------------------------------
553         SUBROUTINE NORMLZ( N,X,BETA )
554         INTEGER    N
555         REAL       X(N),BETA
556   C
557   C     NORMLZ  IS REQUIRED BY SUBROUTINE LSQR.  IT COMPUTES THE
558   C     EUCLIDEAN NORM OF  X  AND RETURNS THE VALUE IN  BETA.
559   C     IF  X  IS NONZERO, IT IS SCALED SO THAT NORM(X) = 1.
```

```
560   C
561   C     FUNCTIONS AND SUBROUTINES
562   C
563   C     BLAS       SNRM2,SSCAL
564   C
565         REAL       ONE,SNRM2,ZERO
566   C
567   C
568         ZERO = 0.0
569         ONE  = 1.0
570         BETA = SNRM2( N,X,1 )
571         IF (BETA .GT. ZERO) CALL SSCAL( N,(ONE/BETA),X,1 )
572         RETURN
573   C
574   C     END OF NORMLZ
575         END
```

6.3.8 SQRTT.FOR

```
1       function sqrtt (z)              ! assure proper branch
2       complex  sqrtt, z, cmplx
3       data  pi / 3.1415926E0 /
4
5       x =  real (z)
6       y = aimag (z)
7       call polar (x,y,r,a,1)          ! [0,2)
8       if (r.eq.0.0) then
9            sqrtt = cmplx (0.0, 0.0)
10      else
11           r = sqrt (r)
12           a = a*pi*0.5               ! [0, pi)
13           sqrtt = cmplx (r*cos(a), r*sin(a))
14      end if
15      return
16      end
```

6.3.9 POLAR.FOR

```
c-------------------------------------------------------------
      subroutine polar (x, y, r, a, icase)
c       given:  x,y
c     discern:  r,a
c       where:  r>0,  x=r*cos(a*pi), y=r*sin(a*pi)
c      case 1:  a ~ [ 0, 2)
c           2:  a ~ (-1, 1]
c           3:  a ~ [ 0, 2*pi)              !  radians
c           4:  a ~ [ 0, 360)               !  degrees

      data  pi / 3.14159265E0 /

      if (y .eq. 0.0) then                  !  x axis
          if (x .lt. 0.0) then
              a = 1.0
              r = -x
          else
              a = 0.0
              r = x
          end if
          goto 1
      end if
      if (x .eq. 0.0) then                  !  y axis
          if (y .lt. 0.0) then
              a = 1.5
              r = -y
          else
              a = 0.5
              r = y
          end if
          goto 1
      end if
      xx = abs (x)
      yy = abs (y)
      if (xx .lt. yy) then
          a = x/y                           !  ratio < 1
          r = yy * sqrt (a*a+1.0)
          a = 0.5 - atan (a) /pi            !  top
          if (y .lt. 0.0)  a=a+1.0          !  bottom
      else
          a = y/x                           !  ratio < 1
          r = xx * sqrt (a*a+1.0)
          a = atan (a) /pi
          if (x .lt. 0.0) then              !  lhs
              a = a+1.0
          else if (y .lt. 0.0) then
              a = a+2.0                     !  rhs
          end if
      end if

```

```
51      1 if (icase .eq. 1) return
52        if (icase .eq. 2) then
53            if (a .gt. 1.0)  a=a-2.0
54            return
55        end if
56        if (icase .eq. 3) then                ! radians
57            a = a*pi
58        else                                  ! degrees
59            a = a*180.0
60        end if
61
62        return
63        end
```

6.3.10 IINDEX.FOR

```
1   c     Discern the sequential index (k) for the (i,j) matrix element,
2   c     where the matrix is symmetric,  a(i,j) = a(j,i) = a(k),
3   c     but where only the upper/lower triangle is stored,  k=k(i,j).
4   c     N ˝ size of the square matrix, i.e.,   A ˝ a(i,j)   is NxN.
5   c     L ˝ indicates the storage format, 1-4.
6
7         function iindex (l,n,i,j)
8         integer  iindex, l,n,i,j
9
10        if (l.lt.1 .or. n.lt.1 .or. i.lt.1 .or. j.lt.1 .or.
11       &    l.gt.6 .or.             i.gt.n .or. j.gt.n     ) then
12            call exit (2)                ! 2˝error,  4˝severe error
13        end if
14
15        mn = min (i,j)
16        mx = max (i,j)
17        goto (1,2,3,4,5,6), l
18
19  c     Symmetric Matix ------------------------------------
20
21  c     Lower triangular matrix, j<=i, stored column by column.
22       1 iindex = mx + ((n+n-mn)*(mn-1))/2
23        return
24
25  c     Lower triangular matrix, j<=i, stored row by row.
26       2 iindex = mn + ((mx-1)*mx)/2
27        return
28
29  c     Upper triangular matrix, i<=j, stored column by column.
30       3 iindex = mn + ((mx-1)*mx)/2
31        return
32
33  c     Upper triangular matrix, i<=j, stored row by row.
34       4 iindex = mx + ((n+n-mn)*(mn-1))/2
35        return
36
37  c     Asymmetric Matrix -----------------------------------
38
39  c     Full matrix, stored column by column.
40       5 iindex = i + (j-1)*n
41        return
42
43  c     Full matrix, stored row by row.
44       6 iindex = j + (i-1)*n
45        return
46
47        end
```

6.3.11 ISTIME.FOR

```
1       function istime (i)         ! dummy argument
2       integer  istime, iftime, i, it
3       logical  first
4       data     first /.true./
5
6       entry iftime (i)
7       if (first) then             ! initial
8           first = .false.
9           call times (-1,it)      ! initialize cpu clock
10      end if
11      call times (1,it)           ! query elapsed cpu time (centi-seconds)
12      istime = it*10              ! milli-seconds
13      return
14      end
```

6.3.12 TIMES.FOR

```
      subroutine times (is, it)
      include   'sys$library:libdef.for'
      include   'sys$library:sigdef.for'
c*    include   'sys$library:mthdef.for'
c*    include   'sys$library:fordef.for'

      parameter (io=6,  nh=2)
      integer*4  handle(nh), status
      save       handle

c-----------------------------------------------------------------------
c     This routine maintains or keeps track of several distinct
c          or separate and independent clocks.
c     Each clock is equivalent to any other clock,
c          i.e., among themselves.
c     The number of clocks is restricted by the magnitude
c          or dimension of the array, "handle", i.e., "nh".
c     The clocks are indexed by the magnitude of "is" = |is|.
c     A clock is initalized whenever "is" is negative;
c          all clocks are initalized whenever "is" is zero.
c     Time displacements or intervals, "it", are:
c          a)   evaluated whenever "is" is positive
c          b)   expressed in units of:  centi-seconds
c     ------------------------------------------------------------------

      if (is.eq.0) then                 ! initialize all clocks
          do i=1,nh                     ! scan distinct clocks
              status = lib$init_timer (handle(i))
              if (status.ne.ss$_normal) then
                  write (io,6)
                  stop
              end if
          end do
          return
      end if

      if (iabs(is).gt.nh) then          ! nonexisting clock
          write (io,8) is
          stop
      end if

      if (is.lt.0) then                 ! initialize, clock # |is|.
          status = lib$init_timer (handle(-is))
          if (status.eq.ss$_normal) return
          write (io,6)
          stop
      else                              ! elapsed time, clock # (is).
          status = lib$stat_timer (2,it,handle(is))
          if (status.eq.ss$_normal) return
          write (io,7)
```

```
51            stop
52          end if
53
54        6 format (' times,  error:   clock initalization problem.')
55        7 format (' times,  error:   clock evaluation problem.')
56        8 format (' times,  error:   using non-existing clock #', i2)
57          end
```

6.3.13 LJCHAR.FOR

```
1  c     ------------------------------------------------------------
2  c     Left-justify and compress a character string
3  c     by removing blank characters from the string.
4  c     ------------------------------------------------------------
5        subroutine ljchar (c)
6        character  a, c*(*)
7
8        l = len (c)             ! length, number of characters in string
9        i = index (c,' ')       ! locate first occurance of blank character
10       if (i.eq.0 .or. i.eq.l) return
11
12       ii = i+1                ! index of next character
13       do j=ii,l               ! scan rightmost part of string
14           a = c(j:j)          ! extract next character
15           if (a.ne.' ') then
16              c(i:i) = a       ! fill on left
17              c(j:j) = ' '     ! fill on right
18              i = i+1          ! update index
19           end if
20       end do
21       return
22       end
```

6.3.14 HHLINE.FOR

```
      subroutine hhline (idat, iout)   ! used only by:   INPDAT

      character*50   dlimit, hyphen    ! convenience
      data hyphen/'--------------------------------------------------'/

      read (idat,2)  dlimit
      if (dlimit .eq. hyphen) then
          write (iout,3)  dlimit
      else
          write (iout,4)  dlimit, hyphen
          stop
      end if
      return

    2 format (a)
    3 format (1x, a)
    4 format (/' ... oops,  inadequate delimiter'
     &        /' read:', a          /' want:', a)
      end
```

6.4 Dielectric Functions, Effective Media

6.4.1 DIEFCN.FOR

```
1  c     Evaluate:     dielectric function    of mixture.
2  c     Consider:
3  c       1)    volume fractions, effective medium approximation
4  c       2)    structure fractions, depolarization factors ... (not yet)
5  c       3)    mixture parameters
6  c       4)    ambient parameters
7  c     ---------------------------------------------------------------
8
9         subroutine  diefcn (imbien, imixtr,
10       &                    anglei, wavlen, dielec,
11       &                    dielew, dielff, dielpp, dielpa)
12
13        complex   dielec, dielew, dielff(1), dielpp(1), dielpa(1)
14
15        include  'iounit.'
16        include  'defnit.'
17        include  'filmmm.'
18        include  'handyy.'                ! pi
19
20        complex   diel(nlmnts), diew(nlmnts), ss(nlmnts)
21        integer   iipa(nparms), iipp(nparms)
22        real      rrpa(nparms), rrpp(nparms)
23        integer   llpa(nparms), llpp(nparms)
24        parameter (klmnts=5)              ! convenience
25        complex   ddpa(nparms,klmnts), ddpp(nparms,klmnts)   ! (nparms,nlmnts)
26
27
28        mm = mmixtr+imbien                ! offset, ambient
29
30        mipa = miparm(mm)
31        kipa = kiparm(mm)
32        if (mipa.ne.0) then               ! ambient parameters
33            ki = kipa
34            do m=1,mipa
35                ki = ki+1
36                ip = jiparm(ki)
37                iipa(m) = iiparm(ip)
38            end do
39        end if
40
41        mrpa = mrparm(mm)
42        krpa = krparm(mm)
43        if (mrpa.ne.0) then               ! ambient parameters
44            kr = krpa
45            do m=1,mrpa
46                kr = kr+1
47                ip = jrparm(kr)
48                rrpa(m) = rrparm(ip)
```

```
49              llpa(m) = lrparm(ip)
50          end do
51      end if
52
53      mipp = miparm(imixtr)
54      kipp = kiparm(imixtr)            ! offset ---> jiparm ---> iiparm
55      if (mipp.ne.0) then
56          ki = kipp
57          do m=1,mipp
58              ki = ki+1
59              ip = jiparm(ki)
60              iipp(m) = iiparm(ip)
61          end do
62      end if
63
64      mrpp = mrparm(imixtr)
65      krpp = krparm(imixtr)            ! offset ---> jrparm ---> rrparm
66      if (mrpp.ne.0) then
67          kr = krpp
68          do m=1,mrpp
69              kr = kr+1
70              ip = jrparm(kr)
71              rrpp(m) = rrparm(ip)
72              llpp(m) = lrparm(ip)
73          end do
74      end if
75
76      mmlmn = mmlmnt(imixtr)           ! quantity of elements in mixture
77      kklmn = kklmnt(imixtr)           ! offset ---> iilmnt
78  c   ================================================
79      if (mmlmn.ne.0) then             ! volume fractions
80          if (mmlmn .gt. klmnts) then  ! insure sufficient workspace
81              write (iout,111)  mmlmn
82              stop
83          end if
84
85          kk = kklmn
86          do i=1,mmlmn                 ! scan elements
87              kk = kk+1
88              lmnt = iilmnt(kk)        ! specific element
89  c           frac = fflmnt(kk)        ! volume fraction
90
91              if (mrpp .ne. 0) then    ! initialize
92                  do j=1,mrpp
93                      ddpp(j,i) = cmplx (0.0, 0.0)
94                  end do
95              end if
96              if (mrpa .ne. 0) then    ! initialize
97                  do j=1,mrpa
98                      ddpa(j,i) = cmplx (0.0, 0.0)
99                  end do
100             end if
101
```

```
102                 call dielmn (lmnt, anglei, wavlen, diel(i), diew(i),
103       &                      mipp,iipp,  mrpp,rrpp,llpp,  ddpp(1,i),
104       &                      mipa,iipa,  mrpa,rrpa,llpa,  ddpa(1,i) )
105               end do
106
107               kk = kklmn+1                ! locate volume fractions
108               call dieema (mmlmn, fflmnt(kk), diel, diew, dielff,
109       &                    dielec, dielew,                    ss )
110
111               if (mrpp .ne. 0) then
112                   do j=1,mrpp
113                       do i=1,mmlmn
114                           ss(i) = ddpp(j,i)
115                       end do
116                       call diemad (mmlmn, fflmnt(kk), diel, ss,
117       &                            dielec, dielpp(j)           )
118                   end do
119               end if
120               if (mrpa .ne. 0) then
121                   do j=1,mrpa
122                       do i=1,mmlmn
123                           ss(i) = ddpa(j,i)
124                       end do
125                       call diemad (mmlmn, fflmnt(kk), diel, ss,
126       &                            dielec, dielpa(j)           )
127                   end do
128               end if
129
130           end if                          ! volume fractions
131   c       ======================================================
132   c       Determine:   d (dielectric function_(w)) /d (mixture parameters)
133
134   c*      if (mrpp.ne.0) then             ! mixture parameters
135   c*          kr = krpp
136   c*          do irp=1,mrpp
137   c*              kr = kr+1
138   c*              dielpp(irp) = 0.0
139   c*          end do
140   c*          dielec =     ... when ever necessary
141   c*          dielew =     ... when ever necessary
142   c*      end if
143   c       ======================================================
144   c       Determine:   d (dielectric function_(w)) /d (ambient parameters)
145
146   c*      if (mrpa.ne.0) then             ! ambient parameters
147   c*          kr = krpa
148   c*          do irp=1,mrpa
149   c*              kr = kr+1
150   c*              dielpa(irp) = 0.0
151   c*          end do
152   c*          dielec =     ... when ever necessary
153   c*          dielew =     ... when ever necessary
154   c*      end if
```

```
155  c     ========================================================
156
157         return
158
159     111 format (/' diefcn,    increase the allocation size  '
160        &        /'            of the second index of arrays:'
161        &        /'            ddpp, ddpa.                   '
162        &        /'            to:  ', i5                    )
163
164         end
```

6.4.2 DIEEMA.FOR

```
1  c     ---------------------------------------------------------------
2  c     Evaluate:     dielectric function   of a mixture
3  c       within:     the effective medium approximation
4  c       Method:     Newton iteration
5  c     ---------------------------------------------------------------
6
7        subroutine  dieema (mlmnt, frac, diel, diew, dief,
8       &                    dielec, dielew,           ss)
9
10       real      frac(mlmnt)
11       complex   diel(mlmnt), diew(mlmnt), dief(mlmnt)
12       complex   dielec, dielew
13       complex   ss(mlmnt)                          ! workspace
14
15       complex   cmplx
16       complex   ss1, ss2, step, denom
17       complex   dies, died, fracc, third, one, zero
18
19       include  'iounit.'                           ! iout
20
21
22       third = cmplx (1.0/3.0, 0.0)
23       one  = cmplx (1.0, 0.0)
24       zero = cmplx (0.0, 0.0)
25       ss1  = zero                                  ! initialize
26       ss2  = zero
27
28       do i=1,mlmnt
29           fracc = cmplx (frac(i), 0.0)             ! volume fraction
30           dies = diel(i)                           ! dielectric function
31           ss(i) = fracc*dies                       ! convenience
32           ss1 = ss1+fracc*dies                     ! shielding minimum
33           ss2 = ss2+fracc/dies                     ! shielding maximum
34       end do
35
36       ss2 = one/ss2
37       dielec = (ss1+ss1+ss2)*third                 ! estimate EMA
38
39       if (mlmnt.eq.1) goto 2
40 c     ----------------------------------------
41 c     Newton iteration
42
43       kt = 0
44      1 kt = kt+1                                   ! iteration count
45       dies = zero                                  ! residual,  Q
46       died = zero                                  ! gradient, dQ
47       do i=1,mlmnt
48           denom = diel(i) + dielec+dielec
49           step = ss(i)/denom                       ! convenience
50           dies = dies+step                         ! f(x)
```

```
51          died = died-step/denom                  ! f'(x) /2
52        end do
53        dies =  dies-third                        !  Q = f(x) - f(root)
54        died =  died+died                         ! dQ = df/dx
55        step = -dies/died                         ! Newton step
56
57        sx =  real (step)
58        sy = aimag (step)
59        dx =  real (dielec)
60        dy = aimag (dielec)
61        call polar (sx,sy, sr,sa, 1)              ! [0,2)
62        call polar (dx,dy, dr,da, 1)              ! [0,2)
63
64        if (kt .eq. 200) then                     ! slow convergence
65            write (iout,101)  kt, dies, step, dielec
66            stop
67        end if
68
69        if (sr .gt. 0.1*dr) then                  ! limit stepsize
70            scal =  0.1*dr/sr
71            dielec = dielec + step*cmplx (scal, 0.0)
72            goto 1
73        else
74            dielec = dielec + step                ! update solution
75            if (sr .ge. dr*1.0E-6) goto 1         ! test convergence
76        end if
77  c     ------------------------------------------
78  c     Convergence achieved.
79  c     Evaluate:      (d/dw) (dielectric function  EMA)
80  c          and:      (d/df) (dielectric function  EMA)
81
82      2 continue
83        dies = zero
84        died = zero
85        do i=1,mlmnt                              ! fractions
86            denom = diel(i) + dielec+dielec
87            dief(i) = diel(i)/denom               ! df,  convenience
88            denom = denom*denom
89            fracc = cmplx (frac(i), 0.0)
90            dies = dies +   ss(i)      /denom     ! ds
91            died = died + diew(i)*fracc/denom     ! dw
92        end do
93        dielew = dielec*(died/dies)               ! ds/dw
94        dies = dies+dies
95        do i=1,mlmnt
96            dief(i) = dief(i)/dies                ! ds/df
97        end do
98
99        return
100
101   101 format (/' dieEMA,  slow convergence or divergence, '
102      &        /'          iteration ', i5
103      &        /'           residual ', 1p2e15.6
```

```
104       &        /'        Newton step ' ',    2e15.6
105       &        /'             dielec ' ',    2e15.6 )
106
107        end
```

6.4.3 DIEMAD.FOR

```
1  c     Evaluate:     dielectric function partial derivatives
2  c       within:     the effective medium approximation
3  c     ------------------------------------------------------------------
4
5        subroutine  diemad (mlmnt, frac, diel, diew,
6       &                    dielec, dielew)
7
8        real       frac(mlmnt)
9        complex    diel(mlmnt), diew(mlmnt), dielec, dielew
10       complex    cmplx
11       complex    dies, died, fracc, denom
12
13       dies = cmplx (0.0, 0.0)
14       died = cmplx (0.0, 0.0)
15       do i=1,mlmnt                                  ! fractions
16           denom = diel(i) + dielec+dielec
17           denom = denom*denom
18           fracc = cmplx (frac(i), 0.0)
19           dies = dies + diel(i)*fracc/denom         ! de
20           died = died + diew(i)*fracc/denom         ! dw
21       end do
22       dielew = dielec*(died/dies)                   ! de/dw
23
24       return
25       end
```

6.4.4 DIELMN.FOR

```
1       subroutine dielmn (lmnt,
2      &                   anglei, wavlen, dielec, dielew,
3      &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,   ! mixture
4      &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)   ! ambient
5
6       integer   iipa(1), iipp(1)
7       real      rrpa(1), rrpp(1)
8       integer   llpa(1), llpp(1)
9       complex   ddpa(1), ddpp(1), dielec, dielew
10
11      include  'iounit.'
12      include  'defnit.'
13
14
15      if (lmnt.lt.1 .or. lmnt.gt.nlmnts) then
16          write (iout,102)  lmnt, nlmnts
17          stop
18      end if
19
20      goto ( 1,  2,  3,  4,  5,  6,  7,  8,  9, 10,
21     &      11, 12, 13, 14, 15, 16, 17, 18, 19, 20 ), lmnt
22          write (iout,103)
23          stop
24
25  c   -----------------------------------------------------
26     1 continue     ! vacuum
27      call diel01 (anglei, wavlen, dielec, dielew,
28     &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
29     &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
30      goto 101
31  c   -----------------------------------------------------
32     2 continue     ! air
33      call diel02 (anglei, wavlen, dielec, dielew,
34     &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
35     &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
36      goto 101
37  c   -----------------------------------------------------
38     3 continue     ! Silicon (crystalline)
39      call diel03 (anglei, wavlen, dielec, dielew,
40     &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
41     &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
42      goto 101
43  c   -----------------------------------------------------
44     4 continue     ! Silicon (amorphous)
45      call diel04 (anglei, wavlen, dielec, dielew,
46     &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
47     &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
48      goto 101
49  c   -----------------------------------------------------
50     5 continue     ! Silicon Dioxide (glass)
```

```
51        call diel05 (anglei, wavlen, dielec, dielew,
52       &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
53       &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
54        goto 101
55  c     ------------------------------------------------
56      6 continue     ! Silicon Nitride (noncrystalline)
57        call diel06 (anglei, wavlen, dielec, dielew,
58       &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
59       &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
60        goto 101
61  c     ------------------------------------------------
62      7 continue     ! Germanium (crystalline)
63        call diel07 (anglei, wavlen, dielec, dielew,
64       &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
65       &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
66        goto 101
67  c     ------------------------------------------------
68      8 continue     ! GaAs (crystalline)
69        call diel08 (anglei, wavlen, dielec, dielew,
70       &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
71       &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
72        goto 101
73  c     ------------------------------------------------
74      9 continue     ! Al(x) Ga(1-x) As
75        call diel09 (anglei, wavlen, dielec, dielew,
76       &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
77       &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
78        goto 101
79  c     ------------------------------------------------
80     10 continue     ! Oxides of GaAs,    Ga(2)O(3), etc.
81        call diel10 (anglei, wavlen, dielec, dielew,
82       &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
83       &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
84        goto 101
85  c     ------------------------------------------------
86     11 continue     ! Arsenic (amorphous)
87        call diel11 (anglei, wavlen, dielec, dielew,
88       &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
89       &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
90        goto 101
91  c     ------------------------------------------------
92     12 continue     ! GaP
93        call diel12 (anglei, wavlen, dielec, dielew,
94       &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
95       &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
96        goto 101
97  c     ------------------------------------------------
98     13 continue     ! GaSb
99        call diel13 (anglei, wavlen, dielec, dielew,
100      &            mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
101      &            mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
102       goto 101
103 c     ------------------------------------------------
```

```
104      14 continue      ! InAs
105         call diel14 (anglei, wavlen, dielec, dielew,
106        &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
107        &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
108         goto 101
109   c     ------------------------------------------------------
110      15 continue      ! InP
111         call diel15 (anglei, wavlen, dielec, dielew,
112        &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
113        &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
114         goto 101
115   c     ------------------------------------------------------
116      16 continue      ! InSb
117         call diel16 (anglei, wavlen, dielec, dielew,
118        &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
119        &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
120         goto 101
121   c     ------------------------------------------------------
122      17 continue      ! AlSb
123         call diel17 (anglei, wavlen, dielec, dielew,
124        &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
125        &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
126         goto 101
127   c     ------------------------------------------------------
128      18 continue      ! Si *1.01, Geist
129         call diel18 (anglei, wavlen, dielec, dielew,
130        &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
131        &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
132         goto 101
133   c     ------------------------------------------------------
134      19 continue      !
135   c*    call diel19 (anglei, wavlen, dielec, dielew,
136   c*   &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
137   c*   &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
138         if (.true.) stop
139         goto 101
140   c     ------------------------------------------------------
141      20 continue      ! GaAs (Sell, Horowitz)
142         call diel20 (anglei, wavlen, dielec, dielew,
143        &             mipp,iipp,  mrpp,rrpp,llpp,  ddpp,
144        &             mipa,iipa,  mrpa,rrpa,llpa,  ddpa)
145         goto 101
146   c     ------------------------------------------------------
147     101 continue
148         return
149
150     102 format (' dielmn,   ... oops, ', 2i4, 4x, '" lmnt, nlmnts')
151     103 format (' dielmn,   ... oops,   update argument list of:  GOTO')
152         end
```

6.5 Dielectric Functions, Constituent Media

6.5.1 DIEL01.FOR, vacuum

```
c     Optical properties of:    vacuum

      subroutine diel01 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                ! pi,cccc,wavlev

c*    common / dieoo1 / rfrac, dfrac    ! not necessary


      if (.not.llmnts(1)) then
          llmnts(1) = .true.
          write (iout,111)
      end if

      dielec = cmplx (1.0, 0.0)         ! dielectric function
      dielew = cmplx (0.0, 0.0)         ! d/d(energy 'eV) (dielec)

      return
  111 format (' lmnt '  1,  filename = vacuum')
      end
```

6.5.2 DIEL02.FOR, air

```
1  c     ------------------------------------------------------------------
2  c     Optical properties of:     air
3
4  c     References:    Frank E. Jones,
5  c     1)  "Calculation of Compressibility Factor for Air Over Ranges of
6  c          Pressure, Temperature, and Relative Humidity of Interest in
7  c          Flowmeter Calibration",
8  c          NBSIR 83-2652, National Bureau of Standards, March 1983.
9  c     2)  "The Refractivity of Air",
10 c          Journal of Research of the National Bureau of Standards,
11 c          Volume 86, Number 1, January-February, 1981, pages 27-32.
12 c     3)  "Simplified Equation for Calculating the Refractivity of Air",
13 c          Applied Optics, Volume 19, Number 24, 15 December 1980,
14 c          pages 4129-4130.
15 c     4)  "The Air Density Equation and the Transfer of the Mass Unit",
16 c          Journal of Research of the National Bureau of Standards,
17 c          Volume 83, Number 5, September-October, 1978, pages 419-428.
18
19 c     The optical properties of air is dependent upon:
20 c     a)    molecular composition   (CO2, water vapor, ...)
21 c     b)    pressure,
22 c     c)    temperature.
23 c     ------------------------------------------------------------------
24
25       subroutine diel02 (anglei, wavlen, dielec, dielew,
26      &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
27      &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient
28
29       integer   iipa(1), iipp(1)
30       real      rrpa(1), rrpp(1), anglei, wavlen
31       integer   llpa(1), llpp(1)
32       complex   ddpa(1), ddpp(1), dielec, dielew
33
34       include  'iounit.'
35       include  'defnit.'
36       include  'elmnts.'
37       include  'handyy.'                ! pi,cccc,wavlev
38
39       common / dieoo2 / rfrac, dfrac    ! (n-1), n'
40
41
42       if (.not.llmnts(2)) then
43           llmnts(2) = .true.
44           write (iout,111)
45       end if
46
47       tz = 273.15                       ! zero Celsius expressed in Kelvin
48       ts = tz + 15.0                    ! standard air
49       tr = tz + 20.0                    ! temperature of room  (Kelvin)
50
```

