Semiconductor Measurement Technology:

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

J. F. Marchiondo
The National Institute of Standards and Technology was established by an act of Congress on March 3, 1901. The Institute's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Institute conducts research to assure international competitiveness and leadership of U.S. industry, science and technology. NIST work involves development and transfer of measurements, standards and related science and technology, in support of continually improving U.S. productivity, product quality and reliability, innovation and underlying science and engineering. The Institute's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the National Computer Systems Laboratory, and the Institute for Materials Science and Engineering.

**The National Measurement Laboratory**

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; provides calibration services; and manages the National Standard Reference Data System. The Laboratory consists of the following centers:

<table>
<thead>
<tr>
<th>Center</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Basic Standards(^2)</td>
</tr>
<tr>
<td></td>
<td>Radiation Research</td>
</tr>
<tr>
<td></td>
<td>Chemical Physics</td>
</tr>
<tr>
<td></td>
<td>Analytical Chemistry</td>
</tr>
</tbody>
</table>

**The National Engineering Laboratory**

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

<table>
<thead>
<tr>
<th>Center</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Computing and Applied Mathematics</td>
</tr>
<tr>
<td></td>
<td>Electronics and Electrical Engineering(^3)</td>
</tr>
<tr>
<td></td>
<td>Manufacturing Engineering</td>
</tr>
<tr>
<td></td>
<td>Building Technology</td>
</tr>
<tr>
<td></td>
<td>Fire Research</td>
</tr>
<tr>
<td></td>
<td>Chemical Engineering</td>
</tr>
</tbody>
</table>

**The National Computer Systems Laboratory**

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Laboratory consists of the following divisions:

<table>
<thead>
<tr>
<th>Division</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information Systems Engineering</td>
</tr>
<tr>
<td></td>
<td>Systems and Software Technology</td>
</tr>
<tr>
<td></td>
<td>Computer Security</td>
</tr>
<tr>
<td></td>
<td>Systems and Network Architecture</td>
</tr>
<tr>
<td></td>
<td>Advanced Systems</td>
</tr>
</tbody>
</table>

**The Institute for Materials Science and Engineering**

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-cutting scientific themes such as nondestructive evaluation and phase diagram development; oversees Institute-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Institute consists of the following divisions:

<table>
<thead>
<tr>
<th>Division</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ceramics</td>
</tr>
<tr>
<td></td>
<td>Fracture and Deformation(^3)</td>
</tr>
<tr>
<td></td>
<td>Polymers</td>
</tr>
<tr>
<td></td>
<td>Metallurgy</td>
</tr>
<tr>
<td></td>
<td>Reactor Radiation</td>
</tr>
</tbody>
</table>

\(^1\)Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.

\(^2\)Some divisions within the center are located at Boulder, CO 80303.

\(^3\)Located at Boulder, CO, with some elements at Gaithersburg, MD.
Semiconductor Measurement Technology:

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

J. F. Marchiando

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

July 1989

NOTE: As of 23 August 1988, the National Bureau of Standards (NBS) became the National Institute of Standards and Technology (NIST) when President Reagan signed into law the Omnibus Trade and Competitiveness Act.
Table of Contents

Abstract .................................................. 1

1. Introduction ........................................... 2

2. The Forward Problem ................................... 5
   2.1 Transverse Electric Mode or S-Polarization .......... 7
      2.1.1 Reflection Coefficient .......................... 7
      2.1.2 Partial Derivatives ............................. 11
   2.2 Transverse Magnetic Mode or P-Polarization ........ 15
      2.2.1 Reflection Coefficient .......................... 15
      2.2.2 Partial Derivatives ............................. 17
   2.3 Ellipsometric Angles (Δ, ψ) ......................... 20

3. The Inverse Problem .................................... 23
   3.1 Formulation of the Least-Squares Problem ........... 23
   3.2 Sensitivity Analysis of Model Parameters ........... 26

   4.1 Software Development Considerations ................. 32
      4.1.1 Input/Output Files ............................... 32
      4.1.2 Allocations of Array or Work Space ............... 33
      4.1.3 Library Software ................................. 34
   4.2 Input Data Requirements .............................. 35
      4.2.1 Parameters ......................................... 35
         4.2.1.1 Wavelengths ................................... 35
         4.2.1.2 Optical Media ................................ 37
         4.2.1.3 Layer Thickness ............................... 40
      4.2.2 Sample Characterization or Construction .......... 41
      4.2.3 Measurement Data (Δ, ψ) ......................... 48
      4.2.4 Combining ........................................... 50
   4.3 Command Options ..................................... 51
      4.3.1 Forward Problems, Plots, ... .................... 53
      4.3.2 Search (vary) ...................................... 56
      4.3.3 Search Grid (vary) ................................. 58
      4.3.4 Search Grid (froz,vary) ........................... 59
### 4.3.5 Sensitivity Analysis

- Page 60

### 4.3.6 MAE Plots of Uncertainties

- Page 61

## 5 Worked Examples

- Page 63
  - 5.1 Search (vary) ........................................... 63
  - 5.2 Search Grid (vary) ..................................... 68
  - 5.3 Search Grid (froz, vary) ................................. 71
  - 5.4 Sensitivity Analysis ..................................... 75

## 6. Listing of Software Source Files and Routines

- Page 79
  - 6.1 Named COMMON and BLOCK DATA Statements ........ 79
    - 6.1.1 IUNIT ............................................. 79
    - 6.1.2 DEFNIT .......................................... 80
    - 6.1.3 FILM/MM ......................................... 81
    - 6.1.4 FILMSS ........................................... 82
    - 6.1.5 RSTACK .......................................... 83
    - 6.1.6 WSTACK .......................................... 84
    - 6.1.7 CGNXX1 .......................................... 85
    - 6.1.8 SCANCC .......................................... 86
    - 6.1.9 SEAMX1 .......................................... 87
    - 6.1.10 SEAMX2 ......................................... 88
    - 6.1.11 BLKDAT.FOR .................................... 89
  - 6.2 Source Programs ....................................... 90
    - 6.2.1 MAIN.FOR ......................................... 90
    - 6.2.2 FILEOP.FOR ...................................... 92
    - 6.2.3 INPDAT.FOR ...................................... 93
    - 6.2.4 ARRANG.FOR ...................................... 103
    - 6.2.5 PLTDAT.FOR ...................................... 105
    - 6.2.6 SCAN2.FOR ....................................... 115
    - 6.2.7 SCAN2G.FOR ..................................... 120
    - 6.2.8 SCAN3.FOR ....................................... 126
    - 6.2.9 ZOOM.FOR ........................................ 142
    - 6.2.10 ZOOM2.FOR ...................................... 145
    - 6.2.11 ZOOM3.FOR ...................................... 148
    - 6.2.12 ASMVL.FOR ....................................... 151
<table>
<thead>
<tr>
<th>Section</th>
<th>File</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.13</td>
<td>ASMBL3.FOR</td>
<td>155</td>
</tr>
<tr>
<td>6.2.14</td>
<td>ASMBLX.FOR</td>
<td>159</td>
</tr>
<tr>
<td>6.2.15</td>
<td>SCATR.FOR</td>
<td>165</td>
</tr>
<tr>
<td>6.2.16</td>
<td>FORWRD.FOR</td>
<td>167</td>
</tr>
<tr>
<td>6.2.17</td>
<td>STAT22.FOR</td>
<td>171</td>
</tr>
<tr>
<td>6.2.18</td>
<td>CORLAT.FOR</td>
<td>173</td>
</tr>
<tr>
<td>6.2.19</td>
<td>SEAMA.FOR</td>
<td>177</td>
</tr>
<tr>
<td>6.2.20</td>
<td>SEAMAX.FOR</td>
<td>183</td>
</tr>
<tr>
<td>6.2.21</td>
<td>SEAM2.FOR</td>
<td>193</td>
</tr>
<tr>
<td>6.2.22</td>
<td>SEAM3.FOR</td>
<td>197</td>
</tr>
<tr>
<td>6.2.23</td>
<td>POPLAT.FOR</td>
<td>202</td>
</tr>
<tr>
<td>6.2.24</td>
<td>PLTE.FOR</td>
<td>207</td>
</tr>
</tbody>
</table>

### 6.3 General Utilities

<table>
<thead>
<tr>
<th>Section</th>
<th>File</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.1</td>
<td>DOT.FOR</td>
<td>215</td>
</tr>
<tr>
<td>6.3.2</td>
<td>NORM.FOR</td>
<td>216</td>
</tr>
<tr>
<td>6.3.3</td>
<td>APROD.FOR</td>
<td>217</td>
</tr>
<tr>
<td>6.3.4</td>
<td>SCALII.FOR</td>
<td>218</td>
</tr>
<tr>
<td>6.3.5</td>
<td>SCALJJ.FOR</td>
<td>220</td>
</tr>
<tr>
<td>6.3.6</td>
<td>CGNL.FOR</td>
<td>222</td>
</tr>
<tr>
<td>6.3.7</td>
<td>LSQR.FOR</td>
<td>224</td>
</tr>
<tr>
<td>6.3.8</td>
<td>SQRRT.FOR</td>
<td>236</td>
</tr>
<tr>
<td>6.3.9</td>
<td>POLAR.FOR</td>
<td>237</td>
</tr>
<tr>
<td>6.3.10</td>
<td>IINDEX.FOR</td>
<td>239</td>
</tr>
<tr>
<td>6.3.11</td>
<td>ISTIME.FOR</td>
<td>240</td>
</tr>
<tr>
<td>6.3.12</td>
<td>TIMES.FOR</td>
<td>241</td>
</tr>
</tbody>
</table>

### 7. Acknowledgments

<table>
<thead>
<tr>
<th>Section</th>
<th>File</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>243</td>
</tr>
</tbody>
</table>

### 8. References

<table>
<thead>
<tr>
<th>Section</th>
<th>File</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>244</td>
</tr>
</tbody>
</table>
MAIN1 is a software program for aiding the analysis of ellipsometric measurements. MAIN1 consists of a suite of routines written in FORTRAN that are used to invert the standard reflection ellipsometric equations for simple systems. Here, a system is said to be simple if the solid material sample may be adequately 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 dielectric function within each layer/substrate is isotropic, homogeneous, local, and linear. Each layer is characterized in part by a thickness \( z \), while the optical properties for a given material and wavelength are expressed in terms of a refractive index \( n \) and extinction coefficient \( k \). The ellipsometric equations are formulated as a standard damped nonlinear least-squares problem and then solved by an iterative method when possible. Estimates of the uncertainties associated with assigning numerical values to the model parameters are calculated as well.

Key words: ellipsometry; FORTRAN; measurement; modeling; program; sensitivity; software; uncertainty.
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. The fundamental problem is then to understand those properties which characterize the material medium and are able to induce this measured change in polarization. MAIN1 is a program for aiding the analysis of ellipsometric measurements. MAIN1 consists of a suite of routines written in FORTRAN* that are used to invert a standard set of reflection ellipsometry equations. The systems under consideration here involve those containing flat, thin, solid films atop a substrate. The program has been used in modeling systems as basic as those involving thermally grown oxide films atop silicon and as complicated as those involving SIMOX (Separation by IMplanted OXy-gen).

A system is said to be simple if the solid material sample may be adequately 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 dielectric function within each layer/substrate is isotropic, homogeneous, local, and linear. Such approximations are well known and documented [1–7].

In general, the model assumes three kinds or types of parameters: each layer is characterized in part by a thickness ($z$), while the optical properties of the media (ambient, layers, substrate) are expressed in terms of a scalar refractive index ($n$) and extinction coefficient ($k$) at some given wavelength of light. With only three model parameters to characterize each layer, a sample may be readily assembled layer by layer atop the

* 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.
substrate. Since the natural parameters of the formulation include the thickness and dielectric function, i.e., the model parameters themselves, it is straightforward to analyze distinct samples collectively. Thus, the program is said to have a multiple-sample capability. Due to the method of indexing the model parameters used in the program, it is possible for two distinct layers (media, thickness) to have some model parameter in common, e.g., distinct media but common thickness or vice versa. Also, it is possible for two distinct samples to have model parameters in common, e.g., common layer (media, thickness). Such cases may arise when analyzing samples that differ in thickness resulting from some etch-back process. Thus, the program is said to allow coupling between samples and/or layers.

Since the model parameters contain the optical properties themselves, it is important to note that they involve no further dependences. No provision is made to incorporate distinct physical models into the expression of the dielectric function, e.g., the effective medium approximation.

The phenomenological models and methods of solution are classical. The ellipsometric equations are formulated as a standard nonlinear, damped least-squares problem which is solved by an iterative method appropriate for linear systems. Following the convergence of the iterations to a good solution, i.e., within the resolution of the measurement by the instrumentation, simple estimates of the uncertainties associated with ascribing numerical values to the model parameters are calculated as well.

Regarding the uniqueness of the calculated solution which is found by minimizing a residual in the least-squares sense, it is important to realize that multiple or pseudo solutions may exist, as well as correlation among the modeling parameters. Consequently, while the program does serve as a useful tool in finding solutions, the responsibility of assessing the consistency and appropriateness of the results remains entirely dependent on the user. The collection of modules or subroutines is designed to run in batch mode, as opposed to interactive mode, and is suitable for calculations involving only the physical models mentioned earlier.

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 does refer to the NCAR software graphics package [27,28]. This is mentioned briefly in section 4.1.3. The section also discusses the use of the LINPACK library [17], which is public domain software.

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. The model describing the scattering of light is presented in section 2. The methods of inverting the ellipsometric equations and estimating the uncertainties associated with knowing the modeling parameters are discussed in section 3. Section 4 presents the manual of operations for the software package. This includes an overview of considerations important in using the software package. A brief outline of the internal organization of the subroutines and calling sequences is provided for orientational purposes. This outline includes a short listing of the parameters which specify upper bounds for the necessary allocations of space for the work-storage 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 in exercising operational control over the program, i.e., constraints of procedures during analyses. A selected set of examples involving sample data input and program output is presented in section 5. The listing of the software source files and routines are presented in section 6.

The source code is written predominantly in standard FORTRAN-77. Minor compiler extensions are associated with the DO-loop constructions and formats for OPEN/CLOSE statements; such are appropriate for a VAX computer. The source code contains some in-line comments for the user's convenience. The more expert user may want to modify 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. The program is available upon request by sending the author a floppy disk or a small nine-track magnetic tape.
2. The Forward Problem

Any discussion regarding the description of the scattering of light from a medium must ultimately incorporate some formulation involving Maxwell's equations [1-7]. Here, a brief discussion is presented regarding 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. Within those regions or neighborhoods where the optical properties of the material medium are continuous, isotropic, linear, local, and absent of free charges and surface currents, Maxwell's equations in centimeter-gram-second (cgs) units are

\[
\nabla \times \mathbf{E} = -\frac{1}{c} \dot{\mathbf{B}} = \frac{i\omega}{c} \mu \mathbf{H},
\]

\[
\nabla \times \mathbf{H} = \frac{1}{c} \dot{\mathbf{D}} + \frac{4\pi}{c} \mathbf{J} = -\frac{i\omega}{c} \left( \varepsilon + i\frac{4\pi\sigma}{\omega} \right) \mathbf{E} = -\frac{i\omega}{c} \varepsilon \mathbf{E},\]

\[
\nabla \cdot \mathbf{D} = 0,
\]

\[
\nabla \cdot \mathbf{B} = 0,
\]

with the constitutive relationships

\[
\mathbf{B} = \mu \mathbf{H}, \quad \mathbf{D} = \varepsilon \mathbf{E}, \quad \mathbf{J} = \sigma \mathbf{E},
\]

and the complex dielectric function \( \varepsilon \) is defined by

\[
\varepsilon = \varepsilon + i\frac{4\pi\sigma}{\omega} = (n + ik)^2,
\]

where \( \mathbf{E} \) is the electric field vector, \( \mathbf{H} \) is the magnetic field vector, \( \mathbf{D} \) is the electric displacement, \( \mathbf{B} \) is the magnetic induction, \( \mathbf{J} \) is the induced conduction current, \( \mu \) is the magnetic permeability, \( \sigma \) is the specific or optical conductivity, \( \varepsilon \) is the dielectric
permittivity, \( n \) is the index of refraction, \( k \) is the extinction coefficient, \( \omega \) is the angular frequency of the incident light, \( c \) is the vacuum speed of light, and where the dot denotes differentiation with respect to time, and the time dependence is assumed to be of the form \( e^{-i\omega t} \), which is distinct from the engineering or Nebraska convention that assumes \( e^{+i\omega t} \) and \( \varepsilon = (n - ik)^2 \).

For convenience, 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. Thus, 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 \((z < 0)\). For this configuration of geometry and restricted class of problems, the optical properties are assumed to be of the form

\[
\begin{align*}
\varepsilon &= \varepsilon(\omega, z), & \sigma &= \sigma(\omega, z), & \mu &= \mu(\omega, z) = 1, & z > 0, \\
\varepsilon &= \varepsilon_0(\omega), & \sigma &= 0, & \mu &= 1, & z < 0,
\end{align*}
\]

where allowance is made for dispersive nonmagnetic materials. Then, it is convenient to express eqs (1) and (2) in Cartesian coordinates, i.e.,

\[
\begin{align*}
\nabla_y E_x - \nabla_x E_y &= i \left( \frac{\omega}{c} \right) \mu H_x, \\
\nabla_z E_x - \nabla_x E_z &= i \left( \frac{\omega}{c} \right) \mu H_y, \\
\nabla_z E_y - \nabla_y E_z &= i \left( \frac{\omega}{c} \right) \mu H_z, \\
\nabla_y H_x - \nabla_x H_y &= -i \left( \frac{\varepsilon}{c} \right) \varepsilon E_x, \\
\nabla_z H_x - \nabla_x H_z &= -i \left( \frac{\varepsilon}{c} \right) \varepsilon E_y, \\
\nabla_z H_y - \nabla_y H_z &= -i \left( \frac{\varepsilon}{c} \right) \varepsilon E_z.
\end{align*}
\]

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 \(\phi\) with respect to the surface normal. The plane of incidence is taken to be the \((z, x)\)
plane, and the direction cosines of propagation both being positive. Any arbitrarily polarized wave may be resolved into two waves. With this geometry, it is natural to choose the two waves to be the orthogonal set involving linearly polarized waves: one which maintains its electric vector polarized perpendicular to the plane of incidence, i.e., transverse electric, TE, or s-polarization; and the other which maintains its magnetic vector perpendicular to the plane of incidence, i.e., transverse magnetic, TM, or p-polarization. This set uncouples the system of Maxwell’s equations, and each polarization (s,p) may be treated individually.

The next problem under consideration in modeling the medium is to determine or otherwise calculate the resulting reflected and/or refracted fields which may be subjected to measurement with an ellipsometer. In the following subsections, further discussion is presented regarding the calculation of the reflected and/or refracted waves for each polarization. Here, the problem for each polarization involves a second-order ordinary differential equation (ODE) complete with boundary conditions.

It is convenient to recognize that both ODEs admit plane wave solutions if the dielectric function within each layer is a constant. The profile of the stratified structure of the dielectric function is approximated by a stepped profile of isotropic, homogeneous, uniformly thick layers or laminae. Then, the problem of solving Maxwell’s equations is reduced to one of matching analytic solutions at boundaries between adjacent layers. This involves solving a set of recurrence relations and is the form of the forward problem that is actually solved here. The Jacobian is determined in a like manner. Finally, regarding the quantities that are measured by the ellipsometer and characterize the phase shift induced upon reflection, the ellipsometric angles (Δ, ψ) are defined in terms of the reflected fields.

2.1 Transverse Electric Mode or S-Polarization

2.1.1 Reflection Coefficient

Due to the uncoupling within the system of Maxwell’s equations, there is one solution or polarization where the incident, refracted, and reflected fields may be characterized by electric-field vectors which are aligned in a direction perpendicular to the plane of incidence, i.e., along the y-direction: \( E_z = E_x = 0 \) and \( E_y \neq 0 \). This is called the transverse electric mode or s-polarization. From eq (7), \( H_y = 0 \); from eq (11), \( H_z = H_z(z, x) \);
and from eq (9), $H_z = H_z(z, x)$; so there are no dependences on $y$. Substituting eqs (6) and (8) into eq (10) yields

$$\nabla_z \left( -\frac{c}{i\mu} \nabla_z E_y \right) - \nabla_z \left( -\frac{c}{i\mu} \nabla_z E_y \right) = -i\frac{\omega}{c} E_y$$

or

$$\left[ \nabla_z^2 + \nabla_z^2 - (\nabla_z \ln \mu) \nabla_z + \frac{\omega^2}{c^2} \mu \right] E_y = 0.$$

Assuming a solution by the method of separable variables, let $E_y = Z(z) \chi(x)$ and $k_o = \omega/c$, then

$$\frac{1}{Z} \left[ \nabla_z^2 - (\nabla_z \ln \mu) \nabla_z + k_o^2 \epsilon \mu \right] Z = -\frac{1}{\chi} \nabla_z^2 \chi = \kappa_a^2$$

where $\kappa_a$ is an arbitrary constant and real because of translation invariance within the $(x, y)$ plane. One may write

$$\kappa_a = k_o \alpha,$$

so that

$$\chi = e^{ik_o \alpha}$$

(12)

where in the ambient region,

$$\alpha = \sqrt{\epsilon_a \sin \phi} = \text{constant} \geq 0,$$

(13)

with $\epsilon_a$ being the dielectric function of the ambient region, and $\phi$ being the angle of incidence of the light, i.e., $0 \leq \phi \leq \frac{\pi}{2}$. Here, $(\phi = 0)$ involves a direction normal to the surface.