```
51        ps = 760.0                          ! standard air ~ one atmosphere
52        pr = 760.0                          ! pressure in room (torr ~ mm Hg)
53
54        xs = 0.0003                         ! standard air
55        xr = 0.00043                        ! carbon dioxide in room
56
57        us = 0.0                            ! standard air ~ dry
58        ur = 0.50                           ! relative humidity (water) in room
59
60  c     ---------------------------------
61  c     Enhancement factor (saturation water vapor pressure:  air vs. pure phase)
62
63        f = 1.00070 + (3.113E-8 * 101325.0)*(pr/ps)
64       &            +    5.4E-7 * (tr-tz)**2
65  c     ---------------------------------
66  c     Saturation water vapor pressure in (pure phase ?)
67
68        es = 1.7526E+11 * exp (-5315.56 /tr)       ! Pascals
69  c     ---------------------------------
70  c     Compressibility of air (standard)
71
72        pp = 1.0                            ! pressure in atmospheres
73        ap =  -10.864   + pp*(588.26    - pp*2.7106)
74        bp =  0.33297   - pp*(12.585    - pp*2.0659E-2)
75        cp = -2.4925E-3 + pp*(8.3706E-2 - pp*5.5619E-5)
76        tt = ts-tz                          ! temperature in Celsius
77        zs = ap + tt*(bp + tt*cp)
78        zs = zs - us*(35.0 + us*0.5)        ! relative humidity correction
79        zs = 1.0 - zs*1.0E-6
80  c     ---------------------------------
81  c     Compressibility of air (room)
82
83        pp = pr/ps                          ! pressure in atmospheres
84        ap =  -10.864   + pp*(588.26    - pp*2.7106)
85        bp =  0.33297   - pp*(12.585    - pp*2.0659E-2)
86        cp = -2.4925E-3 + pp*(8.3706E-2 - pp*5.5619E-5)
87        tt = tr-tz                          ! temperature in Celsius
88        zr = ap + tt*(bp + tt*cp)
89        zr = zr - ur*(35.0 + ur*0.5)        ! relative humidity correction
90        zr = 1.0 - zr*1.0E-6
91  c     ---------------------------------
92  c     The source of incident light is characterized by
93  c     wavlen:     a)   -wavelength in units of nano-meters, or
94  c                 b)    energy     in units of electron-volts.
95  c     ---------------------------------
96        if (wavlen .lt. 0.0) then           ! nm, wavelength
97            sigma = -1.0E3/wavlen           !     um**-1
98        else                                ! eV, energy
99            sigma =  1.0E3*wavlen/wavlev    !     um**-1
100       end if
101       ss = sigma*sigma                    ! inverse (micro-meter) **2
102
103       airs = 8342.13 + 2406030./(130.0-ss) + 15997.0/(38.9-ss)
```

```
104         airs = airs * 1.0E-8 * (pr/ps)*(ts/tr)*(zs/zr)       ! standard air
105         airx = airs * (1.0 + 0.540*(xr-xs))                  ! x CO2
106
107         water = ur*f*es* (4.2922E-2 - ss*3.43E-4) *1.0E-8
108         rfrac = airx - water                          ! n-1
109         refrac = 1.0 + rfrac                          ! refractive index
110         dielec = cmplx (refrac*refrac, 0.0)           ! dielectric function
111   c     ---------------------------------
112   c     Derivatives with respect to energy (eV).
113
114         airs = 2406030./((130.0-ss)**2) + 15997.0/((38.9-ss)**2)  ! d/d(ss)
115         airs = airs * 1.0E-8 * (pr/ps)*(ts/tr)*(zs/zr)       ! standard air
116         airx = airs * (1.0 + 0.540*(xr-xs))                  ! x CO2
117
118         water = ur*f*es* 3.43E-12          ! d/d(ss)
119         dfrac = airx - water               ! d/d(ss)             ' (micro-m)**2
120         dfrac = dfrac * sigma*2.0          ! d/d(sigma)          ' (micro-m)
121         dfrac = dfrac / (wavlev*1.0E-3)  ! d/d(energy ' eV)
122         dielew = cmplx (refrac*dfrac*2.0, 0.0)     ! d/d(energy 'eV) (dielec)
123
124   c*    write (iout,103)  rfrac, dfrac
125
126         return
127
128     103 format (' rfrac = ', 1pe15.5, ' = (n-1) '
129        &       /' dfrac = ',   e15.5, ' = (n-1)'' ' d/d(energy) (n-1) ')
130     111 format (' lmnt '  2,  filename = air')
131         end
```

6.5.3 DIEL03.FOR, Si (crystalline)

```
c     ------------------------------------------------------------------
c     Optical properties of:     Silicon  (Si),   crystalline.
c     ------------------------------------------------------------------

      subroutine diel03 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam
      character*4   string                        ! convenience

c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
c     in order to include the first derivatives at each end point.

      parameter   (nx = 50,           ko = 4)
      parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
      common / dieoo3 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
      common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(4), v(2)


      if (llmnts(3))  goto 2

c     Fetch information on the dielectric function.
c     Assume:   1)   no partitions
c               2)   discretization of input data is of either form:
c                    a)  energy (eV),     refractive index,  extinction
c                    b)  energy (eV),  dielectric function,  conductivity

      filnam = 'Si'                    ! silicon crystalline
      write (iout,111) filnam
      filnam = subdir//filnam
      call ljchar (filnam)             ! left-justify characters
      open (iscr, file=filnam, status='old',readonly,shared,err=11)
c*    write (iout,111) filnam

    1 read  (iscr,112,err=12,end=12)  string     ! skip header
c*    write (iout,113)                string
      if (string .ne. '=h==') goto 1

      read  (iscr,  *)  mx                        ! quantity of input data
```

```
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.ne.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(3)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(4)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(3)                             ! refractive index
          c(k,2) = u(4)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(3)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(4)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do       ! x,u
c     ------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                            ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
      end do
```

```
c     ======================================================
      read  (iscr,112,err=12,end=12)  string     ! end of information
c*    write (iout,113)                string
      if (string .eq. '=h==') then
          close (iscr)
      else
          write (iout,114)
          stop
      end if

      llmnts(3) = .true.
      iptr = 1                              ! initialize

    2 if (wavlen .lt. 0.0) then             ! nm, wavelength
          x = -wavlev /wavlen               !     eV <--- nm
      else
          x = wavlen                        !     eV, energy
      end if

      if (x.lt.x1 .or. x.gt.x2) then        ! point exterior domain
          write (iout,132)  x1,x2,x
          stop
      end if

      do i=1,2
         u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
         v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
      end do

      dielec = cmplx (u(1), u(2))                          ! (n+ik)
      dielew = cmplx (v(1), v(2))                          ! (n+ik)'

      dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
      dielec = dielec*dielec                               ! e

      return
c     ======================================================

   11 write (iout,141)
      stop
   12 write (iout,142)
      stop
   13 write (iout,143)
      stop

  111 format ( ' lmnt ' 3,  filename = ', a)
  112 format ( a)
  113 format ( ' string = ', a)
  114 format ( ' oops,  error in end-of-info header')

  121 format ( ' mx = ', i5)
  122 format ( ' oops, inconsistency within:   mx')
  123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
```

```
157    124 format ( ' oops, ordering violation ')
158
159    131 format ( ' oops, error in B-spline formation '
160       &       /   10x, 'info =', i2
161       &       /   10x, '   i =', i2 , ',', 5x, '(1-u, 2-du/dx)' )
162    132 format ( ' oops, evaluation point is exterior domain,'
163       &       / '       x1,x2,x - ',  3f10.3)
164
165    141 format ( ' oops, error in opening input data file, ')
166    142 format ( ' oops, error in locating header card,    ')
167    143 format ( ' oops, error in reading (i,x,u)         ')
168
169        end
```

6.5.4 DIEL04.FOR, Si (amorphous)

```
1  c     ---------------------------------------------------------------
2  c     Optical properties of:     Silicon  (Si),    amorphous.
3  c     ---------------------------------------------------------------
4
5        subroutine diel04 (anglei, wavlen, dielec, dielew,
6       &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
7       &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient
8
9        integer   iipa(1), iipp(1)
10       real      rrpa(1), rrpp(1), anglei, wavlen
11       integer   llpa(1), llpp(1)
12       complex   ddpa(1), ddpp(1), dielec, dielew
13
14       include  'iounit.'
15       include  'defnit.'
16       include  'elmnts.'
17       include  'handyy.'                          ! pi,cccc,wavlev
18
19       character*64  filnam
20       character*4   string                        ! convenience
21
22 c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
23 c     in order to include the first derivatives at each end point.
24
25       parameter   (nx = 27,           ko = 4)
26       parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
27       common / dieoo4 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
28       common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(2), v(2)
29
30
31       if (llmnts(4))  goto 2
32
33 c     Fetch information on the dielectric function.
34 c     Assume:   1)   no partitions
35 c               2)   discretization of input data is of either form:
36 c                    a)  energy (eV),     refractive index,  extinction
37 c                    b)  energy (eV),  dielectric function,  conductivity
38
39       filnam = 'Sia'                    ! silicon amorphous
40       write (iout,111) filnam
41       filnam = subdir//filnam
42       call ljchar (filnam)              ! left-justify characters
43       open (iscr, file=filnam, status='old',readonly,shared,err=11)
44 c*    write (iout,111) filnam
45
46      1 read  (iscr,112,err=12,end=12)  string     ! skip header
47 c*    write (iout,113)                  string
48       if (string .ne. '=h==') goto 1
49
50       read  (iscr,  *)  mx                        ! quantity of input data
```

```
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.ne.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(1)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(2)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(1)                             ! refractive index
          c(k,2) = u(2)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(1)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(2)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do       ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                                ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
      end do
```

```
c     ======================================================
      read  (iscr,112,err=12,end=12)  string     ! end of information
c*    write (iout,113)                string
      if (string .eq. '=h==') then
          close (iscr)
      else
          write (iout,114)
          stop
      end if

      llmnts(4) = .true.
      iptr = 1                          ! initialize

    2 if (wavlen .lt. 0.0) then         ! nm, wavelength
          x = -wavlev /wavlen           !     eV <--- nm
      else
          x = wavlen                    !     eV, energy
      end if

      if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
          write (iout,132)  x1,x2,x
          stop
      end if

      do i=1,2
         u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
         v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
      end do

      dielec = cmplx (u(1), u(2))                          ! (n+ik)
      dielew = cmplx (v(1), v(2))                          ! (n+ik)'

      dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
      dielec = dielec*dielec                               ! e

      return
c     ======================================================

   11 write (iout,141)
      stop
   12 write (iout,142)
      stop
   13 write (iout,143)
      stop

  111 format ( ' lmnt "  4,  filename = ', a)
  112 format ( a)
  113 format ( ' string = ', a)
  114 format ( ' oops,  error in end-of-info header')

  121 format ( ' mx = ', i5)
  122 format ( ' oops, inconsistency within:   mx')
  123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
```

```
157    124 format ( ' oops, ordering violation ')
158
159    131 format ( ' oops, error in B-spline formation, '
160       &       /   10x, 'info =', i2
161       &       /   10x, '   i =', i2 , ',', 5x, '(1~u, 2~du/dx)' )
162    132 format ( ' oops, evaluation point is exterior domain,'
163       &       / '       x1,x2,x ~ ',  3f10.3)
164
165    141 format ( ' oops, error in opening input data file, ')
166    142 format ( ' oops, error in locating header card,    ')
167    143 format ( ' oops, error in reading (i,x,u)         ')
168
169        end
```

6.5.5 DIEL05.FOR, SiO_2 (amorphous)

```
c     ------------------------------------------------------------
c     Optical properties of:   Silicon Dioxide  (SiO2),  amorphous glass.
c     ------------------------------------------------------------

      subroutine diel05 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam
      character*4   string                        ! convenience

c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
c     in order to include the first derivatives at each end point.

      parameter   (nx = 48,           ko = 4)
      parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
      common / dieoo5 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
      common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(2), v(2)


      if (llmnts(5))  goto 2

c     Fetch information on the dielectric function.
c     Assume:   1)   no partitions
c               2)   discretization of input data is of either form:
c                    a)  energy (eV),     refractive index,  extinction
c                    b)  energy (eV),  dielectric function,  conductivity

      filnam = 'Si_O2g'                 ! silicon dioxide (amorphous glass)
      write (iout,111) filnam
      filnam = subdir//filnam
      call ljchar (filnam)              ! left-justify characters
      open (iscr, file=filnam, status='old',readonly,shared,err=11)
c*    write (iout,111) filnam

    1 read  (iscr,112,err=12,end=12)  string      ! skip header
c*    write (iout,113)                string
      if (string .ne. '=h==') goto 1

      read  (iscr,  *)  mx                        ! quantity of input data
```

```
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.ne.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(1)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(2)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(1)                             ! refractive index
          c(k,2) = u(2)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(1)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(2)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do        ! x,u
c     ------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                            ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
      end do
```

```
c     ======================================================
      read  (iscr,112,err=12,end=12)  string     ! end of information
c*    write (iout,113)                string
      if (string .eq. '=h==') then
          close (iscr)
      else
          write (iout,114)
          stop
      end if

      llmnts(5) = .true.
      iptr = 1                          ! initialize

    2 if (wavlen .lt. 0.0) then         ! nm, wavelength
          x = -wavlev /wavlen           !     eV <--- nm
      else
          x = wavlen                    !     eV, energy
      end if

      if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
          write (iout,132)  x1,x2,x
          stop
      end if

      do i=1,2
         u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
         v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
      end do

      dielec = cmplx (u(1), u(2))                          ! (n+ik)
      dielew = cmplx (v(1), v(2))                          ! (n+ik)'

      dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
      dielec = dielec*dielec                               ! e

      return
c     ======================================================

   11 write (iout,141)
      stop
   12 write (iout,142)
      stop
   13 write (iout,143)
      stop

  111 format ( ' lmnt ' 5,  filename = ', a)
  112 format ( a)
  113 format ( ' string = ', a)
  114 format ( ' oops,  error in end-of-info header')

  121 format ( ' mx = ', i5)
  122 format ( ' oops, inconsistency within:   mx')
  123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
```

```
157    124 format ( ' oops, ordering violation ')
158
159    131 format ( ' oops, error in B-spline formation, '
160       &       /   10x, 'info =', i2
161       &       /   10x, '   i =', i2 , ',', 5x, '(1^u, 2^du/dx)' )
162    132 format ( ' oops, evaluation point is exterior domain,'
163       &       / '       x1,x2,x ^ ',  3f10.3)
164
165    141 format ( ' oops, error in opening input data file, ')
166    142 format ( ' oops, error in locating header card,     ')
167    143 format ( ' oops, error in reading (i,x,u)          ')
168
169        end
```

6.5.6 DIEL06.FOR, Si_3N_4 (noncrystalline)

```
c     ---------------------------------------------------------------
c     Optical properties of:   Silicon Nitride  (Si3N4),  non-crystalline.
c     ---------------------------------------------------------------

      subroutine diel06 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam
      character*4   string                        ! convenience

c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
c     in order to include the first derivatives at each end point.

      parameter   (nx = 21,           ko = 4)
      parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
      common / dieoo6 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
      common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(2), v(2)


      if (llmnts(6))  goto 2

c     Fetch information on the dielectric function.
c     Assume:   1)   no partitions
c               2)   discretization of input data is of either form:
c                    a)  energy (eV),     refractive index,  extinction
c                    b)  energy (eV),  dielectric function,  conductivity

      filnam = 'Si3_N4'                 ! Silicon Nitride (non-crystalline)
      write (iout,111) filnam
      filnam = subdir//filnam
      call ljchar (filnam)              ! left-justify characters
      open (iscr, file=filnam, status='old',readonly,shared,err=11)
c*    write (iout,111) filnam

    1 read  (iscr,112,err=12,end=12)  string      ! skip header
c*    write (iout,113)                string
      if (string .ne. '=h==') goto 1

      read  (iscr,  *)  mx                        ! quantity of input data
```

```
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.ne.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          u(2) = alog (u(2))            ! logarithm fit to extinction

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(1)-c(1,1)) /(x-xx(1))  ! refractive index
              c(k,2) = (u(2)-c(1,2)) /(x-xx(1))  ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(1)                             ! refractive index
          c(k,2) = u(2)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(1)-c(k-2,1)) /(x-xx(k-2))   ! refractive index
              c(k,2) = (u(2)-c(k-2,2)) /(x-xx(k-2))   ! extinction
          end if

      end do        ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                            ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
```

```
104           end if
105        end do
106  c     ================================================
107        read  (iscr,112,err=12,end=12)  string     ! end of information
108  c*    write (iout,113)                string
109        if (string .eq. '=h==') then
110           close (iscr)
111        else
112           write (iout,114)
113           stop
114        end if
115
116        llmnts(6) = .true.
117        iptr = 1                          ! initialize
118
119      2 if (wavlen .lt. 0.0) then         ! nm, wavelength
120           x = -wavlev /wavlen            !     eV <--- nm
121        else
122           x = wavlen                     !     eV, energy
123        end if
124
125        if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
126           write (iout,132)  x1,x2,x
127           stop
128        end if
129
130        do i=1,2
131           u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
132           v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
133        end do
134
135  c     Logarithmic fit to extinction coefficient, k
136        u(2) = exp (u(2))                                     ! k  = exp (u)
137        v(2) = u(2)*v(2)                                      ! k' = k * u'
138
139        dielec = cmplx (u(1), u(2))                           ! (n+ik)
140        dielew = cmplx (v(1), v(2))                           ! (n+ik)'
141
142        dielew = dielec*dielew*cmplx (2.0, 0.0)               ! e'
143        dielec = dielec*dielec                                ! e
144
145        return
146  c     ================================================
147
148     11 write (iout,141)
149        stop
150     12 write (iout,142)
151        stop
152     13 write (iout,143)
153        stop
154
155    111 format ( ' lmnt ' 6,  filename = ', a)
156    112 format ( a)
```

```
157    113 format ( ' string = ', a)
158    114 format ( ' oops,  error in end-of-info header')
159
160    121 format ( ' mx = ', i5)
161    122 format ( ' oops, inconsistency within:   mx')
162    123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
163    124 format ( ' oops, ordering violation ')
164
165    131 format ( ' oops, error in B-spline formation '
166       &       /   10x, 'info =', i2
167       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
168    132 format ( ' oops, evaluation point is exterior domain,'
169       &       / '       x1,x2,x " ',  3f10.3)
170
171    141 format ( ' oops, error in opening input data file ')
172    142 format ( ' oops, error in locating header card,   ')
173    143 format ( ' oops, error in reading (i,x,u)         ')
174
175        end
```

6.5.7 DIEL07.FOR, Ge (crystalline)

```
1  c     ------------------------------------------------------------------
2  c     Optical properties of:     Germanium  (Ge),  crystalline.
3  c     ------------------------------------------------------------------
4
5        subroutine diel07 (anglei, wavlen, dielec, dielew,
6       &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
7       &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient
8
9        integer   iipa(1), iipp(1)
10       real      rrpa(1), rrpp(1), anglei, wavlen
11       integer   llpa(1), llpp(1)
12       complex   ddpa(1), ddpp(1), dielec, dielew
13
14       include  'iounit.'
15       include  'defnit.'
16       include  'elmnts.'
17       include  'handyy.'                          ! pi,cccc,wavlev
18
19       character*64  filnam
20       character*4   string                        ! convenience
21
22 c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
23 c     in order to include the first derivatives at each end point.
24
25       parameter   (nx = 93,           ko = 4)
26       parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
27       common / dieoo7 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
28       common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(2), v(2)
29
30
31       if (llmnts(7))  goto 2
32
33 c     Fetch information on the dielectric function.
34 c     Assume:   1)   no partitions
35 c               2)   discretization of input data is of either form:
36 c                    a)  energy (eV),     refractive index,  extinction
37 c                    b)  energy (eV),  dielectric function,  conductivity
38
39       filnam = 'Ge'                    ! Germanium crystalline
40       write (iout,111) filnam
41       filnam = subdir//filnam
42       call ljchar (filnam)             ! left-justify characters
43       open (iscr, file=filnam, status='old',readonly,shared,err=11)
44 c*    write (iout,111) filnam
45
46      1 read  (iscr,112,err=12,end=12)  string     ! skip header
47 c*    write (iout,113)                  string
48       if (string .ne. '=h==') goto 1
49
50       read  (iscr,  *)  mx                        ! quantity of input data
```

```
c*    write (iout,121)  mx                              !   within one partition
      if (mx.lt.3  .or.  mx+2.ne.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                             ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                              ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                             ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(1)-c(1,1)) /(x-xx(1))         ! refractive index
              c(k,2) = (u(2)-c(1,2)) /(x-xx(1))         ! extinction
          end if

          k = k+1
          xx(k) = x                                     ! energy (eV)
          id(k) = 0                                     ! i-th derivative
          c(k,1) = u(1)                                 ! refractive index
          c(k,2) = u(2)                                 ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                     ! energy (eV)
              id(k) = 1                                     ! i-th derivative
              c(k,1) = (u(1)-c(k-2,1)) /(x-xx(k-2))         ! refractive index
              c(k,2) = (u(2)-c(k-2,2)) /(x-xx(k-2))         ! extinction
          end if

      end do        ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                                ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
      end do
```

```
c     ========================================================
      read  (iscr,112,err=12,end=12)  string     ! end of information
c*    write (iout,113)                string
      if (string .eq. '=h==') then
          close (iscr)
      else
          write (iout,114)
          stop
      end if

      llmnts(7) = .true.
      iptr = 1                          ! initialize

    2 if (wavlen .lt. 0.0) then         ! nm, wavelength
          x = -wavlev /wavlen           !     eV <--- nm
      else
          x = wavlen                    !     eV, energy
      end if

      if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
          write (iout,132)  x1,x2,x
          stop
      end if

      do i=1,2
         u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
         v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
      end do

      dielec = cmplx (u(1), u(2))                          ! (n+ik)
      dielew = cmplx (v(1), v(2))                          ! (n+ik)'

      dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
      dielec = dielec*dielec                               ! e

      return
c     ========================================================

   11 write (iout,141)
      stop
   12 write (iout,142)
      stop
   13 write (iout,143)
      stop

  111 format ( ' lmnt ' 7,  filename = ', a)
  112 format ( a)
  113 format ( ' string = ', a)
  114 format ( ' oops,  error in end-of-info header ')

  121 format ( ' mx = ', i5)
  122 format ( ' oops, inconsistency within:   mx')
  123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
```

```
157    124 format ( ' oops, ordering violation ')
158
159    131 format ( ' oops, error in B-spline formation, '
160       &       /   10x, 'info =', i2
161       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
162    132 format ( ' oops, evaluation point is exterior domain ,'
163       &       / '       x1,x2,x " ',  3f10.3)
164
165    141 format ( ' oops, error in opening input data file, ')
166    142 format ( ' oops, error in locating header card,    ')
167    143 format ( ' oops, error in reading (i,x,u)          ')
168
169        end
```

6.5.8 DIEL08.FOR, GaAs (crystalline)

```
c     ------------------------------------------------------------
c     Optical properties of:   Gallium Arsenide  (GaAs),  crystalline.
c     ------------------------------------------------------------

      subroutine diel08 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam
      character*4   string                        ! convenience

c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
c     in order to include the first derivatives at each end point.

      parameter   (nx = 153,          ko = 4)
      parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
      common / dieoo8 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
      common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(2), v(2)


      if (llmnts(8))  goto 2

c     Fetch information on the dielectric function.
c     Assume:   1)   no partitions
c               2)   discretization of input data is of either form:
c                    a)  energy (eV),     refractive index,  extinction
c                    b)  energy (eV),  dielectric function,  conductivity

      filnam = 'Ga_As'                 ! Gallium Arsenide
      write (iout,111) filnam
      filnam = subdir//filnam
      call ljchar (filnam)             ! left-justify characters
      open (iscr, file=filnam, status='old',readonly,shared,err=11)
c*    write (iout,111) filnam

    1 read  (iscr,112,err=12,end=12)  string     ! skip header
c*    write (iout,113)                string
      if (string .ne. '=h==') goto 1

      read  (iscr,  *)  mx                        ! quantity of input data
```

```
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.ne.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(1)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(2)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(1)                             ! refractive index
          c(k,2) = u(2)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(1)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(2)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do        ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                                ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
      end do
```

```
c     ====================================================
      read  (iscr,112,err=12,end=12)  string     ! end of information
c*    write (iout,113)                string
      if (string .eq. '=h==') then
          close (iscr)
      else
          write (iout,114)
          stop
      end if

      llmnts(8) = .true.
      iptr = 1                          ! initialize

    2 if (wavlen .lt. 0.0) then         ! nm, wavelength
          x = -wavlev /wavlen           !     eV <--- nm
      else
          x = wavlen                    !     eV, energy
      end if

      if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
          write (iout,132)  x1,x2,x
          stop
      end if

      do i=1,2
         u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
         v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
      end do

      dielec = cmplx (u(1), u(2))                          ! (n+ik)
      dielew = cmplx (v(1), v(2))                          ! (n+ik)'

      dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
      dielec = dielec*dielec                               ! e

      return
c     ====================================================

   11 write (iout,141)
      stop
   12 write (iout,142)
      stop
   13 write (iout,143)
      stop

  111 format ( ' lmnt ' 8,  filename = ', a)
  112 format ( a)
  113 format ( ' string = ', a4 )
  114 format ( ' oops,  error in end-of-info header')

  121 format ( ' mx = ', i5)
  122 format ( ' oops, inconsistency within:   mx')
  123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
```

```
157    124 format ( ' oops, ordering violation ' )
158
159    131 format ( ' oops, error in B-spline formation, '
160       &       /   10x, 'info =', i2
161       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
162    132 format ( ' oops, evaluation point is exterior domain,'
163       &       / '       x1,x2,x " ', 3f10.3)
164
165    141 format ( ' oops, error in opening input data file ' )
166    142 format ( ' oops, error in locating header card,   ' )
167    143 format ( ' oops, error in reading (i,x,u)        ' )
168
169        end
```

6.5.9 DIEL09.FOR, $Al_xGa_{1-x}As$ (crystalline)

```
c     ------------------------------------------------------------------
c     Optical properties of:     Aluminum Gallium Arsenide
c                                Al(x)    Ga(1-x) As

c     Fetch information on the dielectric function.
c     Here, we assume that:
c     1)   partitions involve distinct composition/stoichiometry
c     2)   each partition has its OWN discretization (energy)
c     3)   discretization of the input data file is of either form:
c           i)  energy (eV),     refractive index,  extinction coeff.
c          ii)  energy (eV),  dielectric function,  conductivity
c     ------------------------------------------------------------------

      subroutine diel09 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam
      character*4   string                        ! convenience
      integer       kkpp(nparms)
      complex       dielc(2), dielw(2)

c     Let:  MX  be the maximum number of grid points in any profile, u(x).
c     If we then add-include the first derivatives at the end points,
c     dudx(1) & dudx(mx),  we need spline fit:  nx=mx+2  grid points.

      parameter   (ny = 9 )       ! number of distinct stoichiometries
c     parameter   (mx = 47)       ! max number of points in any profile
      parameter   (nx = 49,     ko = 4)
      parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)

      common / dieoo9 / t(nx+ko,ny), c(nx,2,ny),
     &                        x1,x2, mx, iptrx(ny), mxsave(ny),
     &                  yy(ny), y1,y2, my, iptry
      common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(4), v(2)


      if (llmnts(9))  goto 2           ! info already in arrays

      filnam = 'Al_Ga_As'              ! Aluminum Gallium Arsenide
```

```
51        write (iout,111) filnam
52        filnam = subdir//filnam
53        call ljchar (filnam)              ! left-justify characters
54        open (iscr, file=filnam, status='old',readonly,shared,err=11)
55  c     write (iout,111) filnam
56      1 read  (iscr,112,err=12,end=12)  string     ! skip header
57  c     write (iout,113)                  string
58        if (string .ne. '=h==') goto 1
59  c     ------------------------------------------
60
61        read  (iscr,  *)  my                        ! quantity of partitions
62  c     write (iout,121)  my                        !    distinct compositions
63        if (my.lt.1 .or. my.ne.ny) then             !    involving Aluminum
64            write (iout,122)  my, ny
65            stop
66        end if
67        read  (iscr,112)  string                    ! skip delimiter
68
69        do iy=1,my                          ! scan partitions, stoichiometry
70            read  (iscr,  *)  mx, y, jy             ! quantity of input data
71  c         write (iout,131)  mx, y, jy             !   within one partition
72
73            if (mx.lt.3   .or.
74       &        y.lt.0.0  .or.  y.gt.1.0  .or.  jy.ne.iy) then
75                write (iout,132)  mx,y,jy,   nx,iy
76                stop
77            end if
78            if (iy.ne.1) then                       ! verify ordering
79                if (y .le. yy(iy-1)) then           !     of monotonically
80                    write (iout,133)  iy, y         !     increasing
81                    stop                            !     stoichiometry
82                end if
83            end if
84
85            yy(iy) = y                              ! retain composition ~ Al
86            mxsave(iy) = mx                         ! retain number of points
87            k = 0                                   ! index knots
88            do i=1,mx                               ! scan    energy (eV)
89                read  (iscr,*,err=15,end=15)  x, u
90  c             write (iout,134)              x, u
91
92                if (i.ne.1) then                    ! verify ordering
93                    if (x .le. xx(k)) then          !     of monotonically
94                        write (iout,135)            !     increasing energy
95                        stop
96                    end if
97                end if
98                if (i.eq.2) then                    ! usual cubic spline
99                    k = k+1                             ! update index label
100                   id(k) = 1                           ! i-th derivative
101                   xx(k) = xx(1)                       ! energy (eV)
102                   h   = x-xx(1)                       ! energy increment
103                   c(k,1,iy) = (u(3)-c(1,1,iy)) /h     ! refractive index'
```

```
104                     c(k,2,iy) = (u(4)-c(1,2,iy)) /h     ! extinction coeff'
105                 end if
106
107                 k = k+1                                 ! update index label
108                 id(k) = 0                               ! i-th derivative
109                 xx(k) = x                               ! energy (eV)
110                 c(k,1,iy) = u(3)                        ! refractive index
111                 c(k,2,iy) = u(4)                        ! extinction coeff
112
113                 if (i.eq.mx) then                   ! usual cubic spline
114                     k = k+1                             ! update index label
115                     id(k) = 1                           ! i-th derivative
116                     xx(k) = x                           ! energy (eV)
117                     h = x-xx(k-2)                       ! energy increment
118                     c(k,1,iy) = (u(3)-c(k-2,1,iy)) /h   ! refractive index'
119                     c(k,2,iy) = (u(4)-c(k-2,2,iy)) /h   ! extinction coeff'
120                 end if
121             end do              ! x,u
122 c           ----------------------------------------
123             x1 = xx(1)                              ! minimum
124             x2 = xx(k)                              ! maximum
125             mx = k                          ! number of points spline fitted
126             do i=1,2
127                 call bindk (xx, c(1,i,iy), id, mx, ko,
128      &                    t(1,iy), c(1,i,iy),          w,iw,info)
129                 if (info.ne.0) then
130                     write (iout,151)  info, i, iy
131                     stop
132                 end if
133             end do
134
135             read  (iscr,112)  string                ! skip delimiter
136         end do                                      ! stoichiometry
137 c       ----------------------------------------
138         read  (iscr,112,err=13,end=13)  string      ! end of info
139 c       write (iout,113)                string
140         if (string .ne. '=h==')  goto 14
141
142         close (iscr)
143         llmnts(9) = .true.                  ! info in arrays
144         do iy=1,my
145             iptrx(iy) = 1                   ! initialize
146         end do
147         iptry = 1
148         y1 = yy( 1)                         ! minimum, convenience
149         y2 = yy(my)                         ! maximum
150 c       ================================================================
151
152       2 if (wavlen .lt. 0.0) then           ! nm, wavelength
153             x = -wavlev /wavlen             !     eV <--- nm
154         else
155             x = wavlen                      !     eV, energy
156         end if
```

```
157       if (x.lt.x1 .or. x.gt.x2) then   ! point exterior domain
158           write (iout,161)  x1, x2, x
159           stop
160       end if
161
162 c     ---------------------------------
163       if (mipp.eq.0 .or.
164      &    mrpp.eq.0     ) then         ! consistency test
165           write (iout,162)
166           stop
167       end if
168       k = 0
169       do i=1,mipp                      ! discern pointer parameter
170           if (iipp(i) .eq. 9) then     ! naive identification-recognition
171               k = k+1                  ! index local parameters
172               kkpp(k) = i
173           end if
174       end do
175       if (k.ne.1 .or. k.gt.mrpp) then  ! precaution, verification
176           write (iout,163)
177           stop
178       end if
179
180       i = kkpp(k)                      ! locate  composition index
181       y = rrpp(i)                      ! specify composition ' Al
182
183       if (y.lt.y1 .or. y.gt.y2) then   ! point exterior domain
184           write (iout,164)  y1, y2, y
185           stop
186       end if
187
188       j = 1                            ! need one composition, default
189       iy = iptry
190       if (y .eq. yy(1)) then           ! lower boundary
191           iy = 1
192           goto 5
193       else if (y .eq. yy(my)) then     ! upper boundary
194           iy = my
195           goto 5
196       end if
197     3 if (y .eq. yy(iy)) goto 5        ! grid point
198       if (y .lt. yy(iy)) then          ! grid point is high
199           iy = iy-1
200           goto 3
201       end if
202     4 if (y .gt. yy(iy)) then          ! grid point is low
203           iy = iy+1
204           if (y .lt. yy(iy)) then      ! success
205               j = 2                    ! need two compositions
206           else
207               goto 3
208           end if
209       end if
```

```
210
211      5 iptry = iy                          !  yy(iy-1)  <  y  <=  yy(iy)
212
213  c     Necessary to evaluate:    de/dy * de/d stoichiometry
214
215        iys = -1                                        ! step down
216        if (j.eq.1 .and. iy.eq.1) iys=1                 ! step up, lower bdry
217        do jj=1,2
218            if (jj.eq.2) iy=iy+iys                      ! step down/up
219
220            mx = mxsave(iy)
221
222            do i=1,2
223                u(i) = bvalu (t(1,iy),c(1,i,iy),mx,ko, 0,x, iptrx(iy),w)
224                v(i) = bvalu (t(1,iy),c(1,i,iy),mx,ko, 1,x, iptrx(iy),w)
225            end do
226
227            dielec = cmplx (u(1), u(2))                 ! (n+ik)
228            dielew = cmplx (v(1), v(2))                 ! (n+ik)'
229            dielew = dielec*dielew*cmplx (2.0, 0.0)     ! e'
230            dielec = dielec*dielec                      ! e
231            dielc(jj) = dielec
232            dielw(jj) = dielew
233        end do
234        iy = iy-iys                                     ! reset
235        diff =  yy(iy) - yy(iy+iys)                     ! > 0, usually
236        i = kkpp(k)                                     ! locate composition
237        ddpp(i) = (dielc(1)-dielc(2)) /cmplx (diff,0.0)      ! de/dy
238
239        if (j.eq.1) then                                ! grid point
240            dielec = dielc(1)
241            dielew = dielw(1)
242        else                                            !  interpolate, j=2
243            frac = (y-yy(iy-1)) /diff                   !  above
244            dielc(1) = dielc(1)*cmplx (frac, 0.0)
245            dielw(1) = dielw(1)*cmplx (frac, 0.0)
246            frac = (yy(iy)-y) /diff                     !  below
247            dielc(2) = dielc(2)*cmplx (frac, 0.0)
248            dielw(2) = dielw(2)*cmplx (frac, 0.0)
249            dielec = dielc(1)+dielc(2)                  ! linear interpolation
250            dielew = dielw(1)+dielw(2)                  ! between compositions
251        end if
252
253        return
254  c     =====================================================
255
256     11 write (iout,115)                    ! opening
257        stop
258     12 write (iout,116)                    ! end of header
259        stop
260     13 write (iout,117)                    ! end of info
261        stop
262     14 write (iout,118)                    ! end of info  *  improper
```

```
      stop
   15 write (iout,119)                ! (i,x,u)
      stop

  111 format ( ' lmnt ' 9,  filename = ', a)
  112 format ( a)
  113 format ( ' string: ', a)

  115 format ( ' oops,  unable to open:   input data file          ')
  116 format ( ' oops,  unable to find:   delimiter of  header     ')
  117 format ( ' oops,  unable to find:   delimiter of  End-of-Info')
  118 format ( ' oops,  improper delimiting string:     End-of-Info')
  119 format ( ' oops,  unable to read:   (i,x,u)                  ')

  121 format ( ' my = ', i5)
  122 format ( ' oops,  inconsistency,   my = ', i5
     &        /'                         ny = ', i5)

  131 format ( 1x, i4, 2x, f10.4, 2x, i4, 10x, '(mx, y, iy)')
  132 format ( ' oops, inconsistency among the values:    mx, y, iy'
     &        / 7x,  i4, 2x, f10.4, 2x, i4
     &        / 7x,  i4,      14x,      i4)
  133 format ( ' oops, violation in ordering:  y'
     &        /'       iy = ', i4
     &        /'        y = ', f10.4)
  134 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
  135 format ( ' oops, violation in ordering:   x')

  151 format ( ' oops,  error in construction of the B-splines '
     &       /  10x, 'info =', i2
     &       /  10x, '   i =', i2, ',', 4x, '(1"u, 2"du/dx)'
     &       /  10x, '  iy =', i2, ',', 4x, '(amount of Aluminum)')

  161 format (' oops, evaluation point is exterior domain, (DIEL09)'
     &       /'       (x1, x2, x) " ',  3(2x, f10.4))
  162 format (' oops, need to pass some parameter, so as to specify '
     &       /'       ... the composition of Aluminum')
  163 format (' oops, need a pointer from  within:    iipp'
     &       /'       to discern the parameter in:    rrpp')
  164 format (' oops, evaluation point is exterior domain, (DIEL09)'
     &       /'       (y1, y2, y) " ',  3(2x, f10.4))

      end
```

6.5.10 DIEL10.FOR, Oxides of GaAs

```
c     ----------------------------------------------------------------
c     Optical properties of:     Gallium Arsenide Oxide.
c     There are several oxide stoichiometries.
c     These include:    Ga(2)O(3),  GaAsO(4),  As(2)O(3).
c     The order of preference is from (most to least), respectively.
c     Reference:     D.E. Aspnes, G.P. Schwartz, G.J. Gualtieri,
c                    A.A. Studna, and B. Schwartz,
c                    Journal Electrochemical Society,
c                    Volume 128,  Number 3, pp. 590-597,  (1981).
c     ----------------------------------------------------------------

      subroutine diel10 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam

      real  eg(4), eo(4)
      data  eg / 0.075, 5.5,  7.5, 16.0 /         ! poles, Note:  5.5
      data  eo / 0.35,  1.62, 5.5, 18.8 /
      data  x1, x2 / 1.5, 4.5 /                   ! domain

      if (.not.llmnts(10)) then                   ! info already in arrays
          llmnts(10) = .true.                     ! info in arrays
          filnam = 'Ga_As_Oxide'                  ! Gallium Arsenide Oxide
          write (iout,111) filnam
      end if
      if (mipp.ne.0 .or. mrpp.ne.0) then          ! no parameters necessary
          write (iout,112)
          stop
      end if

      if (wavlen .lt. 0.0) then                   ! nm, wavelength
          x = -wavlev /wavlen                     !     eV <--- nm
      else
          x = wavlen                              !     eV, energy
      end if
      if (x.lt.x1 .or. x.gt.x2) then              ! point exterior domain
          write (iout,121)  x1, x2, x
          stop
```

```
51        end if
52
53  c     Evaluate analytic expression for the dielectric function.
54
55        diel = 0.0
56        diew = 0.0
57        do i=1,4
58            top = eg(i)*eo(i)
59            bot = eg(i)**2 - x**2
60            rat = top/bot
61            diel = diel + rat                        ! e
62            diew = diew + rat*(x/bot)*2.0            ! e'
63        end do
64
65        dielec = cmplx (1.0+diel, 0.0)               ! e
66        dielew = cmplx (    diew, 0.0)               ! e'
67
68        return
69
70     111 format ( ' lmnt ' 10,  filename = ', a)
71     112 format (' oops, no parameters necessary')
72     121 format (' oops, evaluation point is exterior domain, (DIEL10)'
73       &       /'       (x1, x2, x) ' ',  3(2x, f10.4))
74        end
```

6.5.11 DIEL11.FOR, As (amorphous)

```
c     ------------------------------------------------------------------
c     Dielectric function of amorphous arsenic.
c     Reference:
c        "Optical Properties of Amorphous Arsenic",
c         G.N. Greaves, E.A.Davis, and J.Bordas,
c         Philosophical Magazine, Vol 34, No.2, pp. 265-290, 1976.
c         Specifically, see page 278.
c     ------------------------------------------------------------------
c*    subroutine diel11 (dielec, dielew, energy)
      subroutine diel11 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp, mrpp,rrpp,llpp, ddpp,  ! mixture
     &                   mipa,iipa, mrpa,rrpa,llpa, ddpa)  ! ambient

      integer    iipa(1), iipp(1)
      real       rrpa(1), rrpp(1), anglei, wavlen
      integer    llpa(1), llpp(1)
      complex    ddpa(1), ddpp(1), dielec, dielew
      real       energy                                    ! eV
      complex    plasma

      include   'iounit.'
      include   'defnit.'
      include   'elmnts.'
      include   'handyy.'    ! pi, cccc, wavlev

      data  hbarwo /  3.9 /  ! natural freq, interband transition, (eV).
      data  hbarwp / 18.0 /  ! plasma freq, (eV).
      data  hbtaui /  5.0 /  ! damping, (eV).

c     ---------------------------------
      if (.not.llmnts(11)) then
          llmnts(11) = .true.
          write (iout,11)
      end if

      if (wavlen .lt. 0.0) then        ! nm, wavelength
          energy = - wavlev/wavlen     !     eV <--- nm
      else
          energy = wavlen              !     eV
      end if
c     ---------------------------------

      plasma = cmplx (hbarwp**2, 0.0)

      t1 = (hbarwo-energy)*(hbarwo+energy)
      t2 =  energy*hbtaui
      dielec = cmplx (t1, -t2)         ! denominator

      t1p = energy*2.0                 ! d/d(energy)
      t2p = hbtaui
```

```
51          dielew = (plasma/dielec)*(cmplx (t1p, t2p) /dielec)
52
53          dielec = cmplx (1.0, 0.0) + plasma/dielec
54          return
55
56       11 format ( ' lmnt ' 11,    amorphous arsenic')
57          end
```

6.5.12 DIEL12.FOR, GaP

```
1  c     ------------------------------------------------------------------
2  c     Optical properties of:   Gallium Phosphide (GaP),  crystalline.
3  c     ------------------------------------------------------------------
4
5        subroutine diel12 (anglei, wavlen, dielec, dielew,
6       &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
7       &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient
8
9        integer   iipa(1), iipp(1)
10       real      rrpa(1), rrpp(1), anglei, wavlen
11       integer   llpa(1), llpp(1)
12       complex   ddpa(1), ddpp(1), dielec, dielew
13
14       include  'iounit.'
15       include  'defnit.'
16       include  'elmnts.'
17       include  'handyy.'                          ! pi,cccc,wavlev
18
19       character*64  filnam
20       character*4   string                        ! convenience
21
22 c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
23 c     in order to include the first derivatives at each end point.
24 c     Let:     nx > mx+2.
25
26       parameter   (nx = 158,          ko = 4)
27       parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
28       common / dieo12 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
29       common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(2), v(2)
30
31
32       if (llmnts(12))  goto 2
33
34 c     Fetch information on the dielectric function.
35 c     Assume:   1)   no partitions
36 c               2)   discretization of input data is of either form:
37 c                    a)  energy (eV),     refractive index,  extinction
38 c                    b)  energy (eV),  dielectric function,  conductivity
39
40       filnam = 'Ga_P'                  ! Gallium Phosphide
41       write (iout,111) filnam
42       filnam = subdir//filnam
43       call ljchar (filnam)             ! left-justify characters
44       open (iscr, file=filnam, status='old',readonly,shared,err=11)
45 c*    write (iout,111) filnam
46
47     1 read  (iscr,112,err=12,end=12)  string     ! skip header
48 c*    write (iout,113)                 string
49       if (string .ne. '=h==') goto 1
50
```

```
      read  (iscr,  *)  mx                          ! quantity of input data
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.gt.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(1)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(2)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(1)                             ! refractive index
          c(k,2) = u(2)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(1)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(2)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do        ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                                ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
```

```
104         end do
105   c     ======================================================
106         read  (iscr,112,err=12,end=12)  string     ! end of information
107   c*    write (iout,113)                string
108         if (string .eq. '=h==') then
109             close (iscr)
110         else
111             write (iout,114)
112             stop
113         end if
114
115         llmnts(12) = .true.
116         iptr = 1                          ! initialize
117
118       2 if (wavlen .lt. 0.0) then         ! nm, wavelength
119             x = -wavlev /wavlen           !     eV <--- nm
120         else
121             x = wavlen                    !     eV, energy
122         end if
123
124         if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
125             write (iout,132)  x1,x2,x
126             stop
127         end if
128
129         do i=1,2
130            u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
131            v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
132         end do
133
134         dielec = cmplx (u(1), u(2))                          ! (n+ik)
135         dielew = cmplx (v(1), v(2))                          ! (n+ik)'
136
137         dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
138         dielec = dielec*dielec                               ! e
139
140         return
141   c     ======================================================
142
143      11 write (iout,141)
144         stop
145      12 write (iout,142)
146         stop
147      13 write (iout,143)
148         stop
149
150     111 format ( ' lmnt ' 12,  filename = ', a)
151     112 format ( a)
152     113 format ( ' string = ', a)
153     114 format ( ' oops,  error in end-of-info header')
154
155     121 format ( ' mx = ', i5)
156     122 format ( ' oops, inconsistency within:   mx')
```

```
157    123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
158    124 format ( ' oops, ordering violation ' )
159
160    131 format ( ' oops, error in B-spline formation, '
161       &       /   10x, 'info =', i2
162       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
163    132 format ( ' oops, evaluation point is exterior domain,'
164       &       / '       x1,x2,x " ', 3f10.3)
165
166    141 format ( ' oops, error in opening input data file ' )
167    142 format ( ' oops, error in locating header card,   ' )
168    143 format ( ' oops, error in reading (i,x,u)         ' )
169
170        end
```

6.5.13 DIEL13.FOR, GaSb

```
c     ------------------------------------------------------------
c     Optical properties of:   Gallium Antimonide (GaSb),  crystalline.
c     ------------------------------------------------------------

      subroutine diel13 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam
      character*4   string                        ! convenience

c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
c     in order to include the first derivatives at each end point.
c     Let:     nx > mx+2.

      parameter   (nx =  48,          ko = 4)
      parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
      common / dieo13 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
      common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(4), v(2)


      if (llmnts(13))  goto 2

c     Fetch information on the dielectric function.
c     Assume:   1)   no partitions
c               2)   discretization of input data is of either form:
c                    a)  energy (eV),     refractive index,  extinction
c                    b)  energy (eV),  dielectric function,  conductivity

      filnam = 'Ga_Sb'                 ! Gallium Antimonide
      write (iout,111) filnam
      filnam = subdir//filnam
      call ljchar (filnam)             ! left-justify characters
      open (iscr, file=filnam, status='old',readonly,shared,err=11)
c*    write (iout,111) filnam

    1 read  (iscr,112,err=12,end=12)  string     ! skip header
c*    write (iout,113)                string
      if (string .ne. '=h==') goto 1

```

```
51        read  (iscr,  *)  mx                          ! quantity of input data
52  c*    write (iout,121)  mx                          !   within one partition
53        if (mx.lt.3  .or.  mx+2.gt.nx)  then
54            write (iout,122)
55            stop
56        end if
57
58        k = 0                                         ! index knots
59        do i=1,mx
60            read  (iscr,*,err=13,end=13)  x, u
61  c*        write (iout,123)              x, u
62
63            if (i.ne.1) then                          ! induce ordering
64                if (x .le. xx(k)) then
65                    write (iout,124)
66                    stop
67                end if
68            end if
69
70            if (i.eq.2) then              ! usual case for cubic spline fit
71                k = k+1
72                xx(k) = xx(1)                         ! energy (eV)
73                id(k) = 1                             ! i-th derivative
74                c(k,1) = (u(3)-c(1,1)) /(x-xx(1))     ! refractive index
75                c(k,2) = (u(4)-c(1,2)) /(x-xx(1))     ! extinction
76            end if
77
78            k = k+1
79            xx(k) = x                                 ! energy (eV)
80            id(k) = 0                                 ! i-th derivative
81            c(k,1) = u(3)                             ! refractive index
82            c(k,2) = u(4)                             ! extinction
83
84            if (i.eq.mx) then             ! usual case for cubic spline fit
85                k = k+1
86                xx(k) = x                                   ! energy (eV)
87                id(k) = 1                                   ! i-th derivative
88                c(k,1) = (u(3)-c(k-2,1)) /(x-xx(k-2))       ! refractive index
89                c(k,2) = (u(4)-c(k-2,2)) /(x-xx(k-2))       ! extinction
90            end if
91
92        end do        ! x,u
93  c     ----------------------------------------------------
94        x1 = xx(1)
95        x2 = xx(k)
96        mx = k                                ! quantity of data points
97
98        do i=1,2
99            call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
100           if (info.ne.0) then
101               write (iout,131)  info, i
102               call exit (2)
103           end if
```

```
104         end do
105   c     ====================================================
106         read  (iscr,112,err=12,end=12)  string     ! end of information
107   c*    write (iout,113)                string
108         if (string .eq. '=h==') then
109             close (iscr)
110         else
111             write (iout,114)
112             stop
113         end if
114
115         llmnts(13) = .true.
116         iptr = 1                          ! initialize
117
118       2 if (wavlen .lt. 0.0) then         ! nm, wavelength
119             x = -wavlev /wavlen           !     eV <--- nm
120         else
121             x = wavlen                    !     eV, energy
122         end if
123
124         if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
125             write (iout,132)  x1,x2,x
126             stop
127         end if
128
129         do i=1,2
130            u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
131            v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
132         end do
133
134         dielec = cmplx (u(1), u(2))                          ! (n+ik)
135         dielew = cmplx (v(1), v(2))                          ! (n+ik)'
136
137         dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
138         dielec = dielec*dielec                               ! e
139
140         return
141   c     ====================================================
142
143      11 write (iout,141)
144         stop
145      12 write (iout,142)
146         stop
147      13 write (iout,143)
148         stop
149
150     111 format ( ' lmnt #',13,  filename = ', a)
151     112 format ( a)
152     113 format ( ' string = ', a)
153     114 format ( ' oops,  error in end-of-info header')
154
155     121 format ( ' mx = ', i5)
156     122 format ( ' oops, inconsistency within:   mx')
```

```
157    123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
158    124 format ( ' oops, ordering violation ' )
159
160    131 format ( ' oops, error in B-spline formation, '
161       &       /   10x, 'info =', i2
162       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
163    132 format ( ' oops, evaluation point is exterior domain,'
164       &       / '       x1,x2,x " ', 3f10.3)
165
166    141 format ( ' oops, error in opening input data file ' )
167    142 format ( ' oops, error in locating header card,   ' )
168    143 format ( ' oops, error in reading (i,x,u)         ' )
169
170        end
```

6.5.14 DIEL14.FOR, InAs

```
c     ------------------------------------------------------------------
c     Optical properties of:   Indium Arsenide    (InAs),  crystalline.
c     ------------------------------------------------------------------

      subroutine diel14 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam
      character*4   string                         ! convenience

c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
c     in order to include the first derivatives at each end point.
c     Let:     nx > mx+2.

      parameter   (nx =  48,          ko = 4)
      parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
      common / dieo14 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
      common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(4), v(2)


      if (llmnts(14))  goto 2

c     Fetch information on the dielectric function.
c     Assume:   1)   no partitions
c               2)   discretization of input data is of either form:
c                    a)  energy (eV),     refractive index,  extinction
c                    b)  energy (eV),  dielectric function,  conductivity

      filnam = 'In_As'                 ! Indium Arsenide
      write (iout,111) filnam
      filnam = subdir//filnam
      call ljchar (filnam)             ! left-justify characters
      open (iscr, file=filnam, status='old',readonly,shared,err=11)
c*    write (iout,111) filnam

    1 read  (iscr,112,err=12,end=12)  string     ! skip header
c*    write (iout,113)                string
      if (string .ne. '=h==') goto 1

```

```
      read  (iscr,  *)  mx                          ! quantity of input data
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.gt.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(3)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(4)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(3)                             ! refractive index
          c(k,2) = u(4)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(3)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(4)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do        ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                                ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
```

```
      end do
c     ======================================================
      read  (iscr,112,err=12,end=12)  string     ! end of information
c*    write (iout,113)                string
      if (string .eq. '=h==') then
          close (iscr)
      else
          write (iout,114)
          stop
      end if

      llmnts(14) = .true.
      iptr = 1                          ! initialize

    2 if (wavlen .lt. 0.0) then         ! nm, wavelength
          x = -wavlev /wavlen           !     eV <--- nm
      else
          x = wavlen                    !     eV, energy
      end if

      if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
          write (iout,132)  x1,x2,x
          stop
      end if

      do i=1,2
         u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
         v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
      end do

      dielec = cmplx (u(1), u(2))                          ! (n+ik)
      dielew = cmplx (v(1), v(2))                          ! (n+ik)'

      dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
      dielec = dielec*dielec                               ! e

      return
c     ======================================================

   11 write (iout,141)
      stop
   12 write (iout,142)
      stop
   13 write (iout,143)
      stop

  111 format ( ' lmnt ' 14,  filename = ', a)
  112 format ( a)
  113 format ( ' string = ', a)
  114 format ( ' oops,  error in end-of-info header')

  121 format ( ' mx = ', i5)
  122 format ( ' oops, inconsistency within:   mx')
```

```
157    123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
158    124 format ( ' oops, ordering violation ' )
159
160    131 format ( ' oops, error in B-spline formation, '
161       &       /   10x, 'info =', i2
162       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
163    132 format ( ' oops, evaluation point is exterior domain,'
164       &       / '       x1,x2,x " ', 3f10.3)
165
166    141 format ( ' oops, error in opening input data file ' )
167    142 format ( ' oops, error in locating header card,   ' )
168    143 format ( ' oops, error in reading (i,x,u)         ' )
169
170        end
```

6.5.15 DIEL15.FOR, InP