From the differential representation of Maxwell's equations across a surface of discontinuity in piecewise continuous media, the tangential fields, $E_y$ and $H_z$, are continuous across boundaries between adjacent layers. Consequently, eq (12) becomes valid within each spatial region, i.e., ambient, layers, and substrate. Then, letting

$$\eta = \sqrt{\epsilon \mu - \alpha^2}, \quad \text{where} \quad 0 \leq \arg(\eta) \leq \pi,$$

(14)

the above ODE in $z$ may be written as

$$\left[ \nabla_z^2 + (\nabla_z \ln \mu) \nabla_z + k_o^2 \eta^2 \right] Z = 0.$$

8
Within any layer of the media, the optical properties are assumed to be homogeneous, so the middle gradient term vanishes. The ODE finally becomes
\[ [\nabla_z^2 + k_0^2 \eta^2] \mathcal{Z} = 0, \] (15)
and admits solutions involving the well-known set of damped plane waves,
\[ \mathcal{Z} = E^{(+)} e^{i k_0 \eta} + E^{(-)} e^{-i k_0 \eta}. \]

From eq (6),
\[ \nabla_z E_y = -i \frac{\omega}{c} \mu H_z = -i k_0 \mu H_z, \]
on one finds
\[ -H_z = \frac{\eta}{\mu} \left[ E^{(+)} e^{i k_0 \eta} - E^{(-)} e^{-i k_0 \eta} \right] \mathcal{X}. \]

It follows from the continuity of the tangential fields that the logarithmic derivative of the solution or admittance function
\[ \mathcal{Y} = -\frac{H_z}{E_y} = \frac{\eta}{\mu} \left[ \frac{E^{(+)} e^{i k_0 \eta} - E^{(-)} e^{-i k_0 \eta}}{E^{(+)} e^{i k_0 \eta} + E^{(-)} e^{-i k_0 \eta}} \right] \]
\[ = \frac{\eta}{\mu} \left[ \frac{1 - R e^{-i 2 z k_0 \eta}}{1 + R e^{-i 2 z k_0 \eta}} \right] \]
is a continuous function of \( z \) alone, where \( R \) is the reflection coefficient local to the layer or spatial region. It also follows that coefficient \( E^{(-)} \) vanishes within the substrate region because its basis function diverges for large \( z \). This allows one to evaluate the admittance at the boundary of the substrate,
\[ \mathcal{Y}_s = \frac{\eta_s}{\mu_s} = \frac{\sqrt{\varepsilon_s \mu_s - \varepsilon_a \sin^2 \phi}}{\mu_s} \] (16)
where the subscript \( s \) denotes the substrate region. This will be called the innermost boundary condition for the admittance.

The last part of the problem is the calculation of the reflection coefficient local to the ambient region, \( R_a \), where the subscript denotes the ambient region. Since the optical properties are assumed to be known, the method of solution reduces to that of matching the boundary conditions across each layer that separates the substrate from the ambient region. A reflection coefficient local to each region needs to be determined. This
is accomplished by considering a single layer alone. For convenience, consider the layer (L) adjacent to the substrate, which has complex dielectric function $\epsilon_L$ and thickness $z_L$. Let a local coordinate system be defined on the layer and let it be oriented so that the positive $z$-axis extends from left-to-right. Let layer (L) occupy the spatial region $(0 \leq z \leq z_L)$, while the substrate is located within the unbounded region $(z > z_L)$. Then, from eq (14),

$$\eta_L = \sqrt{\epsilon_L \mu_L - \epsilon_a \sin^2 \phi}$$

and due to continuity, the boundary condition on the right-hand-side (rhs) of the layer (L) is of the form,

$$\mathcal{Y}_L(\text{rhs}) \equiv \mathcal{Y}_L(z_L) = \mathcal{Y}_s$$

$$= \frac{\eta_L}{\mu_L} \left[ \frac{1 - R_L e^{-i2zLk_0\eta_L}}{1 + R_L e^{-i2zLk_0\eta_L}} \right]$$

so that

$$R_L = e^{i2zLk_0\eta_L} \left[ \frac{(\eta_L/\mu_L) - \mathcal{Y}_L(\text{rhs})}{(\eta_L/\mu_L) + \mathcal{Y}_L(\text{rhs})} \right]$$

and

$$\mathcal{Y}_L(\text{lhs}) \equiv \mathcal{Y}_L(z = 0) = \frac{\eta_L}{\mu_L} \left[ \frac{1 - R_L}{1 + R_L} \right].$$

The dependences on the model parameters and (rhs) boundary condition may be expressed more explicitly, by letting

$$\gamma_L = e^{i2zLk_0\eta_L},$$

and substituting to find

$$\mathcal{Y}_L(\text{lhs}) = \frac{\eta_L}{\mu_L} \left[ \frac{(\eta_L/\mu_L) + \mathcal{Y}_L(\text{rhs}) - \gamma_L [(\eta_L/\mu_L) - \mathcal{Y}_L(\text{rhs})]}{(\eta_L/\mu_L) + \mathcal{Y}_L(\text{rhs}) + \gamma_L [(\eta_L/\mu_L) - \mathcal{Y}_L(\text{rhs})]} \right]$$

$$= \frac{\eta_L}{\mu_L} \left[ \frac{(1 - \gamma_L) (\eta_L/\mu_L) + (1 + \gamma_L) \mathcal{Y}_L(\text{rhs})}{(1 + \gamma_L) (\eta_L/\mu_L) + (1 - \gamma_L) \mathcal{Y}_L(\text{rhs})} \right].$$

This equation is applied recursively, matching the boundary conditions and moving toward the left across each and every layer in succession, so that the admittance function
may be evaluated at the outer surface which borders the ambient region, i.e., \( \mathcal{V}(z = 0) \).
The reflection coefficient in the ambient region, \( R_a \), is found from \( \mathcal{V}(0) \) and is given by

\[
R_a = \left[ \frac{(\eta_a/\mu_a) - \mathcal{V}(0)}{(\eta_a/\mu_a) + \mathcal{V}(0)} \right],
\]

(19)

where again, all of the \( \mu \)'s can be set equal to unity, if the material is nonmagnetic.

2.1.2 Partial Derivatives

The forward scattering problem assumes that all of the parameters which characterize the sample are known, as well as the source field which is incident upon the surface of the sample. The forward problem involves calculating the reflected and refracted fields resulting from this incident field. But reflection ellipsometry provides measurements involving only two numbers, i.e., the relative amplitude attenuation and the relative phase shift of the light that is induced upon reflection for a given angle of incidence \( \phi \) and wavelength \( \lambda \). This is discussed later in section 2.3. It is necessary to invert the ellipsometric equations and to determine best estimates to the model parameters that give rise to the reflected fields that manifest the measured phase shifts and amplitude attenuations. This is the inverse problem that is discussed later in section 3. During the inversion process that involves finding improvements to the values of the model parameters, it becomes necessary to estimate the effect that each model parameter induces in the calculated phase shifts or reflected fields. This requires partial derivatives of the fields with respect to the model parameters \([8,9]\). Collectively, these derivatives will be called the Jacobian.

Since the model parameters include refractive indices and extinction coefficients for each layer and substrate, as well as layer thicknesses, each layer is characterized by \((z, n, k)\) and substrate by \((n, k)\). Here, the subscript labeling of the layer or substrate region and the incident wavelength of light has been suppressed. For example, for a sample containing \(m_L\) layers measured by ellipsometry involving \(m_e\) wavelengths, there would generally be \(m_L + 2(m_L + 1)m_e\) model parameters. However, only three types of partial derivatives need to be considered because the reflection coefficient involves a composite function of such basic parameters as thicknesses via \((i2\pi k_o)\), complex dielectric functions \((\varepsilon = (n+ik)^2)\), and the angle of incidence via \((-\varepsilon_a \sin^2 \phi)\). Recalling eq (19) and suppressing references to the \( \mu \)'s, it follows that one may readily express the partial derivatives
of the reflection coefficient associated with these three basic parameters. They are of the following general operator form:

\[
\frac{\partial R_a}{\partial (i2\pi k_o)} \equiv R'_a = -\frac{2\eta_a \gamma'(0)}{[\eta_a + \gamma(0)]^2}
\]  

when the prime is used for convenience as an operator to imply a partial derivative with respect to \(i2\pi k_o\),

\[
\frac{\partial R_a}{\partial \varepsilon} \equiv R'_a = -\frac{2\eta_a \gamma'(0)}{[\eta_a + \gamma(0)]^2}
\]  

when the prime is used to imply a partial derivative with respect to \(\varepsilon\), and

\[
\frac{\partial R_a}{\partial (-\varepsilon_a \sin^2 \phi)} \equiv R'_a = \frac{2[\eta_a \gamma(0) - \eta_a \gamma'(0)]}{[\eta_a + \gamma(0)]^2}
\]  

when the prime is used to imply a partial derivative with respect to \((-\varepsilon_a \sin^2 \phi)\). These need to be calculated for each model parameter undergoing variation.

The derivatives of the admittance function evaluated at the surface may be understood and calculated in a straightforward manner, by considering the effect of a variation in the model parameters within the substrate region. From eq (16),

\[
\gamma_s = \eta_s = \sqrt{\varepsilon_s - \varepsilon_a \sin^2 \phi},
\]

so

\[
\gamma'_s = \frac{\partial \gamma_s}{\partial \varepsilon_s} = \frac{\partial \gamma_s}{\partial (-\varepsilon_a \sin^2 \phi)} = \frac{1}{2\eta_s} = \frac{1}{2\gamma_s}.
\]  

The effect of this derivative on the admittance must then be matched as a boundary condition across each layer that lies between the substrate and the ambient region. This gives rise to four cases to consider during the matching process, depending upon whether the partial derivative operates on a function whose parameter is exterior or interior to the local region of interest. These four cases follow directly from considering the boundary conditions satisfied by any given layer, i.e., eq (18) for the layer \((L)\) adjacent to the substrate,

\[
\gamma_L \text{(lhs)} = \eta_L \left[ \frac{(1 - \gamma_L) \eta_L + (1 + \gamma_L) \gamma_L \text{(rhs)}}{(1 + \gamma_L) \eta_L + (1 - \gamma_L) \gamma_L \text{(rhs)}} \right].
\]
Case 1 considers any partial derivative taken with respect to \((\varepsilon, z)\) that operates exterior to the local region. Here, only the \((\text{rhs})\) boundary condition is affected. Using the chain rule for partial derivatives on the above equation, it follows that

\[
\mathcal{V}_L' (\text{lhs}) = \frac{4\gamma_L \eta_L^2 \mathcal{V}_L' (\text{rhs})}{[(1 + \gamma_L) \eta_L + (1 - \gamma_L) \mathcal{V}_L (\text{rhs})]^2}.
\] (24)

This equation provides the key relationship for transporting derivatives involving matched boundary conditions across any given layer. In general, if the layers of a sample are indexed consecutively, where the top layer \((1)\) is adjacent to the ambient region, and where layer \((L)\) is not the top layer, then

\[
\mathcal{V}_{L-1} (\text{rhs}) = \mathcal{V}_L (\text{lhs}) \quad \text{implies} \quad \mathcal{V}_{L-1}' (\text{rhs}) = \mathcal{V}_L' (\text{lhs}).
\] (25)

So, eqs (24) and (25) are applied repeatedly until the outer boundary at \((z = 0)\) is reached; i.e., determining \(\mathcal{V}'(0)\), then eqs (20) or (21) may be evaluated as appropriate.

Case 2 considers variations in the complex dielectric function \((\varepsilon_L)\) interior to the layer \((L)\). This leaves \(\mathcal{V}_L (\text{rhs})\) unaffected. Again, using the chain rule for the partial derivatives, and taking advantage of the identities,

\[
\eta'_L = \frac{\partial \eta_L}{\partial \varepsilon_L} = \frac{1}{2\eta_L}
\]

and

\[
\gamma'_L = \frac{\partial \gamma_L}{\partial \varepsilon_L} = \frac{\partial \gamma_L}{\partial \eta_L} \frac{\partial \eta_L}{\partial \varepsilon_L} = \left(\frac{iz_L k_o}{\eta_L}\right) \gamma_L,
\]

one finds that the variation may be written as

\[
\mathcal{V}_L' (\text{lhs}) = \frac{C_1^{(2,\text{TE})}}{[(1 + \gamma_L) \eta_L + (1 - \gamma_L) \mathcal{V}_L (\text{rhs})]} - \frac{[(1 - \gamma_L) \eta_L + (1 + \gamma_L) \mathcal{V}_L (\text{rhs})]}{[(1 + \gamma_L) \eta_L + (1 - \gamma_L) \mathcal{V}_L (\text{rhs})]^2} \quad \text{C}_2^{(2,\text{TE})}
\] (26)

where

\[
C_1^{(2,\text{TE})} = (1 - \gamma_L) + \frac{(1 + \gamma_L) \mathcal{V}_L (\text{rhs})}{2 \eta_L} + iz_L k_o \gamma_L [\mathcal{V}_L (\text{rhs}) - \eta_L]
\]

and

\[
C_2^{(2,\text{TE})} = \frac{(1 + \gamma_L)}{2} + iz_L k_o \gamma_L [\eta_L - \mathcal{V}_L (\text{rhs})].
\]
This boundary condition is then matched across every layer positioned between layer \((L)\) and the ambient, by applying case 1 and eq (24) accordingly.

Case 3 considers variations in the layer's thickness via \((i2z_L k_o)\) interior to the local region. Taking the necessary partial derivatives and utilizing the fact that

\[
\gamma'_L = \frac{\partial \gamma_L}{\partial (i2z_L k_o)} = \eta_L \gamma_L,
\]

the variation may be expressed as

\[
\gamma'_L (\text{lhs}) = - \frac{2\gamma_L \eta_L^2 \left[ \eta_L - \gamma_L (\text{rhs}) \right] \left[ \eta_L + \gamma_L (\text{rhs}) \right]}{\left[ \left(1 + \gamma_L \right) \eta_L + \left(1 - \gamma_L \right) \gamma_L (\text{rhs}) \right]^2}.
\]

Again, this boundary condition is transported to the outer surface, by applying case 1 and eq (24) as necessary.

Case 4 considers variations in the angle of incidence via \((-\varepsilon a \sin^2 \phi)\) exterior and interior to the local region under consideration. Recalling that

\[
\eta'_L = \frac{\partial \eta_L}{\partial (-\varepsilon a \sin^2 \phi)} = \frac{1}{2\eta_L}
\]

and

\[
\gamma'_L = \frac{\partial \gamma_L}{\partial \eta_L} \eta'_L = \left( \frac{i z_L k_o}{\eta_L} \right) \gamma_L,
\]

the variation may be expressed as

\[
\gamma'_L (\text{lhs}) = \frac{C_1^{(4,\text{TE})}}{\left[ \left(1 + \gamma_L \right) \eta_L + \left(1 - \gamma_L \right) \gamma_L (\text{rhs}) \right]} - \frac{\left[ \left(1 - \gamma_L \right) \eta_L + \left(1 + \gamma_L \right) \gamma_L (\text{rhs}) \right]}{\left[ \left(1 + \gamma_L \right) \eta_L + \left(1 - \gamma_L \right) \gamma_L (\text{rhs}) \right]^2} C_2^{(4,\text{TE})}
\]

where

\[
C_1^{(4,\text{TE})} = i z_L k_o \gamma_L \left[ \gamma_L (\text{rhs}) - \eta_L \right] + \left(1 - \gamma_L \right) + \frac{(1 + \gamma_L) \gamma_L (\text{rhs})}{2 \eta_L}
\]

and

\[
+ \left(1 + \gamma_L \right) \eta_L \gamma'_L (\text{rhs})
\]

\[
C_2^{(4,\text{TE})} = i z_L k_o \gamma_L \left[ \eta_L - \gamma_L (\text{rhs}) \right] + \frac{(1 + \gamma_L)}{2} + (1 - \gamma_L) \eta_L \gamma'_L (\text{rhs}).
\]
This case is distinct from the others by requiring both $\mathcal{Y}_L(rhs)$ and $\mathcal{Y}'_L(rhs)$ to be in the right-hand-side of the equation as may be seen from the coefficients $C^{(4)}$. Consequently, eq (28) must be used recursively, starting with the boundary conditions for the substrate region, i.e., eq (23).

Following the evaluations of the necessary partial derivatives in eqs (20) to (22), $\mathcal{Y}'(0)$ and $R'_a$, it is necessary to express the derivatives in terms of the angle of incidence $\phi$ and the model parameters $(z, n, k)$ for the appropriate layer or substrate, i.e.,

\[
\frac{\partial R_a}{\partial z} = (i2k_o) \frac{\partial R_a}{\partial (i2z k_o)},
\]

\[
\frac{\partial R_a}{\partial n} = 2(n + i k) \frac{\partial R_a}{\partial \varepsilon},
\]

\[
\frac{\partial R_a}{\partial k} = 2i(n + i k) \frac{\partial R_a}{\partial \varepsilon},
\]

and

\[
\frac{\partial R_a}{\partial \phi} = -2\varepsilon_a \sin \phi \cos \phi \frac{\partial R_a}{\partial (-\varepsilon_a \sin^2 \phi)}.
\]

These expressions determine the Jacobian of the reflected fields. These are then used in section 2.3, where the measured ellipsometric angles $(\Delta, \psi)$ and the associated partial derivatives are defined in terms of the reflected fields.

### 2.2 Transverse Magnetic Mode or $P$-Polarization

#### 2.2.1 Reflection Coefficient

Proceeding in like manner of presentation of the TE mode, the uncoupling of Maxwell's equations provides a solution or polarization where the incident, refracted, and reflected fields may be characterized by electric field vectors which are aligned in a direction parallel to the plane of incidence, i.e., the $(z, x)$ plane. Here, $(H_x = H_z = 0)$, and $(H_y \neq 0)$.

This is called the transverse magnetic mode or $p$-polarization. From eq (10), $E_y = 0$; from eq (6), $E_z = E_z(z, x)$; from eq (8), $E_z = E_z(z, x)$; so there are no dependences on $y$. Substituting eqs (9) and (11) into eq (7) yields

\[
\nabla_z \left( \frac{1}{ik_o \varepsilon} \nabla_z H_y \right) - \nabla_z \left( \frac{-1}{ik_o \varepsilon} \nabla_z H_y \right) = i\mu k_o H_y
\]
or
\[
\left[ \nabla^2_x + \nabla^2_z - (\nabla \ln \varepsilon) \nabla_x + k^2_0 \varepsilon \mu I \right] H_y = 0.
\]

Assuming a solution by the method of separable variables, let \( H_y = \mathcal{Z}(z) e^{izk_0 \alpha} \). Then, restricting the domain to a single layer or substrate region where the optical properties are homogeneous and uniform, the gradient term vanishes, so that the ODE is of the form of eq (15), which has solutions involving a set of damped plane waves, i.e.,
\[
\mathcal{Z} = H^{(+)} e^{izk_0 \eta} + H^{(-)} e^{-izk_0 \eta}.
\]

Again, the tangential component of the magnetic field vector is continuous across boundaries between adjacent layers involving no interfacial currents. Accordingly, the method of solution follows from eq (9),
\[
\nabla_x H_y = i k_0 \varepsilon E_z
\]

so
\[
E_z = \frac{\eta}{\varepsilon} \left[ H^{(+)} e^{izk_0 \eta} - H^{(-)} e^{-izk_0 \eta} \right] \mathcal{X}.
\]

Since the tangential components of the field vectors are continuous across boundaries, the logarithmic derivative of the solution or impedance function
\[
\mathcal{Y} = \frac{E_z}{H_y} = \frac{\eta}{\varepsilon} \left[ H^{(+)} e^{izk_0 \eta} - H^{(-)} e^{-izk_0 \eta} \right]
\]

is a continuous function of \( z \) alone. The innermost boundary condition for the impedance function is
\[
\mathcal{Y}_s = \frac{\eta}{\varepsilon} = \frac{\sqrt{\varepsilon_s \mu_s - \varepsilon_0 \sin^2 \phi}}{\varepsilon_s}
\]

where the subscript \( s \) denotes the substrate region. Starting with the substrate region, the boundary conditions of the impedance function may be matched to the adjacent layer (\( L \)),
\[
\mathcal{Y}_L (\text{rhs}) = \mathcal{Y}_L (z_L) = \mathcal{Y}_s
\]

so that
\[
R_L = e^{i2z_L k_0 \eta_L} \left[ \frac{(\eta_L / \varepsilon_L) - \mathcal{Y}_L (\text{rhs})}{(\eta_L / \varepsilon_L) + \mathcal{Y}_L (\text{rhs})} \right]
\]

16
and
\[ \gamma_L(\text{lhs}) = \gamma_L(z = 0) = \frac{\eta_L}{\varepsilon_L} \left[ \frac{1 - R_L}{1 + R_L} \right], \]
or upon substitution with eq (17),
\[ \gamma_L(\text{lhs}) = \frac{\eta_L}{\varepsilon_L} \left[ \frac{(\eta_L/\varepsilon_L) + \gamma_L(\text{rhs}) - \gamma_L[(\eta_L/\varepsilon_L) - \gamma_L(\text{rhs})]}{(\eta_L/\varepsilon_L) + \gamma_L(\text{rhs}) + \gamma_L[(\eta_L/\varepsilon_L) - \gamma_L(\text{rhs})]} \right] \]
\[ = \frac{\eta_L}{\varepsilon_L} \left[ \frac{(1 - \gamma_L)(\eta_L/\varepsilon_L) + (1 + \gamma_L)\gamma_L(\text{rhs})}{(1 + \gamma_L)(\eta_L/\varepsilon_L) + (1 - \gamma_L)\gamma_L(\text{rhs})} \right] \]
\[ = \frac{\eta_L}{\varepsilon_L} \left[ \frac{(1 - \gamma_L)\eta_L + (1 + \gamma_L)\varepsilon_L\gamma_L(\text{rhs})}{(1 + \gamma_L)\eta_L + (1 - \gamma_L)\varepsilon_L\gamma_L(\text{rhs})} \right]. \tag{34} \]

The reflection coefficient in the ambient region, \( R_a \), is found from the impedance function evaluated at the outer surface \((z = 0)\) and is given by
\[ R_a = \left[ \frac{(\eta_a/\varepsilon_a) - \gamma(0)}{(\eta_a/\varepsilon_a) + \gamma(0)} \right]. \tag{35} \]

### 2.2.2 Partial Derivatives

The discussion presented here parallels section 2.1.2 of the TE mode. Recalling eq (35), it follows that the operator equations regarding partial derivatives are of the form:
\[ \frac{\partial R_a}{\partial (2i\varepsilon k_o)} \equiv R'_a = -\frac{2(\eta_a/\varepsilon_a)'\gamma'(0)}{[(\eta_a/\varepsilon_a) + \gamma(0)]^2} \tag{36} \]
when the prime is used to imply a partial derivative with respect to \((2i\varepsilon k_o)\),
\[ \frac{\partial R_a}{\partial \varepsilon} \equiv R'_a = -\frac{2(\eta_a/\varepsilon_a)\gamma'(0)}{[(\eta_a/\varepsilon_a) + \gamma(0)]^2} \tag{37} \]
when the prime is used to imply a partial derivative with respect to \(\varepsilon\), and
\[ \frac{\partial R_a}{\partial (-\varepsilon_a \sin^2 \phi)} \equiv R'_a = \frac{2[2(\eta_a/\varepsilon_a)'\gamma(0) - (\eta_a/\varepsilon_a)\gamma'(0)]}{[(\eta_a/\varepsilon_a) + \gamma(0)]^2} \tag{38} \]
when the prime is used to imply a partial derivative with respect to \((-\varepsilon_a \sin^2 \phi)\). These need to be calculated for each model parameter undergoing variation.
These derivatives of the impedance function may be understood by considering a variation in the model parameter within the substrate region. From eq (33), one finds that
\[
\frac{\partial y_s}{\partial \varepsilon_s} = y_s' = \left(\frac{\varepsilon_s}{\varepsilon_s'}\right)' = \frac{1}{2\eta_s \varepsilon_s} - \frac{\varepsilon_s - 2\eta_s^2}{2\eta_s \varepsilon_s^2}
\]
and
\[
\frac{\partial y_s}{\partial (-\varepsilon_s \sin^2 \phi)} = y_s' = \frac{1}{2\eta_s \varepsilon_s}.
\]

The effect of each derivative may be viewed as a variation on the impedance function that must be matched as a boundary condition across each layer that lies between the substrate and ambient region. Again, four cases need to be considered, depending upon whether the partial derivative operates on a function whose parameter is exterior or interior to the local region of interest. Consider the adjacent layer \((L)\).

Case 1 considers partial derivatives with respect to \((\varepsilon, z)\) operating exterior to the local region. Again, only the \((\text{rhs})\) boundary condition is affected. Operating upon eq (34) yields
\[
y'_L(\text{lhs}) = \frac{4\gamma_L \eta_s^3 y'_L(\text{rhs})}{[(1 + \gamma_L) \eta_L + (1 - \gamma_L) \varepsilon_L y'_L(\text{rhs})]^2}.
\]
The above equation is applied repeatedly until the outer boundary at \((z = 0)\) is reached; i.e., \(y'(0)\), so that eq (36) or (37) may be evaluated where appropriate.

Case 2 considers variations in the complex dielectric function \((\varepsilon_L)\) interior to layer \((L)\). This leaves \(y_L(\text{rhs})\) unaffected. The partial derivative may be expressed rather simply in unreduced form,
\[
y'_L(\text{lhs}) = \left(\frac{\varepsilon_L - 2\eta_L^2}{2\eta_L \varepsilon_L^2}\right) \left[\frac{[(1 - \gamma_L) \eta_L + (1 + \gamma_L) \varepsilon_L y_L(\text{rhs})]}{[(1 + \gamma_L) \eta_L + (1 - \gamma_L) \varepsilon_L y_L(\text{rhs})]}\right]
+ \left(\frac{\eta_L}{\varepsilon_L}\right) \frac{C_1^{(2, \text{TM})}}{[(1 + \gamma_L) \eta_L + (1 - \gamma_L) \varepsilon_L y_L(\text{rhs})]^2}
- \left(\frac{\eta_L}{\varepsilon_L}\right) \frac{[(1 - \gamma_L) \eta_L + (1 + \gamma_L) \varepsilon_L y_L(\text{rhs})]}{[(1 + \gamma_L) \eta_L + (1 - \gamma_L) \varepsilon_L y_L(\text{rhs})]^2} C_2^{(2, \text{TM})} \]
where
\[
C_1^{(2, \text{TM})} = \left(\frac{1 - \gamma_L}{2\eta_L}\right) + (1 + \gamma_L) y_L(\text{rhs}) + i z_L k_0 \gamma_L \left[\left(\frac{\eta_L}{\varepsilon_L}\right) y_L(\text{rhs}) - 1\right]
\]
and
\[ C_2^{(2, \text{TM})} = \left( \frac{1 + \gamma_L}{2\eta_L} \right) + (1 - \gamma_L) \gamma_L (\text{rhs}) + iz_L k_0 \gamma_L \left[ 1 - \left( \frac{\eta_L}{\epsilon_L} \right) \gamma_L (\text{rhs}) \right]. \]

This boundary condition may then be matched across any layer positioned between layer \((L)\) and the ambient, by repeated use of case 1 and eq (41) as appropriate.

Case 3 considers variations in the layer's thickness via \((i2z_L k_0)\) interior to the local region. The partial derivative is of the form

\[ \mathcal{Y}_L' (\text{lhs}) = - \left( \frac{\eta_L}{\epsilon_L} \right) \frac{2\gamma_L \eta_L \left[ \eta_L - \epsilon_L \mathcal{Y}_L (\text{rhs}) \right] \left[ \eta_L + \epsilon_L \mathcal{Y}_L (\text{rhs}) \right]}{\left[ (1 + \gamma_L) \eta_L + (1 - \gamma_L) \epsilon_L \mathcal{Y}_L (\text{rhs}) \right]^2}. \] (43)

Again, this boundary condition is transported to the outer surface, by applying case 1 and eq (41) as necessary.

Case 4 considers variations in the angle of incidence via \((-\epsilon_a \sin^2 \phi)\) exterior and interior to the local region under consideration. The partial derivative may be expressed as

\[ \mathcal{Y}_L' (\text{lhs}) = \frac{1}{2\eta_L \epsilon_L} \left[ \frac{(1 - \gamma_L) \eta_L + (1 + \gamma_L) \epsilon_L \mathcal{Y}_L' (\text{rhs})}{(1 + \gamma_L) \eta_L + (1 - \gamma_L) \epsilon_L \mathcal{Y}_L (\text{rhs})} \right] \\
+ \left( \frac{\eta_L}{\epsilon_L} \right) \frac{C_1^{(4, \text{TM})}}{\left[ (1 + \gamma_L) \eta_L + (1 - \gamma_L) \epsilon_L \mathcal{Y}_L (\text{rhs}) \right]} \\
- \left( \frac{\eta_L}{\epsilon_L} \right) \frac{\left[ (1 - \gamma_L) \eta_L + (1 + \gamma_L) \epsilon_L \mathcal{Y}_L (\text{rhs}) \right]}{\left[ (1 + \gamma_L) \eta_L + (1 - \gamma_L) \epsilon_L \mathcal{Y}_L (\text{rhs}) \right]^2} \frac{C_2^{(4, \text{TM})}}{C_1^{(4, \text{TM})}} \] (44)

where

\[ C_1^{(4, \text{TM})} = \left( \frac{1 - \gamma_L}{2\eta_L} \right) + (1 + \gamma_L) \epsilon_L \mathcal{Y}_L' (\text{rhs}) + iz_L k_0 \gamma_L \left[ \left( \frac{\eta_L}{\epsilon_L} \right) \mathcal{Y}_L (\text{rhs}) - 1 \right] \]

and

\[ C_2^{(4, \text{TM})} = \left( \frac{1 + \gamma_L}{2\eta_L} \right) + (1 - \gamma_L) \epsilon_L \mathcal{Y}_L' (\text{rhs}) + iz_L k_0 \gamma_L \left[ 1 - \left( \frac{\eta_L}{\epsilon_L} \right) \mathcal{Y}_L (\text{rhs}) \right]. \]

Again, case 4 is distinct from the other three by requiring both, \(\mathcal{Y}_L (\text{rhs})\) and \(\mathcal{Y}_L' (\text{rhs})\), as may be seen from the coefficients \(C^{(4, \text{TM})}\). Consequently, eq (44) must be used recursively, starting with the boundary condition of the substrate region, i.e., eq (40).

The partial derivatives with respect to the model parameters \((z, n, k)\), for each appropriate layer, substrate, and wavelength are required later, and are of the same form as that presented in eqs (29) to (32).
2.3 Ellipsometric Angles \((\Delta, \psi)\)

In sections 2.1 and 2.2, a physical model was proposed for a restricted class of media, and the forward electromagnetic problem was solved where the reflection coefficients local to the ambient region are complex amplitudes, i.e., from eq (19)

\[ R_{a,s} = |R_{a,s}| \exp (i\delta_s), \]  
\[ R_{a,p} = |R_{a,p}| \exp (i\delta_p), \]

and from eq (35),

where subscripts \((s, p)\) refer to polarization, and \(\delta\) refers to the phase. The ellipsometric angles \((\Delta, \psi)\) are defined in terms of the complex variable \(\rho\) which is formed by the ratio of the reflection coefficients,

\[ \rho = \frac{R_{a,p}}{R_{a,s}} = \left| \frac{R_{a,p}}{R_{a,s}} \right| \exp \{i(\delta_p - \delta_s)\} = \tan \psi \exp (i\Delta), \]

where the explicit relationship of the phases involves

\[ \Delta = \delta_p - \delta_s, \quad \text{such that} \quad 0 \leq \Delta < 2\pi, \]

while the modulus involves

\[ \tan \psi = \left| \frac{R_{a,p}}{R_{a,s}} \right| = |\rho|, \quad \text{where} \quad 0 \leq \psi \leq \frac{\pi}{2}, \]

so

\[ \psi = \tan^{-1} |\rho|. \]

Further, since the ellipsometric angles refer exclusively to the reflected fields in the ambient region, it is convenient to omit further subscriptive references to the ambient.

It is important to draw attention to two subtleties arising from the choice of convention used in defining the ellipsometric angles. First, the \(R_s\) was defined in terms of the electric field, whereas \(R_p\) used the magnetic field. If electric fields are used in the definitions exclusively, then \(R_{p,E} = -R_{p,H}\), so that only the \(\Delta\) is affected, i.e.,

\[ \Delta_p = \Delta_H + \pi \pmod{2\pi}. \]
Second, the time dependence here was chosen from the physics convention, as opposed to the engineering or Nebraska convention. The complex amplitudes found with one convention are related to that with the other convention by simple complex conjugation. This affects only the phase or sign of $\Delta$, i.e.,

$$\Delta_{\text{engineering}} = 2\pi - \Delta_{\text{physics}} \pmod{2\pi}. \quad (51)$$

Now that the ellipsometric angles have been defined, it will be necessary to formulate the Jacobian or partial derivatives of the ellipsometric angles in terms of the reflected fields and their partial derivatives. These are important to the general formulation of the inverse problem, which involves inverting the ellipsometric equations discussed later in section 3.

For convenience, let all further primes used within this section refer to an implied partial derivative with respect to some model parameter, e.g., $(z_L, n_L, k_L, n_s, k_s, \phi)$, where the subscripts refer to some layer or substrate region. Since the ellipsometric angles involve phase and modulus information of the complex variable $\rho$, it will be necessary to develop relationships which express partial derivatives of the modulus and phase of a complex variable in terms of the partial derivatives of the real and imaginary parts of that same complex variable.

This may be accomplished in the usual manner by considering the notation involving a single complex variable, denoted by $R$, which has modulus $r$, phase $\delta$, and real and imaginary parts, $x$ and $y$, respectively, i.e.,

$$R = re^{i\delta} = x + iy \quad \text{where} \quad r = |R| = \sqrt{x^2 + y^2}.$$ 

Then,

$$\frac{R}{R^*} = e^{i2\delta} \quad \text{and} \quad i2\delta = \ln \left( \frac{R}{R^*} \right),$$

so that by forming

$$(R^*R) \left[ \frac{R'}{R} - \frac{R'^*}{R^*} \right] = R^*R' - RR'^*$$

$$r^2 \left[ \ln \left( \frac{R}{R^*} \right) \right]' = (x - iy)(x' + iy') - (x + iy)(x' - iy')$$

$$2i r^2 \delta' = 2i (xy' - yx'),$$
one finds that
\[ \delta' = \frac{xy' - yx'}{x^2 + y^2}. \]  
(52)

Proceeding in like manner,
\[ (r^2)' = (x^2 + y^2)' = (R^* R)' \]
= 2 \cdot 2 (xx' + yy') ,

one finds that
\[ r' = \frac{xx' + yy'}{\sqrt{x^2 + y^2}}. \]
(53)

Here, eqs (52) and (53) applies to each polarization individually; i.e., one finds
(\delta'_s, \delta'_p, r'_s, r'_p).

Now, consider the definitions of the ellipsometric angles. From eq (49) and using the
trigonometric identity involving derivatives of arctangents, it follows that
\[ \psi' = \frac{|\rho|'}{1 + |\rho|^2} = \frac{(r_p/r_s)'}{1 + (r_p/r_s)^2} \]
= \frac{r_s^2}{r_s^2 + r_p^2} \left[ \frac{r_p'}{r_s} - \frac{r_p r_s'}{r_s^2} \right]
= \frac{r_s r_p' - r_p r_s'}{r_s^2 + r_p^2}, \]
(54)

where the right-hand-side of this expression is used in conjunction with eq (53) for each
polarization.

Proceeding in like manner with eq (48), it follows that
\[ \Delta' = \delta'_p - \delta'_s, \]
(55)

where the right-hand-side of this expression is used in conjunction with eq (52) for each
polarization.

Eqs (52) to (55) provide the basic set of equations necessary for evaluating the various
partial derivatives of the ellipsometric angles in terms of the reflection coefficients and
their partial derivatives.
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, i.e., eqs (48) and (49). These equations describe how the material optical properties of the 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. One common approach to performing such inversions is to formulate them as nonlinear least-squares problems [10–15]. Here, one considers a sequence of forward problems, where each increment of the sequence involves three distinct steps. The steps include: 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 are of the form:

\[
\Delta = \Delta(\lambda, \phi, b) \tag{56}
\]

\[
\psi = \psi(\lambda, \phi, b) \tag{57}
\]

where \(\lambda\) is the vacuum wavelength of the incident light; \(\phi\) is the angle of incidence; \(b\) is an array where the components specify the model parameters, i.e., layer thicknesses, indices of refraction and extinction coefficients of the layers and substrates, e.g., \((z_L, n_L^\lambda, k_L^\lambda, n_s^\lambda, k_s^\lambda)\), where the subscripts refer to a layer or substrate region; and where the superscripts refer to the wavelength dependence of the optical properties. The standard procedure involves minimizing some non-negative scalar error expression containing the deviations between experiment and model of \(\Delta\) and \(\psi\) in the least-squares sense, e.g.,

\[
G(b) = \frac{1}{2M} \sum_{i=1}^{M} \left[ (\Delta_i^e - \Delta_i^m)^2 + (\psi_i^e - \psi_i^m)^2 \right] \tag{58}
\]

\[
= \sum_{i=1}^{2M} g_i^2 = g^T g = ||g||^2 \tag{59}
\]
where superscripts (e,m) refer to experiment and model, respectively, \( M \) denotes the number of measurements of (\( \Delta, \psi \)) from experiment, \( g \) is a column array of the deviations, i.e.,

\[
g_{2i-1} = (\Delta_i^e - \Delta_i^m) / \sqrt{2M},
\]
\[
g_{2i} = (\psi_i^e - \psi_i^m) / \sqrt{2M},
\]

where \( 1 \leq i \leq M \), and the superscript \( T \) denotes transposition. Here, the error expression assumes equal weighting factors or uncertainties in the measurements of \( \Delta \) and \( \psi \). This is a simplification because it is well known that the measurement uncertainty in \( \Delta \) and \( \psi \) may differ by more than a factor of two [26].

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, \( g = Jv \), where \( v \) is a column array (Newton step) for improving the model parameters that were selected to undergo variation, i.e., \( v_j \sim \delta b_j \), and \( J \) is the sparse matrix involving the Jacobian which is not necessarily square, i.e., \( J_{ij} \sim (1/\sqrt{2M}) \left( \partial \Delta_i / \partial b_j \right) \).

Such matrix equations are common to optimization problems and involve only linear algebra. It is well known that additional numerical stability may result when requesting the norm of \( v \) to be minimized as well. This may be accomplished by modifying the error expression to

\[
G = (g - Jv)^T (g - Jv) + \kappa v^T v,
\]

where \( \kappa \) is a positive scalar parameter subjectively chosen between 0.01 and 1.0 for our calculations. Of course, the final solution \( v \) should be independent of \( \kappa \), but here, \( \kappa \) simply moderates the rate of convergence.

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

\[
S_{jj} = \sqrt{(J^T J)_{jj}}.
\]
Then letting

$$\tilde{J} = JS^{-1},$$  \hspace{1cm} (64)

$$\tilde{v} = Sv,$$  \hspace{1cm} (65)

$$r = g - Jv = g - \tilde{J}\tilde{v},$$  \hspace{1cm} (66)

a suitable error expression may be defined by

$$G = (g - \tilde{J}\tilde{v})^T(g - \tilde{J}\tilde{v}) + \kappa \tilde{v}^T\tilde{v}$$  \hspace{1cm} (67)

$$= r^T r + \kappa \tilde{v}^T\tilde{v}.$$  \hspace{1cm} (68)

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

$$\begin{bmatrix} 1 & \tilde{J} \\ \tilde{J}^T & -\kappa 1 \end{bmatrix} \begin{bmatrix} r \\ \tilde{v} \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix},$$  \hspace{1cm} (69)

and must then be solved for \( \tilde{v} \).

Because of the sparsity of the Jacobian, it is expedient to utilize an iterative method for solving the above matrix algebra problem. Algorithms [16,17] exist that specifically address this type of problem, one [18,19] of which utilizes a relatively stable Lanczos process (Krylov space decomposition) in formulating the method of steepest-descents. Essentially, the method requires each updating vector be orthogonal to the previous update vectors. Further details may be found elsewhere [18–22]. From this procedure, one finds \( \tilde{v} \), which leads to \( v \), which is then used to improve the estimate of the values for the model parameters, e.g.,

$$b_j^{new} = b_j^{old} + v_j.$$  

Using this improved \( b \), the sequence is repeated again until either: \(|g| \) becomes sufficiently small, of the order of a few millidegrees, e.g., the resolution of the measurement, or until \(|\tilde{v}| \) becomes sufficiently small so that \(|g| \) 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 pseudo-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 were 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 \(|g|\) 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 \(G\), 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 \(|g|\) 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., \(|g|\), 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 its use of the \(F\)-statistic in assessing the so-called goodness-of-fit test [23]. 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 vast literature available in statistics.

### 3.2 Sensitivity Analysis of Model Parameters

In the discussion of the least-squares problem in section 3.1, the main emphasis was on obtaining accurate numerical values for the model parameters. It follows naturally that the next step should involve some assessment of the uncertainties associated with those values. Fortunately, for a restricted class of linearizable problems, one may ascribe estimates to these uncertainties by utilizing a formulism similar to that used in the least-squares problem [2, 10–15, 23–25].
Consider again the functional representation of the model and its expansion about the critical or fixed point solution, e.g., using variations out to first order, i.e.,

$$
\Delta_i^e - \Delta_i^m = \sum_{j=1}^{N} \frac{\partial \Delta_i}{\partial b_j} \bigg|_o \delta b_j + \frac{\partial \Delta_i}{\partial \phi} \bigg|_o \delta \phi + \tilde{\epsilon}_{\Delta,i} \tag{70}
$$

and

$$
\psi_i^e - \psi_i^m = \sum_{j=1}^{N} \frac{\partial \psi_i}{\partial b_j} \bigg|_o \delta b_j + \frac{\partial \psi_i}{\partial \phi} \bigg|_o \delta \phi + \tilde{\epsilon}_{\psi,i} \tag{71}
$$

where \(N\) refers to the number of distinct model parameters, and \(\tilde{\epsilon}_i\) refers to the uncertainty associated with individual measurements of \((\Delta, \psi)\) performed by the instrument. It may also be the case during the course of the analysis of finding the critical point solution discussed in section 3.1, that some of the model parameters will have had values and uncertainties assigned to them by some earlier experiment or measurement that is external to and distinct from those being analyzed here, e.g., values found or taken from the literature; these values are assumed as given and remained unchanged throughout the calculations. Consequently, the set of model parameters may be partitioned into two disjoint sets, those \((\delta b_j)\) that remain unchanged \((u_j)\) and those \((\delta b_j)\) that are allowed to vary \((v_j)\). This allows both of the above expansions to be combined and expressed as

$$
g_i = J_{u,i}v_j + J_{u,i}u_j + J_{\phi,i}\tilde{\phi}_i + \tilde{\epsilon}_i \tag{72}
$$

or in matrix notation as

$$
g = J_u v + J_u u + J_{\phi} \tilde{\phi} + \tilde{\epsilon} \tag{73}
$$

where \(g_i\) refers to the deviations similar to eqs (60) and (61) but without including the scaling factor of \((1/\sqrt{2M})\), \(J\) refers to the Jacobian or the array of partial derivatives that is appropriate to the partitioning and evaluated at the critical point, \(\tilde{\phi}_i\) 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 procedures of assigning values of uncertainty in knowing \(v\).

From the representation of the error expression \(G\) presented in eqs (62) or (67), it follows that the minimum value of \(G\) is attributed to residuals that are due to measurement uncertainties \(\tilde{\epsilon}_i\) 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 to be of the form

$$G_0 = |g|^2 = |e|^2,$$  \hspace{1cm} (74)

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

$$
\tilde{G} = G - G_0 = \left| g - J_v \hat{v} - J_u \hat{u} - J_\phi \hat{\Phi} \right|^2 - |g|^2.
$$ \hspace{1cm} (75)

Before presenting a formal solution for $v$, it is worthwhile to consider the fact that round-off error affects the numerical accuracy of a calculation, and this is often dependent on the scaling of the problem. Again, it is convenient to follow eqs (63) to (65) and scale the numerical problem by defining the diagonal matrix $S$, where

$$S_{jj} = \sqrt{(J_v^T J_v)_{jj}},$$ \hspace{1cm} (76)

with

$$\tilde{v} = S v$$ \hspace{1cm} (77)

and

$$\tilde{J}_v = J_v S^{-1},$$ \hspace{1cm} (78)

so that eq (75) may be written as a function of $\tilde{v}$, i.e.,

$$\tilde{G} = \left| g - \tilde{J}_v \tilde{v} - J_u \hat{u} - J_\phi \hat{\Phi} \right|^2 - |g|^2.$$ \hspace{1cm} (79)

Recalling the criteria of a critical point, i.e., $(\partial \tilde{G}/\partial \tilde{v}_j) = 0$, one finds a system of equations for $\tilde{v}$, i.e.,

$$\tilde{J}_v^T \left( g - \tilde{J}_v \tilde{v} - J_u \hat{u} - J_\phi \hat{\Phi} \right) = 0.$$ \hspace{1cm} (80)

Defining the square matrix

$$A = \tilde{J}_v^T \tilde{J}_v,$$ \hspace{1cm} (81)

and the nonsquare matrix

$$B = S^{-1} A^{-1} \tilde{J}_v^T,$$ \hspace{1cm} (82)
and substitutes eq (82) into eq (80), a formal solution of \( v \) may be written as

\[
v = B (g - J_u u - J_v \tilde{\Phi}).
\]  

(83)

Of course, this is contingent upon the rank or invertibility of \( A \). This equation involves a measure of both types of errors, i.e., random error \((g, \tilde{\Phi})\) and systematic error \((u)\). However, due to the assumption of stochastic independence, each type of error may be considered separately.