```
c     ------------------------------------------------------------------
c     Optical properties of:   Indium Phosphide   (InP),   crystalline.
c     ------------------------------------------------------------------

      subroutine diel15 (anglei, wavlen, dielec, dielew,
     &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
     &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient

      integer   iipa(1), iipp(1)
      real      rrpa(1), rrpp(1), anglei, wavlen
      integer   llpa(1), llpp(1)
      complex   ddpa(1), ddpp(1), dielec, dielew

      include  'iounit.'
      include  'defnit.'
      include  'elmnts.'
      include  'handyy.'                          ! pi,cccc,wavlev

      character*64  filnam
      character*4   string                        ! convenience

c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
c     in order to include the first derivatives at each end point.
c     Let:     nx > mx+2.

      parameter   (nx =  48,          ko = 4)
      parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
      common / dieo15 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
      common / dieoco / xx(nx), id(nx), iw(nx), w(nw), u(4), v(2)


      if (llmnts(15))  goto 2

c     Fetch information on the dielectric function.
c     Assume:   1)   no partitions
c               2)   discretization of input data is of either form:
c                    a)  energy (eV),     refractive index,  extinction
c                    b)  energy (eV),  dielectric function,  conductivity

      filnam = 'In_P'                  ! Indium Phosphide
      write (iout,111) filnam
      filnam = subdir//filnam
      call ljchar (filnam)             ! left-justify characters
      open (iscr, file=filnam, status='old',readonly,shared,err=11)
c*    write (iout,111) filnam

    1 read  (iscr,112,err=12,end=12)  string     ! skip header
c*    write (iout,113)                string
      if (string .ne. '=h==') goto 1

```

```
      read  (iscr,  *)  mx                          ! quantity of input data
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.gt.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(3)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(4)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(3)                             ! refractive index
          c(k,2) = u(4)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                     ! energy (eV)
              id(k) = 1                                     ! i-th derivative
              c(k,1) = (u(3)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(4)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do        ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                                ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
```

```
104         end do
105   c     ====================================================
106         read  (iscr,112,err=12,end=12)  string     ! end of information
107   c*    write (iout,113)                string
108         if (string .eq. '=h==') then
109             close (iscr)
110         else
111             write (iout,114)
112             stop
113         end if
114
115         llmnts(15) = .true.
116         iptr = 1                          ! initialize
117
118       2 if (wavlen .lt. 0.0) then         ! nm, wavelength
119             x = -wavlev /wavlen           !     eV <--- nm
120         else
121             x = wavlen                    !     eV, energy
122         end if
123
124         if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
125             write (iout,132)  x1,x2,x
126             stop
127         end if
128
129         do i=1,2
130            u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
131            v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
132         end do
133
134         dielec = cmplx (u(1), u(2))                          ! (n+ik)
135         dielew = cmplx (v(1), v(2))                          ! (n+ik)'
136
137         dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
138         dielec = dielec*dielec                               ! e
139
140         return
141   c     ====================================================
142
143      11 write (iout,141)
144         stop
145      12 write (iout,142)
146         stop
147      13 write (iout,143)
148         stop
149
150     111 format ( ' lmnt ' 15,  filename = ', a)
151     112 format ( a)
152     113 format ( ' string = ', a)
153     114 format ( ' oops,  error in end-of-info header')
154
155     121 format ( ' mx = ', i5)
156     122 format ( ' oops, inconsistency within:   mx')
```

```
157    123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
158    124 format ( ' oops, ordering violation ' )
159
160    131 format ( ' oops, error in B-spline formation, '
161       &       /   10x, 'info =', i2
162       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
163    132 format ( ' oops, evaluation point is exterior domain,'
164       &       / '       x1,x2,x " ', 3f10.3)
165
166    141 format ( ' oops, error in opening input data file ' )
167    142 format ( ' oops, error in locating header card,   ' )
168    143 format ( ' oops, error in reading (i,x,u)         ' )
169
170        end
```

6.5.16 DIEL16.FOR, InSb

```
1  c     ------------------------------------------------------------
2  c     Optical properties of:   Indium Antimonide  (InSb),  crystalline.
3  c     ------------------------------------------------------------
4
5        subroutine diel16 (anglei, wavlen, dielec, dielew,
6       &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
7       &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient
8
9        integer   iipa(1), iipp(1)
10       real      rrpa(1), rrpp(1), anglei, wavlen
11       integer   llpa(1), llpp(1)
12       complex   ddpa(1), ddpp(1), dielec, dielew
13
14       include  'iounit.'
15       include  'defnit.'
16       include  'elmnts.'
17       include  'handyy.'                          ! pi,cccc,wavlev
18
19       character*64  filnam
20       character*4   string                        ! convenience
21
22 c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
23 c     in order to include the first derivatives at each end point.
24 c     Let:     nx > mx+2.
25
26       parameter   (nx =  48,          ko = 4)
27       parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
28       common / dieo16 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
29       common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(4), v(2)
30
31
32       if (llmnts(16))  goto 2
33
34 c     Fetch information on the dielectric function.
35 c     Assume:   1)   no partitions
36 c               2)   discretization of input data is of either form:
37 c                    a)  energy (eV),     refractive index,  extinction
38 c                    b)  energy (eV),  dielectric function,  conductivity
39
40       filnam = 'In_Sb'                 ! Indium Antimonide
41       write (iout,111) filnam
42       filnam = subdir//filnam
43       call ljchar (filnam)             ! left-justify characters
44       open (iscr, file=filnam, status='old',readonly,shared,err=11)
45 c*    write (iout,111) filnam
46
47     1 read  (iscr,112,err=12,end=12)  string     ! skip header
48 c*    write (iout,113)                 string
49       if (string .ne. '=h==') goto 1
50
```

```
      read  (iscr,  *)  mx                          ! quantity of input data
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.gt.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(3)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(4)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(3)                             ! refractive index
          c(k,2) = u(4)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(3)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(4)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do        ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                                ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
```

```
104         end do
105   c     ====================================================
106         read  (iscr,112,err=12,end=12)  string     ! end of information
107   c*    write (iout,113)                string
108         if (string .eq. '=h==') then
109             close (iscr)
110         else
111             write (iout,114)
112             stop
113         end if
114
115         llmnts(16) = .true.
116         iptr = 1                          ! initialize
117
118       2 if (wavlen .lt. 0.0) then         ! nm, wavelength
119             x = -wavlev /wavlen           !     eV <--- nm
120         else
121             x = wavlen                    !     eV, energy
122         end if
123
124         if (x.lt.x1 .or. x.gt.x2) then    ! point exterior domain
125             write (iout,132)  x1,x2,x
126             stop
127         end if
128
129         do i=1,2
130            u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
131            v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
132         end do
133
134         dielec = cmplx (u(1), u(2))                          ! (n+ik)
135         dielew = cmplx (v(1), v(2))                          ! (n+ik)'
136
137         dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
138         dielec = dielec*dielec                               ! e
139
140         return
141   c     ====================================================
142
143      11 write (iout,141)
144         stop
145      12 write (iout,142)
146         stop
147      13 write (iout,143)
148         stop
149
150     111 format ( ' lmnt ' 16,  filename = ', a)
151     112 format ( a)
152     113 format ( ' string = ', a)
153     114 format ( ' oops,  error in end-of-info header')
154
155     121 format ( ' mx = ', i5)
156     122 format ( ' oops, inconsistency within:   mx')
```

```
157    123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
158    124 format ( ' oops, ordering violation ' )
159
160    131 format ( ' oops, error in B-spline formation, '
161       &       /   10x, 'info =', i2
162       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
163    132 format ( ' oops, evaluation point is exterior domain,'
164       &       / '       x1,x2,x " ', 3f10.3)
165
166    141 format ( ' oops, error in opening input data file ' )
167    142 format ( ' oops, error in locating header card,   ' )
168    143 format ( ' oops, error in reading (i,x,u)         ' )
169
170        end
```

6.5.17 DIEL17.FOR, AlSb

```
1  c     ------------------------------------------------------------------
2  c     Optical properties of:   Aluminum Antimonide  (AlSb),  crystalline.
3  c     ------------------------------------------------------------------
4
5        subroutine diel17 (anglei, wavlen, dielec, dielew,
6       &                   mipp,iipp,  mrpp,rrpp,llpp,  ddpp,  ! mixture
7       &                   mipa,iipa,  mrpa,rrpa,llpa,  ddpa)  ! ambient
8
9        integer   iipa(1), iipp(1)
10       real      rrpa(1), rrpp(1), anglei, wavlen
11       integer   llpa(1), llpp(1)
12       complex   ddpa(1), ddpp(1), dielec, dielew
13
14       include  'iounit.'
15       include  'defnit.'
16       include  'elmnts.'
17       include  'handyy.'                          ! pi,cccc,wavlev
18
19       character*64  filnam
20       character*4   string                        ! convenience
21
22 c     Note:    nx = mx+2,          because of:   dudx(1), dudx(mx)
23 c     in order to include the first derivatives at each end point.
24 c     Let:     nx > mx+2.
25
26       parameter   (nx =  47,          ko = 4)
27       parameter   (nw = nx*(3*ko-2) + ko*ko + ((ko+1)*(ko+2))/2)
28       common / dieo17 / t(nx+ko), c(nx,2), x1,x2, mx, iptr
29       common / dieooo / xx(nx), id(nx), iw(nx), w(nw), u(4), v(2)
30
31
32       if (llmnts(17))  goto 2
33
34 c     Fetch information on the dielectric function.
35 c     Assume:   1)   no partitions
36 c               2)   discretization of input data is of either form:
37 c                    a)  energy (eV),     refractive index,  extinction
38 c                    b)  energy (eV),  dielectric function,  conductivity
39
40       filnam = 'Al_Sb'                 ! Aluminum Antimonide
41       write (iout,111) filnam
42       filnam = subdir//filnam
43       call ljchar (filnam)             ! left-justify characters
44       open (iscr, file=filnam, status='old',readonly,shared,err=11)
45 c*    write (iout,111) filnam
46
47      1 read  (iscr,112,err=12,end=12)  string     ! skip header
48 c*    write (iout,113)                  string
49       if (string .ne. '=h==') goto 1
50
```

```
      read  (iscr,  *)  mx                          ! quantity of input data
c*    write (iout,121)  mx                          !   within one partition
      if (mx.lt.3  .or.  mx+2.gt.nx)  then
          write (iout,122)
          stop
      end if

      k = 0                                         ! index knots
      do i=1,mx
          read  (iscr,*,err=13,end=13)  x, u
c*        write (iout,123)              x, u

          if (i.ne.1) then                          ! induce ordering
              if (x .le. xx(k)) then
                  write (iout,124)
                  stop
              end if
          end if

          if (i.eq.2) then              ! usual case for cubic spline fit
              k = k+1
              xx(k) = xx(1)                         ! energy (eV)
              id(k) = 1                             ! i-th derivative
              c(k,1) = (u(3)-c(1,1)) /(x-xx(1))     ! refractive index
              c(k,2) = (u(4)-c(1,2)) /(x-xx(1))     ! extinction
          end if

          k = k+1
          xx(k) = x                                 ! energy (eV)
          id(k) = 0                                 ! i-th derivative
          c(k,1) = u(3)                             ! refractive index
          c(k,2) = u(4)                             ! extinction

          if (i.eq.mx) then             ! usual case for cubic spline fit
              k = k+1
              xx(k) = x                                 ! energy (eV)
              id(k) = 1                                 ! i-th derivative
              c(k,1) = (u(3)-c(k-2,1)) /(x-xx(k-2))     ! refractive index
              c(k,2) = (u(4)-c(k-2,2)) /(x-xx(k-2))     ! extinction
          end if

      end do       ! x,u
c     ------------------------------------------------------
      x1 = xx(1)
      x2 = xx(k)
      mx = k                            ! quantity of data points

      do i=1,2
          call bindk (xx,c(1,i),id,mx,ko,  t,c(1,i),  w,iw,info)
          if (info.ne.0) then
              write (iout,131)  info, i
              call exit (2)
          end if
```

```
104         end do
105   c     ========================================================
106         read  (iscr,112,err=12,end=12)  string     ! end of information
107   c*    write (iout,113)                string
108         if (string .eq. '=h==') then
109             close (iscr)
110         else
111             write (iout,114)
112             stop
113         end if
114
115         llmnts(17) = .true.
116         iptr = 1                            ! initialize
117
118       2 if (wavlen .lt. 0.0) then           ! nm, wavelength
119             x = -wavlev /wavlen             !     eV <--- nm
120         else
121             x = wavlen                      !     eV, energy
122         end if
123
124         if (x.lt.x1 .or. x.gt.x2) then      ! point exterior domain
125             write (iout,132)  x1,x2,x
126             stop
127         end if
128
129         do i=1,2
130            u(i) = bvalu (t,c(1,i),mx,ko,  0,x,  iptr,w)      ! (n+ik)
131            v(i) = bvalu (t,c(1,i),mx,ko,  1,x,  iptr,w)      ! (n+ik)'
132         end do
133
134         dielec = cmplx (u(1), u(2))                          ! (n+ik)
135         dielew = cmplx (v(1), v(2))                          ! (n+ik)'
136
137         dielew = dielec*dielew*cmplx (2.0, 0.0)              ! e'
138         dielec = dielec*dielec                               ! e
139
140         return
141   c     ========================================================
142
143      11 write (iout,141)
144         stop
145      12 write (iout,142)
146         stop
147      13 write (iout,143)
148         stop
149
150     111 format ( ' lmnt ' 17,  filename = ', a)
151     112 format ( a)
152     113 format ( ' string = ', a)
153     114 format ( ' oops,  error in end-of-info header')
154
155     121 format ( ' mx = ', i5)
156     122 format ( ' oops, inconsistency within:   mx')
```

```
157    123 format ( 1x, i4, 2x, 3f10.4, 10x, '(j,x,u)' )
158    124 format ( ' oops, ordering violation ' )
159
160    131 format ( ' oops, error in B-spline formation, '
161       &       /   10x, 'info =', i2
162       &       /   10x, '   i =', i2 , ',', 5x, '(1"u, 2"du/dx)' )
163    132 format ( ' oops, evaluation point is exterior domain,'
164       &       / '       x1,x2,x " ', 3f10.3)
165
166    141 format ( ' oops, error in opening input data file ' )
167    142 format ( ' oops, error in locating header card,   ' )
168    143 format ( ' oops, error in reading (i,x,u)         ' )
169
170        end
```

6.6 Databases

6.6.1 W.SI

```
1  Optical Properties of Silicon
2
3  D.E. Aspnes  &  A.A. Studna,
4  Physical Review B,  Volume 27,  Number 2, 1983, January, 15.
5  Pages:   985-1009.
6
7  ------------------------------------------------------------------------
8  E(eV),     <e1>,    <e2>,     n,       k,       R,    alpha*1.0E3 (cm**-1)
9  ------------------------------------------------------------------------
10 =h==                 ' indicate end of header information
11  48                  ' quantity of lines of data
12  1.500    13.488    0.038    3.673    0.005    0.327       0.78
13  1.600    13.793    0.057    3.714    0.008    0.331       1.25
14  1.700    14.079    0.078    3.752    0.010    0.335       1.80
15  1.800    14.413    0.099    3.796    0.013    0.340       2.38
16  1.900    14.797    0.126    3.847    0.016    0.345       3.15
17  1.959308 15.0684   0.14646  3.88185  0.019                          Aspnes
18  2.000    15.254    0.172    3.906    0.022    0.351       4.47
19  2.100    15.754    0.236    3.969    0.030    0.357       6.32
20  2.200    16.334    0.260    4.042    0.032    0.364       7.17
21  2.27037  16.7912   0.35981  4.09794  0.044                          Aspnes
22  2.300    16.994    0.396    4.123    0.048    0.372      11.19
23  2.400    17.761    0.508    4.215    0.060    0.380      14.65
24  2.500    18.661    0.630    4.320    0.073    0.390      18.48
25  2.600    19.724    0.803    4.442    0.090    0.400      23.81
26  2.700    20.987    1.193    4.583    0.130    0.412      35.63
27  2.800    22.565    1.548    4.753    0.163    0.426      46.22
28  2.900    24.574    2.017    4.961    0.203    0.442      59.75
29  3.000    27.197    2.807    5.222    0.269    0.461      81.73
30  3.100    30.874    4.321    5.570    0.387    0.486     121.62
31  3.200    36.355    7.636    6.062    0.630    0.518     204.28
32  3.300    43.264   17.717    6.709    1.320    0.561     441.68
33  3.400    35.224   35.282    6.522    2.705    0.592     932.13
34  3.500    22.394   33.818    5.610    3.014    0.575    1069.19
35  3.600    19.124   31.632    5.296    2.987    0.564    1089.90
36  3.700    17.231   31.527    5.156    3.058    0.563    1146.67
37  3.800    15.531   32.229    5.065    3.182    0.568    1225.46
38  3.900    13.965   33.567    5.016    3.346    0.577    1322.69
39  4.000    12.240   35.939    5.010    3.586    0.591    1454.11
40  4.100     9.364   39.947    5.020    3.979    0.614    1653.60
41  4.200     2.371   45.348    4.888    4.639    0.652    1974.84
42  4.300   -12.404   44.095    4.087    5.395    0.703    2351.38
43  4.400   -18.818   33.350    3.120    5.344    0.726    2383.23
44  4.500   -19.815   24.919    2.452    5.082    0.740    2317.99
45  4.600   -17.931   18.601    1.988    4.678    0.742    2181.15
46  4.700   -15.190   15.094    1.764    4.278    0.728    2038.07
47  4.800   -13.087   13.193    1.658    3.979    0.710    1936.06
48  4.900   -11.507   11.974    1.597    3.749    0.693    1862.11
```

```
49   5.000   -10.242   11.195   1.570    3.565    0.675    1806.67
50   5.100    -9.291   10.776   1.571    3.429    0.658    1772.70
51   5.200    -8.724   10.655   1.589    3.354    0.646    1767.66
52   5.300    -8.751   10.586   1.579    3.353    0.647    1801.26
53   5.400    -9.168    9.907   1.471    3.366    0.663    1842.63
54   5.500    -9.106    8.846   1.340    3.302    0.673    1840.59
55   5.600    -8.726    7.999   1.247    3.206    0.675    1820.07
56   5.700    -8.325    7.400   1.186    3.120    0.673    1802.31
57   5.800    -7.987    6.898   1.133    3.045    0.672    1789.99
58   5.900    -7.721    6.460   1.083    2.982    0.673    1783.51
59   6.000    -7.443    5.877   1.010    2.909    0.677    1769.27
60  =h==                ' indicate end of information
```

6.6.2 W.SIA

```
1  Optical Properties of Silicon (Amorphous) (a-Si).
2
3  Reference:
4    Handbook of Optical Constants,   Edited by:   Edward D. Palik,
5    (Academic Press, Inc.;  Orlando, 1985),
6    Subpart 2,    Section:  Silicon (Amorphous) (a-Si),
7    pages: 571-586,   compiled by:   H. Piller,
8    literature:    D.T. Pierce and W.E. Spicer,
9                   Physical Review B 5, 3017, (1972).
10
11 ------------------------------------------------------------------------
12 E(eV),        n,        k,        n(H),       k(H)
13 ------------------------------------------------------------------------
14 =h==          ' indicate end of header information
15   25          ' quantity of lines of data
16  1.500       3.86      0.0812      4.13        2.00E-4       (xxx?)
17  1.600       3.93      0.136       4.25        2.37E-2       (xxix)
18  1.700       4.01      0.199       4.37                      (xxi?)
19  1.800       4.09      0.271       4.49                      (xxi?)
20  1.900       4.17      0.363       4.59                      (xxi?)
21  2.000       4.23      0.461       4.71        0.217         (xxxx)
22  2.200       4.36      0.690       4.926       0.4           (xxii)
23  2.400       4.46      0.969       5.142       0.6           (xxii)
24  2.500       4.47      1.12        5.25        0.992         (xxxx)
25  2.600       4.49      1.28                    0.850         (xxii)
26  2.800       4.47      1.64                                  (xxii)
27  3.000       4.38      2.02        5.43        2.19          (xxxx)
28  3.200       4.17      2.38                                  (xxii)
29  3.400       3.90      2.66                                  (xxii)
30  3.500       3.73      2.79        4.59        3.38          (xxxx)
31  3.600       3.55      2.88
32  3.800       3.21      3.00
33  4.000       2.87      3.06        3.36        3.92          (xxxx)
34  4.200       2.56      3.04                                  (xxii)
35  4.400       2.30      2.99                                  (xxii)
36  4.600       2.07      2.93                                  (xxii)
37  4.800       1.86      2.85                                  (xxii)
38  5.000       1.69      2.76        1.66        3.38          (xxxx)
39  5.500       1.35      2.51        1.23        2.99          (xxxx)
40  6.000       1.11      2.28        0.961       2.65          (xxxx)
41 =h==                 ' indicate end of information
```

6.6.3 W.SI_O2G

```
1  Optical Properties of Silicon Dioxide (glass,amorphous) (SiO2-g).
2
3  Reference:
4    Handbook of Optical Constants,   Edited by:   Edward D. Palik,
5    (Academic Press, Inc.;  Orlando, 1985),
6    Subpart 2,    Section:  Silicon Dioxide (Glass) (SiO2),
7    pages: 749-763,   compiled by:   H.R. Phillip.
8
9  ------------------------------------------------------------------------
10 E(eV),           n,        k
11 ------------------------------------------------------------------------
12 =h==          ' indicate end of header information
13   46          ' quantity of lines of data
14  0.91018        1.44621    0.0
15  1.0985         1.44888    0.0
16  1.1449         1.44941    0.0
17  1.2228         1.45025    0.0
18  1.3863         1.45185    0.0
19  1.4550         1.45248    0.0
20  1.7549         1.45515    0.0
21  1.8566         1.45608    0.0
22  1.8892         1.45637    0.0
23  1.9257         1.45671    0.0
24  1.95930784     1.457018   0.0          Malitson
25  2.1041         1.45841    0.0
26  2.1102         1.45847    0.0
27  2.1411         1.45877    0.0
28  2.1489         1.45885    0.0
29  2.2705         1.46008    0.0
30  2.4379         1.46187    0.0
31  2.5504         1.46313    0.0
32  2.6503         1.46429    0.0
33  2.8448         1.46669    0.0
34  3.0640         1.46961    0.0
35  3.3967         1.47453    0.0
36  3.4340         1.47512    0.0
37  3.5770         1.47746    0.0
38  3.6427         1.47858    0.0
39  3.7105         1.47976    0.0
40  3.7542         1.48053    0.0
41  4.1034         1.48719    0.0
42  4.1784         1.48873    0.0
43  4.2848         1.49099    0.0
44  4.4226         1.49404    0.0
45  4.5040         1.49592    0.0
46  4.5940         1.49805    0.0
47  4.6751         1.50004    0.0
48  4.9939         1.50841    0.0
49  5.1674         1.51338    0.0
50  5.2131         1.51474    0.0
```

```
51   5.3858          1.52009     0.0
52   5.4680          1.52276     0.0
53   5.7819          1.53371     0.0
54   5.7976          1.53429     0.0
55   6.0             1.543       0.0
56   6.25            1.554       0.0
57   6.5             1.567       0.0
58   6.75            1.582       0.0
59   7.0             1.600       0.0
60  =h==                 ' indicate end of information
```

6.6.4 W.SI3_N4

```
1  Optical Properties of Silicon Nitride (non-crystalline) (Si3N4).
2
3  Reference:
4    Handbook of Optical Constants,   Edited by:   Edward D. Palik,
5    (Academic Press, Inc.;  Orlando, 1985),
6    Subpart 2,    Section:  Silicon Nitride (non-crystalline) (Si3N4),
7    pages: 771-774,   compiled by:   H.R. Phillip.
8
9  Note:    k should be fitted logarithmically within:   diel06
10 ------------------------------------------------------------------------
11 E(eV),           n,        k
12 ------------------------------------------------------------------------
13 =h==          ' indicate end of header information
14   19          ' quantity of lines of data
15  1.0            1.998      1.0E-10   ' 0.0
16  1.5            2.008      1.0E-10
17  2.0            2.022      1.0E-10
18  2.5            2.041      1.0E-10
19  3.0            2.066      1.0E-10
20  3.5            2.099      1.0E-10
21  4.0            2.141      1.0E-10
22  4.25           2.167      1.0E-10   ' 0.0
23  4.5            2.198      2.2E-4
24  4.75           2.234      1.2E-3
25  5.0            2.278      4.9E-3
26  5.25           2.331      1.1E-2
27  5.5            2.393      2.9
28  5.75           2.464      5.7E-2
29  6.0            2.541      0.102
30  6.25           2.620      0.174
31  6.5            2.682      0.273
32  6.75           2.724      0.380
33  7.0            2.752      0.493
34 =h==               ' indicate end of information
```

6.6.5 W.GE

```
1  ------------------------------------------------------------------------
2  Optical Properties of:     Germanium  (Ge)
3
4  1)    Handbook of Optical Constants of Solids,
5        edited by   Edward D. Palik,
6        (Academic Press, Inc., Orlando, 1985)
7
8  2)    D.E. Aspnes  &  A.A. Studna,
9        Physical Review B,  Volume 27,  Number 2, 1983, January, 15.
10       Pages:   985-1009.
11
12 ------------------------------------------------------------------------
13 E(eV),         n,         k,         R,      alpha*1.0E3 (cm**-1)
14 ------------------------------------------------------------------------
15 =h==                    ' indicate end of header information
16    91                   ' quantity of lines of data
17  1.500        4.653       0.298      0.419      45.30
18  1.54         4.684       0.316
19  1.600        4.763       0.345      0.428      55.97
20  1.64         4.816       0.364
21  1.700        4.897       0.401      0.439      69.06
22  1.74         4.961       0.430
23  1.800        5.067       0.500      0.453      91.25
24  1.84         5.152       0.537
25  1.900        5.294       0.638      0.471     122.96
26  1.95         5.5         0.66
27  2.000        5.588       0.933      0.495     189.12
28  2.04         5.708       1.150
29  2.100        5.748       1.634      0.523     347.91
30  2.14         5.554       1.924
31  2.200        5.283       2.049      0.516     456.93
32  2.24         5.198       2.125
33  2.300        5.062       2.318      0.519     540.33
34  2.34         4.883       2.434
35  2.400        4.610       2.455      0.508     597.26
36  2.44         4.482       2.429
37  2.500        4.340       2.384      0.492     604.15
38  2.54         4.267       2.353
39  2.600        4.180       2.309      0.480     608.62
40  2.64         4.133       2.281
41  2.700        4.082       2.240      0.471     613.12
42  2.74         4.058       2.215
43  2.800        4.035       2.181      0.464     619.05
44  2.84         4.030       2.162
45  2.900        4.037       2.140      0.461     629.17
46  2.94         4.052       2.135
47  3.000        4.082       2.145      0.463     652.25
48  3.04         4.107       2.164
49  3.100        4.141       2.215      0.471     696.14
50  3.14         4.153       2.262
```

51	3.200	4.157	2.340	0.481	758.90
52	3.24	4.150	2.392		
53	3.300	4.128	2.469	0.490	825.88
54	3.34	4.106	2.517		
55	3.400	4.070	2.579	0.497	888.81
56	3.44	4.048	2.615		
57	3.500	4.020	2.667	0.502	946.01
58	3.54	4.005	2.702		
59	3.600	3.985	2.759	0.509	1006.86
60	3.64	3.974	2.799		
61	3.700	3.958	2.863	0.517	1073.71
62	3.74	3.949	2.909		
63	3.800	3.936	2.986	0.527	1150.29
64	3.84	3.930	3.043		
65	3.900	3.920	3.137	0.539	1240.23
66	3.94	3.916	3.209		
67	4.000	3.905	3.336	0.556	1352.55
68	4.04	3.896	3.436		
69	4.100	3.869	3.614	0.579	1501.84
70	4.14	3.837	3.756		
71	4.200	3.745	4.009	0.612	1706.55
72	4.24	3.633	4.207		
73	4.300	3.338	4.507	0.659	1964.58
74	4.34	3.038	4.653		
75	4.400	2.516	4.669	0.705	2082.48
76	4.44	2.226	4.543		
77	4.500	1.953	4.297	0.713	1960.14
78	4.54	1.839	4.147		
79	4.600	1.720	3.960	0.702	1846.52
80	4.64	1.659	3.852		
81	4.700	1.586	3.709	0.690	1767.14
82	4.74	1.546	3.624		
83	4.800	1.498	3.509	0.677	1707.30
84	4.84	1.470	3.439		
85	4.900	1.435	3.342	0.664	1659.73
86	4.94	1.417	3.282		
87	5.000	1.394	3.197	0.650	1620.15
88	5.04	1.382	3.146		
89	5.100	1.370	3.073	0.636	1588.72
90	5.14	1.366	3.031		
91	5.200	1.364	2.973	0.622	1567.25
92	5.24	1.366	2.938		
93	5.300	1.371	2.897	0.609	1556.15
94	5.34	1.377	2.877		
95	5.400	1.383	2.854	0.600	1562.04
96	5.44	1.387	2.849		
97	5.500	1.380	2.842	0.598	1584.57
98	5.54	1.372	2.842		
99	5.600	1.360	2.846	0.602	1615.23
100	5.64	1.345	2.852		
101	5.700	1.310	2.866	0.613	1656.06
102	5.74	1.273	2.874		
103	5.800	1.209	2.873	0.632	1689.07