Consider first the contributions associated with the random error. From eq (83), the random component assumes

\[
v_j = B_{ji} \left( g_i - J_{*i} \tilde{\phi}_i \right).
\]  

(84)

The measurement uncertainties \((\tilde{e}_i)\), as well as the deviations found from fitting the model to the data \((g_i)\), are assumed to behave as statistical random variables, which are mutually stochastically independent and identically distributed. Of course, this assumes that \( \Delta \) and \( \psi \) are uncorrelated. Then, each random variable involves the same probability distribution function (PDF), and the variance may be estimated from \( g \). Recall that \( g \) involves \( 2M \) measurements and \( N \) model parameters, and \( \tilde{\Phi} \) involves \( M \) model parameters. The PDF for \( g_i^2 \) is assumed to behave as a \( \chi^2 \) distribution with \( 2M - N \) degrees of freedom, where the variance is estimated by

\[
s^2_g = s^2(g_i) = \frac{\mathbf{g}^T \mathbf{g}}{2M - N}.
\]  

(85)

The PDF for \( \tilde{\phi}_i^2 \) is assumed to behave as a \( \chi^2 \) distribution with \( 2M - M \) degrees of freedom, where the variance is estimated by

\[
s^2_\phi = s^2(\tilde{\phi}_i^2) = \frac{\tilde{\Phi}^T \tilde{\Phi}}{2M - M},
\]  

(86)

where the \( \tilde{\phi}_i \) were estimated from a set of calibration data, which is external to the set of measurements of the ellipsometric angles, and not estimated from fitting the model to the measurement of \((\Delta, \psi)\). Often, \( \tilde{\phi}_i \) may be of the order of a few millidegrees. Given the estimated variance of the statistical PDF for the random variables \((g_i, \tilde{\phi}_i)\), and given the linear relationship between the model parameters and the random variables, i.e., eq
one may then proceed and estimate the variance of the PDF for the model parameters \( \mathbf{v} \). The expectation values are of the form

\[
\langle g_{i} g_{l} \rangle = s_{g}^{2} \delta_{il}, \quad (87)
\]

\[
\langle \phi_{i} \phi_{l} \rangle = s_{\phi}^{2} \delta_{il}, \quad (88)
\]

\[
\langle g_{i} \phi_{l} \rangle = 0, \quad (89)
\]

where the bra-ket notation \( \langle \rangle \) refers to an expectation value, and \( \delta \) refers to the Kronecker delta. The variance of the model parameters \( \mathbf{v} \) is then estimated by considering the expectation value of the following quadratic form,

\[
v_{j} v_{k} = B_{ji} \left( g_{i} - J_{\phi,i} \phi_{l} \right) B_{kl} \left( g_{l} - J_{\phi,l} \phi_{l} \right). \quad (90)
\]

This yields the form

\[
\langle v_{j} v_{k} \rangle = B_{ji} B_{kl} \left( s_{g}^{2} + s_{\phi}^{2} J_{\phi,i}^{2} \right), \quad (91)
\]

or expressed in matrix notation as

\[
\langle \mathbf{v v}^{T} \rangle = s_{g}^{2} \mathbf{B B}^{T} + s_{\phi}^{2} \mathbf{B J}_{\phi} \mathbf{J}_{\phi}^{T} \mathbf{B}. \quad (92)
\]

Further, it may be seen that the variance of that involving some linear combination of the model parameters, e.g.,

\[
\mathcal{V} = \alpha_{j} v_{j} \quad (93)
\]

where \( \alpha_{j} \) are real scalar coefficients, is of the form

\[
\langle \mathcal{V}^{2} \rangle = \alpha_{j} \alpha_{k} \langle v_{j} v_{k} \rangle. \quad (94)
\]

Also, it follows from eqs (78), (81), and (82), that

\[
\mathbf{B B}^{T} = \mathbf{S}^{-1} \mathbf{A}^{-1} \mathbf{J}_{\phi}^{T} \mathbf{J}_{\phi} \mathbf{A}^{-1} \mathbf{S}^{-1}
\]

\[
= \mathbf{S}^{-1} \mathbf{A}^{-1} \mathbf{S}^{-1}
\]

\[
= \mathbf{S}^{-1} \left[ \mathbf{S}^{-1} \mathbf{J}_{\phi}^{T} \mathbf{J}_{\phi} \mathbf{S}^{-1} \right]^{-1} \mathbf{S}^{-1}
\]

\[
= \mathbf{S}^{-1} \left[ \mathbf{S} \left( \mathbf{J}_{\phi}^{T} \mathbf{J}_{\phi} \right)^{-1} \mathbf{S} \right] \mathbf{S}^{-1}
\]

\[
= \left( \mathbf{J}_{\phi}^{T} \mathbf{J}_{\phi} \right)^{-1}, \quad (95)
\]
which is well known.

Next, consider the contributions associated with systematic error. From eq (83), the systematic component assumes

$$v_j = B_{ji} J_{u,ik} u_k.$$  \hspace{1cm} (96)

By applying absolute value signs and the triangle inequality, a strict upper bound for $|v_j|$ may be found and is given by

$$|v_j| \leq |B_{ji} J_{u,ik}| u_k|. \hspace{1cm} (97)$$

Assuming the total magnitude of uncertainty that is assigned to a model parameter involves a direct sum of both types of errors, then one may combine eqs (91) and (97) to yield the form

$$|v_j| \leq \sqrt{B_{ji}^2 \left(s_g^2 + s_i^2 J_{s,i}^2 \right)} + |B_{ji} J_{u,ik}| u_k|, \hspace{1cm} (98)$$

where summation is implied on the $(i, k)$ indices. And, the uncertainty associated with a linear combination of the model parameters from eq (93) is assumed to be given by

$$|V| \leq \sqrt{\alpha_j \alpha_k B_{ji} B_{ki} \left(s_g^2 + s_i^2 J_{s,i}^2 \right)} + |\alpha_j | |B_{ji} J_{u,ik}| u_k|, \hspace{1cm} (99)$$

where summation is implied on the $(i, j, k)$ indices.

In order to use MAIN1, it is necessary to: (1) specify the model parameters which characterize the layered structure of the sample, (2) collect the necessary measurement data of ellipsometric angles, (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, and (5) understand the limitations and capabilities of the software package and its utility for implementing strategies to analyze the measurement data. The filename convention assumes a filename of up to six characters and an extension of up to three characters; the standard format is given by: filename.ext. This section presents a brief overview of these considerations.

4.1 Software Development Considerations

The main program, MAIN, performs a small set of functions. First, it calls a subroutine to open the necessary data files. Second, it requests a routine 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

MAIN1 makes use of four input/output files. One file serves for entering the input data, while three files serve for collecting output data.

X.DAT is the input data file. It contains: the model parameters which characterize the sample, the configuration of the layered structure of the sample, the ellipsometric angles of the measurement data, and the command options that provide control of the program.

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.

X.SOUT is the solution output data file that collects the breakpoint information during any grid scan of the model parameters. This information is written onto disk 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.DAT, 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.DAT before resuming calculations. Consequently, it is important to replace any earlier breakpoint data in X.DAT, as appropriate.

X.PLOT is the output file whose data are formatted in a manner that is intended to be amenable for later reading and plotting.

These files are opened initially by the subroutine FILEOP. Since breakpointing involves opening/closing files, other occurrences of the open/close statements may be found in subroutines: SCAN2, SCAN2G, SCAN3, SEAMAX. 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 interior of the program by the named common block statement located in IOUNIT.

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

4.1.2 Allocations 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. Consequently, if any changes are made to these assignments, it is usually necessary to re-compile and re-link 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.

nsamp1, the total number of samples allowed for analysis, where the sample involves some finite number of films/layers atop a substrate.

nfilms, the total number of films/layers allowed atop the substrate of any given sample.
n1mnts, the total number of distinct indices of refraction and extinction coefficients that are allowed during analysis.

nwaves, the total number of distinct wavelengths of light that are allowed to be incident on the sample.

nbient, the total number of distinct ambients (air, vaccum, etc.) which are external to the layers/substrate of the sample.

nrrepeat, the total number of allowed repeats of measurements (per wavelength) on a sample.

nanglx, the total number of allowed angles of incidence (per wavelength) of incident light.

nanglm, the total number of allowed angles of incidence (per wavelength) used during multiple-angle error analysis. This is discussed in section 4.3, but involves setting the command option=6. As may be seen in the main program, MAIN, this option has been suppressed; i.e., this constant is not currently used.

In some cases, the array allocations seem needlessly overestimated. The algorithm that estimates the array allocations from the eight constants assumes a worst-case scenario with the largest size possible being calculated. This occurs especially when sizing the arrays for the Jacobian. Consequently, it is convenient to assign some upper limit to these arrays. This is the purpose of the constant named nnjaaa. Further, a simple internal check-test is also provided to ensure that the indices remain within bounds of the dimensions of these arrays.

4.1.3 Library Software

During the course of analyses, it is often necessary to determine the sensitivity of dependence of the calculation upon the model parameters. As discussed earlier in section 3.2 from eqs (82) and (99), 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 is accomplished or performed by subroutines from the LINPACK library of mathematical software [17]. Reference calls to these subroutines may be found in files: CORLAT.FOR, SEAMA.FOR, and SEAMAX.FOR.
Finally, several of the subroutine calls occurring in subroutine POPLAT refer to the NCAR graphics library [27,28]. But again, this has been suppressed because it refers to command option=6, which is discussed in section 4.3.

4.2 Input Data Requirements

As mentioned in section 4.1.1, the program reads all of its input data from file X.DAT. From section 2, it follows that the file contains: the model parameters, the characterization of the layered structure of the samples, and the measurement data of ellipsometric angles. This section discusses a line by line construction of file X.DAT. An example of a partially completed input file is shown in section 4.2.4. The first line in the file is reserved for entering a command option. This is discussed later in section 4.3.

The main program uses subroutine INPDAT to read the data that are presented in the following subsections. These data are stored in arrays located in the named common areas that are in file FILMMM.

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 numerical values that are associated with optical properties of media at particular wavelengths.

4.2.1 Parameters

As discussed earlier in section 2, the forward problem involves at least two kinds of parameters, those which characterize the source, i.e., wavelength and angle of incidence, and those which characterize the material sample, i.e., indices of refraction and extinction coefficient at each wavelength for each distinct type of optical media and distinct layer thicknesses. In the following subsections, the parameters are grouped and presented in the order that they ought to appear in the input file. Each subsection presents one particular type of parameter and the general format appropriate for that type of input data. Each format is demonstrated by a worked example.

4.2.1.1 Wavelengths

The first group of parameters mentioned above involves wavelengths. Here, the first line of data is the number of distinct wavelengths that are associated with the measurements
collected on all of the samples. This positive integer is denoted by $m_{\lambda}$. This line of data is next followed by $m_{\lambda}$ lines of data which contain the information about the wavelengths. Each of these lines contains two numbers, one index label and one wavelength. The wavelengths are expressed in units of nanometers. The labels are indexed consecutively from 1 to $m_{\lambda}$. The lines are ordered so that the index label is monotonically increasing while the wavelengths are monotonically decreasing.

An example of this format is shown below for a simple case, where the measurements involve only two distinct wavelengths, such as that resulting from the use of two different lasers and a collection of multiple-angle of incidence data on one or more samples. The format would then involve three lines of data, e.g.,

```
2 ! mwaves ' total number of distinct wavelengths.
1  632.8 ! HeNe ' wavelength ' nanometers
2  441.6 ! HeCd  ' laser line
```

This format may be expressed succinctly by the following form:

$$m_{\lambda} / (i, \lambda_i)$$

where:

$m_{\lambda}$ is the total number of distinct wavelengths;

$i$ is an integer that indexes the wavelengths consecutively in unit increments, i.e., $i = 1, 2, 3, \ldots, m_{\lambda}$; and

$\lambda_i$ is the wavelength of the incident light that is measured in free-space and expressed in units of nanometers. It is required that the wavelengths be ordered monotonically decreasing with increasing index, i.e., $(\lambda_i > \lambda_{i+1})$. Wavelengths associated with lower energy are entered or listed first.

The above notation is a suggestive adaptation of the argument list for a READ statement in FORTRAN. 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. In general, this notation will be assumed throughout the remainder of the document, as appropriate, i.e., unless stated otherwise from context.

4.2.1.2 Optical Media

4.2.1.2.1 Ambient

The ambient region refers to that spatial region which lies external to the layered structure of the sample, i.e., the (layers/substrate) system. Distinct refractive indices may be due to dispersion or fluid that is inside some filled cell containing the submersed sample. Although the corrections for the window of the cell would be important, such considerations are beyond the scope of presentation here, and the reader is well advised to refer to the outside literature. The format for entering the indices of refraction for the ambient is of the same form as that discussed for entering wavelengths, i.e.,

\[ m_a / (i, n_i) \]

where:

- \( m_a \) is the total number of distinct refractive indices in the ambient region;
- \( i \) is an integer that indexes the refractive indices consecutively in unit increments, i.e., \( i = 1, 2, 3, \ldots, m_a \); and
- \( n_i \) is the index of refraction of the ambient at one particular wavelength.

To demonstrate the use of the above format, consider a case where ellipsometric measurement data were collected on a sample that had been placed in at least two distinct ambients, e.g., air and vacuum. Further, let the incident wavelengths be those mentioned in the previous example. Then, the format could be the following:

```
3                  ! ambient - number of distinct ambients
1  1.0              !   vacuum
2  1.00027           !   air at 632.8 nm, HeNe laser line
3  1.00027           !   air at 441.6 nm, HeCd
```

This involves four lines of input data. Apart from the comments that are appended to the above lines of data, the format does not associate optical properties with wavelength. This
will be accomplished later in section 4.2.2, where the construction of the sample’s layered structure from the constituent model parameters is discussed.

4.2.1.2.2 Layers/Substrate

For each optical medium that contributes toward the construction of the layered structure of the sample, one requires the numerical value of the index of refraction and extinction coefficient for each wavelength of incident light. For convenience, the word 'element' or its mnemonic 'lmnt' is used to associate some form of indexing of the numerical values that are assigned to the optical properties at distinct wavelengths.

As discussed 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.

The format for entering the optical properties of the layers/substrate is:

\[ m_{lmnt} / (i, n_i, \delta n_i, v_i) \]

where:

- \( m_{lmnt} \) is the total number of distinct 'elements' involving refractive indices and extinction coefficients for each distinct type of optical media and wavelength;
- \( i \) is an integer that indexes the line entries of data consecutively in unit increments, i.e., \( i = 1, 2, 3, \ldots, m_{lmnt} \);
$n_i$ is the numerical value that is assigned to either an index of refraction or extinction coefficient of some particular optical medium at some particular wavelength;

$\delta n_i$ is the magnitude of uncertainty that is assigned to the numerical value of the model parameter $n_i$; and

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

To demonstrate this format, consider the well-known case involving a silicon substrate that supports a thermally grown oxide layer. Further, assume that the wavelengths associated with the measurements were those mentioned earlier from the example in section 4.2.1.1. The format could then be something like the following:

```
7
1 3.882 0.002 0 ! nmnts " (n,k | wavelength, element)
2 0.019 0.002 0 ! n,nu,ivary " Silicon at 632.8 nm, HeNe
3 1.457 0.002 0 ! n,nu,ivary " SiO2 amorphous glass
4 0.0 0.0 1 ! k,ku,ivary " SiO2 " at HeNe or HeCd
5 4.753 0.002 0 ! n,nu,ivary " Silicon at 441.6 nm, HeCd
6 0.163 0.002 0 ! k,ku,ivary " Silicon
7 1.466 0.002 0 ! n,nu,ivary " SiO2 amorphous glass
```

which involves only eight lines of input data. This example involves four sets of information. It contains the optical properties of crystalline silicon and amorphous silicon dioxide at two distinct wavelengths. This is a simple tabulation of numerical values; ordering is not important among the last seven line entries. Note that no reference to wavelength is indicated here; thus the specifications regarding the optical properties are still incomplete. The manner in which the optical properties are associated with the appropriate wavelength will be discussed later in section 4.2.2. (Again, that section discusses how the layered structure of the samples is constructed layer by layer from the constituent model parameters as a function of wavelength. That section will conclude the characterization of the sample.)

Lastly, the example shows that one model parameter was selected to have its numerical value undergo variation, i.e., the extinction coefficient for amorphous silicon dioxide at either wavelength. Note also that the uncertainty value was set to zero. This condition is check-tested upon input, so the user would be notified. (Setting the uncertainty value to
a nonzero value circumvents any 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.

4.2.1.3 Layer Thickness

The program also requires the thicknesses of the layers that contribute to the layered structure of the samples. The unit of thickness is nanometers. Again, in similar fashion as that presented in the previous subsection, the format is of the form:

\[ m_z / (i, z_i, \delta z_i, v_i) \]

where:

- \( m_z \) is the total number of distinct thicknesses that contribute to the layered structure of the sample;
- \( i \) 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;
- \( \delta z_i \) is the magnitude of uncertainty in knowing the thickness \( z_i \); and
- \( v_i \) is the integer (froz/vary) switch with value 0 or 1, respectively.

To demonstrate this format, let the following example build upon information presented from previous examples. Consider a sample involving two layers atop a substrate of silicon, where one layer is atop the other layer. Let the layer adjacent to the substrate be a thermally grown oxide, and let the layer adjacent to the ambient region be a layer of silicon. Then, the layered structure is of the ordered form:

(ambient / silicon / oxide / silicon substrate).
Suppose further that the top layer of silicon is 50 nm thick, that the oxide layer is 100 nm thick, and that a reasonable initial estimate of the uncertainty is subjectively chosen to be 2 nm. This construct would require three lines of input data. An example of the format may be as the following:

```
2               ! mfilm ' thicknesses
1  50.0  2.0   0   ! z,mi,ivy ' top layer, Si
2  100.0  2.0   1  ! z,mi,ivy ' bottom layer, Si02
```

This tabulation would include the distinct thickness from all samples; thus ordering among thicknesses is not important here. Further, note that this format does not associate thickness with a sample or with ordering of layers on a sample. The ordering of the layers is discussed in the next section. And lastly, one may see from the above example that the oxide thickness has been judiciously chosen to undergo variation.

### 4.2.2 Sample Characterization or Construction

Since the basic constituent model parameters have been presented, it is now possible and necessary to discuss how: (1) the layered structure of each sample is constructed from the constituent model parameters, (2) the wavelengths are associated with the optical properties, and (3) the ordering of the wavelengths affects the ordering of the input of the measurement data. The last item is discussed in the next section, i.e., section 4.2.3.

First, it is important to discuss the meaning of the word ‘sample.’ A sample is defined as being that structure of the (layers/substrate) system that is subjected to ellipsometric measurement. Characterization of the sample refers to the index-labeling procedure that associates the spatial regions of the (ambient/layers/substrate) system with the appropriate model parameters at given wavelengths, so that the forward scattering problem is completely defined. Measurements on a sample may involve one or more wavelengths, as well as one or more ambients. For any given sample, the optical properties may change as a function of wavelength, i.e., exhibit dispersion, whereas layer thicknesses remain unchanged. Here, the representation of the optical media involves a distinct model parameter per distinct wavelength. A sample is constructed in a layer-by-layer fashion by specifying the optical properties for each spatial region for each incident wavelength. Admittedly,
this parameterization of optical media is by design suitable only for analyzing media involving very few wavelengths, very few ambients, but many-angle of incidence measurement data.

Recall, the inverse scattering 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. Accordingly, 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 nsamp1, as mentioned in section 4.1.2.

Recall again that the previous section discussed formats for inputting model parameters. Each model parameter has a distinct index label; each line entry following the first line is indexed. The actual ordering among the line entries within any given set of model parameters is relatively unimportant, i.e., apart from wavelength. Here, the index labels are used for specifying the configuration of the (ambient, wavelength, sample) system. The sample is characterized by an ordered set of integers that point into arrays that store the numerical values associated with the model parameters.

For convenience, the integers are ordered in a format compatible with the collection algorithm of the laboratory instrumentation. Here, measurements are collected by scanning the angles of incidence for each given wavelength. 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:

\[(\text{angles, repeats, ambients, wavelengths, samples})\]

where:

- \textit{samples} indexes the set of distinct samples that were subjected to ellipsometric measurement;

- \textit{wavelengths} indexes the set of distinct wavelengths used during measurement on a given sample;

- \textit{ambients} indexes the set of distinct ambients used during measurement involving a given wavelength and sample;
repeats indexes the sets of distinct repeats of multiple-angle measurements performed on a system of given ambient, wavelength, and sample; and

angles indexes the sets of distinct angles of incidence used during measurement on a system of given repeat index, ambient, wavelength, and sample.

This form suggests that the samples are considered individually. First, the sample is specified; then, the incident wavelengths are specified; this is followed by the specification of the ambient, i.e., the value of its refractive index at the associated wavelength. Last, the angle of incidence is specified to completely define the forward problem. The repeats serve as convenient (or artificial) partitioning of the multiple-angle of incidence data, e.g., distinguishing data collected on different days.

To implement the above organization and thereby characterize the (ambient, wavelength, sample) system, the program reads a tabulated set of integers line by line. The first line of data specifies the number of samples; this is denoted by \( m_{\text{samples}} \). This is followed by a series of lines that are grouped in a sample-by-sample manner.

For convenience, consider the lines that are associated with the \( j^{\text{th}} \) sample. The first line of data for this subset should specify: the index label, the number of layers, and the number of wavelengths. These are denoted, respectively, by: \( j_{\text{sample}} \), \( m_{z,j} \), and \( m_{\lambda,j} \). The next line should specify the wavelength and number of ambients that have been involved with this sample and wavelength. The two integers are denoted, respectively, by: \( \lambda_{i,j} \) and \( m_{\alpha,ij} \), where the first entry is the index label for the wavelength, and \( i \) is a local index of the ordered wavelengths, i.e., \( i = 1, 2, 3, \ldots, m_{\lambda,j} \).

Now that the sample and wavelength have been indicated, consider next the structure of the layers/substrate. The layers/substrate are considered in succession, i.e., starting with the top layer or layer 1, which is adjacent to the ambient region. Each layer/substrate region is associated with a line of integer data. The integers include the index labels for the layer, refractive index, extinction coefficient, and thickness, in that order. This is indicated, respectively, by: \( l, \tilde{n}_{l,ij}, \tilde{k}_{l,ij}, \) and \( \tilde{z}_{l,ij} \), where the tilde refers to that implying index labels. The substrate assumes a layer index label of \( m_{z,j} + 1 \), but no index label for thickness is required. This completes the specification of the layers/substrate system for the \( j^{\text{th}} \) sample.
The next set of $m_{a,ij}$ lines of data is used to specify the ambients and repeats, i.e., one line for each ambient. Each line should contain two integers; they are the index label of the ambient at associated wavelength and the number of repeats of the multiple-angle of incidence data. These are denoted, respectively, by: $a_{\ell,ij}$ and $r_{\ell,ij}$, where $\ell$ is a local index of the ambients, i.e., $\ell = 1, 2, 3, \ldots, m_{a,ij}$. Here, a repeat value of 1 is associated with one set of measurements, i.e., not two. This completes the specification of the (ambient/layers/substrate) system, as well as the implied organization of the collection of measurement data.