```
104   5.84        1.167       2.862
105   5.900       1.108       2.831       0.644     1693.31
106   5.94        1.073       2.801
107   6.000       1.023       2.774       0.653     1686.84
108  =h==                    * indicate end of information
```

6.6.6 W.GA_AS

```
1  ------------------------------------------------------------------------
2  Optical Properties of:     Gallium Arsenide  (GaAs)
3
4  1)   Handbook of Optical Constants of Solids,
5       edited by Edward D. Palik,
6       (Academic Press, Inc.,  Orlando, 1985).
7
8  2)   D.E. Aspnes  &  A.A. Studna,
9       Physical Review B,  Volume 27,  Number 2, 1983 January 15,
10      Pages:   985-1009.
11
12 ------------------------------------------------------------------------
13 E(eV),         n,       k,       R,    alpha*1.0E3 (cm**-1)
14 ------------------------------------------------------------------------
15 =h==                  ' indicate end of header information
16  151                  ' quantity of lines of data
17  1.500       3.666    0.080    0.327      12.21
18  1.52        3.672    0.083
19  1.54        3.679    0.085
20  1.56        3.685    0.087
21  1.58        3.693    0.089
22  1.600       3.700    0.091    0.330      14.83
23  1.62        3.707    0.093
24  1.64        3.716    0.097
25  1.66        3.725    0.101
26  1.68        3.734    0.105
27  1.700       3.742    0.112    0.335      19.28
28  1.72        3.752    0.118
29  1.74        3.762    0.127
30  1.76        3.772    0.134
31  1.78        3.779    0.152
32  1.800       3.785    0.151    0.339      27.49
33  1.82        3.792    0.158
34  1.84        3.799    0.168
35  1.86        3.809    0.173
36  1.88        3.817    0.173
37  1.900       3.826    0.179    0.344      34.45
38  1.92        3.836    0.183
39  1.94        3.846    0.187
40  1.96        3.856    0.196
41  1.98        3.867    0.203
42  2.000       3.878    0.211    0.349      42.79
43  2.02        3.890    0.213
44  2.04        3.902    0.223
45  2.06        3.914    0.228
46  2.08        3.927    0.232
47  2.100       3.940    0.240    0.356      51.15
48  2.12        3.954    0.245
49  2.14        3.968    0.251
50  2.16        3.983    0.257
```

51	2.18	3.998	0.266		
52	2.200	4.013	0.276	0.363	61.46
53	2.22	4.029	0.285		
54	2.24	4.045	0.294		
55	2.26	4.063	0.301		
56	2.28	4.082	0.308		
57	2.300	4.100	0.320	0.372	74.56
58	2.32	4.120	0.327		
59	2.34	4.141	0.337		
60	2.36	4.162	0.347		
61	2.38	4.183	0.359		
62	2.400	4.205	0.371	0.382	90.34
63	2.42	4.229	0.385		
64	2.44	4.254	0.398		
65	2.46	4.279	0.411		
66	2.48	4.305	0.426		
67	2.500	4.333	0.441	0.395	111.74
68	2.52	4.362	0.458		
69	2.54	4.392	0.476		
70	2.56	4.423	0.497		
71	2.58	4.456	0.517		
72	2.600	4.492	0.539	0.410	142.02
73	2.62	4.525	0.569		
74	2.64	4.567	0.595		
75	2.66	4.605	0.626		
76	2.68	4.649	0.659		
77	2.700	4.694	0.696	0.429	190.53
78	2.72	4.741	0.739		
79	2.74	4.793	0.789		
80	2.76	4.845	0.846		
81	2.78	4.902	0.912		
82	2.800	4.959	0.991	0.456	281.33
83	2.82	5.015	1.088		
84	2.84	5.065	1.206		
85	2.86	5.102	1.353		
86	2.88	5.107	1.529		
87	2.900	5.052	1.721	0.490	505.75
88	2.92	4.917	1.885		
89	2.94	4.755	1.960		
90	2.96	4.626	1.967		
91	2.98	4.550	1.952		
92	3.000	4.509	1.948	0.472	592.48
93	3.02	4.483	1.961		
94	3.04	4.462	1.988		
95	3.06	4.439	2.029		
96	3.08	4.413	2.082		
97	3.100	4.373	2.146	0.477	674.17
98	3.12	4.313	2.212		
99	3.14	4.229	2.270		
100	3.16	4.126	2.304		
101	3.18	4.023	2.307		
102	3.200	3.938	2.288	0.468	742.21
103	3.22	3.871	2.260		

104	3.24	3.818	2.232		
105	3.26	3.776	2.207		
106	3.28	3.740	2.183		
107	3.300	3.709	2.162	0.447	723.09
108	3.32	3.681	2.142		
109	3.34	3.657	2.123		
110	3.36	3.635	2.107		
111	3.38	3.614	2.091		
112	3.400	3.596	2.076	0.434	715.28
113	3.42	3.580	2.062		
114	3.44	3.566	2.049		
115	3.46	3.553	2.036		
116	3.48	3.541	2.024		
117	3.500	3.531	2.013	0.425	714.20
118	3.54	3.513	1.992		
119	3.600	3.495	1.965	0.419	717.14
120	3.64	3.489	1.950		
121	3.700	3.485	1.931	0.415	724.14
122	3.74	3.488	1.920		
123	3.800	3.501	1.909	0.414	735.28
124	3.84	3.512	1.905		
125	3.900	3.538	1.904	0.416	752.62
126	3.94	3.559	1.907		
127	4.000	3.601	1.920	0.421	778.65
128	4.04	3.634	1.935		
129	4.100	3.692	1.969	0.430	818.23
130	4.14	3.736	2.001		
131	4.200	3.810	2.069	0.444	880.86
132	4.24	3.864	2.132		
133	4.300	3.939	2.260	0.466	984.86
134	4.34	3.981	2.368		
135	4.400	4.015	2.563	0.494	1143.26
136	4.44	4.004	2.715		
137	4.500	3.913	2.919	0.521	1331.28
138	4.54	3.850	3.018		
139	4.600	3.769	3.169	0.540	1477.66
140	4.64	3.708	3.280		
141	4.700	3.598	3.452	0.565	1644.29
142	4.74	3.511	3.574		
143	4.800	3.342	3.770	0.596	1834.18
144	4.84	3.187	3.898		
145	4.900	2.890	4.047	0.633	2009.92
146	4.94	2.654	4.106		
147	5.000	2.273	4.084	0.668	2069.81
148	5.04	2.044	3.992		
149	5.100	1.802	3.795	0.676	1961.86
150	5.14	1.699	3.660		
151	5.200	1.599	3.484	0.661	1836.14
152	5.24	1.552	3.384		
153	5.300	1.499	3.255	0.644	1748.74
154	5.34	1.468	3.181		
155	5.400	1.430	3.079	0.628	1685.29
156	5.44	1.410	3.019		

157	5.500	1.383	2.936	0.613	1636.68
158	5.54	1.367	2.885		
159	5.600	1.349	2.815	0.599	1597.99
160	5.64	1.339	2.772		
161	5.700	1.325	2.710	0.584	1565.73
162	5.74	1.321	2.673		
163	5.800	1.311	2.625	0.571	1543.07
164	5.84	1.304	2.592		
165	5.900	1.288	2.556	0.562	1528.86
166	5.94	1.287	2.523		
167	6.000	1.284	2.472	0.550	1503.20
168	=h==		' indicate end of information		

6.6.7 W.AL_GA_AS

Optical Properties of: Aluminium Gallium Arsenide,
Al(x) Ga(1-x) As,
where: 0 < x < .8

D.E.Aspnes, S.M.Kelso, R.A.Logan, R.Bhat,
Journal Applied Physics, Volume 60, Number 2, 1986-January-15,
Pages: 754-767.

E(eV),	<e1>,	<e2>,	n,	k,	R,	alpha*1.0E-3 (cm**-1)
=h==			' indicate end of header information			
9			' quantity of distinct compositions			
----------			' quantity of lines, composition of Aluminum			
46	0.0	1	(mx, y, iy)			
1.500	13.435	0.436	3.666	0.059	0.327	9.03
1.600	13.683	0.687	3.700	0.093	0.330	15.06
1.700	13.991	0.922	3.742	0.123	0.335	21.22
1.800	14.307	1.192	3.786	0.157	0.340	28.72
1.900	14.607	1.367	3.826	0.179	0.344	34.40
2.000	14.991	1.637	3.878	0.211	0.349	42.79
2.100	15.463	1.893	3.940	0.240	0.356	51.15
2.200	16.031	2.212	4.013	0.276	0.363	61.46
2.300	16.709	2.622	4.100	0.320	0.372	74.56
2.400	17.547	3.123	4.205	0.371	0.382	90.34
2.500	18.579	3.821	4.333	0.441	0.395	111.74
2.600	19.885	4.841	4.492	0.539	0.410	142.02
2.700	21.550	6.536	4.694	0.696	0.429	190.53
2.800	23.605	9.830	4.959	0.991	0.456	281.33
2.900	22.558	17.383	5.052	1.721	0.490	505.75
3.000	16.536	17.571	4.509	1.948	0.472	592.48
3.100	14.519	18.765	4.373	2.146	0.477	674.17
3.200	10.271	18.022	3.938	2.288	0.468	742.21
3.300	9.086	16.037	3.709	2.162	0.447	723.09
3.400	8.626	14.929	3.596	2.076	0.434	715.28
3.500	8.413	14.216	3.531	2.013	0.425	714.20
3.600	8.355	13.739	3.495	1.965	0.419	717.14
3.700	8.419	13.459	3.485	1.931	0.415	724.14
3.800	8.611	13.365	3.501	1.909	0.414	735.28
3.900	8.890	13.470	3.538	1.904	0.416	752.62
4.000	9.279	13.832	3.601	1.920	0.421	778.65
4.100	9.754	14.538	3.692	1.969	0.430	818.23
4.200	10.235	15.767	3.810	2.069	0.444	880.86
4.300	10.412	17.803	3.939	2.260	0.466	984.86
4.400	9.545	20.582	4.015	2.563	0.494	1143.26
4.500	6.797	22.845	3.913	2.919	0.521	1331.28
4.600	4.163	23.891	3.769	3.169	0.540	1477.66
4.700	1.030	24.835	3.598	3.452	0.565	1644.29
4.800	-3.045	25.196	3.342	3.770	0.596	1834.17
4.900	-8.022	23.394	2.890	4.047	0.633	2009.91
5.000	-11.514	18.564	2.273	4.084	0.668	2069.81

51	5.100	-11.156	13.677	1.802	3.795	0.676	1961.86
52	5.200	-9.578	11.143	1.599	3.484	0.661	1836.15
53	5.300	-8.350	9.758	1.499	3.255	0.644	1748.74
54	5.400	-7.435	8.806	1.430	3.079	0.628	1685.29
55	5.500	-6.705	8.123	1.383	2.936	0.613	1636.68
56	5.600	-6.107	7.593	1.349	2.815	0.599	1597.99
57	5.700	-5.589	7.182	1.325	2.710	0.584	1565.73
58	5.800	-5.171	6.882	1.311	2.625	0.571	1543.08
59	5.900	-4.876	6.587	1.288	2.557	0.562	1528.86
60	6.000	-4.511	6.250	1.264	2.472	0.550	1503.21
61	----------						
62	46	0.099	2	(mx, y, iy)			
63	1.500	12.758	0.000	3.572	0.000	0.316	0.00
64	1.600	13.401	0.434	3.661	0.059	0.326	9.62
65	1.700	13.521	0.600	3.678	0.082	0.328	14.06
66	1.800	13.798	0.738	3.716	0.099	0.332	18.13
67	1.900	14.237	0.962	3.775	0.127	0.338	24.54
68	2.000	14.563	1.305	3.820	0.171	0.343	34.63
69	2.100	14.981	1.540	3.876	0.199	0.349	42.29
70	2.200	15.471	1.864	3.940	0.237	0.356	52.74
71	2.300	16.067	2.216	4.018	0.276	0.364	64.29
72	2.400	16.796	2.627	4.111	0.320	0.373	77.73
73	2.500	17.662	3.225	4.220	0.382	0.384	96.83
74	2.600	18.732	4.023	4.353	0.462	0.397	121.79
75	2.700	20.080	5.196	4.518	0.575	0.413	157.38
76	2.800	21.744	7.213	4.725	0.763	0.433	216.63
77	2.900	23.411	11.184	4.968	1.126	0.461	330.89
78	3.000	20.038	17.765	4.838	1.836	0.483	558.25
79	3.100	16.095	17.384	4.460	1.949	0.469	612.35
80	3.200	13.306	18.598	4.253	2.187	0.475	709.22
81	3.300	10.072	17.026	3.864	2.203	0.458	737.04
82	3.400	9.205	15.494	3.690	2.100	0.441	723.61
83	3.500	8.846	14.619	3.601	2.030	0.430	720.13
84	3.600	8.699	14.060	3.552	1.979	0.423	722.19
85	3.700	8.690	13.736	3.532	1.945	0.419	729.33
86	3.800	8.809	13.609	3.537	1.924	0.417	740.99
87	3.900	9.028	13.692	3.566	1.920	0.419	758.97
88	4.000	9.342	14.017	3.618	1.937	0.423	785.31
89	4.100	9.734	14.664	3.697	1.983	0.431	824.19
90	4.200	10.146	15.766	3.801	2.074	0.444	882.91
91	4.300	10.364	17.571	3.922	2.240	0.464	976.34
92	4.400	9.851	20.145	4.017	2.507	0.489	1118.25
93	4.500	7.811	23.065	4.010	2.876	0.519	1311.72
94	4.600	4.236	24.731	3.829	3.229	0.546	1505.64
95	4.700	0.537	25.535	3.611	3.536	0.572	1684.43
96	4.800	-4.119	25.136	3.267	3.846	0.604	1871.41
97	4.900	-8.604	22.369	2.772	4.036	0.637	2004.32
98	5.000	-10.991	17.583	2.207	3.983	0.662	2018.51
99	5.100	-10.408	13.478	1.819	3.704	0.664	1914.68
100	5.200	-9.116	11.220	1.634	3.433	0.651	1809.48
101	5.300	-8.043	9.870	1.531	3.223	0.635	1731.43
102	5.400	-7.227	8.928	1.459	3.059	0.621	1674.24
103	5.500	-6.550	8.225	1.408	2.921	0.607	1628.38

104	5.600	-5.963	7.677	1.371	2.800	0.593	1589.56
105	5.700	-5.468	7.256	1.345	2.698	0.579	1558.54
106	5.800	-5.032	6.937	1.330	2.608	0.565	1533.15
107	5.900	-4.702	6.692	1.318	2.538	0.554	1517.71
108	6.000	-4.321	6.442	1.311	2.457	0.539	1494.49
109	----------						
110	47	0.198	3	(mx, y, iy)			
111	1.500	11.950	0.000	3.457	0.000	0.304	0.00
112	1.600	12.502	0.017	3.536	0.002	0.313	0.39
113	1.700	13.213	0.013	3.635	0.002	0.323	0.30
114	1.719	13.423	0.02	3.664	0.003		
115	1.800	13.423	0.601	3.662	0.082	0.326	14.97
116	1.900	13.682	0.696	3.700	0.094	0.330	18.11
117	2.000	14.119	0.887	3.759	0.118	0.337	23.92
118	2.100	14.529	1.260	3.815	0.165	0.343	35.16
119	2.200	14.946	1.567	3.871	0.202	0.349	45.13
120	2.300	15.464	1.909	3.940	0.242	0.356	56.49
121	2.400	16.096	2.313	4.022	0.288	0.364	69.94
122	2.500	16.845	2.812	4.118	0.341	0.374	86.51
123	2.600	17.767	3.460	4.235	0.409	0.386	107.66
124	2.700	18.893	4.363	4.375	0.499	0.399	136.45
125	2.800	20.272	5.787	4.547	0.636	0.417	180.59
126	2.900	21.877	8.225	4.757	0.865	0.439	254.16
127	3.000	22.682	13.063	4.943	1.322	0.467	401.85
128	3.100	17.781	17.110	4.607	1.857	0.472	583.43
129	3.200	15.264	17.325	4.379	1.978	0.467	641.61
130	3.300	11.927	17.808	4.084	2.180	0.466	729.24
131	3.400	9.908	16.040	3.792	2.115	0.447	728.85
132	3.500	9.315	14.923	3.668	2.034	0.434	721.70
133	3.600	9.061	14.254	3.602	1.979	0.426	721.98
134	3.700	8.987	13.874	3.572	1.942	0.421	728.35
135	3.800	9.035	13.714	3.568	1.922	0.419	740.28
136	3.900	9.199	13.758	3.588	1.917	0.420	757.87
137	4.000	9.458	14.047	3.633	1.933	0.423	783.89
138	4.100	9.795	14.630	3.701	1.976	0.431	821.32
139	4.200	10.163	15.627	3.795	2.059	0.443	876.48
140	4.300	10.429	17.254	3.911	2.206	0.460	961.42
141	4.400	10.147	19.686	4.018	2.449	0.485	1092.43
142	4.500	8.568	22.715	4.053	2.803	0.514	1278.33
143	4.600	5.008	25.295	3.924	3.223	0.547	1502.86
144	4.700	0.459	26.010	3.638	3.575	0.576	1702.90
145	4.800	-4.468	25.047	3.238	3.867	0.607	1881.50
146	4.900	-8.500	21.861	2.734	3.997	0.634	1985.28
147	5.000	-10.433	17.302	2.210	3.914	0.655	1983.56
148	5.100	-9.895	13.595	1.860	3.654	0.654	1889.12
149	5.200	-8.795	11.429	1.677	3.407	0.643	1795.77
150	5.300	-7.837	10.058	1.567	3.208	0.629	1723.61
151	5.400	-7.075	9.089	1.490	3.049	0.615	1668.90
152	5.500	-6.426	8.346	1.433	2.912	0.602	1623.42
153	5.600	-5.865	7.785	1.393	2.794	0.588	1585.91
154	5.700	-5.360	7.346	1.366	2.688	0.574	1553.11
155	5.800	-4.943	7.013	1.349	2.600	0.561	1528.73
156	5.900	-4.611	6.776	1.339	2.531	0.549	1513.36

157	6.000	-4.261	6.551	1.333	2.457	0.536	1494.40
158	---						
159	47	0.315	4	(mx, y, iy)			
160	1.500	11.585	0.000	3.404	0.000	0.298	0.00
161	1.600	11.945	0.000	3.456	0.000	0.304	0.00
162	1.700	12.312	0.000	3.509	0.000	0.310	0.00
163	1.800	12.901	0.058	3.592	0.008	0.319	1.46
164	1.864	13.423	0.1	3.664	0.014		
165	1.900	13.309	0.812	3.650	0.111	0.325	21.43
166	2.000	13.597	1.070	3.690	0.145	0.330	29.38
167	2.100	14.036	1.250	3.750	0.167	0.336	35.47
168	2.200	14.516	1.543	3.815	0.202	0.343	45.08
169	2.300	14.937	1.755	3.872	0.227	0.349	52.83
170	2.400	15.495	2.038	3.945	0.258	0.356	62.83
171	2.500	16.167	2.460	4.032	0.305	0.365	77.30
172	2.600	16.963	3.037	4.135	0.367	0.367	96.76
173	2.700	17.928	3.794	4.258	0.446	0.388	121.93
174	2.800	19.089	4.900	4.404	0.556	0.403	157.89
175	2.900	20.478	6.614	4.582	0.722	0.421	212.14
176	3.000	21.832	9.680	4.781	1.012	0.445	307.84
177	3.100	20.854	15.031	4.825	1.558	0.469	489.44
178	3.200	16.329	16.746	4.456	1.879	0.465	609.42
179	3.300	13.867	17.334	4.246	2.041	0.464	682.67
180	3.400	10.827	16.737	3.922	2.134	0.455	735.39
181	3.500	9.653	15.298	3.724	2.054	0.439	728.58
182	3.600	9.242	14.445	3.633	1.988	0.428	725.50
183	3.700	9.094	13.968	3.589	1.946	0.422	729.80
184	3.800	9.095	13.734	3.575	1.921	0.419	739.75
185	3.900	9.212	13.739	3.588	1.914	0.419	756.76
186	4.000	9.427	13.973	3.625	1.927	0.422	781.41
187	4.100	9.723	14.492	3.686	1.966	0.429	816.92
188	4.200	10.065	15.387	3.772	2.040	0.440	868.37
189	4.300	10.362	16.868	3.883	2.172	0.456	946.63
190	4.400	10.266	19.139	3.999	2.393	0.479	1067.23
191	4.500	9.029	22.204	4.062	2.733	0.509	1246.64
192	4.600	5.766	25.299	3.982	3.177	0.544	1481.12
193	4.700	0.383	26.540	3.669	3.617	0.579	1722.91
194	4.800	-4.850	24.805	3.196	3.881	0.609	1888.25
195	4.900	-8.451	21.238	2.684	3.957	0.633	1965.08
196	5.000	-9.954	16.908	2.198	3.845	0.648	1948.84
197	5.100	-9.460	13.534	1.878	3.604	0.647	1862.86
198	5.200	-8.520	11.454	1.696	3.376	0.637	1779.44
199	5.300	-7.656	10.078	1.581	3.187	0.624	1712.04
200	5.400	-6.940	9.074	1.497	3.030	0.612	1658.54
201	5.500	-6.308	8.313	1.437	2.893	0.598	1613.01
202	5.600	-5.765	7.734	1.393	2.776	0.585	1575.68
203	5.700	-5.255	7.296	1.367	2.669	0.570	1541.99
204	5.800	-4.814	6.967	1.352	2.577	0.556	1514.99
205	5.900	-4.471	6.698	1.338	2.502	0.544	1496.52
206	6.000	-4.156	6.583	1.347	2.443	0.531	1486.00
207	---						
208	46	0.419	5	(mx, y, iy)			
209	1.500	11.160	0.000	3.341	0.000	0.291	0.00

210	1.600	11.412	0.000	3.378	0.000	0.295	0.00
211	1.700	11.709	0.000	3.422	0.000	0.300	0.00
212	1.800	12.106	0.000	3.479	0.000	0.306	0.00
213	1.900	12.669	0.018	3.559	0.003	0.315	0.49
214	2.000	13.423	0.431	3.664	0.059	0.326	11.91
215	2.100	13.578	0.735	3.686	0.100	0.329	21.21
216	2.200	14.022	1.003	3.747	0.134	0.335	29.86
217	2.300	14.562	1.357	3.820	0.178	0.343	41.40
218	2.400	15.015	1.700	3.881	0.219	0.350	53.29
219	2.500	15.582	2.120	3.957	0.268	0.353	67.88
220	2.600	16.279	2.581	4.047	0.319	0.367	84.03
221	2.700	17.103	3.202	4.154	0.385	0.378	105.49
222	2.800	18.100	4.037	4.280	0.472	0.391	133.82
223	2.900	19.273	5.279	4.430	0.596	0.406	175.12
224	3.000	20.591	7.236	4.605	0.786	0.425	238.89
225	3.100	21.579	10.694	4.778	1.119	0.448	351.61
226	3.200	19.457	15.439	4.706	1.640	0.466	532.03
227	3.300	15.873	16.459	4.401	1.870	0.461	625.45
228	3.400	13.237	17.043	4.172	2.042	0.460	703.85
229	3.500	10.814	16.101	3.887	2.071	0.448	734.87
230	3.600	9.940	14.982	3.736	2.005	0.435	731.61
231	3.700	9.622	14.365	3.668	1.958	0.427	734.32
232	3.800	9.516	14.059	3.640	1.931	0.424	743.92
233	3.900	9.549	14.005	3.640	1.924	0.423	760.47
234	4.000	9.696	14.201	3.667	1.936	0.425	785.08
235	4.100	9.942	14.663	3.719	1.971	0.431	819.31
236	4.200	10.229	15.482	3.794	2.040	0.441	868.66
237	4.300	10.519	16.846	3.897	2.161	0.456	941.92
238	4.400	10.530	18.973	4.014	2.363	0.477	1053.94
239	4.500	9.587	22.083	4.103	2.691	0.507	1227.63
240	4.600	6.468	25.594	4.054	3.157	0.543	1471.88
241	4.700	0.644	27.217	3.733	3.646	0.582	1736.73
242	4.800	-5.043	25.014	3.200	3.909	0.611	1901.83
243	4.900	-8.234	21.217	2.695	3.937	0.630	1955.15
244	5.000	-9.614	17.074	2.234	3.822	0.643	1936.79
245	5.100	-9.238	13.861	1.926	3.598	0.642	1860.13
246	5.200	-8.428	11.775	1.740	3.384	0.633	1783.80
247	5.300	-7.643	10.329	1.613	3.201	0.622	1719.58
248	5.400	-6.964	9.277	1.523	3.047	0.611	1667.58
249	5.500	-6.334	8.469	1.456	2.908	0.598	1621.01
250	5.600	-5.765	7.828	1.406	2.783	0.584	1579.54
251	5.700	-5.259	7.365	1.377	2.675	0.570	1545.40
252	5.800	-4.787	6.985	1.357	2.574	0.555	1513.42
253	5.900	-4.465	6.767	1.350	2.507	0.543	1499.39
254	6.000	-4.124	6.602	1.353	2.440	0.529	1483.96
255	---	---	---	---	---	---	---
256	46	0.491	6	(mx, y, iy)			
257	1.500	10.780	0.000	3.283	0.000	0.284	0.00
258	1.600	11.080	0.000	3.329	0.000	0.289	0.00
259	1.700	11.344	0.000	3.368	0.000	0.294	0.00
260	1.800	11.676	0.000	3.417	0.000	0.299	0.01
261	1.900	12.089	0.000	3.477	0.000	0.306	0.02
262	2.000	12.659	0.016	3.558	0.002	0.315	0.46

263	2.100	13.423	0.647	3.665	0.088	0.327	18.80
264	2.200	13.639	0.985	3.696	0.133	0.330	29.72
265	2.300	14.115	1.234	3.761	0.164	0.337	38.25
266	2.400	14.685	1.571	3.838	0.205	0.345	49.79
267	2.500	15.176	1.914	3.903	0.245	0.352	62.12
268	2.600	15.797	2.331	3.985	0.292	0.361	77.07
269	2.700	16.525	2.900	4.081	0.355	0.371	97.24
270	2.800	17.411	3.596	4.195	0.429	0.382	121.66
271	2.900	18.445	4.623	4.328	0.534	0.396	156.98
272	3.000	19.629	6.132	4.483	0.684	0.413	207.96
273	3.100	20.801	8.624	4.654	0.926	0.433	291.11
274	3.200	20.801	12.735	4.753	1.340	0.455	434.48
275	3.300	17.419	15.823	4.525	1.748	0.461	584.80
276	3.400	14.746	16.502	4.294	1.922	0.458	662.21
277	3.500	12.074	16.529	4.034	2.049	0.454	726.82
278	3.600	10.534	15.419	3.822	2.017	0.440	736.17
279	3.700	9.992	14.670	3.724	1.969	0.431	738.60
280	3.800	9.771	14.294	3.680	1.942	0.427	748.01
281	3.900	9.737	14.190	3.671	1.933	0.425	764.12
282	4.000	9.821	14.346	3.688	1.945	0.427	788.49
283	4.100	9.995	14.768	3.730	1.980	0.433	822.67
284	4.200	10.237	15.540	3.798	2.046	0.442	871.00
285	4.300	10.502	16.815	3.894	2.159	0.456	941.02
286	4.400	10.547	18.847	4.009	2.351	0.476	1048.33
287	4.500	9.752	21.915	4.107	2.668	0.505	1216.88
288	4.600	6.678	25.824	4.072	3.147	0.543	1467.13
289	4.700	0.636	27.202	3.731	3.645	0.582	1736.48
290	4.800	-5.044	24.850	3.187	3.899	0.611	1896.85
291	4.900	-8.024	20.971	2.686	3.904	0.627	1938.84
292	5.000	-9.278	17.038	2.250	3.787	0.639	1919.09
293	5.100	-9.003	13.967	1.951	3.579	0.637	1850.21
294	5.200	-8.301	11.910	1.763	3.378	0.630	1780.31
295	5.300	-7.573	10.444	1.632	3.199	0.620	1718.78
296	5.400	-6.923	9.330	1.532	3.045	0.609	1666.52
297	5.500	-6.293	8.487	1.462	2.903	0.596	1618.56
298	5.600	-5.733	7.849	1.412	2.780	0.583	1577.75
299	5.700	-5.205	7.351	1.379	2.666	0.568	1540.12
300	5.800	-4.755	7.018	1.364	2.572	0.553	1512.15
301	5.900	-4.344	6.790	1.363	2.490	0.537	1489.38
302	6.000	-3.983	6.608	1.366	2.418	0.523	1470.81
303	----------						
304	46	0.590	7	(mx, y, iy)			
305	1.500	10.480	0.000	3.237	0.000	0.279	0.00
306	1.600	10.721	0.000	3.274	0.000	0.283	0.00
307	1.700	10.973	0.000	3.313	0.000	0.288	0.00
308	1.800	11.247	0.000	3.354	0.000	0.292	0.00
309	1.900	11.591	0.000	3.405	0.000	0.298	0.00
310	2.000	12.017	0.000	3.467	0.000	0.305	0.01
311	2.100	12.575	0.034	3.546	0.005	0.314	1.03
312	2.200	13.380	0.458	3.658	0.063	0.326	13.95
313	2.300	13.599	0.931	3.690	0.126	0.329	29.43
314	2.400	14.097	1.181	3.758	0.157	0.337	38.23
315	2.500	14.682	1.574	3.837	0.205	0.345	51.97

316	2.600	15.208	2.047	3.909	0.262	0.353	69.00
317	2.700	15.837	2.532	3.992	0.317	0.362	86.80
318	2.800	16.600	3.140	4.092	0.384	0.372	108.87
319	2.900	17.493	3.936	4.208	0.468	0.384	137.44
320	3.000	18.517	5.070	4.343	0.584	0.399	177.51
321	3.100	19.653	6.784	4.497	0.754	0.416	237.00
322	3.200	20.560	9.557	4.649	1.028	0.436	333.35
323	3.300	19.663	13.530	4.665	1.450	0.454	485.03
324	3.400	16.542	15.536	4.429	1.754	0.456	604.41
325	3.500	14.137	16.255	4.224	1.924	0.455	682.65
326	3.600	11.804	15.923	3.977	2.002	0.447	730.58
327	3.700	10.713	15.077	3.822	1.973	0.437	739.80
328	3.800	10.287	14.581	3.750	1.944	0.431	748.73
329	3.900	10.131	14.417	3.725	1.935	0.428	765.00
330	4.000	10.113	14.518	3.729	1.947	0.430	789.30
331	4.100	10.210	14.870	3.758	1.978	0.434	822.15
332	4.200	10.403	15.554	3.815	2.038	0.442	867.72
333	4.300	10.647	16.723	3.903	2.142	0.455	933.67
334	4.400	10.746	18.612	4.015	2.318	0.474	1033.77
335	4.500	10.194	21.592	4.127	2.616	0.502	1193.05
336	4.600	7.325	25.601	4.120	3.107	0.541	1448.53
337	4.700	1.071	27.219	3.762	3.617	0.579	1723.21
338	4.800	-4.574	24.901	3.221	3.866	0.607	1880.90
339	4.900	-7.553	21.268	2.740	3.881	0.623	1927.50
340	5.000	-8.921	17.385	2.304	3.772	0.634	1911.84
341	5.100	-8.797	14.375	2.007	3.581	0.633	1851.25
342	5.200	-8.246	12.245	1.805	3.392	0.627	1787.75
343	5.300	-7.589	10.682	1.661	3.217	0.619	1727.95
344	5.400	-6.945	9.504	1.553	3.059	0.609	1674.41
345	5.500	-6.321	8.600	1.475	2.915	0.596	1625.05
346	5.600	-5.732	7.920	1.422	2.785	0.582	1580.61
347	5.700	-5.195	7.413	1.389	2.669	0.567	1542.06
348	5.800	-4.701	7.030	1.370	2.565	0.551	1507.87
349	5.900	-4.301	6.810	1.370	2.485	0.535	1486.39
350	6.000	-3.936	6.704	1.385	2.420	0.520	1471.56
351	---						
352	46	0.700	8	(mx, y, iy)			
353	1.500	9.940	0.000	3.153	0.000	0.269	0.00
354	1.600	10.161	0.000	3.188	0.000	0.273	0.00
355	1.700	10.398	0.000	3.225	0.000	0.277	0.00
356	1.800	10.636	0.000	3.261	0.000	0.282	0.00
357	1.900	10.928	0.000	3.306	0.000	0.287	0.00
358	2.000	11.294	0.000	3.361	0.000	0.293	0.00
359	2.100	11.728	0.001	3.425	0.000	0.300	0.03
360	2.200	12.252	0.003	3.500	0.000	0.309	0.09
361	2.300	12.924	0.016	3.595	0.002	0.319	0.51
362	2.400	13.653	0.513	3.696	0.069	0.330	16.89
363	2.500	14.012	0.969	3.746	0.129	0.335	32.78
364	2.600	14.582	1.408	3.823	0.184	0.344	48.54
365	2.700	15.200	1.918	3.906	0.245	0.353	67.19
366	2.800	15.806	2.446	3.987	0.307	0.361	87.05
367	2.900	16.538	3.058	4.084	0.374	0.371	110.04
368	3.000	17.393	3.857	4.198	0.460	0.383	139.75

369	3.100	18.378	4.965	4.325	0.574	0.397	180.35
370	3.200	19.450	6.576	4.471	0.735	0.413	238.51
371	3.300	20.334	9.049	4.615	0.980	0.432	327.96
372	3.400	19.925	12.658	4.665	1.357	0.450	467.51
373	3.500	17.454	15.104	4.502	1.678	0.456	595.12
374	3.600	15.128	16.217	4.319	1.877	0.457	685.06
375	3.700	12.864	16.355	4.103	1.993	0.453	747.43
376	3.800	11.550	15.837	3.947	2.006	0.446	772.81
377	3.900	10.928	15.538	3.868	2.009	0.442	793.97
378	4.000	10.620	15.516	3.836	2.023	0.442	820.09
379	4.100	10.487	15.765	3.835	2.055	0.444	854.10
380	4.200	10.502	16.330	3.868	2.111	0.450	898.74
381	4.300	10.595	17.349	3.932	2.206	0.461	961.51
382	4.400	10.632	19.053	4.028	2.365	0.478	1054.80
383	4.500	10.160	21.908	4.142	2.645	0.504	1206.30
384	4.600	7.253	26.106	4.144	3.150	0.544	1468.60
385	4.700	0.897	27.338	3.758	3.637	0.581	1732.65
386	4.800	-4.518	24.762	3.214	3.853	0.606	1874.51
387	4.900	-7.292	21.512	2.777	3.873	0.620	1923.76
388	5.000	-8.809	17.839	2.354	3.788	0.632	1919.97
389	5.100	-8.906	14.834	2.049	3.620	0.634	1871.32
390	5.200	-8.526	12.599	1.829	3.445	0.632	1815.84
391	5.300	-7.924	10.863	1.662	3.269	0.626	1756.01
392	5.400	-7.215	9.572	1.545	3.098	0.616	1695.93
393	5.500	-6.541	8.613	1.462	2.946	0.603	1642.26
394	5.600	-5.908	7.903	1.407	2.809	0.589	1594.16
395	5.700	-5.337	7.391	1.375	2.688	0.573	1553.18
396	5.800	-4.796	7.046	1.365	2.581	0.554	1517.15
397	5.900	-4.348	6.811	1.366	2.493	0.537	1490.74
398	6.000	-3.991	6.683	1.377	2.426	0.523	1475.64
399	----------						
400	46	0.804	9	(mx, y, iy)			
401	1.500	9.761	0.000	3.124	0.000	0.265	0.00
402	1.600	9.902	0.000	3.147	0.000	0.268	0.00
403	1.700	10.068	0.000	3.173	0.000	0.271	0.00
404	1.800	10.251	0.000	3.202	0.000	0.275	0.00
405	1.900	10.472	0.000	3.236	0.000	0.279	0.00
406	2.000	10.739	0.000	3.277	0.000	0.283	0.00
407	2.100	11.038	0.000	3.322	0.000	0.289	0.00
408	2.200	11.408	0.000	3.378	0.000	0.295	0.00
409	2.300	11.834	0.019	3.440	0.003	0.302	0.64
410	2.400	12.382	0.025	3.519	0.004	0.311	0.87
411	2.500	13.217	0.095	3.635	0.013	0.323	3.30
412	2.600	13.964	0.778	3.738	0.104	0.334	27.42
413	2.700	14.315	1.217	3.787	0.161	0.340	43.96
414	2.800	14.948	1.586	3.872	0.205	0.349	58.12
415	2.900	15.613	2.190	3.961	0.276	0.358	81.26
416	3.000	16.281	2.859	4.050	0.353	0.368	107.31
417	3.100	17.072	3.632	4.155	0.437	0.379	137.32
418	3.200	18.001	4.625	4.277	0.541	0.392	175.38
419	3.300	19.007	6.042	4.413	0.685	0.407	228.97
420	3.400	20.017	8.123	4.562	0.890	0.425	306.82
421	3.500	20.339	11.193	4.667	1.199	0.444	425.45

422	3.600	18.845	14.405	4.613	1.561	0.456	569.68
423	3.700	16.603	16.240	4.462	1.820	0.462	682.44
424	3.800	14.152	17.181	4.267	2.013	0.463	775.50
425	3.900	12.255	17.059	4.078	2.092	0.459	826.82
426	4.000	11.207	16.793	3.962	2.119	0.456	859.20
427	4.100	10.643	16.737	3.904	2.144	0.455	890.89
428	4.200	10.392	17.002	3.893	2.183	0.458	929.51
429	4.300	10.334	17.723	3.928	2.256	0.465	983.42
430	4.400	10.323	19.127	4.004	2.389	0.479	1065.33
431	4.500	9.941	21.704	4.112	2.639	0.503	1203.83
432	4.600	7.082	25.692	4.107	3.128	0.542	1458.44
433	4.700	1.235	26.869	3.751	3.582	0.576	1706.48
434	4.800	-3.720	24.345	3.233	3.765	0.597	1831.69
435	4.900	-6.524	21.615	2.833	3.815	0.612	1894.58
436	5.000	-8.274	18.256	2.426	3.763	0.625	1907.01
437	5.100	-8.758	15.338	2.110	3.635	0.631	1878.88
438	5.200	-8.628	12.909	1.857	3.475	0.633	1831.74
439	5.300	-8.089	10.939	1.661	3.293	0.629	1769.26
440	5.400	-7.300	9.489	1.528	3.104	0.619	1699.08
441	5.500	-6.517	8.495	1.447	2.935	0.604	1636.04
442	5.600	-5.837	7.808	1.399	2.792	0.587	1584.53
443	5.700	-5.237	7.307	1.370	2.667	0.570	1540.88
444	5.800	-4.722	6.931	1.354	2.560	0.552	1505.06
445	5.900	-4.267	6.729	1.360	2.473	0.534	1479.16
446	6.000	-3.931	6.589	1.368	2.409	0.520	1464.84

447 ----------

448 =h==

6.6.8 W.GA_P

```
1  ------------------------------------------------------------------------
2  Optical Properties of:   Gallium Phosphide (GaP)
3
4  1)    Handbook of Optical Constants,
5        edited by  Edward D. Palik,
6        (Academic Press, Inc.,  Orlando, 1985)
7
8  2)    D.E. Aspnes  &  A.A. Studna,
9        Physical Review B, Volume 27,  Number 2,  1983 January 15.
10       Pages:   985-1009.
11
12 ------------------------------------------------------------------------
13 E(eV),        n,         k,         R,       alpha*1.0E3 (cm**-1)
14 ------------------------------------------------------------------------
15 =h==                    ' indicate end of header information
16  156                    ' quantity of lines of data
17  1.500        3.178       0.0        0.272       0.0
18  1.54         3.191       0.0
19  1.600        3.209       0.0        0.275       0.0
20  1.64         3.219       0.0
21  1.700        3.234       0.0        0.278       0.0
22  1.74         3.245       0.0
23  1.800        3.262       0.0        0.282       0.0
24  1.84         3.275       0.0
25  1.900        3.295       0.0        0.286       0.0
26  1.94         3.311       0.0
27  2.000        3.334       0.0        0.290       0.0
28  2.04         3.350       0.0
29  2.100        3.375       0.0        0.295       0.01
30  2.14         3.393       0.0
31  2.18         3.411       2.8E-7
32  2.200        3.421       1.4E-6     0.3         0.04
33  2.22         3.430       3.1E-6
34  2.24         3.441       2.7E-5
35  2.26         3.452       6.2E-5
36  2.28         3.463       1.3E-4
37  2.300        3.474       2.5E-4     0.306       0.36
38  2.34         3.497       5.5E-4
39  2.400        3.535       1.1E-3     0.312       0.86
40  2.44         3.561       1.6E-3
41  2.48         3.590       2.47E-3
42  2.500        3.605       2.54E-3    0.320       1.63
43  2.54         3.638       3.1E-3
44  2.600        3.691       5.5E-3     0.329       2.94
45  2.64         3.730       1.5E-2
46  2.700        3.805       2.7E-2     0.341       7.52
47  2.72         3.835       3.5E-2
48  2.74         3.869       5.7E-2
49  2.76         3.896       8.5E-2
50  2.78         3.904       0.103
```

51	2.800	3.919	0.118	0.352	33.42
52	2.82	3.936	0.135		
53	2.84	3.952	0.152		
54	2.86	3.964	0.165		
55	2.88	3.976	0.176		
56	2.900	3.990	0.183	0.360	53.90
57	2.92	4.006	0.194		
58	2.94	4.023	0.202		
59	2.96	4.041	0.208		
60	2.98	4.060	0.218		
61	3.000	4.081	0.224	0.369	68.26
62	3.02	4.102	0.233		
63	3.04	4.124	0.244		
64	3.06	4.147	0.253		
65	3.08	4.171	0.264		
66	3.100	4.196	0.275	0.380	86.28
67	3.12	4.222	0.285		
68	3.14	4.249	0.298		
69	3.16	4.278	0.310		
70	3.18	4.308	0.323		
71	3.200	4.339	0.337	0.394	109.25
72	3.22	4.372	0.353		
73	3.24	4.406	0.369		
74	3.26	4.442	0.388		
75	3.28	4.479	0.407		
76	3.300	4.518	0.426	0.410	142.64
77	3.32	4.560	0.449		
78	3.34	4.604	0.475		
79	3.36	4.651	0.503		
80	3.38	4.700	0.534		
81	3.400	4.751	0.568	0.431	195.80
82	3.42	4.805	0.605		
83	3.44	4.861	0.649		
84	3.46	4.920	0.697		
85	3.48	4.983	0.752		
86	3.500	5.050	0.819	0.458	290.40
87	3.52	5.121	0.893		
88	3.54	5.194	0.982		
89	3.56	5.268	1.089		
90	3.58	5.339	1.217		
91	3.600	5.406	1.368	0.496	499.04
92	3.62	5.454	1.550		
93	3.64	5.472	1.766		
94	3.66	5.437	2.010		
95	3.68	5.328	2.260		
96	3.700	5.149	2.451	0.530	919.21
97	3.72	4.927	2.585		
98	3.74	4.700	2.646		
99	3.76	4.497	2.649		
100	3.78	4.328	2.615		
101	3.800	4.196	2.562	0.500	986.69
102	3.82	4.095	2.502		
103	3.84	4.021	2.443		

104	3.86	3.966	2.389		
105	3.88	3.923	2.343		
106	3.900	3.890	2.303	0.467	910.55
107	3.94	3.840	2.240		
108	4.000	3.790	2.171	0.452	880.10
109	4.04	3.769	2.137		
110	4.100	3.752	2.100	0.444	872.59
111	4.14	3.748	2.082		
112	4.200	3.754	2.063	0.441	878.38
113	4.24	3.765	2.057		
114	4.300	3.792	2.058	0.442	896.99
115	4.34	3.817	2.066		
116	4.400	3.867	2.090	0.449	932.14
117	4.44	3.907	2.117		
118	4.500	3.978	2.180	0.461	994.27
119	4.54	4.031	2.241		
120	4.600	4.113	2.371	0.482	1105.61
121	4.62	4.137	2.427		
122	4.64	4.158	2.491		
123	4.66	4.173	2.560		
124	4.68	4.181	2.634		
125	4.700	4.181	2.712	0.511	1291.94
126	4.72	4.173	2.791		
127	4.74	4.157	2.871		
128	4.76	4.132	2.949		
129	4.78	4.100	3.024		
130	4.800	4.062	3.096	0.539	1506.17
131	4.82	4.018	3.162		
132	4.84	3.970	3.218		
133	4.86	3.923	3.269		
134	4.88	3.880	3.316		
135	4.900	3.844	3.358	0.557	1667.57
136	4.92	3.806	3.404		
137	4.94	3.774	3.454		
138	4.96	3.739	3.509		
139	4.98	3.701	3.567		
140	5.000	3.661	3.631	0.580	1839.99
141	5.02	3.615	3.698		
142	5.04	3.561	3.768		
143	5.06	3.498	3.839		
144	5.08	3.424	3.909		
145	5.100	3.342	3.975	0.614	2054.74
146	5.12	3.248	4.036		
147	5.14	3.144	4.086		
148	5.16	3.036	4.123		
149	5.18	2.930	4.150		
150	5.200	2.825	4.170	0.647	2197.82
151	5.22	2.718	4.185		
152	5.24	2.607	4.197		
153	5.26	2.490	4.201		
154	5.28	2.368	4.192		
155	5.300	2.248	4.168	0.678	2239.10
156	5.32	2.134	4.128		

157	5.34	2.028	4.077		
158	5.36	1.936	4.019		
159	5.38	1.851	3.955		
160	5.400	1.778	3.889	0.689	2128.87
161	5.44	1.660	3.752		
162	5.500	1.543	3.556	0.677	1982.53
163	5.54	1.494	3.441		
164	5.600	1.444	3.297	0.657	1871.33
165	5.64	1.418	3.211		
166	5.700	1.385	3.096	0.637	1788.84
167	5.74	1.368	3.028		
168	5.800	1.348	2.934	0.618	1724.82
169	5.84	1.342	2.882		
170	5.900	1.327	2.803	0.600	1676.14
171	5.94	1.327	2.766		
172	6.000	1.309	2.690	0.583	1635.71
173	=h==		* indicate end of information		

6.6.9 W.GA_SB

```
1  ------------------------------------------------------------------------
2  Optical Properties of:   Gallium Antimonide (GaSb)
3
4  D.E. Aspnes  &  A.A. Studna,
5  Physical Review B,  Volume 27,  Number 2, 1983, January, 15.
6  Pages:   985-1009.
7
8  ------------------------------------------------------------------------
9  E(eV),       <e1>,     <e2>,     n,       k,       R,    alpha*1.0E3 (cm**-1)
10 ------------------------------------------------------------------------
11 =h==                  ' indicate end of header information
12  46                   ' quantity of lines of data
13  1.500      19.135      3.023     4.388    0.344    0.398      52.37
14  1.600      20.137      3.752     4.507    0.416    0.409      67.51
15  1.700      21.322      4.503     4.643    0.485    0.421      83.56
16  1.800      22.826      5.889     4.817    0.611    0.437     111.53
17  1.900      24.836      8.373     5.052    0.829    0.458     159.59
18  2.000      25.545     14.442     5.239    1.378    0.487     279.43
19  2.100      18.883     16.963     4.705    1.803    0.474     383.74
20  2.200      17.386     15.794     4.521    1.747    0.461     389.54
21  2.300      16.980     16.069     4.492    1.789    0.461     416.97
22  2.400      16.521     17.708     4.513    1.962    0.473     477.22
23  2.500      13.367     19.705     4.312    2.285    0.484     579.07
24  2.600      10.676     18.172     3.984    2.280    0.470     600.94
25  2.700       9.328     16.966     3.836    2.211    0.457     605.14
26  2.800       9.484     16.216     3.760    2.157    0.449     612.05
27  2.900       9.399     15.810     3.728    2.121    0.445     623.34
28  3.000       9.479     15.738     3.732    2.109    0.444     641.20
29  3.100       9.628     16.070     3.766    2.134    0.448     670.45
30  3.200       9.558     16.797     3.800    2.210    0.456     716.82
31  3.300       9.121     17.658     3.808    2.319    0.465     775.62
32  3.400       8.490     18.440     3.794    2.430    0.475     837.48
33  3.500       7.852     19.267     3.785    2.545    0.485     902.86
34  3.600       7.011     20.306     3.774    2.690    0.497     981.54
35  3.700       5.853     21.453     3.748    2.862    0.512    1073.44
36  3.800       4.281     22.719     3.701    3.069    0.530    1182.10
37  3.900       2.058     24.057     3.620    3.323    0.553    1313.68
38  4.000      -1.374     25.138     3.450    3.643    0.583    1477.21
39  4.100      -6.203     24.648     3.100    3.976    0.620    1652.40
40  4.200     -10.699     20.831     2.522    4.130    0.658    1758.30
41  4.300     -11.435     15.607     1.989    3.923    0.673    1709.93
42  4.400     -10.196     12.500     1.723    3.628    0.665    1618.09
43  4.500      -8.989     10.763     1.586    3.392    0.651    1547.17
44  4.600      -8.031      9.642     1.503    3.208    0.637    1495.68
45  4.700      -7.249      8.823     1.444    3.055    0.623    1455.44
46  4.800      -6.594      8.244     1.408    2.928    0.608    1424.76
47  4.900      -6.079      7.846     1.387    2.829    0.595    1405.00
48  5.000      -5.693      7.529     1.369    2.751    0.585    1394.02
49  5.100      -5.365      7.290     1.358    2.685    0.575    1387.91
50  5.200      -5.156      7.173     1.356    2.645    0.568    1394.03
```

```
 5.300      -5.151    7.099    1.345    2.638    0.568    1417.39
 5.400      -5.353    6.890    1.299    2.653    0.578    1452.17
 5.500      -5.527    6.410    1.212    2.645    0.592    1474.51
 5.600      -5.497    5.866    1.127    2.602    0.601    1476.67
 5.700      -5.297    5.385    1.062    2.535    0.602    1464.56
 5.800      -5.102    5.070    1.022    2.479    0.601    1457.65
 5.900      -5.002    4.814    0.985    2.444    0.603    1461.49
 6.000      -4.962    4.520    0.935    2.416    0.610    1469.28
=h==                 ' indicate end of information
```

6.6.10 W.IN_AS

```
1  ------------------------------------------------------------------------
2  Optical Properties of:   Indium Arsenide  (InAs)
3
4  D.E. Aspnes  &  A.A. Studna,
5  Physical Review B,  Volume 27,  Number 2, 1983, January, 15.
6  Pages:   985-1009.
7
8  ------------------------------------------------------------------------
9  E(eV),       <e1>,     <e2>,     n,       k,       R,     alpha*1.0E3 (cm**-1)
10 ------------------------------------------------------------------------
11 =h==                  ' indicate end of header information
12  46                   ' quantity of lines of data
13  1.500      13.605     3.209    3.714    0.432    0.337      65.69
14  1.600      13.884     3.478    3.755    0.463    0.342      75.11
15  1.700      14.181     3.744    3.798    0.493    0.347      84.94
16  1.800      14.545     4.083    3.851    0.530    0.353      96.72
17  1.900      15.015     4.481    3.917    0.572    0.361     110.16
18  2.000      15.558     5.062    3.995    0.634    0.370     128.43
19  2.100      16.205     5.820    4.088    0.712    0.380     151.51
20  2.200      16.957     6.905    4.199    0.822    0.394     183.33
21  2.300      17.776     8.582    4.331    0.991    0.411     230.98
22  2.400      18.298    11.458    4.466    1.283    0.433     312.08
23  2.500      15.856    15.592    4.364    1.786    0.454     452.64
24  2.600      12.611    15.160    4.021    1.885    0.441     496.84
25  2.700      11.229    15.766    3.911    2.016    0.445     551.66
26  2.800       8.276    16.010    3.626    2.208    0.448     626.53
27  2.900       6.603    14.211    3.337    2.129    0.428     625.87
28  3.000       6.083    13.003    3.197    2.034    0.412     618.46
29  3.100       5.831    12.162    3.108    1.957    0.400     614.80
30  3.200       5.736    11.540    3.051    1.891    0.389     613.30
31  3.300       5.735    11.082    3.018    1.836    0.381     614.18
32  3.400       5.820    10.753    3.004    1.790    0.375     616.84
33  3.500       5.973    10.550    3.008    1.754    0.371     622.13
34  3.600       6.197    10.471    3.030    1.728    0.370     630.47
35  3.700       6.478    10.529    3.069    1.715    0.370     643.29
36  3.800       6.835    10.754    3.129    1.719    0.374     661.95
37  3.900       7.254    11.187    3.208    1.743    0.382     689.18
38  4.000       7.744    11.919    3.313    1.799    0.393     729.23
39  4.100       8.273    13.130    3.449    1.903    0.411     791.03
40  4.200       8.663    15.173    3.615    2.099    0.437     893.42
41  4.300       8.000    18.639    3.761    2.478    0.478    1080.14
42  4.400       4.024    22.171    3.644    3.042    0.527    1356.76
43  4.500      -1.663    22.006    3.194    3.445    0.566    1571.19
44  4.600      -5.509    19.372    2.705    3.581    0.593    1669.74
45  4.700      -7.921    15.762    2.205    3.575    0.617    1703.11
46  4.800      -7.961    12.077    1.803    3.349    0.622    1629.16
47  4.900      -6.905     9.909    1.608    3.081    0.605    1530.10
48  5.000      -5.923     8.752    1.524    2.871    0.583    1455.26
49  5.100      -5.264     8.107    1.484    2.732    0.565    1412.38
50  5.200      -4.942     7.600    1.436    2.646    0.556    1394.86
```