Regarding the organization for these specification data, the general format is given by the following form:
\( m_{\text{samples}} \)

\[
\begin{array}{cccc}
1 & \cdots & m_{z,i} & m_{\lambda,i} \\
\vdots & & \vdots & \vdots \\
(j-1)_{\text{sample}} & \cdots & \tilde{\lambda}_{1,i} & \cdots \\
\vdots & & \vdots & \vdots \\
 j_{\text{sample}} & m_{z,i} & m_{\lambda,i} & m_{a,i} \tilde{\lambda}_{1,i} \tilde{\lambda}_{2,i} \cdots \tilde{\lambda}_{m_{\lambda},i} \\
\vdots & \vdots & \vdots & \vdots \\
 m_{z,i} & \tilde{\eta}_{i,i} & \tilde{k}_{i,i} & \tilde{z}_{i,i} \\
(m_{z,i}+1) & \tilde{\eta}_{m_{z},i} & \tilde{k}_{m_{z},i} & \tilde{z}_{m_{z},i} \\
\tilde{\alpha}_{1,i} & r_{1,i} \tilde{r}_{1,i} \tilde{r}_{1,i} \cdots \tilde{r}_{m_{\alpha},i} & \tilde{\alpha}_{m_{\alpha},i} & r_{m_{\alpha},i} \\
\vdots & \vdots & \vdots & \vdots \\
(j+1)_{\text{sample}} & \cdots & \cdots & \cdots \\
\vdots & \vdots & \vdots & \vdots \\
m_{\text{sample}} & \cdots & \cdots & \cdots \\
\end{array}
\]
where:

$m_{\text{samples}}$ is the total number of distinct samples that are/were subject to ellipsometric measurement.

$j_{\text{sample}}$ is an integer that indexes the samples consecutively in unit increments, i.e., 
\[ j = 1, 2, 3, \ldots, m_{\text{samples}}. \]

$m_{z,j}$ is the total number of layers that lie atop the substrate of sample $j$. The top layer is adjacent to the ambient and is indexed locally as being equal to one. The bottom layer is adjacent to the substrate and is indexed locally as being equal to $m_{z,j}$.

$m_{\lambda,j}$ is the total number of distinct wavelengths that are used in the measurements involving sample $j$.

$\lambda_{i,j}$ is the integer index label of the appropriate wavelength that is associated with the $i^{\text{th}}$ wavelength incident on sample $j$. Note that $i$ is indexed locally, i.e., 
\[ i = 1, 2, 3, \ldots, m_{\lambda,j}. \]

Further, it is requested that the set of measurement data be ordered upon input so that the wavelengths be decreasing with increasing index, or equivalently, that the index label $\lambda$ be monotonically increasing with index, i.e., 
\[ (\lambda_{i,j} < \lambda_{i+1,j}). \]

$m_{a,ij}$ is the total number of distinct ambients that involve the $i^{\text{th}}$ wavelength incident on sample $j$.

$l$ is an integer local index of the layers and substrate of the sample, i.e., 
\[ l = 1, 2, 3, \ldots, m_{z,j}, m_{z,j}+1. \]

The layers are indexed consecutively; the layer adjacent to the ambient is labeled 1; the layer adjacent to the substrate is labeled $m_{z,j}$, and the substrate is labeled $m_{z,j}+1$.

$\hat{n}_{i,ij}$ is the integer index label of the appropriate index of refraction that involves the $l^{\text{th}}$ layer and the $i^{\text{th}}$ wavelength incident on sample $j$.

$\hat{k}_{l,ij}$ is the integer index label of the appropriate extinction coefficient that involves the $l^{\text{th}}$ layer and the $i^{\text{th}}$ wavelength incident on sample $j$.

$\hat{\delta}_{l,ij}$ is the integer index label of the appropriate thickness of the $l^{\text{th}}$ layer of sample $j$. 

46
\( \tilde{n}_{s,ij} \) is the integer index label of the appropriate index of refraction of the substrate that involves the \( i^{th} \) wavelength incident on sample \( j \).

\( \tilde{k}_{s,ij} \) is the integer index label of the appropriate extinction coefficient of the substrate that involves the \( i^{th} \) wavelength incident on sample \( j \).

\( \tilde{\alpha}_{s,ij} \) is the integer index label of the index of refraction of the \( \ell^{th} \) ambient that is used with the \( i^{th} \) wavelength incident on sample \( j \). Here, \( \ell \) is indexed locally, i.e., \( (\ell = 1, 2, 3, \ldots, m_{s,ij}) \). Furthermore, it is requested that the measurement data be ordered upon input so that \( \tilde{\alpha}_{s,ij} < \tilde{\alpha}_{s+1,ij} \).

\( r_{s,ij} \) is the total number of sets of \textit{repeated} measurements that involve the \( \ell^{th} \) ambient with the \( i^{th} \) wavelength incident on sample \( j \). Note that a \textit{repeat} value of 1 relates to reading one set of data, i.e., not two. Each set involves a collection of multiple-angle measurements.

Again, the optical properties become associated with the appropriate wavelength via explicit use of index label. This simplifies any inherent coupling between distinct samples. Distinct samples may have one or more model parameters in common.

To demonstrate this format, consider again the example presented in the previous section that discussed layer thicknesses, i.e., section 4.2.1.3. There, the sample involved two layers, i.e., a top layer of silicon, a layer of amorphous silicon dioxide, and a substrate of silicon. Suppose that this sample is measured by ellipsometry using wavelengths from the two aforementioned laser lines. Assume further that sets of multiple-angle measurements were performed in air, as well as in vacuum. After performing these measurements, suppose that this sample is subjected to some etchant, which is able to remove the top layer and leave the oxide layer unaffected. Let this become the ‘second’ sample. Let this ‘second’ sample be subjected to a similar set of ellipsometric measurements, but only in air. Note that the model parameters of the oxide layer and substrate are common to both samples. The format for this example could be as the following.

\[
\begin{array}{ccccccc}
2 & 1 & 2 & 2 & m_{\text{sample}}^{\text{number of samples}} \\
1 & 2 & n_{\text{sample}}, m_{\text{film}}, m_{\text{wave}} \\
1 & n_{\text{ambient}} \quad \text{\textit{(HeNe)}} \\
1 & 1 & 2 & 1 & \text{\textit{Si}, 50 nm} \\
2 & 3 & 4 & 2 & \text{\textit{SiO}_2, 100 nm} \\
3 & 1 & 2 & \text{\textit{Si}, substrate} \\
1 & 1 & \text{\textit{ambient}, mrpeat \quad \text{\textit{vacuum}}} \\
\end{array}
\]
<table>
<thead>
<tr>
<th>ambient, mrpeat</th>
<th>air</th>
<th>iwave, mbient</th>
<th>HeCd</th>
<th>n,k,z</th>
<th>Si, 50 nm</th>
<th>Si02, 100 nm</th>
<th>n,k</th>
<th>Si, substrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>imbieitiit</td>
<td>2 2</td>
<td>1</td>
<td>2</td>
<td>3 5</td>
<td>3 1</td>
<td>2 1 2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5 6 1</td>
<td>2</td>
<td>4 2</td>
<td>1</td>
<td>1</td>
<td>7 4 2</td>
<td>2</td>
<td>5 6</td>
</tr>
<tr>
<td>isample, mfilm, mwave</td>
<td>1 1 2</td>
<td>3 4 2</td>
<td>1 1 2</td>
<td>1 1 2</td>
<td>1 7 4 2</td>
<td>2 5 6</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
\( \alpha \) is an integer that indexes the multiple-angle measurements consecutively in unit increments, i.e., \( \alpha = 1, 2, 3, \ldots, m_{\phi, \delta \psi} \).

\( \phi_\alpha \) is the angle of incidence measured in degrees, where a value of zero relates to that of normal incidence.

\( \delta \phi_\alpha \) is the magnitude of uncertainty in the angle of incidence as determined from calibration of laboratory instrumentation.

\( \Delta_\alpha \) is measured in degrees, i.e., \( 0 \leq \Delta < 360 \), assuming the Nebraska convention, i.e., \( R_{p,H} \) and eq (51).

\( \delta \Delta_\alpha \) is the magnitude of uncertainty associated with measuring \( \Delta_\alpha \).

\( \psi_\alpha \) is measured in degrees, i.e., \( 0 \leq \psi \leq 90 \).

\( \delta \psi_\alpha \) is the magnitude of uncertainty associated with measuring \( \psi_\alpha \).

To demonstrate this format, let the measurements be those found or generated by solving the forward scattering problem for the sample configurations presented in the previous example. Let the measurements involve only a few angles of incidence data. These measurements would be exact. Let the uncertainty associated with \( \phi, \Delta, \) and \( \psi \) be given by 0.01, 0.05, and 0.05 degrees, respectively. Although these values naturally depend upon the instrumentation, their magnitudes are chosen subjectively here for convenience. The format could then be as the following:

\[
\begin{array}{cccccc}
2 & 1 & 1 & m_{\alpha} & m_{\beta} & m_{\psi}(\phi, \delta \psi) (\text{angle}, \text{d_angle}) & ! \text{vacuum} \\
1 & 65.0000 & 0.0100 & 173.5043 & 0.0500 & 37.0579 & 0.0500 & ! \text{HeNe} \\
2 & 70.0000 & 0.0100 & 172.2664 & 0.0500 & 34.3707 & 0.0500 & ! \text{HeNe} \\
3 & 2 & 1 & m_{\alpha} & m_{\beta} & m_{\psi}(\phi, \delta \psi) (\text{angle}, \text{d_angle}) & ! \text{air} \\
1 & 65.0000 & 0.0100 & 173.5101 & 0.0500 & 37.0527 & 0.0500 & ! \text{HeNe} \\
2 & 70.0000 & 0.0100 & 172.2750 & 0.0500 & 34.3633 & 0.0500 & ! \text{HeNe} \\
3 & 75.0000 & 0.0100 & 170.0354 & 0.0500 & 30.1573 & 0.0500 & ! \text{HeNe} \\
2 & 2 & 1 & m_{\alpha} & m_{\beta} & m_{\psi}(\phi, \delta \psi) (\text{angle}, \text{d_angle}) & ! \text{repeat} \\
1 & 67.5000 & 0.0100 & 172.9647 & 0.0500 & 35.8405 & 0.0500 & ! \text{HeNe} \\
2 & 72.5000 & 0.0100 & 171.3529 & 0.0500 & 32.5214 & 0.0500 & ! \text{HeNe} \\
4 & 3 & 2 & m_{\alpha} & m_{\beta} & m_{\psi}(\phi, \delta \psi) (\text{angle}, \text{d_angle}) & ! \text{air} \\
1 & 67.5000 & 0.0100 & 286.8467 & 0.0500 & 30.2644 & 0.0500 & ! \text{HeCd} \\
2 & 70.0000 & 0.0100 & 297.5219 & 0.0500 & 31.2799 & 0.0500 & ! \text{HeCd} \\
3 & 72.5000 & 0.0100 & 307.5732 & 0.0500 & 32.6322 & 0.0000 & ! \text{HeCd} \\
4 & 75.0000 & 0.0100 & 316.8687 & 0.0500 & 34.2225 & 0.0000 & ! \text{HeCd} \\
2 & 2 & 1 & m_{\alpha} & m_{\beta} & m_{\psi}(\phi, \delta \psi) (\text{angle}, \text{d_angle}) & ! \text{air} \\
1 & 70.0000 & 0.0100 & 79.8092 & 0.0500 & 41.0455 & 0.0500 & ! \text{HeNe} \\
2 & 73.0000 & 0.0100 & 68.2502 & 0.0500 & 41.0304 & 0.0500 & ! \text{HeNe} \\
\end{array}
\]
Incidentally, these data are retained in arrays that are located in the named common area, exprmt.

4.2.4 Combining

The input data file is composed of at least 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. The input data file is constructed, i.e., in part, by simply combining the previous formats in the order as presented. Such involves the wavelengths, the model parameters, the characterization of the samples, and the measurement data. No intervening blank lines are allowed.

Combining all the examples as presented thus far, and indexing the lines for convenience, i.e., the data file X.DAT should not actually contain such indexing, the combined format would be:

```plaintext
1 2 3 2 * mangle_inplanation, isample, iwave/(detla, psi) (angle, d_angle) ! air
1 70.0000 0.0100 211.2235 0.0500 77.3289 0.0500 ! HeCd
2 73.0000 0.0100 216.1656 0.0500 84.4928 0.0500

50
```
The last type of input data that must be included in the file X.DAT is that associated with directing the use of the software package. To direct the execution of the software program package, a menu-driven decision-tree of command options is made available to the user. The first level of the tree involves a menu of five options. Here, one option is selected per execution of the program, and the selection of that option forms the first line of the input to the data file X.DAT, as mentioned previously in section 4.2. The program reads this line and the following lines shown in the previous subsection (4.2.4), i.e., model

4.3 Command Options

<table>
<thead>
<tr>
<th>Line</th>
<th>Command Line</th>
<th>Vacuum</th>
<th>HeNe</th>
<th>Air</th>
<th>Repeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>imbient, mrpeat / repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... reserved for further/other option information
parameters, sample characterization, and measurement data, where appropriate. After reading this input of data, the program branches to the appropriate subroutines. Upon completion from these routines (at level one), the main program stops. Again, as was mentioned earlier, the main program does not loop back to a previous condition where a request is made of the user to select another option at level one.

Before reading the first line of the input file, the program writes to the output file X.DAT the following menu of available options:

```
options:  1, forward problems, plots, ...
         2, search ( vary)
         3, search grid ( vary)
         4, search grid (froz, vary)
         5, sensitivity analysis
         6, MAE plots * uncertainty
```

Enter: option

Where the options (level one):

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 forward scattering problem. No iterative calculations are induced, 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.

4, requests the program to invert the ellipsometric equations by performing a grid scan over a selected set of model parameters, where the grid scans are held local to the individual sample. Hence, the coupling between samples is suppressed. A
series of constrained optimization calculations is initiated at each point in the grid scan. This is discussed further in section 4.3.4.

5, requests the program to initiate a sensitivity analysis of selected model parameters. The uncertainty associated with assigning a numerical value to a model parameter is estimated from the covariance matrix elements and the relative contributions of uncertainty from the frozen model parameters. This is discussed further in section 4.3.5.

6, requests the program to determine which selection of multiple angles (of incidence) minimizes the calculated magnitude of uncertainty in the numerical value of some selected model parameter. This involves a grid scan over groups/sets of incident angles. The magnitude of uncertainty is calculated as a function of these groups/sets. This is discussed further in section 4.3.6.

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, option five is more useful towards the conclusion of analysis, while options two and three form the primary tools during analysis. Option four provides a measure of convenience with uncoupling samples. This not only reduces the sizes of the matrices, but also this tends to remove redundant calculations from the grid scan. Since the grid scans are constructed from a simple nesting of DO-loops, a reduction may be expected to occur whenever the grid scans involve model parameters which are not common to every sample. This reduction and savings can become significant.

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, ...

Upon selection of option one (at level one), the main program calls subroutine PLTDAT. The routine is partitioned to serve five distinct tasks. Hence, another menu of options is afforded to the user. Regarding the calling sequence of routines as initiated from the main program, this involves level two. Accordingly, the menu of options is:

\texttt{PLTDAT, forward scattering problems, plots, ...}

This menu of options is at level two.
options: 1, plot (z,n,k)  
2, plot deviations of fit,  \[ g = \text{experiment-model} \]  
3, model \[ \rightarrow \text{experiment/measurement data}, \]  
\quad use model to simulate measurement data, \newline  \quad \text{format is suitable to reading as input,}  \quad \newline  \quad (\text{angles, repeats, ambient, waves, samples})  
4, use model at measurement/scan points,  
\quad request another menu to format output,  
\quad output \quad ^{\prime} \quad (\text{delta, psi, intensity, R, dR, ...})  
5, scan grid, plot \mid g \mid, \quad 1\text{D or 2D}  

Enter: \quad \text{option} \quad ^{\prime} \quad  

where the options (level two):  

1, requests the program to output the profile of the index of refraction and extinction coefficient as a function of depth through the layered structure of the sample. The simple format is intended to be suitable for later plotting. The output is written to the plot file X.PLOT.  

2, 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.  

3, requests the program to use the model to simulate measurement data. The format is suitable for use in file X.DAT. The output is written to both files, X.OUT and X.PLOT.  

4, requests the program to use the model to calculate some selected field quantity. The selections include: the ellipsometric angles, reflected field intensities, and matrix elements of the Jacobian. The output is written to file X.PLOT. To assist the selection process, two menus are provided. These are presented later in this section.  

5, requests the program to initiate a grid scan of model parameters and evaluate the error expression \[ |g| \] for each point on the grid. Here, the grid is either one or two dimensional; i.e., only one or two model parameters may undergo variation. Here, it is necessary to specify the domain of the grid. The format for this input is presented later in this subsection. The output is written to file X.PLOT.  

As mentioned above, regarding options four and five, both present menus to assist the se-
lection process in the decision-tree of options. These menus are presented next.

For the case regarding option four (level two), the program responds by presenting the first of two menus. The first menu presents a list of selected field quantities that are available for formatted output.

**PLTDAT**, select the format of the output table.
This menu of options is at level three.

**options:**

1. \((i, ai, \text{delta}, \text{psi}, \text{intensity})\)
2. \((i, ai, |R_s|, |R_p|, |R_s|^2, |R_p|^2)\)
3. \((i, ai, |R_s|, |R_p|, \text{intensity})\)
4. \((i, ai, \text{d/d parameter} (\text{delta}, \text{psi}))\)

Enter: option

where:

1. involves the ellipsometric angles and the intensity \(I\), where \(I = \frac{1}{2}(|R_s|^2 + |R_p|^2)\).
2. involves the magnitude and intensity of the field amplitude for each polarization.
3. involves the magnitude of the field amplitudes and the total intensity.
4. involves the matrix elements of the Jacobian. The partial derivative is specified by the (froz/vary) switch of the model parameter. Either zero or one model parameters may be selected as undergoing variation. For the case involving zero model parameters undergoing variation, the partial derivative is taken with respect to the angle of incidence. Here, the Jacobian matrix elements are expressed in the physics convention, i.e., not the engineering convention. Only that associated with \(\Delta\) is affected, i.e., inducing a minus sign.

\(i\), refers to the local indexing of the multiple angles of incidence, which involves one given configuration of the system, i.e., \((\text{repeat, ambient, wavelength, sample})\).

\(ai\), refers to the angle of incidence measured in degrees.

Continuing the case regarding option four (level two), the second menu of options (level three) involves selecting the appropriate grid of incident angles used in the tabulation.

**PLTDAT**, select the domain of incident angles.
This menu of options is at level three.
options: 1, at experiment/measurement points
        2, grid scan, incident angles *(1,89)
        
Enter: option

where the options refer to the incident angles associated with

1, the measurement data of ellipsometric angles \((\Delta, \psi)\), and

2, a grid, indexed from 1 to 89, with stepsize of one (degree). Such is usually more
than adequate for plotting the field as a function of incident angle.

Finally, consider the case involving option five (level two). Here, the domain of the grid
needs to be specified. Since each dimension of the grid involves one of the distinct model
parameters, it is necessary to specify: the model parameters, their bounds, and their step-
sizes. The model parameters are selected via the (froz/vary) switch; only those indicated
as varying contribute to the grid. Since the plots are either one or two dimensional, only
one or two model parameters may have numerical values being indicated as varying. The
format for reading this grid information follows the convention of DO-loop specification,
and is given by the form:

\[ i_p, p_1, p_2, p_3, \]

where \( i_p \) refers to the index label of the model parameter \( p \), and \( p_j \) refers to numerical
value of the model parameter associated with the initial value, final value, and stepsize.
Further, it is required that the lines be ordered according to the input data requirements
presented earlier in section 4.2, i.e., optical media first, then thicknesses.

4.3.2 Search (vary)

Upon selection of option two (level one), the main program calls subroutine ZOOM. This
routine initiates and maintains the search for a minimum to the error expression as an un-
constrained optimization problem. 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 \((\Delta, \psi)\), so that their associated rows in the Ja-
cobian 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 sec-
tion 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. Such has been discussed earlier in section 3.

The following presents a brief orientation regarding the internal organization of the routine ZOOM. 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, breakpointing is not necessary, and so it was not incorporated into the routine.

Upon completion of the above tasks, 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 PLTDAT. Statistics regarding the deviations are provided by the routine STAT22. It reports the statistical means, standard deviations, and the correlation coefficient of the deviations $g$. Here, the mean involving $\Delta$ is defined by

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

the 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, and the covariance is defined by

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

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 $g_{*,i}$.

A correlation matrix, as defined by eq (81), is calculated by subroutine CORLAT. Being a symmetric matrix, the upper triangle is reported to the output file, X.OUT. The output file also reports the condition number of this matrix [17], as well as the diagonal elements
of the renormalization matrix as defined by eq (63). Such helps identify which model parameters may be correlated and aids the decision process regarding the selection of frozen model parameters during a series of calculations. Upon completing these tasks, the program stops.

4.3.3 Search Grid (vary)

Upon selection of option three (level one), the main program calls subroutine SCAN2 to invert the ellipsometric equations. It 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 dimension to the grid, i.e., a hyper-cube. Note that efficient use of the grid occurs when the selected model parameters are common to all samples.

Again, each dimension of the grid involves four items: 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 request/require that the input format to be of the form:

\[ i_p, p_1, p_2, p_3, \]

where \( i_p \) refers to the index label associated with 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. Also, it is required that the lines of data be ordered according to the input data requirements presented earlier in section 4.2, i.e., optical media first, then thicknesses.

For each point on the multidimensional grid, the calculation initiates a series of unconstrained optimization problems, except that the range of numerical values of the selected model parameters is restricted within the bounds of the grid. At the conclusion of the grid scan, when the program reports its best fixed-point solution, the program also reports any components of the solution which 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.
Since grid specification includes the stepsize, i.e., number of steps, it is possible for calculations to become very time-consuming. For this reason, the routine provides breakpoint information to the output file X.SOUT at intervals of 15 cpu-minutes. This is discussed in section 4.1.1. The mechanism 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.DAT, without any intervening blank lines. Upon starting any grid scan calculation, the program will attempt reading a set of breakpoint information. If restarting is not intended by the user, the input data file should be absent of excess lines. The program does initiate some measure of fail-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 connected line of four equal signs.

Again, as presented in the previous subsection, i.e., section 4.3.2, the routine reports only a basic set of statistics regarding deviations of the fit between the measurement data and that of the model.