```
51   5.300      -4.665    6.980    1.366    2.555    0.550    1372.78
52   5.400      -4.278    6.425    1.312    2.449    0.537    1340.56
53   5.500      -3.851    6.008    1.282    2.344    0.521    1306.62
54   5.600      -3.424    5.738    1.276    2.248    0.501    1275.94
55   5.700      -3.006    5.595    1.293    2.163    0.479    1249.73
56   5.800      -2.642    5.602    1.333    2.102    0.459    1235.70
57   5.900      -2.430    5.764    1.383    2.084    0.448    1246.25
58   6.000      -2.403    6.055    1.434    2.112    0.448    1284.15
59  =h==                 ' indicate end of information
```

6.6.11 W.IN_P

```
1  ------------------------------------------------------------------------
2  Optical Properties of:   Indium Phosphide (InP)
3
4  D.E. Aspnes  &  A.A. Studna,
5  Physical Review B,  Volume 27,  Number 2, 1983, January, 15.
6  Pages:   985-1009.
7
8  ------------------------------------------------------------------------
9  E(eV),       <e1>,     <e2>,     n,       k,       R,    alpha*1.0E3 (cm**-1)
10 ------------------------------------------------------------------------
11 =h==                  " indicate end of header information
12  46                   " quantity of lines of data
13  1.500       11.904     1.400    3.456    0.203    0.305      30.79
14  1.600       11.972     1.509    3.467    0.218    0.307      35.30
15  1.700       12.022     1.680    3.476    0.242    0.308      41.64
16  1.800       12.120     1.889    3.492    0.270    0.310      49.34
17  1.900       12.284     2.062    3.517    0.293    0.313      56.44
18  2.000       12.493     2.252    3.549    0.317    0.317      64.32
19  2.100       12.734     2.488    3.585    0.347    0.322      73.87
20  2.200       13.026     2.755    3.629    0.380    0.327      84.65
21  2.300       13.382     3.060    3.682    0.416    0.333      96.89
22  2.400       13.812     3.425    3.745    0.457    0.341     111.25
23  2.500       14.313     3.904    3.818    0.511    0.349     129.56
24  2.600       14.899     4.524    3.903    0.579    0.360     152.71
25  2.700       15.585     5.337    4.004    0.667    0.372     182.41
26  2.800       16.365     6.482    4.121    0.786    0.386     223.21
27  2.900       17.188     8.205    4.256    0.964    0.404     283.32
28  3.000       17.759    10.962    4.395    1.247    0.427     379.23
29  3.100       16.483    15.325    4.415    1.735    0.454     545.30
30  3.200       11.211    17.043    3.976    2.143    0.458     695.23
31  3.300        7.911    15.797    3.576    2.209    0.446     738.76
32  3.400        6.639    13.592    3.299    2.060    0.419     709.95
33  3.500        6.400    12.443    3.193    1.948    0.403     691.21
34  3.600        6.312    11.731    3.133    1.872    0.391     683.12
35  3.700        6.330    11.266    3.103    1.816    0.383     680.92
36  3.800        6.432    10.974    3.095    1.773    0.378     682.96
37  3.900        6.616    10.841    3.108    1.744    0.376     689.47
38  4.000        6.874    10.871    3.141    1.730    0.376     701.54
39  4.100        7.205    11.088    3.196    1.735    0.380     720.91
40  4.200        7.620    11.539    3.275    1.762    0.387     750.02
41  4.300        8.119    12.358    3.384    1.826    0.400     795.80
42  4.400        8.644    13.739    3.527    1.948    0.419     868.69
43  4.500        8.891    16.161    3.697    2.186    0.449     996.95
44  4.600        7.484    20.039    3.800    2.637    0.493    1229.49
45  4.700        2.292    22.948    3.560    3.223    0.543    1535.24
46  4.800       -3.469    20.989    2.984    3.517    0.577    1711.26
47  4.900       -5.868    17.894    2.546    3.514    0.591    1745.40
48  5.000       -7.678    14.896    2.131    3.495    0.613    1771.52
49  5.100       -7.787    11.483    1.745    3.291    0.620    1701.26
50  5.200       -6.668     9.399    1.558    3.016    0.601    1589.64
```

```
51   5.300      -5.654    8.308    1.482    2.802    0.577    1505.35
52   5.400      -4.915    7.717    1.455    2.652    0.554    1451.50
53   5.500      -4.528    7.308    1.426    2.562    0.542    1428.14
54   5.600      -4.280    6.832    1.375    2.484    0.534    1410.02
55   5.700      -3.924    6.317    1.325    2.383    0.522    1376.99
56   5.800      -3.509    5.924    1.299    2.280    0.504    1340.27
57   5.900      -3.073    5.680    1.301    2.183    0.483    1305.47
58   6.000      -2.681    5.644    1.336    2.113    0.461    1285.10
59  =h==                 ' indicate end of information
```

6.6.12 W.IN_SB

```
1  ------------------------------------------------------------------------
2  Optical Properties of:   Indium Antimonide (InSb)
3
4  D.E. Aspnes  &  A.A. Studna,
5  Physical Review B,  Volume 27,  Number 2, 1983, January, 15.
6  Pages:   985-1009.
7
8  ------------------------------------------------------------------------
9  E(eV),       <e1>,     <e2>,     n,       k,       R,     alpha*1.0E3 (cm**-1)
10 ------------------------------------------------------------------------
11 =h==                  ' indicate end of header information
12  46                   ' quantity of lines of data
13  1.500      19.105      5.683    4.418    0.643    0.406      97.79
14  1.600      20.302      6.838    4.568    0.749    0.421     121.39
15  1.700      21.699      9.019    4.754    0.949    0.441     163.46
16  1.800      22.148     13.707    4.909    1.396    0.467     254.73
17  1.900      16.144     16.603    4.433    1.873    0.463     360.65
18  2.000      14.448     14.875    4.194    1.773    0.443     359.46
19  2.100      13.974     14.643    4.136    1.770    0.439     376.79
20  2.200      13.674     15.302    4.135    1.850    0.445     412.62
21  2.300      12.653     16.936    4.111    2.060    0.458     480.25
22  2.400       9.377     17.480    3.822    2.287    0.463     556.30
23  2.500       7.811     15.856    3.570    2.221    0.447     562.77
24  2.600       7.278     14.787    3.447    2.145    0.434     565.33
25  2.700       7.069     14.069    3.377    2.083    0.425     570.01
26  2.800       7.044     13.617    3.345    2.036    0.419     577.71
27  2.900       7.150     13.395    3.342    2.004    0.415     589.12
28  3.000       7.354     13.421    3.366    1.994    0.416     606.27
29  3.100       7.627     13.779    3.419    2.015    0.420     633.20
30  3.200       7.742     14.572    3.482    2.093    0.431     678.77
31  3.300       7.507     15.631    3.525    2.217    0.445     741.69
32  3.400       6.782     16.678    3.520    2.369    0.459     816.31
33  3.500       5.995     17.673    3.511    2.517    0.474     892.82
34  3.600       4.830     18.854    3.485    2.705    0.492     987.01
35  3.700       3.147     20.102    3.427    2.933    0.514    1099.83
36  3.800       0.534     21.064    3.287    3.204    0.541    1234.25
37  3.900      -2.838     21.177    3.044    3.479    0.572    1375.21
38  4.000      -6.722     19.443    2.632    3.694    0.608    1497.79
39  4.100      -8.911     15.595    2.127    3.666    0.633    1523.33
40  4.200      -8.580     12.296    1.791    3.433    0.634    1461.59
41  4.300      -7.678     10.382    1.618    3.209    0.623    1398.45
42  4.400      -6.910      9.191    1.515    3.034    0.610    1353.08
43  4.500      -6.297      8.351    1.443    2.894    0.598    1320.24
44  4.600      -5.788      7.690    1.385    2.776    0.586    1294.36
45  4.700      -5.324      7.160    1.341    2.669    0.574    1271.51
46  4.800      -4.912      6.761    1.312    2.576    0.562    1253.17
47  4.900      -4.534      6.492    1.301    2.495    0.548    1239.36
48  5.000      -4.250      6.378    1.307    2.441    0.537    1237.01
49  5.100      -4.190      6.360    1.309    2.430    0.534    1255.97
50  5.200      -4.359      6.207    1.270    2.444    0.543    1288.04
```

```
51   5.300      -4.505     5.815    1.194    2.435    0.556    1308.25
52   5.400      -4.487     5.345    1.116    2.394    0.563    1310.58
53   5.500      -4.325     4.931    1.057    2.333    0.563    1300.55
54   5.600      -4.126     4.664    1.025    2.275    0.558    1291.44
55   5.700      -3.995     4.470    1.000    2.235    0.555    1291.22
56   5.800      -3.945     4.282    0.969    2.210    0.558    1299.21
57   5.900      -3.925     4.029    0.922    2.185    0.565    1306.73
58   6.000      -3.835     3.681    0.861    2.139    0.572    1300.85
59  =h==                 ' indicate end of information
```

6.6.13 W.AL_SB

```
1  ------------------------------------------------------------------------
2  Optical Properties of:     Aluminum Antimonide (AlSb)
3
4  S. Zollner, C. Lin, E. Schonherr, A. Bohringer, and M. Cardona
5  Journal of Applied Physics, Volume 66, Number 1, 1 July 1989.
6  Pages:  383-387.
7  ------------------------------------------------------------------------
8  E(eV),       <e1>,     <e2>,     n,       k,       R,     alpha*1.0E3 (cm**-1)
9  ------------------------------------------------------------------------
10 =h==              ' indicate end of header information
11  45               ' quantity of lines of data
12  1.400      12.23     0.0007    3.50     0.0001   0.308       0.02
13  1.500      12.56     0.001     3.54     0.0002   0.312       0.03
14  1.600      12.91     0.002     3.60     0.0003   0.319       0.05
15  1.700      13.30     0.007     3.66     0.001    0.325       0.2
16  1.800      13.93     0.015     3.73     0.002    0.333       0.3
17  1.900      14.53     0.02      3.81     0.003    0.341       0.6
18  2.000      15.24     0.03      3.90     0.004    0.350       0.8
19  2.100      16.08     0.05      4.01     0.006    0.361       1.3
20  2.200      17.54     0.08      4.20     0.01     0.378       2.4
21  2.300      18.50     2.11      4.31     0.24     0.390      57.1
22  2.400      19.62     2.99      4.44     0.33     0.402      82.0
23  2.500      20.97     4.25      4.61     0.46     0.417     117.0
24  2.600      22.76     6.09      4.81     0.63     0.436     166.8
25  2.700      24.98     9.35      5.08     0.92     0.462     252.1
26  2.800      25.30    16.70      5.27     1.58     0.496     449.3
27  2.900      17.41    19.19      4.66     2.06     0.486     605.7
28  3.000      16.54    17.82      4.52     1.97     0.473     599.0
29  3.100      16.30    19.43      4.57     2.12     0.485     668.0
30  3.200      14.24    22.35      4.51     2.47     0.505     803.1
31  3.300       9.91    22.32      4.14     2.69     0.508     900.8
32  3.400       8.77    21.00      3.97     2.64     0.498     911.2
33  3.500       8.31    21.33      3.95     2.69     0.503     957.7
34  3.600       7.72    22.32      3.96     2.81     0.513    1028.7
35  3.700       6.63    23.35      3.91     2.97     0.525    1113.8
36  3.800       4.53    24.12      3.81     3.16     0.540    1218.3
37  3.900       2.16    25.22      3.71     3.40     0.560    1345.0
38  4.000      -1.56    26.00      3.50     3.71     0.588    1506.1
39  4.100      -6.14    25.24      3.15     4.00     0.621    1665.6
40  4.200     -11.16    21.99      2.60     4.22     0.662    1799.7
41  4.300     -12.73    15.94      1.96     4.07     0.691    1776.3
42  4.400     -11.20    11.81      1.59     3.70     0.688    1653.6
43  4.500      -9.41     9.81      1.45     3.39     0.669    1547.4
44  4.600      -8.11     8.71      1.38     3.16     0.648    1475.0
45  4.700      -7.17     7.97      1.33     2.99     0.629    1425.2
46  4.800      -6.40     7.42      1.30     2.84     0.611    1384.6
47  4.900      -5.76     7.06      1.29     2.72     0.592    1354.9
48  5.000      -5.29     6.85      1.30     2.64     0.576    1338.8
49  5.100      -5.00     6.77      1.31     2.59     0.565    1339.4
50  5.200      -4.95     6.74      1.31     2.58     0.563    1360.3
```

```
51   5.300      -5.15      6.53      1.26      2.59      0.574     1394.7
52   5.400      -5.30      6.02      1.17      2.58      0.589     1413.2
53   5.500      -5.18      5.51      1.09      2.52      0.593     1407.9
54   5.600      -5.00      5.12      1.04      2.46      0.594     1399.9
55   5.700      -4.84      4.88      1.01      2.42      0.592     1398.7
56   5.800      -4.81      4.78      1.00      2.40      0.593     1415.3
57  =h==                 ' indicate end of information
```

7. Acknowledgments

It is the author's pleasure to acknowledge Dr. George Candela for his suggestions and support during the course of this project. The author appreciates the questions and suggestions from Deane Chandler-Horowitz and Pradip Dutta, who have been among the first to apply the software package. Somewhere during the middle of the development of this software, the author had the expressed good fortune of receiving an invitation from Drs. Paul G. Snyder and John A. Woollam to attend a week-long seminar regarding ellipsometry that they were organizing at the University of Nebraska-Lincoln. The author appreciates the many fine presentations of and discussions with the staff and invited speakers, as well as the hospitality that was extended to all of the participants.

Regarding the database of dielectric functions for the constituent media, these were collected from the literature and serve so vitally in advancing various applications of spectroscopic ellipsometry. Such collections of data are products of much labor and incorporate much relevant physics. It would be a serious omission to fail to include special mention of appreciation of those who have made such databases available. Among them, the author especially appreciates the work reported by Dr. David E. Aspnes and associates.

Finally, the author appreciates the assistance of Dr. Ronald Boisvert in making available various subroutines from the GAMS software library, and the editorial assistance of Ms. Jane Walters and Jo Gonzalez.

8. References

[1] R. M. A. Azzam and N. M. Bashara, *Ellipsometry and Polarized Light*, paperback edition (North-Holland, Amsterdam, 1987).

[2] *Fundamentals and Applications of Ellipsometry* ed. by J. A. Woollam, Department of Electrical Engineering, (University of Nebraska, Lincoln, Nebraska, June 15–19, 1987).

[3] J. F. Marchiando, "*Semiconductor Measurement Technology:* A Software Program for Aiding the Analysis of Ellipsometric Measurements, Simple Models," *NIST Spec. Publ. 400-83*, Natl. Inst. Stand. Technol. (U.S. Department of Commerce), 1989.

[4] *Handbook of Optical Constants of Solids*, ed. by E. W. Palik (Academic Press, Orlando, 1985).

[5] F. Wooten, *Optical Properties of Solids* (Academic, New York, 1972).

[6] Handbook on Semiconductors, Volume 2, *Optical Properties of Solids*, ed. by M. Balkanski, (North-Holland, Amsterdam, 1980).

[7] P. G. Snyder, J. E. Oh, J. A. Woolam, and R. E. Owens, "Ellipsometric Analysis of Built-in Electric Fields in Semiconductor Heterostructures," *Appl. Phys. Lett.* **51**, 770–772 (1987).

[8] F. A. Hopf and G. I. Stegeman, *Applied Classical Electrodynamics, Volume 1: Linear Optics* (John Wiley & Sons, Inc., New York, 1985).

[9] F. E. Jones, "The Refractivity of Air," *J. Res. Natl. Bur. Stand. (U.S.)* **86**, 27–32 (January–February 1981).

[10] D. A. G. Bruggeman, "Calculation of the Various Physical Constants of Heterogeneous Substances," *Ann. Physik*, Series 5, **24**, 636–664 (1935).

[11] C. G. Granqvist and O. Hunderi, "Optical Properties of Ultrafine Gold Particles," *Phys. Rev. B*, **16**, 3513–3534 (1977).

[12] D. E. Aspnes, J. B. Theeten, and F. Hottier, "Investigation of Effective-Medium Models of Microscopic Surface Roughness by Spectroscopic Ellipsometry," *Phys. Rev. B*, **20**, 3292–3302 (1979).

[13] D. E. Aspnes and J. B. Theeten, "Dielectric Function of Si-SiO_2 and Si-Si_3N_4 Mixtures," *J. Appl. Phys.*, **50**, 4928–4935 (1979).

[14] R. F. Boisvert, S. E. Howe, and D. K. Kahaner, *Guide to Available Mathematical Software*, Center for Applied Mathematics, National Bureau of Standards (U.S. Department of Commerce), 1985.

[15] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK Users' Guide* (Society for Industrial and Applied Mathematics, Philadelphia, 1979).

[16] C. C. Paige and M. A. Saunders, "LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares," *ACM Trans. Math. Softw.* **8**, 43–71 (1982).

[17] C. C. Paige and M. A. Saunders, "Algorithm 583, LSQR: Sparse Linear Equations and Least Squares Problems," *ACM Trans. Math. Softw.* **8**, 195–209 (1982).

[18] G. H. Bu-Abbud, N. M. Bashara, and J. A. Woolam, "Variable Wavelength, Variable Angle Ellipsometry Including a Sensitivities Correlation Test," *Thin Solid Films* **138**, 27–41 (1986).

[19] G. H. Bu-Abbud and N. M. Bashara, "Parameter Correlation and Precision in Multiple-Angle Ellipsometry," *Appl. Opt.* **20**, 3020–3026 (1981).

[20] M. M. Ibrahim and N. M. Bashara, "Parameter-Correlation and Computational Considerations in Multiple-Angle Ellipsometry," *J. Opt. Soc. Am.* **61**, 1622–1629 (1971).

[21] S. Y. Kim and K. Vedam, "Proper Choice of the Error Function in Modeling Spectroellipsometric Data," *Appl. Opt.* **25**, 2013–2021 (1986).

[22] "The SCD Graphics Utilities," ed. by G. R. McArthur, NCAR Technical Note 166+IA, Scientific Computing Division, National Center for Atmospheric Research, Boulder, Colorado (1981).

[23] "The System Plot Package," edited by G. R. McArthur, NCAR Technical Note 162+IA, Scientific Computing Division, National Center for Atmospheric Research, Boulder, Colorado (1981).

Appendix

Quick Reference Guide
for
NIST Special Publication 400–84

Semiconductor Measurement Technology:
A Software Program for Aiding the Analysis of Ellipsometric Measurements, Simple Spectroscopic Models

J. F. Marchiando
Semiconductor Electronics Division
National Institute of Standards and Technology
Gaithersburg, Maryland 20899

Table of Contents

1. Introduction

MAIN2 is a software program for analyzing spectroscopic ellipsometric measurements. The solid material sample is assumed to be nonmagnetic and to exhibit depth-dependent optical properties, such as one with layered structure atop a substrate, where the substrate behaves like a semi-infinite half-space, and where the layers are flat and of uniform thickness. The ambient region refers to that region of space that lies external to the layers-substrate region of the sample. The optical medium within each ambient/layer/substrate region is assumed to be isotropic, homogeneous, local, and linear. The dielectric function of an optical medium is represented by the Bruggeman effective medium approximation (EMA). An effective medium is represented as a mixture of distinct constituent bulk media, where the optical properties of each constituent medium is known a priori. The ellipsometric equations are solved for the model parameters by using a damped least-squares method.

The program involves at least five data files. These include: X.DAT, an input data file of the model parameters and measurement data; X.INN, an input data file of command options controlling the program; X.OUT, an output data file journaling the activity of the program; X.PLOT, an output data file of results suitable for later plotting; and W.*, the data files that contain the dispersion relationships of the distinct constituent media that are used in representing the effective media of the sample.

2. Input Data Requirements

The input data file X.DAT contains both the model parameters that characterize the layered structure of the sample and the ellipsometric measurement data. Each subsection presents a format convention that is used in forming the input file X.DAT for one particular subset of these input data. The subsections are presented in the order that the subsets are to appear in the input file. Each subset of entries is separated by a demaraction line that is made of connected hyphens and is at least 50 characters in length. Parameters that are shown to be capped with a tilde refer to index labels of array elements.

2.1 Database Information

The first line in file X.DAT specifies the location of the database files. The database files contain the optical properties or complex dielectric functions as functions of energy or optical frequency of the distinct constituent media. These constituent media are used in rep-

resenting the effective media of the ambient/layers/substrate regions. The format convention for specifying the location of the database files is of the form:

disk:[directory]w.

where *'disk'* refers to the name of the mass-storage device, *'directory'* refers to the name of the directory that contains the database files of the constituent media, and *'w'* is the filename of the various database files. The database file of a particular constituent medium has a distinct *extension* in its filename specification; the appropriate *extension* is supplied by the program.

2.2 Layer Thicknesses

The format convention for specifying the layer thicknesses is of the form:

$$m_z \;/\; (i,\; z_i,\; \delta z_i,\; v_i)$$

where:

m_z is the number of distinct thicknesses;

i is an integer that indexes the line entries consecutively in unit increments, i.e., $(i = 1, 2, 3, \ldots, m_z)$;

z_i is the thickness of a layer measured in nanometers;

δz_i is the uncertainty that is assigned to the numerical value of z_i; and

v_i is the integer (froz/vary) switch with value 0 or 1, respectively.

The forward slash mark '/' delimits the first line of input data. The parentheses bound items that ought to appear on each subsequent line of data until the implied DO-loop has been satisfied. Incidentally, if $(m_z = 0)$, this is the only line of information that ought to be entered for this set of data, i.e., apart from the short line of connected hyphens for demarcation.

2.3 Supplementary Parameters (Integer)

The format convention for specifying the integer supplementary parameters is of the form:

$$m_I \;/\; (i,\; p_i)$$

where:

m_I is the number of distinct integer supplementary parameters;

i is an integer that indexes the line entries of data consecutively in unit increments, i.e., $(i = 1, 2, 3, \ldots, m_I)$; and

p_i is the integer supplementary parameter.

Since integer parameters are not subjected to optimization, no uncertainties are included regarding their numerical value.

2.4 Supplementary Parameters (Floating-Point)

The format convention for specifying the floating-point supplementary parameters is of the form:

$$m_R \; / \; (i,\; p_i,\; \delta p_i,\; v_i)$$

where:

m_R is the number of distinct floating-point supplementary parameters;

i is an integer that indexes the line entries of data consecutively in unit increments, i.e., $(i = 1, 2, 3, \ldots, m_R)$;

p_i is the floating-point supplementary parameter;

δp_i is the uncertainty that is assigned to the numerical value of p_i; and

v_i is the integer (froz/vary) switch with value of either 0 or 1, respectively.

2.5 Effective Media or Mixtures

For each effective medium used in the ambient/layers/substrate regions, one must specify the constituent media and their volume fractions. The format convention for specifying the effective media or mixtures is of the form:

m_{media}

$\vdots$

$m_{f,j},\ m_{I,j},\ m_{R,j}$ $\quad / (i_f,\ \tilde{\varepsilon}_i,\ f_i,\ \delta f_i,\ v_i)$

$/ (i_I,\ \tilde{p}_{I,i})$

$/ (i_R,\ \tilde{p}_{R,i})$

$\vdots$

where:

m_{media} is the number of distinct effective media, i.e., $(m_{\text{media}} \geq 2)$;

j is an integer that indexes the distinct effective media, i.e., $(j = 1, 2, \ldots, m_{\text{media}})$;

$m_{f,j}$ is the number of distinct constituent media (or volume fractions) that are associated with the j^{th} effective medium;

$m_{I,j}$ is the number of integer supplementary parameters that are associated with the j^{th} effective medium;

$m_{R,j}$ is the number of floating-point supplementary parameters that are associated with the j^{th} effective medium;

i_f is an integer that locally indexes the distinct constituent media (or volume fractions) of the j^{th} effective medium, i.e., $(i_f = 1, 2, \ldots, m_{f,j})$;

$\tilde{\varepsilon}_i$ is an integer index label of the appropriate constituent medium that is associated with the $i_f{}^{\text{th}}$ constituent medium of the j^{th} effective medium;

f_i is the volume fraction of the $i_f{}^{\text{th}}$ constituent medium of the j^{th} effective medium;

δf_i is the uncertainty that is assigned to the numerical value of f_i;

v_i is the integer (froz/vary) switch for f_i with value 0 or 1, respectively;

i_I is an integer that locally indexes the integer supplementary parameters that are associated with the j^{th} effective medium, i.e., $(i_I = 1, 2, \ldots, m_{I,j})$;

$\tilde{p}_{I,i}$ is the integer index label of the appropriate integer supplementary parameter that is associated with the $i_I{}^{\text{th}}$ integer supplementary parameter of the j^{th} effective medium, i.e., $(1 \leq \tilde{p}_{I,i} \leq m_I)$;

i_R is an integer that locally indexes the floating-point supplementary parameters that are associated with the j^{th} effective medium, i.e., $(i_R = 1, 2, \ldots, m_{R,j})$; and

$\tilde{p}_{R,i}$ is the integer index label of the appropriate floating-point supplementary parameter that is associated with the $i_R{}^{\text{th}}$ floating-point supplementary parameter of the j^{th} effective medium, i.e., $(1 \leq \tilde{p}_{R,i} \leq m_R)$.

2.6 Ambient and External Parameters

The ambient region refers to that spatial region that lies external to the layered structure of the sample. The indexed listing associates distinct effective media with distinct subsets of the supplementary parameters. The format convention for specifying the ambient and the external parameters is of the form:

m_{ambients}

$\vdots$

$j,\ \tilde{\varepsilon}_j,\ m_{I,j},\ m_{R,j}$ / $(i_I,\ \tilde{p}_{I,i})$

/ $(i_R,\ \tilde{p}_{R,i})$

$\vdots$

where:

m_{ambients} is the number of distinct ambients which involve associating effective media with subsets of the supplementary parameters;

j is an integer that indexes the distinct ambients, i.e., $(j = 1, 2, \ldots, m_{\text{ambients}})$;

$\tilde{\varepsilon}_j$ is an integer index label of the appropriate effective (*not* constituent) medium that is associated with the j^{th} ambient, i.e., $(1 \leq \tilde{\varepsilon}_j \leq m_{\text{media}})$;

$m_{I,j}$ is the number of integer supplementary parameters that are associated with the j^{th} ambient;

$m_{R,j}$ is the number of floating-point supplementary parameters that are associated with the j^{th} ambient;

i_I is an integer that locally indexes the integer supplementary parameters that are associated with the j^{th} ambient, i.e., $(i_I = 1, 2, \ldots, m_{I,j})$;

$\tilde{p}_{I,i}$ is the integer index label of the appropriate integer supplementary parameter that is associated with the $i_I{}^{\text{th}}$ integer supplementary parameter of the j^{th} ambient, i.e., $(1 \leq \tilde{p}_{I,i} \leq m_I)$;

i_R is an integer that locally indexes the floating-point supplementary parameters that are associated with the j^{th} ambient, i.e., $(i_R = 1, 2, \ldots, m_{R,j})$; and

$\tilde{p}_{R,i}$ is the integer index label of the appropriate floating-point supplementary parameter that is associated with the $i_R{}^{\text{th}}$ floating-point supplementary parameter of the j^{th} ambient, i.e., $(1 \leq \tilde{p}_{R,i} \leq m_R)$.

2.7 Sample Characterization or Construction

The samples are constructed or assembled in a layer-by-layer fashion. Each spatial region requires an effective medium, and if the region is a layer, a thickness is required as well. Index labels are used to specify the configuration of the (layers/substrate) system. The format convention for specifying the layered structure of the samples is of the form:

$$
\begin{array}{lll}
m_{\text{samples}} & & \\
 & \vdots & \\
 & m_{z,j} \qquad / \ (i,\ \tilde{\varepsilon}_i,\ \tilde{z}_i) & \\
 & \qquad\quad / \ m_{z,j}+1,\ \ \tilde{\varepsilon}_{\text{substrate},j} & \\
 & \vdots &
\end{array}
$$

where:

m_{samples} is the number of distinct material samples being analyzed.

j is an integer that indexes the distinct samples, i.e., $(j = 1, 2, \ldots, m_{\text{samples}})$.

$m_{z,j}$ is the number of layers that lie atop the substrate of the j^{th} sample.

i is an integer that locally indexes the distinct layers of the j^{th} sample, i.e., $(i = 1, 2, \ldots, m_{z,j})$. The layer adjacent to the ambient is indexed 1, and the layer adjacent to the substrate is indexed $m_{z,j}$.

$\tilde{\varepsilon}_i$ is the integer index label of the appropriate effective medium that is associated with the i^{th} layer of the j^{th} sample, i.e., $(1 \leq \tilde{\varepsilon}_i \leq m_{\text{media}})$.

$\tilde{z}_i$ is the integer index label of the appropriate thickness that is associated with the i^{th} layer of the j^{th} sample, i.e., $(1 \leq \tilde{z}_i \leq m_z)$.

2.8 Measurement Data (Δ, ψ)

The measurement data of ellipsometric angles are organized in the same fashion that the samples are constructed, i.e., sample by sample. The data structure is of the form:

(*angles & wavelengths, repeats, ambients, samples*)

where:

samples indexes the set of distinct samples;

ambients indexes the set of distinct ambients involving a given sample;

repeats indexes the sets of distinct repeats of multiple-angle of incidence and multiple-optical frequency measurements of a system involving a given ambient and sample; and

angles & wavelengths indexes the set of distinct angles of incidence and optical frequencies (or wavelengths) used during the measurements of a system involving a given repeat index, ambient, and sample.

The measurement data are grouped sample by sample. For each sample, they are grouped ambient by ambient. For each ambient, they are grouped by their repeat index label. For each repeat index, they are indexed by both source variables, i.e., the angle of incidence and the optical frequency (or wavelength in vacuum or corresponding energy) of the light. The format convention for associating the measurement data with the sample, e.g., sample s, is of the form:

m_{as}

$\vdots$

m_{ras} $\quad \tilde{a}_s$

$\vdots$

$m_{\lambda\phi,ras}$ / $(i,\ \lambda_i,\ \phi_i,\ \Delta_i,\ \psi_i\ /\ \delta\lambda_i,\ \delta\phi_i,\ \delta\Delta_i,\ \delta\psi_i)_{ras}$

$\vdots$

$\vdots$

where:

s is an integer that indexes the samples, i.e., $(s = 1, 2, \ldots, m_{\text{samples}})$.

m_{as} is the number of ambients associated with sample s.

a_s is an integer that locally indexes the ambients associated with sample s, i.e., $(a_s = 1, 2, \ldots, m_{as})$.

$\tilde{a}_s$ is the integer index label of the appropriate ambient that is associated with the $a_s{}^{\text{th}}$ ambient on sample s.

m_{ras} is the number of sets of repeat measurements of (Δ, ψ) involving the $a_s{}^{\text{th}}$ ambient on sample s.

$m_{\lambda\phi,ras}$ is the number of measurements of ellipsometric angles (Δ, ψ) involving multiple-angles of incidence and multiple-optical frequencies (or wavelengths in vacuum or associated energies) that involve the $r_{as}{}^{\text{th}}$ repeat set of measurements, i.e., $(r_{as} = 1, 2, \ldots, m_{ras})$, and the $a_s{}^{\text{th}}$ ambient of sample s.

i is an integer that locally indexes the multiple-angle of incidence and multiple-optical frequency measurement data consecutively with unit increments, i.e., $(i = 1, 2, \ldots, m_{\lambda\phi,ras})$.

λ_i is a floating-point source variable that may be of either sign depending upon how one chooses to characterize the optical frequency of light. When the value is negative, the unit of measure is nanometers. When the value is positive, the unit of measure is electron-volts. The program check-tests for either $(-1240 \leq \lambda_i \leq -200)$ or $(1.0 \leq \lambda_i \leq 6.0)$.

$\delta\lambda_i$ is the uncertainty that is assigned to the numerical value of λ_i.

ϕ_i is the angle of incidence measured in degrees, where a value of zero relates to that of normal incidence, i.e., $(0 \leq \phi_i \leq 90)$.

$\delta\phi_i$ is the uncertainty that is assigned to the numerical value of ϕ_i.

Δ_i is an ellipsometric angle measured in degrees, i.e., $(0 \leq \Delta_i < 360)$, assuming the engineering or Nebraska convention, i.e., $R_{p,H}$.

$\delta\Delta_i$ is the uncertainty that is assigned to the numerical value of Δ_i.

ψ_i is an ellipsometric angle measured in degrees, i.e., $(0 \leq \psi_i \leq 90)$.

$\delta\psi_i$ is the uncertainty that is assigned to the numerical value of ψ_i.

Within any set of $m_{\lambda\phi,ras}$ measurement data, ordering among the angles of incidence or among the optical frequencies (or wavelengths in vacuum or associated energies) is not necessarily important.

To construct the final set of data that includes all of the samples, one simply concatenates the data of the individual samples together, i.e., without any intervening lines of demarcation.

3. Command Options

The input data file X.INN contains the list of command options that direct the control of the software package. To direct the execution of the program, a menu-driven decision tree of command options is made available to the user. The subsections are presented in the order that the options are listed at level one. The first level of the tree involves a menu of three options.