4.3.4 Search Grid (froz, vary)

Option four (level one) is one example of a specialty algorithm. It is tailored to a specific need of the user regarding the analysis of a particular class of problems and, as such, may find limited applicability with general problems, apart from providing an outline for the user to modify the program to address one’s own particular needs.

Here, the program calls subroutine SCAN3 to invert the ellipsometric equations by constructing two nested blocks of distinct grid scans, where the inner-block grid assumes that the samples are uncoupled. The first/inner block of grids is similar to that mentioned earlier in section 4.3.3, the so-called vary grid. Each sample is scanned independently of other samples, so the coupling between samples is suppressed, while coupling within the sample is fully accounted. At each point of the grid and sample, the routine initiates a calculation of the constrained optimization problem.

The outer block of grids involves a selected subset of model parameters whose numerical values are considered frozen. For each point of the so-called froz grid, the program scans the appropriate vary grid of each sample, and the results of calculation are reported to the output file, X.OUT. Consequently, this can lead to substantial listings of output. Note,
the coupling between samples involving model parameters contributing to the *froz* grid is fully accounted.

Breakpointing capability is provided with the routine. Upon completion of the above tasks, a basic set of statistics regarding deviations of the fit between measurement and the model is provided. If coupling exists between samples regarding the *vary* grid of model parameters, the simple statistics that are generated at conclusion of calculation will generally be meaningless, because the samples were treated independently; i.e., distinct samples maintain distinct values associated with the same model parameter.

The capability provided by this option has been found toward generating surfaces of the error expression for purposes of later graphics. Note, it is a simple matter to dupe the program and induce an uncoupling among the samples during analysis by selecting one model parameter to compose the *froz* grid and set its lower/upper bound to the same numerical value.

To specify which model parameters contribute to the *froz* grid, one need merely set the (*froz*/*vary*) switch to an integer value of two and include the specification of its grid within the ordering of the input file as discussed earlier in section 4.3.3.

**4.3.5 Sensitivity Analysis**

Option five (level one) induces the program to initiate a sensitivity analysis regarding the model parameters whose numerical values were selected to undergo variation, as discussed in section 3.2. This calculation is performed by the subroutine SEAMA.

The output includes the following, where:

\( <gg> \), refers to the statistical estimate of the variance of the deviations involving both \( \Delta \) and \( \psi \) as defined by eq (85). The standard deviation is included for convenience.

\( <aa> \), refers to the statistical estimate of the variance of the uncertainty in the angle of incidence associated with laboratory instrumentation during measurement, as defined by eq (86). The standard deviation is included as well.
\( |Bg g B|_{jk} \), refers to covariance matrix elements involving the deviations as defined by the first term in eq (91).

\( |B J a a J B|_{jk} \), refers to covariance matrix elements involving the uncertainty in the angles of incidence as defined by the second term in eq (91).

The so-called total uncertainty reported for the vary model parameter is defined by eq (98), which combines contributions from random and systematic errors. Here, the random component involves one of the diagonal elements from the covariance matrix. Also listed are the relative contributions of systematic errors entering into this sum.

\[4.3.6\] MAE Plots of Uncertainties

Option six (level one) is another example of a specialty algorithm. This option serves the purpose of finding the optimum set of multiple angles of incidence which may induce the minimum magnitude of uncertainty associated with some selected model parameter. To accomplish this, the routine scans a grid of incident angles and calculates the magnitude of uncertainty associated with the appropriate model parameter. The formulation of the problem regarding uncertainties is that as reported in the literature [24], which is distinct from that presented in section 3.2. The algorithm combines contributions utilizing absolute values, and so it overestimates the calculated magnitudes. Further, as may be expected, any effects due to correlation here will naturally frustrate the exercise.

This option is delegated to the subroutine SEAMAX. It calls subroutine SEAM2 to construct a table involving the forward problem, i.e., providing predetermined magnitudes to the deviations \( g \), while properly calculating/storing the elements of the Jacobian, for the entire grid of incident angles. As the multidimensional grid is scanned across this collection of multiple angles of incidence, subroutine SEAM3 constructs the forward problem for the appropriate set of incident angles. The associated magnitudes of uncertainty \( |v_j| \) are calculated and written to the output file, X.SOUT. At conclusion of the scan, subroutine POPLAT is used to plot the data. Here, since the calculated magnitudes were found to vary by several orders of magnitude, it was convenient to request subroutine POPLAT to induce a logarithmic transformation onto the magnitudes, where magnitudes are mapped into the range from \(-5\) to \(4\). For the case involving more that two multiple angles of incidence, the routine graphs the so-called Density of States profile associated with the magnitudes. The domain of the graph involves the angle of incidence.

61
The necessary input is read by subroutine INPDAT. No repeat data are allowed; i.e., repeat values are set equal to one. No measurement data are necessary in the input file, X.DAT. The first line of input, associated with this option, is the integer one. The input format for the next following lines is of the form:

\[ h_{\phi,\ell_{ij}}, m_{\phi,\ell_{ij}}, \tilde{a}_{\ell_{ij}}, \tilde{\lambda}_{ij}, j \]

where \( h_{\phi,\ell_{ij}} \) refers to the integer increment or step of the DO-loop associated with the construction of the grid of incident angles, involving the \( \ell \)th repeat, the \( \ell \)th ambient, and the \( \ell \)th wavelength on the \( j \)th sample. The unit step is measured in degrees, where a reasonable integer value is something like 2 degrees. The other symbols are the same as that referred to in section 4.2.3. Note that \( 1 \leq m_{\phi,\ell_{ij}} \leq \text{nanglm} \) as was mentioned earlier in section 4.1.2.

Again, subroutine SEAM2 sets up a tabulation, and subroutine SEAM3 uses it. This design removes redundant calculations of the forward problem, i.e., to hasten execution time, but at the expense of array space. The tabulation requires the use of large arrays, and the algorithms within SCANCC and SEAMX2 tend to overestimate the sizes necessary for said arrays. For these reasons and that due to limited applications of this option, it has been suppressed from the main program.
5. Worked Examples

The following subsections present worked examples of using the program. Each subsection presents one 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, the output lines are shown as indexed for convenience. As mentioned earlier in section 4.1.1, this output file is named X.OUT.

5.1 Search (vary)

This option is discussed in section 4.3.2. The input data file X.DAT is similar to that presented in section 4.2.4, i.e., apart from some minor modifications as discussed below. This option is selected by setting the first line of data in the input file X.DAT to be the integer value '2.' This is shown on line 9 of the output file listing below. Regarding the output on line 31, the model parameter whose numerical value is selected to undergo variation is that associated with thickness of the layer of oxide atop the substrate. The numerical value of the model parameter is set to 110 nm, i.e., the initial value solution as entered in X.DAT.

Recall from section 4.2.3 that the measurement data are exact for the model parameters with numerical values as presented in section 4.2.4. Note that on line 19 of the example presented in section 4.2.4, the oxide thickness is 100 nm, i.e., the true solution. Hence, the initial value solution is displaced from the true solution by 10 nm. The purpose here is to show the rate of convergence of the program. The output below written on line 141 shows that 54 iterations were used to converge to the correct final solution shown on line 144. Attention is directed less toward optimality of iterations than with that necessary to follow convergence.

Again, the output file 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 file X.OUT is given below.

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

<table>
<thead>
<tr>
<th>Enter</th>
<th>option</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>options</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>options</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

**mwaves** = 2, number of distinct wavelengths

| 1 | 632.8000 |
| 2 | 441.6000 |

**mbient** = 3, number of distinct ambient environments and waves.

| 1 | 1.000000 |
| 2 | 1.000270 |
| 3 | 1.000270 |

**mlmnts** = 7, number of distinct: \( n+ik \)

| 1 | 3.8820 | 0.0020 | 0 |
| 2 | 0.0190 | 0.0020 | 0 |
| 3 | 1.4630 | 0.0020 | 0 |
| 4 | 0.0000 | 0.0000 | 0 |
| 5 | 4.7530 | 0.0020 | 0 |
| 6 | 0.1630 | 0.0020 | 0 |
| 7 | 1.4630 | 0.0020 | 0 |

**mfilm** = 2, number of distinct film widths

| 1 | 50.0000 | 2.0000 | 0 |
| 2 | 110.0000 | 2.0000 | 1 |

**msampl** = 2, number of distinct samples

| 1 | 2 | 2 |
| 1 | 2 |
| 1 | 1 | 2 | 1 |
| 2 | 3 | 4 | 2 |
| 3 | 1 | 2 |
| 1 | 1 |
| 2 | 2 |
| 2 | 1 |
| 1 | 5 | 6 | 1 |
| 2 | 7 | 4 | 2 |
| 3 | 5 | 6 |
| 3 | 1 |
| 2 | 1 | 2 |
| 1 | 1 |
| 1 | 3 | 4 | 2 |
| 2 | 1 | 2 |
| 2 | 1 |
| 1 | 7 | 4 | 2 |
| 2 | 5 | 6 |
| 3 | 1 |

64
|   | 2 | 1 | 1 | 1 | 1 | mangl, repeat, ambient, wave, sample/ (phi, delta, psi) |
|---|---|---|---|---|---|---|---|
| 1 | 65.0000 | 0.0100 | 173.5043 | 0.0500 | 37.0579 | 0.0500 |
| 2 | 70.0000 | 0.0100 | 172.2664 | 0.0500 | 34.3707 | 0.0500 |
| 3 | 75.0000 | 0.0100 | 170.0354 | 0.0500 | 30.1573 | 0.0500 |
| 2 | 72.5000 | 0.0100 | 171.3529 | 0.0500 | 32.5214 | 0.0500 |
| 4 | 75.0000 | 0.0100 | 316.8887 | 0.0500 | 34.2225 | 0.0000 |

|   | 2 | 1 | 2 | 1 | mangl, repeat, ambient, wave, sample/ (phi, delta, psi) |
|---|---|---|---|---|---|---|---|
| 1 | 67.5000 | 0.0100 | 286.8467 | 0.0500 | 30.2644 | 0.0500 |
| 2 | 70.0000 | 0.0100 | 297.5219 | 0.0500 | 31.2799 | 0.0500 |
| 3 | 72.5000 | 0.0100 | 307.5732 | 0.0500 | 32.6322 | 0.0000 |
| 4 | 75.0000 | 0.0100 | 316.8887 | 0.0500 | 34.2225 | 0.0000 |

|   | 2 | 1 | 3 | 2 | mangl, repeat, ambient, wave, sample/ (phi, delta, psi) |
|---|---|---|---|---|---|---|---|
| 1 | 70.0000 | 0.0100 | 79.8092 | 0.0500 | 41.0455 | 0.0500 |
| 2 | 73.0000 | 0.0100 | 68.2502 | 0.0500 | 41.0304 | 0.0500 |

|   | 2 | 1 | 3 | 2 | mangl, repeat, ambient, wave, sample/ (phi, delta, psi) |
|---|---|---|---|---|---|---|---|
| 1 | 70.0000 | 0.0100 | 211.2235 | 0.0500 | 77.3289 | 0.0500 |
| 2 | 73.0000 | 0.0100 | 216.1556 | 0.0500 | 84.4928 | 0.0500 |

zoom: loop, ratio of reduction, |g| (rel) (total) (degrees)

<p>|   |   |   | 1.487E+01 |
|---|---|---|---|---|
| 87 | 0 | 9.034E-01 | 9.034E-01 | 1.344E+01 |
| 88 | 1 | 8.729E-01 | 7.888E-01 | 1.173E+01 |
| 89 | 2 | 8.138E-01 | 6.418E-01 | 9.546E+00 |
| 90 | 3 | 7.986E-01 | 5.128E-01 | 7.623E+00 |
| 91 | 4 | 7.926E-01 | 4.061E-01 | 6.041E+00 |
| 92 | 5 | 7.893E-01 | 3.206E-01 | 4.768E+00 |
| 93 | 6 | 7.907E-01 | 2.353E-01 | 3.770E+00 |
| 94 | 7 | 7.918E-01 | 2.007E-01 | 2.985E+00 |
| 95 | 8 | 7.937E-01 | 1.593E-01 | 2.369E+00 |
| 96 | 9 | 7.953E-01 | 1.267E-01 | 1.884E+00 |
| 97 | 10 | 7.961E-01 | 1.008E-01 | 1.500E+00 |
| 98 | 11 | 7.971E-01 | 8.038E-02 | 1.196E+00 |
| 99 | 12 | 7.976E-01 | 6.411E-02 | 9.536E-01 |
| 100 | 13 | 7.982E-01 | 5.117E-02 | 7.612E-01 |
| 101 | 14 | 7.986E-01 | 4.087E-02 | 6.079E-01 |
| 102 | 15 | 7.989E-01 | 3.265E-02 | 4.856E-01 |
| 103 | 16 | 7.991E-01 | 2.609E-02 | 3.880E-01 |
| 104 | 17 | 7.994E-01 | 2.085E-02 | 3.102E-01 |
| 105 | 18 | 7.995E-01 | 1.667E-02 | 2.480E-01 |
| 106 | 19 | 7.995E-01 | 1.333E-02 | 1.983E-01 |
| 107 | 20 | 7.995E-01 | 1.066E-02 | 1.585E-01 |
| 108 | 21 | 7.999E-01 | 8.525E-03 | 1.268E-01 |
| 109 | 22 | 7.999E-01 | 6.525E-03 | 1.268E-01 |</p>
<table>
<thead>
<tr>
<th>model parameter value along the minimum:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 100.0000  0.00000  for:  2, (z ), estimated uncertainty</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statistics of deviations “ experiment-model ” g</th>
</tr>
</thead>
<tbody>
<tr>
<td>where:  g “ column array of length “ 2M</td>
</tr>
<tr>
<td>let: () “ (psi or delta) ” (1 or 2)</td>
</tr>
</tbody>
</table>

| 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 / [std dev (psi) * std dev (delta)] |

| psi:  0.0000  0.0000  (degrees) |

---
delta:  0.000  0.000
        0.595 "correlation coefficient" <psi|delta>

J(T)*J:  (renormalized for correlation)
         1)  1.00000

Normalization coefficients:  \sqrt{[J(T)*J](i,i)}
         8.35E-02

rcond=  1.000E+00, condition number

elapsed cpu-time = 38 centi-seconds
         + 16 seconds
5.2 Search Grid (vary)

This option is discussed in section 4.3.3. The option is selected by entering an integer value of ‘3’ on the first line in the input data file X.DAT. This is indicated on line 9 of the output file listing below. Again, only the oxide layer thickness is subjected to variation. Line 87 shows the necessary parameters governing the formation of the grid. Note that the grid values overstep the correct solution. For each grid point, the program initiates a series of iterations similar to that shown in the previous subsection. The solution is presented on line 96; it is the correct solution. The contents of the output file X.OUT is given below.

options: 1, forward problems, plots, ...
2 2, search ( vary)
3 3, search grid ( vary)
4 4, search grid (froz,vary)
5 5, sensitivity analysis
6 6, MAE plots * uncertainty

Enter: option ”
    option ” 3

mwaves = 2, number of distinct wavelengths
1 632.8000
2 441.6000

ambient = 3, number of distinct ambient environments and waves.
1 1.000000
2 1.000270
3 1.000270

mlmnts = 7, number of distinct: n+ik
1 3.8820 0.0020 0
2 0.0190 0.0020 0
3 1.4670 0.0020 0
4 0.0000 0.0000 0
5 4.7530 0.0020 0
6 0.1630 0.0020 0
7 1.4680 0.0020 0

mfilm = 2, number of distinct film widths
1 50.0000 2.0000 0
2 100.0000 2.0000 1

msampl = 2, number of distinct samples
1 2 2  sample, mfilm, mwave
1 2  iwave, mbien
1 1 2 1 i,n,k,z
2 3 4 2 i,n,k,z

68
Scan a grid of model parameters.

Grid info: initial, final, increment

1)  95.0000  105.0000  10.0000  for:  2, (z )

Note: NO attempt was made to restart.
number of grid points scanned, \( k_t = 2 \)
population along the minimum, \( k_{ts} = 1 \)
norm of residual, \( |g| = 5.30715E-05 \) (degrees)

Model parameter value along the minimum:
1) \( 1.000000E+02 \) for: 2, \( \alpha \)

----------------------------------------------

Statistics of deviations = experiment-model \( g \)

\[
\text{where: } g \text{ is column array of length } 2M \\
\text{let: } () \text{ is } (\psi \text{ or } \delta) \text{ (1 or 2)} \\
\text{mean () } = m() = <g()> = (1/M) \text{ sum: } g() \\
\text{variance () } = <[g()-m()]^2 > \\
\text{covariance } = <[g(1)-m(1)][g(2)-m(2)] > \\
\text{std dev } = \sqrt{\text{variance}} \\
\text{correlat coef } = \text{covariance } / [\text{std dev } (\psi) \ast \text{std dev } (\delta)] \\
\]

\[
\begin{align*}
\text{mean,} & \quad \text{std dev (degrees)} \\
\psi: & \quad 0.000 \quad 0.000 \\
\delta: & \quad 0.000 \quad 0.000 \\
\text{0.274 correlation coefficient } & \quad <\psi|\delta> \\
\end{align*}
\]

\( J(T)^*J: \) (renormalized for correlation)
1) \( 1.00000 \)

Normalization coefficients: \( \sqrt{J(T)^*J(i,i)} \)
8.35E-02

\( \text{rcond= 1.000E+00, condition number} \)

\( \text{elapsed cpu-time } = 15 \text{ centi-seconds} + 32 \text{ seconds} \)
5.3 Search Grid (froz,vary)

This option is discussed in section 4.3.4. To select this option, the first item entered on the first line of the input data file X.DAT ought to be the integer value '4.' This is indicated on line 9 in the listing presented below. Again, the thickness of the oxide layer is subjected to variation, a vary model parameter, i.e., line 31. Regarding line 24, it is seen that the integer (froz/vary) switch has been set to two, i.e., a grid involving a frozen model parameter. This model parameter is the extinction coefficient of the oxide layer. The associated grid parameters are shown on lines 87 and 88. Here, for convenience, the oxide extinction coefficient is set to consider only a single value, i.e., zero. The correct solution is presented on line 131. The output file is listed below.

```
1 options: 1, forward problems, plots, ...
2 2, search ( vary)
3 3, search grid ( vary)
4 4, search grid (froz, vary)
5 5, sensitivity analysis
6 6, MAE plots ' uncertainty
7
8 Enter: option '
9 option ' 4
10
11 mwaves = 2, number of distinct wavelengths
12 1 632.8000
13 2 441.6000
14
15 mbient = 3, number of distinct ambient environments and waves.
16 1 1.000000
17 2 1.000270
18 3 1.000270
19
20 mlmnts = 7, number of distinct: n+ik
21 1 3.8820 0.0020 0
22 2 0.0190 0.0020 0
23 3 1.4570 0.0020 0
24 4 0.0000 0.0000 2
25 5 4.7530 0.0020 0
26 6 0.1630 0.0020 0
27 7 1.4660 0.0020 0
28
29 mfilm = 2, number of distinct film widths
30 1 50.0000 2.0000 0
31 2 100.0000 2.0000 1
32
33 msamp = 2, number of distinct samples
34 1 2 2 " sample, mfilm, mwave
35 1 2 " iwave, mbien
```
Scan a grid of model parameters.

Grid info: initial, final, increment

1) 0.0000 0.0000 0.0000 for: 4, (n+ik)
2) 95.0000 105.0000 10.0000 for: 2, (z)
Range of do-loops: froz
  1) 1 grid points for: 4, (n+ik)

Range of do-loops: vary
  1) 2 grid points for: 2, (z )

Couplings between samples involve: 1 distinct model parameters
  2, (z )

Note: The existence of coupling between samples due to the VARY model parameters --
induces an unusual interpretation unto the residual |g|, because, while providing coupling, they are NOT
necessarily of similar value on different samples.

Attempt to read breakpoint information

Note: NO attempt was made to restart.

# 1 "grid: froz
  1) 1 0.0000000E+00 "model parameter for: 4, (n+ik)

  sample "grid: vary
  1 3.947890E-05 "|g|
    1.0000000E+02 "for: 2, (z )
  2 1.996443E-04 "|g|
    1.0000000E+02 "for: 2, (z )
    1.084976E-04 "|g| "summed over samples

number of FROZ grid points scanned, ktu = 1
population along the minimum, ktum = 1
norm of residual, |g| = 1.084994E-04 (degrees)

Model parameter value along the minimum: froz
  0.0000000E+00 "for: 4, (n+ik) " boundary

Model parameter value along the minimum: vary
  1.0000000E+02 "for: 2, (z )

Note that the minimum point is near a boundary.

Note: The existence of coupling between samples due to the VARY model parameters --
induces an unusual interpretation unto the residual |g|, because, while providing coupling, they are NOT
necessarily of similar value on different samples.
Statistics of deviations "experiment-model" $g$

where: $g$ "column array of length 2M

let: () "(psi or delta)" (1 or 2)

mean () = $m() = \langle g() \rangle = (1/N) \sum g()$

variance () = $\langle [g()-m()]^2 \rangle$

covariance = $\langle [g(1)-m(1)][g(2)-m(2)] \rangle$

std dev = $\sqrt{\text{variance}}$

correlat coef = $\text{covariance} / \text{std dev (psi) * std dev (delta)}$

mean, std dev (degrees)

<table>
<thead>
<tr>
<th></th>
<th>psi:</th>
<th>std dev</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>delta:</th>
<th>std dev</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

0.595 "correlation coefficient" $\langle \text{psi|delta} \rangle$

$J(T)J$: (renormalized for correlation)

1) 1.00000

Normalization coefficients: $\sqrt{[J(T)J](i,i)}$

8.35E-02

rcond= 1.000E+00, condition number

elapsed cpu-time = 29 centi-seconds

+ 42 seconds
5.4 Sensitivity Analysis

This option is discussed in section 4.3.5. To select this option, the first item on the first line in the input data file X.DAT ought to contain the integer value '5.' Regarding the output presented below, lines 30 and 31 reveal that both layer thicknesses are the only vary model parameters. Note that no iterations are performed by the program; it is assumed that the critical-point or fixed-point solution has already been found, and is given by those entered upon input. The total uncertainty associated with knowing the thicknesses of the the silicon and oxide layers is given, respectively, by lines 125 and 142, i.e., 4.69E-02 and 3.03E-01. The output file is given by the following.

```plaintext
options: 1, forward problems, plots, ...
  2, search ( vary)
  3, search grid ( vary)
  4, search grid (froz,vary)
  5, sensitivity analysis
  6, MAE plots " uncertainty

Enter: option 5

mwaves = 2, number of distinct wavelengths
  1 632.8000
  2 441.6000

mbient = 3, number of distinct ambient environments and waves.
  1 1.000000
  2 1.000270
  3 1.000270

mlmnts = 7, number of distinct: n+ik
  1 3.8820 0.0020 0
  2 0.0190 0.0020 0
  3 1.4570 0.0020 0
  4 0.0000 0.0000 0
  5 4.7530 0.0020 0
  6 0.1630 0.0020 0
  7 1.4660 0.0020 0

mfilm = 2, number of distinct film widths
  1 50.0000 2.0000 1
  2 100.0000 2.0000 1

msamp = 2, number of distinct samples
  1 2 2 " sample, mfilm, mwave
  1 2 " iwave, mbien
  1 1 2 1 " i,n,k,z
  2 3 4 2 " i,n,k,z
```
38 3 1 2 "i,n,k
39 1 1 "imbien, mrpeat
40 2 2 "imbien, mrpeat
41 2 1 "iwave, mbien
42 1 5 6 1 "i,n,k,z
43 2 7 4 2 "i,n,k,z
44 3 5 6 "i,n,k
45 3 1 "imbien, mrpeat
46 2 1 2 "sample, mfilm, mwave
47 1 1 "iwave, mbien
49 1 3 4 2 "i,n,k,z
50 2 1 2 "i,n,k
51 2 1 "imbien, mrpeat
52 2 1 "iwave, mbien
53 1 7 4 2 "i,n,k,z
54 2 5 6 "i,n,k
55 3 1 "imbien, mrpeat
56
57 2 1 1 1 1 "mangl, repeat, ambient, wave, sample/ (phi, delta, psi)
58 1 65.0000 0.0100 173.5043 0.0500 37.0579 0.0500
59 2 70.0000 0.0100 172.2664 0.0500 34.3707 0.0500
60
61 3 1 2 1
62 1 65.0000 0.0100 173.5101 0.0500 37.0527 0.0500
63 2 70.0000 0.0100 172.2750 0.0500 34.3633 0.0500
64 3 75.0000 0.0100 170.0354 0.0500 30.1673 0.0500
65
66 2 2 2 1
67 1 67.6000 0.0100 172.9647 0.0500 35.8405 0.0500
68 2 72.6000 0.0100 171.3529 0.0500 32.5214 0.0500
69
70 4 1 3 2
71 1 67.5000 0.0100 286.8467 0.0500 30.2844 0.0500
72 2 70.0000 0.0100 297.5219 0.0500 31.2799 0.0500
73 3 72.6000 0.0100 307.5732 0.0500 32.6322 0.0000
74 4 75.0000 0.0100 316.8687 0.0500 34.2226 0.0000
75
76 2 1 2 1 2 "mangl, repeat, ambient, wave, sample/ (phi, delta, psi)
77 1 70.0000 0.0100 79.8092 0.0500 41.0455 0.0500
78 2 73.0000 0.0100 68.2502 0.0500 41.0304 0.0500
79
80 2 1 3 2
81 1 70.0000 0.0100 211.2235 0.0500 77.3289 0.0500
82 2 73.0000 0.0100 216.1556 0.0500 84.4928 0.0500
83
84
85 Statistics of deviations " experiment-model " g
86
87 where: g " column array of length ' 2M
88 let: () " (psi or delta) " (1 or 2)
89
76
mean() = m() = \langle g() \rangle = (1/N) \sum g()
variance() = \langle [g() - m()]^2 \rangle
\text{covariance} = \langle [g(1) - m(1)]*[g(2) - m(2)] \rangle
\text{std dev} = \sqrt{\text{variance}}
\text{correlat coef} = \text{covariance} / [\text{std dev (psi)} \times \text{std dev (delta)}]

mean, std dev (degrees)

\begin{align*}
\text{psi}: & \quad 0.000 \quad 0.000 \\
\text{delta}: & \quad 0.000 \quad 0.000
\end{align*}

\text{correlation coefficient}\quad <\text{psi}|\text{delta}>

\text{mean}, \text{std dev (degrees)}

<table>
<thead>
<tr>
<th>\text{parameter}</th>
<th>\text{total}</th>
<th>\text{initial}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1</td>
<td>1.86E-12</td>
<td>2.36E-06</td>
</tr>
<tr>
<td></td>
<td>1.36E-06</td>
<td>1.54E-03</td>
</tr>
</tbody>
</table>

\text{B} = [J(T)\times J]^{-1} \times J(T), \quad \text{(nonsquare J)}

\text{rcond} = 8.477E-01, \text{ condition number}

\text{Discern: Uncertainty in model parameters}
\text{where: } |v| = \sqrt{B B(T) \langle gg \rangle + (B J)(B J(T) \langle aa \rangle)}
\quad + |B J| |u|,

\text{B} \times [J(T)\times J]^{-1} \times J(T), \quad \text{(nonsquare J)}

\text{vary} \quad \text{total} \quad \text{initial} \quad \text{parameter}

\begin{align*}
1 & 1 & 1.86E-12 & 2.36E-06 \quad (j,i), |B g g B|, |B J a \times J B| \\
& & 1.36E-06 & 1.54E-03 \quad |B g| , |B J a|
2 & 1 & 4.69E-02 & 2.00E+00 \quad |B g g B|, |B J a \times J B| \\
& & 1.54E-03 & 1.36E-06 \quad |B g| , |B J a|
\end{align*}

\begin{align*}
1) & 2.29E-03 \quad \text{systematic} \quad |B J u| \quad 1, (n+i k) \\
2) & 9.22E-04 \quad \text{systematic} \quad |B J u| \quad 2, (n+i k) \\
3) & 2.54E-03 \quad \text{systematic} \quad |B J u| \quad 3, (n+i k) \\
4) & 0.00E+00 \quad \text{systematic} \quad |B J u| \quad 4, (n+i k) \\
5) & 2.16E-02 \quad \text{systematic} \quad |B J u| \quad 5, (n+i k) \\
6) & 8.66E-03 \quad \text{systematic} \quad |B J u| \quad 6, (n+i k) \\
7) & 9.40E-03 \quad \text{systematic} \quad |B J u| \quad 7, (n+i k) \\
2 & 1 & -1.44E-13 & -2.66E-07 \quad (j,i), |B g g B|, |B J a \times J B| \\
& & 3.80E-07 & 5.18E-04 \quad |B g| , |B J a|
2 & 2 & 1.64E-12 & 3.03E-06 \quad (j,i), |B g g B|, |B J a \times J B| \\
& & 1.28E-06 & 1.74E-03 \quad |B g| , |B J a|
\end{align*}

\begin{align*}
2) & 3.03E-01 & 2.00E+00 \quad |B g g B|, |B J a \times J B| \\
& & 1.74E-03 & 1.28E-06 \quad |B g| , |B J a|
\end{align*}
<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Description</th>
<th>Symbol</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>3)</td>
<td>$1.03 \times 10^{-4}$</td>
<td>systematic</td>
<td>$</td>
<td>\mathbf{B}_J\mathbf{u}</td>
</tr>
<tr>
<td>4)</td>
<td>$2.51 \times 10^{-5}$</td>
<td>systematic</td>
<td>$</td>
<td>\mathbf{B}_J\mathbf{u}</td>
</tr>
<tr>
<td>5)</td>
<td>$2.46 \times 10^{-4}$</td>
<td>systematic</td>
<td>$</td>
<td>\mathbf{B}_J\mathbf{u}</td>
</tr>
<tr>
<td>6)</td>
<td>$0.00 \times 10^{0}$</td>
<td>systematic</td>
<td>$</td>
<td>\mathbf{B}_J\mathbf{u}</td>
</tr>
<tr>
<td>7)</td>
<td>$6.31 \times 10^{-3}$</td>
<td>systematic</td>
<td>$</td>
<td>\mathbf{B}_J\mathbf{u}</td>
</tr>
<tr>
<td>8)</td>
<td>$2.90 \times 10^{-1}$</td>
<td>systematic</td>
<td>$</td>
<td>\mathbf{B}_J\mathbf{u}</td>
</tr>
</tbody>
</table>

---------------------------------------------------------------------

Elapsed CPU-time: 62 centi-seconds
6. Listing of Software Source Files and Routines

This section presents a listing of the source files for MAIN1. For convenience, the files are partitioned into three classes. The first class involves files associated with storage arrays, the second class involves source programs that govern the scattering problem and hosts the user's options, and the third class involves general utilities that provide basic tasks for the source programs of class two.

6.1 Named COMMON and BLOCK DATA Statements

6.1.1 IOUNIT.

1 c Named common of the logical unit assignments associated with
2 c reading/writing of input/output files on the hardware disk.
3
4 common /iounit / in, iout, idat, isout, iplt
6.1.2 DEFINIT.

Each sample is characterized by the film/substrate:

- geometry: air / z(1) / ... / z(nfilms) / substrate
- parameters:
  - three data (t,n,k) for each film, may vary.
  - two data (t,n,k) for the substrate, may vary.
  - one data ( ,n, ) for the air, may not vary.

Special case for: analyzing Standard Reference Materials

- parameters:
  - (nsampl=150) ! sample configurations "films/substrate
  - (nfilms= 4) ! distinct films on a sample
  - (nlmnts= 16) ! (n,k| w,lmnt) " Si, SiO2, etc.
  - (nwaves= 1) ! wavelengths of incident probe
  - (nbient= 1) ! ambient environment of samples
  - (nrpeat= 20) ! repeats of any given experiment
  - (nanglx= 2) ! angles at one wavelength to fit experiment
  - (nanglm= 2) ! multi-angle error analysis at one wavelength

Usual case:

- parameters:
  - (nsampl= 8) ! sample configurations "films/substrate
  - (nfilms= 8) ! distinct films on a sample
  - (nlmnts=300) ! (n,k| w,lmnt) " Si, SiO2, etc.
  - (nwaves=100) ! wavelengths of incident probe
  - (nbient= 1) ! ambient environment of samples
  - (nrpeat= 1) ! repeats of any given experiment
  - (nanglx= 20) ! angles at one wavelength to fit experiment
  - (nanglm= 2) ! multi-angle error analysis at one wavelength

- parameters:
  - (nrows =nfilms*3+2 ) ! (z,n,k)/(n,k) " 1 sample
  - (nfilms=nsampl) ! z
  - (mrowss=nsampl*nfilms*nlmnts) ! (z,n,k) " n samples
  - (mrowss=nsampl*nwaves*nbient*nrpeat*nanglx*2)
  - (nexpts=nsampl*nwaves*nbient*nrpeat*nanglx )
  - (nseam3=neampl*nwaves*nbient* nanglm )

- parameters:
  - (mnjaaa = 1000000) ! convenience
  - (mnjaaa = mrowss*nrows) ! aa,ja, aaa, jaa

complex cmplx, conjg, sqrtt

Note: nanglm < nanglx ! SEAMAX
6.1.3 FILMMM.

real ambient(nwaves*nbient)
real waveln(nwaves), waveqq(nwaves), waveww(nwaves)
real diefcn(nlmnts), uncerl(nlmnts)
real widths(nfilm), uncerz(nfilm)
ingter lvaryl(nlmnts), lvaryz(nfilm)
ingter nncfilm(nsamp1), iifilm(nsamp1*nwaves*nrows)
ingter nncave(nsamp1), iicave(nsamp1*nwaves)
ingter nncbent(nsamp1*nwaves), iicent(nsamp1*nwaves*nbient)
ingter nnpeat(nsamp1*nwaves*nbient)
ingter psiis(nexpts), deltas(nexpts), angles(nexpts)
ingter psiiu(nexpts), deltau(nexpts), angleu(nexpts)
ingter mangle(nnpeat*nbient*nwaves*nsamp1)
ingter isteps(nbient*nwaves*nsamp1)
ingter aa(nnjaaa), bb(mrowss), cc(mrowss), xx(nrowss)
ingter ja(nnjaaa), ia(mrowss+1)
ingter iptu(nrowss), iptv(nrowss), iptw(nrowss)
ing logical llnorm

common /wvaves / waveln, waveqq, waveww, nwaves

common /filmmm / ambient, diefcn, widths,
& iifilm, nnfilm, iicave, nncave,
& iicent, nnbent, nnpeat,
& mbient, nlmnts, mfilm, nsamp1

common /choose / uncerl, uncerz, lvaryl, lvaryz

common /exprmt / psiis, deltas, angles, psiis, deltau, angleu,
& mangle, isteps, method

common /arrays / aa, bb, cc, xx, ja, ia, iptu, iptv, iptw,
& meqns, nvary, mfoz, llnorm
Each sample is characterized by the film/substrate:
geometry: air / z(1) / ... / z(nfilms) / substrate
parameters: three data (z,n,k) for each film, may vary.
two data ( ,n,k) for the substrate, may vary.
one data ( ,n, ) for the air, may not vary.

epsilon = electric permittivity = dielectric function
sigma = specific conductivity
mu = magnetic permeability
omega = angular frequency
die_r = epsilon * mu
die_i = 4*pi * sigma * mu / omega

complex die(nfilms+l)
real zzz(nfilms ), air
integer mfilm

common /filmss / die, zzz, air, mfilm
6.1.5 RSTACK.

1 complex Rs, Rp ! reflection coeff in ambient
2 complex dRs(nrows), dRp(nrows) ! Jacobians
3 complex dRs, dRp ! d/d(angle)
4
5 common / rstack / Rs,Rp, dRs,dRp, dRs, dRp
6.1.6 WSTACK.

1 parameter (naat=(nrowss*(nrowss+1))/2) ! upper triangle + diagonal
2 real aat (naat) ! A(T)A, Inverse
3 real aats(nrowss) ! A(T)A, scale factors
4 integer ipvt(nrowss) ! workspace, LINPACK
5
6 common / wstack / aat, aats, ipvt
6.1.7 CGNXX1.

1 c Storage allocation required by: CGN
2 c This storage is needed only in: ZOOM, ZOOM2
3
4 real p(mrowss), u(mrowss)
5 real v(nrowss), w(nrowss), xw(nrowss), se(nrowss)
6
7 common / cgnxx1 / p, u, v, w, xw, se
6.1.8 SCANCC.

Used only by: SCAN2, SCAN3, SCAN2G.

character bufft*8, buffd*9 ! convenience, label breakpoint

Nested do-loops associated with multi-parameter grid scan

integer iiii(niii), iiii2(niii)
real pppp(niii), ppp1(niii), ppp2(niii), ppp3(niii)
real psav(niii)
6.1.9 SEAMX1.

1     c     Sensitivity analysis for multiple: angle, ambient, wave, sample.
2
3     integer  iptx(nrows), ipty(nrows)           ! vary
4     integer  kptx(nrows), kpty(nrows)           ! frozen
5
6     real     aaa(nnjaaa), bbb(mrows), xxx(nrows) ! frozen
7     integer  jaa(nnjaaa), iaa(mrows+1)
8
9     common / seamx1 / iptx, ipty, kptx, kpty,
10 &       aaa, bbb, xxx, jaa, iaa
Sensitivity analysis for multiple: angle, ambient, wave, sample.

Storage of forward scattering problem at various incident angles.
For the error analysis, mrpeat=1, ... in all cases.

parameter (ndegr = 89) ! grid of angles
parameter (ndegs = ndegr*nbient*nwaves*nsample) ! (ndegr,)
real psi(ndegs), del(ndegs)
real psia(ndegs), dela(ndegs) ! dR/d(phi)
real psid(nrows,ndegs), deld(nrows,ndegs) ! dR/d(e) "Jacobian

Arrays associated with the nesting of do-loops within SEAM,
Sensitivity/error analysis of multiple:
angle, ambient, wave, sample.

integer iii(nseams), iii1(nseams), iii2(nseams)
common / seamx2 / psi, del, psia, dela, psid, deld,
& iii1, iii, iii2, mraws, maraws,
& mm, raddeg, uang
block data blkdat
include 'iounit.'
data in,iout,idat,isout,iplt / 5,6,7,8,9 /
end
6.2 Source Programs

6.2.1 MAIN.FOR

```
program main
include 'iounit.'
call fileop ! open relevant files
write (iout,101)
read (idat,*,err=12,end=12) ichoic
write (iout,102) ichoic
if (ichoic.lt.1 .or. ichoic.gt.5) then
  write (iout,122)
  stop
end if

call inpdat (ichoic)
call arrang
it1 = istime (i)
goto (1,2,3,4,5,6), ichoic

1 continue ! forward problem, plots, ...
call pltdata (0)
goto 11

2 continue ! unconstrained optimization
call zoom
call corlat
c* call seama
goto 11

3 continue ! grid scan ( ,vary)
call scan2
call corlat
c* call seama
goto 11

4 continue ! grid scan (froz,vary)
call scan3
call corlat
c* call seama
goto 11

5 continue ! sensitivity analysis
call corlat
call seama
goto 11

6 continue ! sensitivity analysis scans
c* call seamax
11 goto 11
11 it2 = iftime (i)
it = (it2-it1)/10
its = 100
itm = its*60
```


ith = item*60 ! / hour
itd = ith*24 ! / day
id = it/itd ! days
it = it-id*itd
ih = it/ith ! hours
it = it-ih*ith
im = it/itm ! minutes
it = it-im*itm
is = it/its ! seconds
it = it-is*its ! centi-seconds

if (id.ne.0) then
  write (iout,111) it,is,im,ih,id
else if (ih.ne.0) then
  write (iout,111) it,is,im,ih
else if (im.ne.0) then
  write (iout,111) it,is,im
else
  write (iout,111) it,is
end if

close (iout)
stop
12 write (iout,121)
stop
101 format (" options: 1, forward problems, plots, ... ",
& " 2, search ( vary) ",
& " 3, search grid ( vary) ",
& " 4, search grid (froz, vary) ",
& " 5, sensitivity analysis ",
& " 6, MAE plots uncertainity ",
& " Enter: option ", )
102 format (" option ", i1 )
111 format (" elapsed cpu-time = ", i3, " centi-seconds", :
& " + , i3, " seconds ", :
& " + , i3, " minutes ", :
& " + , i3, " hours ", :
& " + , i3, " days ")
121 format (" ... oops, unable to discern: option")
122 format (" ... oops, inconsistent value: option")
end
subroutine fileop
  include 'iounit.'

  open (idat, file='x.dat', status='old', readonly, shared)

  open (iout, file='x.out', status='old', disp='delete', err=12)
  close(iout)
  goto 11

  open (iout, file='x.out', status='new')

  open (isout, file='x.sout', status='old', disp='delete', err=14)
  close(isout)
  goto 13

  continue ! open (isout, file='x.sout', status='new')

  open (iplt, file='x.plot', status='old', disp='delete', err=16)
  close(iplt)
  goto 15

  open (iplt, file='x.plot', status='new')

  return

end
INPDAT.FOR

6.2.3

subroutine inpdat (ichoic)
include 'iounit.'
include 'defnit.'
include 'filinmm.'
logical first

data pi / 3.14159265E+00 /
data cccc / 2.99792458E+17 / ! speed of light, (nm/sec)
data cccc / 2.99792458E+14 / ! speed of light, (micro-m/sec)
data cccc / 2.99792458E+10 / ! speed of light, (cm/sec)

! Note: 1.0 micro-m = 1E3 nano-m = 1E4 Angstrom
1E-3 micro-m = 1 nano-m = 10 A
1E-4 micro-m = 1E-1 nano-m = 1 A

raddeg = pi/180.0
first = .false.

! Distinct wavelengths (free space) incident on sample.
read (idat, *) mwaves
write (iout,101) mwaves
if (mwaves.le.1 .or. mwaves.gt.nwaves) then
 write (iout,102) mwaves
 stop
end if
do i=1,mwaves
 read (idat, *) j, wavlen
 write (iout,103) j, wavlen
 if (j.ne.i .or. wavlen.le.0.0) then
  write (iout,104)
  stop
 end if
if (i.ne.1) then
 if (wavlen(i-1).le. wavlen) then ! impose ordering
  longer waves first,
  waves be distinct.
  write (iout,105)
  stop
 end if
 end if
wavlen(i) = wavlen ! nano-meters
waveqq(i) = 2.*pi/ wavlen
waveww(i) = 2.*pi* (cccc/wavlen)
end do

! Distinct refractive indices of ambients *(n| ambient,wave)
read (idat, *) mbient
write (iout,111) mbient
mbnwav = mbient*mwaves ! ambients, waves
if (mbient.lt.1 .or. mbient.gt.nbnwav) then
    write (iout,112) nbnwav
    stop
end if

do i=1,mbient
    read (idat, *) j, air ! refractive index
    write (iout,113) j, air
    if (j.ne.i .or. air.gt.1.0) then
        write (iout,114)
        stop
    else
        ambient(i) = air
    end if
end do