```
Enter:   option
           1,  forward problems, plots, ...
           2,  search      (vary)
           3,  search grid (vary)
```

where the options at level one include:

1, requests the program to perform one of several simple tasks, such as providing a set of tabulated output that is amenable for plotting or initiating a series of calculations of the direct or forward problem. No iterative calculations are considered, i.e., no minimizations or inversions of the ellipsometric equations.

2, requests the program to invert the ellipsometric equations by performing a series of unconstrained optimization calculations.

3, requests the program to invert the ellipsometric equations by performing a grid scan over a selected set of model parameters. A series of constrained optimization calculations is initiated at each point of the grid of model parameters.

3.1 Forward Problems, Plots, ...

This option provides simulations of the forward problem, i.e., option '1' at level one. The next set of options requests the type of grid used for the source variables (λ, ϕ), i.e., the optical frequency (or wavelength in vacuum or corresponding energy) and the angle of incidence. The menu of options at level two is:

```
Enter:   choice of incident (energies, angles)
           1,   measurement data,     x.dat
           2,   grid scan.
```

where the options at level two involve:

1, requests the program to use as source variables only those sets of ordered pairs (λ, ϕ) that are specified in the input file X.DAT for the measurement data of (Δ, ψ).

2, requests the program to use as source variables only those sets of ordered pairs (λ, ϕ) that lie on a two-dimensional grid that is specified later in a following prompt-request. The program prompts or requests information regarding the lower bound, the upper bound, and the stepsize for each dimension of the grid.

The next step in the program involves specifying which field quantities are to be calculated and written to the output file X.PLOT. Two alternative sets of options are presented in succession.

Following the selection of option '1' at level two, the menu of options at level three is:

```
Enter:   choice of output suitable for:
           1,   input data,     x.dat,
           2,   plotting (Delta, psi),
           3,   plotting (Delta, psi) deviations,
                 deviation = measurement - model.
           4,   plotting |g| ^ rms deviation,  unscaled,
                 on a 1D or 2D grid of model parameters.
```

where:

1, requests the program to calculate (Δ, ψ) using (λ, ϕ) as supplied by the file X.DAT. The output is written to the plot file X.PLOT. The format is suitable for use in file X.DAT.

2, requests the program to calculate (Δ, ψ) using (λ, ϕ) as supplied by the file X.DAT. The output is written to the plot file X.PLOT. The format is suitable for later graphics.

3, requests the program to calculate and tabulate the deviations between the measurement data and that calculated by the model. The output is written to the plot file X.PLOT.

4, requests the program to initiate a grid scan of model parameters and evaluate the error expression |g| for each point on the grid. For convenience, the grid may be either one or two dimensional; i.e., only one or two model parameters may undergo variation. To specify the grid of 'vary' model parameters, see section 3.3, but note that there is no option regarding optimization here. The output is written to file X.PLOT.

Alternatively, following the selection of option '2' at level two, the menu of options at level three is:

```
Enter:   choice of output suitable for:
           1,   dielectric function,   media,
           2,   (Delta, psi),          x.dat,
           3,   (Delta, psi),       plotting,
           4,   d/db,             b^(z,f,p),
           5,   d/da,     angle of incidence,
           6,   d/dE,                 energy.
```

where:

1, requests the program to calculate the dielectric function for an effective medium that is specified later. The format is suitable for later graphics.

2, requests the program to calculate (Δ, ψ) for a grid of (λ, ϕ). The format is suitable for use in file X.DAT.

3, requests the program to calculate (Δ, ψ) for a grid of (λ, ϕ). The format is suitable for later graphics.

4, requests the program to calculate partial derivatives of (Δ, ψ) with respect to one of the model parameters, i.e., thickness, volume fraction, or supplementary parameter. Since volume fractions involve a linear constraint, the partial derivative with

respect to $\hat{f}_j$ is calculated according to eq (10), where $(k < j)$. Again, if the selection involves volume fractions, two volume fractions must be selected to undergo variation; that which is associated with $\hat{f}_j$, as given by eq (10), is written to the file X.PLOT. The model parameter is selected by setting the (froz/vary) switch to '1' in X.DAT. The unit of measure is degree per unit of the 'vary' model parameter.

5, requests the program to calculate partial derivatives of (Δ, ψ) with respect to the angle of incidence, ϕ. The measure is unitless, i.e., degree per degree.

6, requests the program to calculate partial derivatives of (Δ, ψ) with respect to the corresponding energy of light. The unit measure is degree per electron-volt.

If the dielectric function of option '1' is selected from the above list, then the associated effective medium must be specified next.

```
Enter:   kmix ' effective medium of dielectric function
```

where *'kmix'* is the integer index label of the particular effective medium.

Finally, regarding the grid of source variables (λ, ϕ), the grid is established by specifying for each axis of the grid the lower bound, the upper bound, and the stepsize.

The first axis involves the grid of optical frequencies, wavelengths in vacuum, or associated energies.

```
Enter:    range of incident energies (eV)
  or:                  - wavelengths (nm)
Enter:    ev1, ev2, ev3
```

where *ev1* is the lower bound, *ev2* is the upper bound, and *ev3* is the stepsize. Negative values are associated with wavelength in units of nanometers, and positive values are associated with frequency as expressed by energy in units of electron-volts.

The second axis involves the grid of angles of incidence.

```
Enter:    range of incident angles (degrees)
Enter:    angle1, angle2, angle3
```

where *angle1* is the lower bound, *angle2* is the upper bound, and *angle3* is the stepsize. Numerical values are positive and are in units of degrees.

3.2 Search (vary)

This option lets the program invert the ellipsometric equations by minimizing the *error* expression as an unconstrained optimization problem, i.e., option '2' at level one. No further options are required here. At least one model parameter must be set to undergo variation in file X.DAT for each set of measurement data being analyzed. Regarding the 'vary' model parameters, the stepsizes of the Newton steps are scaled by the numerical values of the uncertainties.

3.3 Search Grid (vary)

This option lets the program invert the ellipsometric equations by minimizing the *error* expression as an constrained optimization problem, i.e., option '3' at level one. This involves scanning a grid of model parameters; the grid is scanned from a set of nested DO-loops; the grid is established as a direct product of its axes. Each axis involves one distinct model parameter and is characterized by its lower bound, upper bound, and stepsize. The program issues the following prompt-request:

```
Scan a grid of model parameters:  (z,p,f).
Grid info:   DO-loop parameters
Grid info:    i,        initial,      final,     increment
```

For each model parameter selected to undergo variation, the format convention for specifying the range of the DO-loop to characterize the axis of the grid is of the form:

$$i_p,\ p_1,\ p_2,\ p_3,$$

where i_p refers to the local integer index label of the model parameter p, and p_j refers to the numerical value of the model parameter that is associated with the initial value, final value, and stepsize, as appropriate. The model parameters are entered in the order of their placement in the input file X.DAT. Regarding the grid of volume fractions for a given effective medium, the grid specification *skips* the one with the smallest index label.

For each point of the multidimensional grid of model parameters, the program evaluates the *error* expression |g|. The program may then move to the next grid point, i.e., no optimization, or else use Newton steps to minimize |g|, i.e., full optimization. Accordingly, the program prompts the following list of options:

```
Enter:   option  regarding the grid scan:
            0,     no optimization, |g| only,
            1,   full optimization, Jacobian.
```

The format convention for indicating the type of optimization is of the form:

$$i_o$$

where i_o is an integer with value 0 or 1, accordingly.

4. Constituent Media

The effective medium of a ambient/layer/substrate region is modeled as a mixture composed of distinct constituent media. A constituent medium is characterized by its optical properties, i.e., complex dielectric function as a function of optical frequency. These dielectric functions of the constituent media are known a priori, and serve as basis functions in expressing the effective media. The following is the current indexed listing of constituent media.

1) vacuum

2) air

3) Si (crystalline)

4) Si (amorphous)

5) SiO_2 (amorphous)

6) Si_3N_4 (noncrystalline)

7) Ge (crystalline)

8) GaAs (crystalline)

9) $Al_xGa_{1-x}As$ (crystalline)

10) Oxides of GaAs (amorphous)

11) As (amorphous)

12) GaP (crystalline)

13) GaSb (crystalline)

14) InAs (crystalline)

15) InP (crystalline)

16) InSb (crystalline)

17) AlSb (crystalline)

5. Example, SiO_2 on Si

This section presents an example of input files, X.DAT and X.INN, for a case of modeling a flat sample of silicon with a thermally grown oxide layer. For convenience, let the layered structure of the sample involve two layers and a substrate. Let the top layer be amorphous SiO_2 and be 100 nm thick; let the transition region be a 50-50 mixture of amorphous SiO_2 and crystalline Si and be 2 nm thick, and let the substrate be crystalline Si. An example of the input file X.DAT is the following:

```
drb1:[data_bases]w.
-------------------------------------------------
2                                ! mfilmz ' thicknesses /(i,z,zu,ivary)
  1    100.0    2.0    0        !    i,z,zu,ivary '    top layer, SiO2
  2      2.0    2.0    1        !    i,z,zu,ivary ' bottom layer, SiO2+Si
-------------------------------------------------
0                                ! mipars / (i,ip)
-------------------------------------------------
0                                ! mrpars / (i,rp,up,ivary)
-------------------------------------------------
4                                ! mixtures ' number of effective media
1  0  0                          !    mlmnt,mipar,mrpar       #1
   1  2  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ' air
1  0  0                          !    mlmnt,mipar,mrpar       #2
   1  3  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ' Si
1  0  0                          !    mlmnt,mipar,mrpar       #3
   1  5  1.0  0.0   0            !       j,lmnt,frac,ufrac,ivary  ' SiO2
2  0  0                          !    mlmnt,mipar,mrpar       #4
   1  3  0.5  0.02  1            !       j,lmnt,frac,ufrac,ivary  ' Si
   2  5  0.5  0.02  1            !       j,lmnt,frac,ufrac,ivary  ' SiO2
```

```
21 ------------------------------------------------
22 1                              ! mbient ' number of ambients
23    1  1  0  0                  !    j,imix,mipar,mrpar          ' air
24 ------------------------------------------------
25 1                              ! msampl   ' number of samples analyzed
26 2                              !    mfilm ' number of layers on sample #1
27    1  3  1                     !       j,imix,iz     ' SiO2    , top layer
28    2  4  2                     !       j,imix,iz     ' SiO2+Si, transition region
29    3  2                        !       j,imix        ' Si      , substrate
30 ------------------------------------------------
31 1                              ! mbien ' number of ambients on sample #1
32 1  1                           !    mrpeat,imbien
33   10                           !       mexpt ' number of measurements
34    1    1.50000E+00  7.00000E+01  8.04309E+01  3.08875E+01  (i,E,a,  d,p)
35         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
36    2    2.00000E+00  7.00000E+01  7.97129E+01  4.31275E+01  (i,E,a,  d,p)
37         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
38    3    2.50000E+00  7.00000E+01  1.02227E+02  6.93018E+01  (i,E,a,  d,p)
39         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
40    4    3.00000E+00  7.00000E+01  2.50415E+02  6.09989E+01  (i,E,a,  d,p)
41         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
42    5    3.50000E+00  7.00000E+01  2.62427E+02  4.00376E+01  (i,E,a,  d,p)
43         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
44    6    4.00000E+00  7.00000E+01  2.47313E+02  3.31054E+01  (i,E,a,  d,p)
45         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
46    7    4.50000E+00  7.00000E+01  1.92800E+02  3.34557E+01  (i,E,a,  d,p)
47         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
48    8    5.00000E+00  7.00000E+01  1.24499E+02  3.21867E+01  (i,E,a,  d,p)
49         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
50    9    5.50000E+00  7.00000E+01  8.77637E+01  3.66717E+01  (i,E,a,  d,p)
51         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
52   10    6.00000E+00  7.00000E+01  7.10687E+01  4.18238E+01  (i,E,a,  d,p)
53         1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
```

where the lines have been indexed or numbered for convenience and readability; the input file should *not* contain such indexing.

Line 1 specifies the location of the database files for the constituent media. Lines 3 to 5 refer to the distinct layer thicknesses. Line 7 refers to the integer supplementary parameters. Line 9 refers to the floating-point supplementary parameters. Lines 11 to 20 specify the effective media in terms of the constituent media. Lines 22 and 23 refer to the ambient and external parameters. Lines 25 to 29 specify the layered structure of the sample. Lines 31 to 53 refer to the collected set of simulated measurement data.

This example shows that three model parameters are selected to undergo variation, a thickness and two volume fractions; see lines 5, 19, and 20, respectively; the (froz/vary) switches are set to '1.' Hence, their numerical values will undergo variation if so requested from the

command options in file X.INN. These three model parameters are associated with the transition region; see line 28.

Regarding the selection of command options in input file X.INN, suppose now that a grid search is to be used to find the *best* solution for this set of *vary* model parameters. For convenience, let the thickness range from 1.0 to 3.0 nm in steps of 2.0 nm. Regarding volume fractions (f_i), the volume fraction with the smallest index label value of the effective medium is *not* entered in the specification of the grid. From lines 19 and 20, note that the volume fraction for crystalline silicon (f_{Si}) is listed *before* the volume fraction for amorphous SiO_2 (f_{SiO_2}); thus no grid information is entered for f_{Si}. Let f_{SiO_2} range from 0.3 to 0.7 in steps of 0.4. Lastly, let the grid search use Newton steps in the calculations. The input file X.INN would then be as follows:

```
3                                   ! search grid
2    1.0     3.0     2.0            ! thickness                  transition region
2    0.3     0.7     0.4            ! volume fraction, oxide, transition region
1                                   ! full optimization, use the Jacobian
```

The output file X.OUT journals the activity of the program. The output file X.PLOT would contain the deviations between the measurement and the model.

6. Example, $Al_xGa_{1-x}As$ on GaAs

This section presents an example of using the supplementary parameters. Consider a flat sample of GaAs with a layer of $Al_xGa_{1-x}As$. Let the layered structure involve only the layer and the substrate. Let the layer be 50 nm thick, and let ($x = 0.3$) specify the mole fraction of aluminum. An example of the input file X.DAT is the following:

```
 1  drb1:[data_bases]w.
 2  ------------------------------------------------
 3  1                               ! mfilmz / (j,z,zu,ivary)
 4     1   50.   1.0    1           !   thickness, top layer
 5  ------------------------------------------------
 6  1                               ! mipars / (i,ip)
 7     1    9                       !   lmnt ~ 9,   constituent ~ AlGaAs
 8  ------------------------------------------------
 9  1                               ! mrpars / (i,rp,up,ivary)
10     1   0.3   0.01   1           !    stoichiometry ~ x,    Al(x)Ga(1-x)As
11  ------------------------------------------------
12  3                               ! mixtures ~ number of effective media
13  1  0  0                        !    mlmnt,mipar,mrpar       #1
```

```
14      1  2  1.0   0.0    0          !       j,lmnt,frac,ufrac,ivary  ' air
15  1   0  0                          !    mlmnt,mipar,mrpar       #2
16      1  8  1.0   0.0    0          !       j,lmnt,frac,ufrac,ivary  ' GaAs
17  1   1  1                          !    mlmnt,mipar,mrpar       #3
18      1  9  1.0   0.0    0          !       j,lmnt,frac,ufrac,ivary  ' AlGaAs
19      1  1                          !       i, iip
20      1  1                          !       i, irp
21  ------------------------------------------------
22  1                                 ! mbient ' number of distinct ambients
23  1   1   0   0                     !    j,imix,mipar,mrpar          ' air
24  ------------------------------------------------
25  1                                 ! msampl ' number of samples
26  1                                 !    mfilm ' number of layers on sample #1
27  1   3   1                         !       j,imix,iz                ' AlGaAs
28  2   2                             !       j,imix                   ' GaAs substrate
29  ------------------------------------------------
30  1                                 ! mbien ' number of ambients for sample #1
31  1   1                             !    mrpeat,imbien
32      12                            !       mexpt ' number of measurements
33       1   1.50000E+00  6.50000E+01  1.76446E+02  1.24387E+01  (i,E,a,  d,p)
34           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
35       2   2.00000E+00  6.50000E+01  1.79401E+02  1.57825E+01  (i,E,a,  d,p)
36           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
37       3   2.50000E+00  6.50000E+01  1.77396E+02  1.95307E+01  (i,E,a,  d,p)
38           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
39       4   3.00000E+00  6.50000E+01  1.66208E+02  2.33677E+01  (i,E,a,  d,p)
40           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
41       5   3.50000E+00  6.50000E+01  1.49379E+02  2.43128E+01  (i,E,a,  d,p)
42           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
43       6   4.00000E+00  6.50000E+01  1.48705E+02  2.35367E+01  (i,E,a,  d,p)
44           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
45       7   1.50000E+00  7.00000E+01  1.69646E+02  4.65575E+00  (i,E,a,  d,p)
46           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
47       8   2.00000E+00  7.00000E+01  1.78701E+02  8.31343E+00  (i,E,a,  d,p)
48           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
49       9   2.50000E+00  7.00000E+01  1.75534E+02  1.26304E+01  (i,E,a,  d,p)
50           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
51      10   3.00000E+00  7.00000E+01  1.58723E+02  1.75083E+01  (i,E,a,  d,p)
52           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
53      11   3.50000E+00  7.00000E+01  1.35016E+02  1.96919E+01  (i,E,a,  d,p)
54           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
55      12   4.00000E+00  7.00000E+01  1.33538E+02  1.88574E+01  (i,E,a,  d,p)
56           1.00000E-01  1.00000E-02  1.00000E-02  1.00000E-02  (uncertainty)
```

where the lines have been indexed for convenience and readability; the input file should *not* contain such indexing.

Lines 3 and 4 refer to the thickness of the layer. Lines 6 and 7 refer to the integer supplementary parameters; the integer parameter is used to specify a distinct constituent medium. Lines 9 and 10 refer to the floating-point supplementary parameters; the floating-

point parameter is used to specify the mole fraction of aluminum. Lines 19 and 20 are used to provide the necessary correspondence between the effective medium and the supplementary parameters. Two model parameters are shown to be selected for optimization, the layer thickness and the mole fraction of aluminum; two (froz/vary) switches are set to one; see lines 4 and 10. Lines 30 to 56 refer to simulated measurement data.

Regarding command options, suppose that a grid search is used to find the *best* solution. Let the thickness range from 40 to 60 nm in steps of 2 nm. Let the aluminum mole fraction range from 0.2 to 0.4 in steps of 0.02. Since the axes of the grid are specified according to the order that the model parameters are presented in the input file X.DAT, the specification for layer thickness precedes that for the mole fraction of aluminum; see lines 4 and 10 of file X.DAT. Let the grid search evaluate |g| only at the grid points, i.e., no Newton steps. The input file X.INN would be as follows:

```
3                                   ! search grid
1   40.0    60.0    2.0             ! thickness, top layer, Al(x)Ga(1-x)As
1    0.2     0.4    0.02            ! stoichiometry,    x = Al
0                                   ! no optimization, grid points only
```


☆ U.S. GOVERNMENT PRINTING OFFICE: 1990 — 2 6 1 - 9 1 3 20593

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NIST/SP-400/84

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
April 1990

4. TITLE AND SUBTITLE
Semiconductor Measurement Technology: A Software Program for Aiding the Analysis of Ellipsometric Measurements, Simple Spectroscopic Models

5. AUTHOR(S)
J. F. Marchiando

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)
U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED
Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)
Same as Item #6

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

MAIN2 is a software program for analyzing spectroscopic ellipsometric measurements. MAIN2 consists mainly of subroutines written in FORTRAN that are used to invert the standard reflection ellipsometry equations for simple systems. Here, a system is said to be simple if the solid material sample is characterized by models which assume at least the following: (1) materials are nonmagnetic; (2) samples exhibit depth-dependent optical properties, such as one with layered or laminar structure atop a substrate that behaves like a semi-infinite half-space; (3) layers are flat and of uniform thickness; and (4) the optical medium within each ambient/layer/substrate is isotropic, homogeneous, local, and linear. The ambient region refers to that region of space which lies external to the layer/substrate structure of the sample. Usually, the ambient region involves a medium of air or vacuum. Each layer is characterized by a thickness and a dielectric function. The dielectric function of a region, i.e., ambient, layer, or substrate, is represented by the Bruggeman effective medium approximation (EMA). Within the EMA, the effective medium of a region is characterized by an aggregate mixture of constituent media, and the dielectric function of each constituent medium is known a priori. The constituent dielectric functions are taken from the literature. The ellipsometric equations are formulated as a standard damped nonlinear least-squares problem and then solved by an iterative method when possible. The program is sufficiently modular to allow one to modify some of the models used in the calculations.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)
ellipsometry; EMA; FORTRAN; modeling; software; spectroscopic models

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[X] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
362

15. PRICE

ELECTRONIC FORM

NIST Technical Publications

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NIST by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW., Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NIST research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NIST publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency Reports (NISTIR)—A special series of interim or final reports on work performed by NIST for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Institute of Standards and Technology
(formerly National Bureau of Standards)
Gaithersburg, MD 20899