----------------------------------------

c

Distinct material elements: (n,k| wave,samp)

read (idat, *) mlmnts
write (iout,121) mlmnts
if (mlmnts.lt.1 .or. mlmnts.gt.nlmnts) then
    write (iout,122) nlmnts
    stop
end if

do i=1,mlmnts
    read (idat, *) j, dielec, uncert, ivary
    write (iout,123) j, dielec, uncert, ivary
    if (j.ne.i .or. 
    & dielec.lt.0.0 .or. uncert.lt.0.0 .or. 
    & ivary.lt.0 .or. ivary.gt.2 ) then
        write (iout,124)
        stop
    end if
    if (ivary.eq.1 .and. uncert.eq.0.0) then
        write (iout,125)
        stop
    end if
    diefcn(i) = dielec ! n,k
    uncerl(i) = uncert
    lvaryl(i) = ivary
    iptw(i) = -1 ! discern utilization
    if (ivary.eq.2 .and. ichoic.ne.4) then ! convenience
        first = .true.
    end if
end do

c
----------------------------------------

Distinct film thickness-es: z
read (idat, *) mfilmm
write (iout,131) mfilmm
if (mfilmm.lt.0 .or. mfilmm.gt.nfilmm) then
    write (iout,132) nfilmm
stop
end if
if (mfilmm.gt.0) then
  do i=1,mfilmm
    read (idat, *) j, width, uncert, ivary
    write (iout,133) j, width, uncert, ivary
    if (j.ne.i .or. width.lt.0.0 .or. uncert.lt.0.0 .or. ivary.lt.0 .or. ivary.gt.2 ) then
      write (iout,134)
      stop
    end if
    if (ivary.eq.1 .and. uncert.eq.0.0) then
      write (iout,135)
      stop
    end if
    widths(i) = width
    uncerz(i) = uncert
    lvaryz(i) = ivary
    j = mlnmts+i
    iptw(j) = -1
    if (ivary.eq.2 .and. ichoic.ne.4) then
      first = .true.
    end if
  end do
end if
if (first) then
  write (iout,137)
  write (iout,136)
end if
if (first) then
  write (iout,137)
  ! impose constraint:
  ! ivary=2
  ! be allowed only for:
  ! ichoic=4
  !
------------------------------------------------------------------------
Distinct sample configurations: (films/substrate)
read (idat, *) msampl
write (iout,141) msampl
if (msampl.lt.1 .or. msampl.gt.msampl) then
  write (iout,142) msampl
  stop
end if
i = 0
iws = 0
iaws = 0
do is=1,msampl
  read (idat, *) js, mfilm, mwave
  write (iout,143) js, mfilm, mwave
  if (js.ne.is .or. mfilm.lt.0 .or. mfilm.gt.mfilms .or. mwave.lt.1 .or. mwave.gt.mwaves ) then
    write (iout,144) nfilsms, mwaves
  end if
  if (msampl.gt.0 .or. mfilm.gt.mfilms .or. mwave.gt.mwaves ) then
    write (iout,144) nfilsms, mwaves
  end if
157 stop
158 end if
159 mfilms = mfilm+1 ! films/substrate
160 nnfilm(is) = mfilm ! number of films on sample
161 nnwave(is) = mwave ! number of waves on sample
162 do iw=1,mwave ! scan distinct waves on sample
163    read (idat, *) iwave, mbien
164    write (iout,145) iwave, mbien
165    if (iwave.lt.1 .or. iwave.gt.mwaves .or.
166        &        mbien.lt.1 .or. mbien.gt.mbient ) then
167        write (iout,146)
168        stop
169    end if
170    if (iw.ne.1) then ! impose ordering
171        if (iiwave(iws).ge.iwave) then
172            write (iout,147)
173            stop
174        end if
175    end if
176    iws = iws+1
177    iiwave(iws) = iwave ! specify the distinct wave
178    nnbent(iws) = mbien ! number of distinct ambients
179    do m=1,mfilms ! films/substrate
180       if (m.eq.mfilms) then ! substrate
181          read (idat, *) j, n,k
182          write (iout,148) j, n,k
183       else
184          read (idat, *) j, n,k,iz ! film
185          write (iout,149) j, n,k,iz
186          if (iz.lt.l .or. iz.gt.mfilmm) then
187              write (iout,150)
188              stop
189          end if
190          i = i+1
191          iiifilm(i) = iz
192       end if
193       if (j.ne.m .or. n.eq.k .or.
194           &        n.lt.1 .or. n.gt.mlmnts .or.
195           &        k.lt.1 .or. k.gt.mlmnts ) then
196          write (iout,150)
197          stop
198       end if
199       i = i+1
200       iiifilm(i) = n
201       i = i+1
202       iiifilm(i) = k
203       j = mlmnts+iz ! indicate utilization
204       iptw(j) = 0
205       iptw(n) = 0
206       iptw(k) = 0
207
end do ! film

if (iw.ne.1) then ! test widths "z
  ii = i+1 - (mfilm*3+2)
  jj = i+1 - (mfilm*3+2)*2
  do iz=1,mfilm
    if (iifilm(jj).ne.iifilmi(iii)) then
      write (iout,151) iz, iifilm(jj), iifilm(ii)
      stop
    end if
    ii = ii+3
    jj = jj+3
  end do
end if ! test z

do mbn=1,mbien ! scan distinct ambients
  read (idat, *) mbien, mrpeat
  write (iout,152) mbien, mrpeat
  if (mbien.lt.1 .or. mbien.gt.mbien .or.
     mrpeat.lt.1 .or. mrpeat.gt.mrpeat ) then
    write (iout,153)
    stop
  end if
  if (mbn.ne.1) then ! impose ordering
    if (mbien .le. iibent(iaws)) then
      write (iout,154)
      stop
    end if
  end if
  iaws = iaws+1
  iibent(iaws) = mbien ! specify distinct ambient
  nnpeat(iaws) = mrpeat ! repeats of experiment
end do ! ambient
end do ! wave
write (iout,175)
end do ! sample

if (ichoic .eq. 1) then ! plot: (z,n,k)
c* call pltdat (1)
c* return
  c*
end if

k = 0 ! discern utilization of
mm = mlmnts+mfilmn ! the model parameters
do i=1,mm
  if (iptw(i) .eq. -1) then ! not utilized
    k = k+1
    if (i .le. mlmnts) then
      write (iout,156) '(n+ik)', i
    else
      j = i-mlmnts
      write (iout,156) '(z )', j
    end if
  end if

end if
do
if (k.ne.0) then
  ! impose constraint that
  ! all model parameters
  ! be of use.
c
then
  impose constraint that
  stop
  all model parameters
end if!
be of use.

C

if (ichoic.eq.6) goto 30
! sensitivity analysis

Experimental, measurement, or target data.

(.psi,delta | phi,repeat,ambient,wave,sample)

i = 0
! index measured data
ii = 0
! overflow indicator
iws = 0
iaws = 0
iraws = 0
do is=1,msamp! scan samples
  mwave = mwave(is)
  first = .true.
do iw=1,mwave
    iws = iws+1
    iwave = iwave(iws)
    mbien = nnbent(iws)
do mbn=1,mbien
    iaws = iaws+1
    imbien = iibent(iaws)
do mrpeat=1,mrpeat
    iraws = iraws+1
mangle(iraws) = mangl
do iangle=1,mangl
  ! incident angles
  read (idat, *) j,angle,au,delta,du,psi,pu
  write (iout,174) j,angle,au,delta,du,psi,pu
  if (delta.lt.0.0) then ! [0, 360)
    delta = delta + 360.0
  end if
The above incoming measurement data (delta, psi) is assumed to be of the Nebraska convention.

- Engineering assumes: n-ik --> (psi, delta) (Nebraska)
- Physics assumes: n+ik --> (psi, delta) (this software program)

Both assume: TM mode or p-polarization uses the H field.

Consequently, this affects only the Delta.

The transformation is: (physics) <-- conjg (Nebraska)

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

psi = abs (psi ) ! [0, 90)
pu = abs (pu )
delta = abs (delta) ! [0, 360)
du = abs (du )
angle = abs (angle) ! [0, 90)
a = abs (a )

psi = a mod (psi , 90.0) ! degrees
pu = a mod (pu , 90.0)
delta = a mod (delta, 360.0)
du = a mod (du , 360.0)
angle = a mod (angle, 90.0)
a = a mod (a , 90.0)

psi = psi * raddeg ! radians
pu = pu * raddeg
delta = delta * raddeg
du = du * raddeg
angle = angle * raddeg
a = a * raddeg

i = i+1
if (i.gt.nexpts) then
  i = i-1
  ii = ii+1
end if

psiis(i) = psi
psiiu(i) = pu
deltas(i) = delta
deltau(i) = du
angles(i) = angle
angleu(i) = a

end do ! angle
write (iout,175)
end do ! repeat
end do ! ambient
end do ! wave
end do ! sample
```
if (ii.gt.0) then
  i = i+ii
  write (iout,176) nexpts, i
  stop
end if

return

C ==========================================================================

C Sensitivity/error analysis of multiple: angle, ambient, wave, sample.
C No need for experimental data, here. But rather
C we need: manglm, isteps.
C Assume: mrpeat = 1, ... in all cases.

30 read (idat, *) method
  write (iout,161) method
  if (method.lt.1 .or. method.gt.3) then
    write (iout,162)
    stop
  end if
  i = 0 ! index measured data
  ii = 0 ! overflow indicator
  iws = 0
  iaws = 0
  iraws = 0
  do is=1,msampl ! scan samples
    mwave = mnwave(is)
    first = .true.
    do iw=1,mwave ! scan wavelengths
      iws = iws+1
      iwave = iiwave(iws) ! specify wavelength
      mbien = nnbent(iws)
      do mbn=1,mbien ! scan ambients
        iaws = iaws+1
        imbien = iibent(iaws) ! specify ambient
        mrpeat = nnpeat(iaws)
        if (mrpeat.gt.1) then ! consistency
          write (iout,181) mrpeat
          stop
        end if
      do irpeat=1,mrpeat ! repeats of experiment
        read (idat, *) istep,mangl, jmbn,jw,js
        if (first) then
          first = .false.
          write (iout,182) istep,mangl, jmbn,jw,js
        else
          write (iout,182) istep,mangl, jmbn,jw
        end if
      if (istep.lt.1 .or. istep.gt.10) .or.
        mangl.lt.1 .or. mangl.gt.manigm .or.
        jmbn.ne.imbien .or.
        jw.ne.iwave .or. js.ne.is ) then
write (iout,163) nanglm,imbien,iwave,is
stop
end if
irows = irows+1
mangle(irows) = nangl  ! multiple angles
isteps(irows) = istep  ! increment of grid
c*    end do     ! repeat
end do     ! ambient
end do     ! wave
end do     ! sample
return

101 format(/' mwaves =', i4, ', number of distinct wavelengths')
102 format (' mwaves =', i4, ', <-------- oops')
103 format ('', i4, fl2.4)
104 format (' oops, ... card info inconsistent')
105 format (' oops, ... wavelengths not: ordered,distinct.')
106 format(/' mbien =', i4, ', number of distinct ambient', &
         ' environments and waves.' )
107 format (' mbwav =', i4, ', <-------- oops')
108 format ('', i4, if12.6)
109 format (' oops, ... card info inconsistent.')
110 format(/' nlmnts =', i4, ', number of distinct: n+ik')
111 format(/' nlmnts =', i4, ', <-------- oops')
112 format ('', i4, 2fl12.4, i6)
113 format (' oops, ... card info inconsistent')
114 format (' oops, since:  ivary = 1, '
   & /' then:  uncert /= 0.0 ')
115 format(/' nfilm =', i4, ', number of distinct film widths')
116 format (' nfilm =', i4, ', <-------- oops')
117 format ('', i4, 2fl12.4, i6)
118 format (' oops, ... card info inconsistent')
119 format (' oops, since: ivary = 1, '
   & /' then:  uncert /= 0.0 ')  ! may one be allowed to use: ivary = 2' )
120 format(/' nsampl =', i4, ', number of distinct samples')
121 format (' nsampl =', i4, ', <-------- oops')
122 format ('', i4, 3fl4, 8x, ' sample, nfilm, mwave')
123 format (' oops, nfilms=', i4, ', mwaves=', i4)
124 format ('', 2fl4, 12x, ' iwave, mbienv')
125 format (' oops, ... card info inconsistent')
126 format (' oops, ... distinct, ordered ')  ! consistent
127 format ( 9x, 3i4, 8x, ' i,n,k ')
128 format ( 9x, 4i4, 4x, ' i,n,k,z')
129 format (' oops, ... card info inconsistent')
130 format (' oops, ... card info inconsistent')
131 format (' oops, ... card info inconsistent')
132 format (' oops, compare corresponding widths: '
   & /
   iz, iifilm(jj), iifilm(ii) =', 3i5)
133 format ('', 2fl4, 12x, ', imbienv, mrpeat')
134 format (' oops, ... card info inconsistent')
135 format (' oops, ... impose ordering on the: ambient')
136 format (' oops, parameter not in sample configuration, ',
& a6, ' ', i3)
161 format (' method =', i4, ', (1 ' normal eqns, '
162 format (' oops, ... card info inconsistent')
172 format (1x, 5i4, ' ' mangl, repeat, ambient, wave, sample',
173 format (' oops, ... card info inconsistent'
174 format (1x, i4, 6f12.4)
175 format (' ')
176 format (' oops, enlarge: nexpts=', i4, ' ----> ', i4)
181 format (' oops, repeats =', i4, ', ----> 1')
182 format (1x, 5i4, ' ' istep, mangl, ambient, wave, sample')
183 format (' oops, ... card info inconsistent'
188 format (13x, 5i4, 4x, ', ' (ambient, wave, sample'))
end
This subroutine sets up the necessary pointers for indexing/ordering the matrix elements in the sparse matrix of the Jacobian.

```fortran
subroutine arrang
  include 'defnit.'
  include 'filmm.'

  ! The sparse matrix format for the model parameters is of the form:
  ! [(n/k)_1, ..., (n/k)_(mlmnts)]  diefcn (w)
  ! [ z_(1), ..., z_(mfilms)]  widths

  ! Ordering of indices in IPTU:  [vary--->  <--froz]

  mm = mlnmns+mfilmm  ! model parameters
  i = 0  ! vary " compress
  k = 0  ! froz " compress
  do m=1,mlmnts
    iptu(m) = 0  ! local uniqueness
    if (lvary(i).eq. 1) then
      i = i+1
    end if
    iptu(i) = m
    iptv(m) = i
  else
    k = k+1
    km = mm+1-k
    iptu(km) = m
    iptv( m) = km
  end if
  end do

  ! vary
  ! vary (full)
  ! vary (full)
  ! froz
  ! compress
  ! backwards
  ! full (froz)
  ! froz (full)

  do m=1,filmm
    j = mlnmns+m
    iptu(j) = 0
    if (lvary(j).eq. 1) then
      i = i+1
    end if
    iptu(i) = j
    iptv(j) = i
  else
    k = k+1
    km = mm+1-k
    iptu(km) = j
    iptv( j) = km
  end if
  end do

  ! vary
  ! vary (full)
  ! vary (full)
  ! froz
  ! compress
  ! backwards
  ! full (froz)
  ! froz (full)

  mvary = i  ! compress
```

6.2.4 ARRANG.FOR
47    mfooz = k
48
49    return
50    end

! compress
subroutine pltdat (ichoic)
include 'iounit.'
include 'defnit.'
include 'filmm.'
include 'filmss.'
include 'rstack.'
include 'seanxl.'
real a (nrows*2), b(2), c(2)
real uz(nrows*2), un(nrows*2), uk(nrows*2)
character*12 labelo ! convenience label
data pi / 3.14159265 /
The sparse matrix format for the model parameters is of the form:
\[ \left[ \frac{(n/k)}{1}, \ldots, \left(\frac{n/k}{m\text{mmts}}\right) \right] \text{ diefcn \( (w) \)}
\[ \left[ \frac{z(1)}{1}, \ldots, \frac{z(m\text{films})}{w} \right] \text{ widths} \]

j = ichoic
if (j.lt.1 .or. j.gt. 5) then
write (iout,101)
read (idat, *) j
write (iout,102) j
if (j.lt.1 .or. j.gt.5) then
stop
end if
end if
if (j.eq. 1) goto 1
if (j.eq. 2) goto 2
if (j.eq. 3) goto 3
if (j.eq. 4) goto 4
if (j.eq. 5) goto 5
================================================================
Plot distribution: \((z,n,k)\) for those samples
which have one or more films.
continue
i = 0
iws = 0
iws = 0
do is=1,msampl
mfilm = nnfilm(is)
mwave = nnwave(is)
if (mfilm.ne.0) then
mfilm = mfilm+1 ! films/substrate
do iw=1,mwave
iws = iws+1
iwave = iwave(iws)
mbien = mnbent(iws)
qq = waveqq(iwave) ! FILMSS

k = 2 ! index of plot array
uk(1) = 0.0 ! extinction of ambient
uk(2) = 0.0
uz(2) = 0.0 ! surface

do m=1,mfilms ! films/substrate
  if (m.ne.mfilms) then ! z
    i = i+1
    iz = iifilm(i)
    zzz(m) = widths(iz)
    uz(k+1) = uz(k)
    uz(k+2) = uz(k) + zzz(m)
    if (m.eq.1) then
      thin = zzz(m)
    else
      thin = aminl(thin, zzz(m))
    end if
  end if
  do ink=1,2 ! n+ik
    i = i+1
    nk = iifilm(i)
    c(ink) = diefcm(nk)
  end do
  un(k+1) = c(1) ! LHS
  un(k+2) = c(1) ! RHS
  uk(k+1) = c(2) ! LHS
  uk(k+2) = c(2) ! RHS
  k = k+2
end do ! mfilms

uz( 1) = -thin
uz(k-1) = uz(k-2) ! substrate/film interface
uz( k ) = uz(k-2) + thin ! substrate

do mbn=1,mbien ! ambients
  iaws = iaws+1
  imbien = iibent(iaws)
  air = ambient(imbien) ! FILMSS
  un(1) = air ! refractive index
  un(2) = air

  nu = 2 ! n+ik
  write (iplt,111) k,nu,mbn,iw,is
  do j=1,k
    write (iplt,112) j, uz(j), un(j), uk(j)
  end do
end do ! ambient
end do ! wave
else ! advance indices
  i = i + (mfilm*3+2)*mwave
  do iw=1,mwave

106
iws = iws+1
mbien = nnbent(iws)
iaws = iaws+mbien
end do
end if
end do
! sample
return

c============================================================================
c Plot deviations of the fit: b = experiment-model

c 2 continue
llnorm = .false.
call asml ! bb

rd = 180.0/\pi ! radians --> degrees
ngl = 0
ii = 0
ivs = 0
iaws = 0
iraws = 0
do is=l,msaml
mwave = nnwave(is)
do iw=1,mwave
iws = iws+1
mbien = nnbent(iws)
do mbn=1,mbien
iaws = iaws+1
mrpeat = nnpeat(iaws)
do irpeat=1,mrpeat
iraws = iraws+1
mangl = mangle(iraws)

nu = 2 ! (psi, delta)
write (iplt,211) mangl,nu,irpeat,mbn,iw,is
do iangl=1,mangl
ngl = ngl+1
angl = angles(ngl) ! radians
psixm = bb(ii+1)
delxm = bb(ii+2)
angl = angl*rd ! degrees
psixm = psixm*rd
delxm = delxm*rd
write (iplt,212) iangl,angl,delxm,psixm
ii = ii+2
end do ! angle
end do ! repeat
end do ! ambient
end do ! wave
end do ! sample
return

c============================================================================
Output model experimental data suitable for use as input data.

3 continue  ! experiment <--- model
  call arrang
mws = 0
iws = 0
iaws = 0
iaw = 0
ngl = 0  ! measured data
  do is=1,msampl
    mfilm = nnfilm(is)
    mwave = nnwave(is)
    mfims = mfilm+1
    nrow = mfilm*3+2
    do iw=1,mwave
      iws = iws+1
      iwave = iiwave(iws)
      mbien = mnbent(iws)
      qq = waveqq(iwase)
    end do
    do m=1,mfilms  ! films/substrate
      if (m.ne.mfilms) then
        mws = mws+1
        iz = iiifilm(mws)
        zzz(m) = widths(iz)  ! FILMSS
      end if
      do ink=1,2  ! n+ik
        mws = mws+1
        nk = iiifilm(mws)
        c(ink) = diecn(nk)  ! n,k
      end do
      die(m) = cmplx (c(1),c(2))**2  ! FILMSS
    end do  ! mfilms
  end do  ! is
  do mbn=1,mbien  ! ambient
    iaws = iaws+1
    imbien = iiibent(iaws)
    mrpeat = mmpeat(iaws)
    air = ambient(imbien)  ! FILMSS
    do irpeat=1,mrpeat
      iaw = iaw+1
      mangl = mangle(iaw)
      write (iout,311) mangl,imbien,iwae
      write (iplt,311) mangl,imbien,iwae
      do iangl=1,mangl  ! incident angles
        ngl = ngl+1
        angl = angles(ngl)
        call scatr (qq,angl,b,a,c)
        psi = b(1)*180.0/pi  ! psi
        delta = b(2)*180.0/pi  ! delta
        angl = angl*180.0/pi
      end do
    end do
  end do  ! mbn
end do  ! is
au = angleu(ngl) *180.0/pi
pu = psiuu(ngl) *180.0/pi
du = deltamu(ngl) *180.0/pi

if (delta.lt.0.0) then  ! [0,360)
delta = delta + 360.0  
end if

if (delta.ne.0.0) then  ! phase shift
delta = 360.0 - delta
end if

write (iout,312) iangl, angl, au,
delta, du, psi, pu

write (iplt,312) iangl, angl, au,
delta, du, psi, pu

end do  ! angle
end do  ! repeat
end do  ! ambient
end do  ! wave
end do  ! sample
return

==========================================================================

c Output model experimental data, (delta, psi, R, |R|, ...)  
c Scan incident angles.
4 continue
write (iout,411)
read (idat, *) ichoos
write (iout,412) ichoos
if (ichoos.lt.1 .or. ichoos.gt.4) then
    write (iout,413) 'choose'
    stop
end if
write (iout,421)
read (idat, *) iselct
write (iout,422) iselct
if (iselct.lt.1 .or. iselct.gt.2) then
    write (iout,413) 'select'
    stop
end if
write (labelo,423) ichoos, iselct  ! label output

call arrange
mws = 0
iws = 0
iaws = 0
iraws = 0
ngl = 0
do is=1,msample
    mfilm = nnfilm(is)  ! FILMSS
    mwave = nnwave(is)

109
mfilm = mfilm + 1  ! films/substrate
nrow = mfilm * 3 + 2

do iw = 1, nwave
    iws = iws + 1
    iwave = iw + iws
    mbi = mmnber(iws)
    qq = waveqq(iwave)  ! FILMSS

end do

do i = 1, nrow
    iptx(i) = 0
    ipty(i) = 0
end do

iv = 0
kv = 0
mv = 0

do m = 1, mfilms  ! films/substrate
    if (m .ne. mfilms) then  ! films ~ z
        mws = mws + 1
        iz = iiifilm(mws)
        zzz(m) = widths(iz)  ! FILMSS
        iv = iv + 1
        if (lvaryNZ(iz).eq.1) then
            j = mlmnts + iz
            i = iptv(j)
            kv = kv + 1
            iptx(kv) = i
            ipty(kv) = iv
            if (iptw(i).eq.0) then ! unique
                mv = mv + 1
                iptw(i) = mv
            end if
        end if
    end if
end do

nk = iifilm(mws)
c(nk) = diefcn(nk)  ! n,k
iv = iv + 1
if (lvary(nk).eq.1) then
    i = iptv(nk)
    kv = kv + 1
    iptx(kv) = i
    ipty(kv) = iv
    if (iptw(i).eq.0) then
        mv = mv + 1
        iptw(i) = mv
    end if
end if
end do

die(m) = cmplx(c(1),c(2))**2  ! FILMSS
end do  ! mfilms
if (ichoos.eq.4 .and. mv.gt.1) then
    write (iout,431)
    stop
end if

do mbn=1,mbien
  iaws = iaws+1
  imbien = iibent(iaws)
  mrpeat = mrpeat(iaws)
  air = ambient(imbien)
  do irpeat=1,mrpeat
    iraws = iraws+1
    if (iselct.eq.1) then
      mangle = mangle(iraws)
    else
      mangle = 89
    end if
    if (ichoos.eq.1) ncolmn=3
    if (ichoos.eq.2) ncolmn=4
    if (ichoos.eq.3) ncolmn=3
    if (ichoos.eq.4) ncolmn=2
    write (iplt,432) mangle,ncolmn,imbien,iwave,is,
        labelo
    do iangle=1,mangle
      if (iselct.eq.1) then
        ngl = ngl+1
        angl = angles(ngl)
        phi = angl*(180.0/pi)
      else
        phi = float (iangle)
        angl = phi*(pi/180.0)
      end if
      call forwrd (qq,angl, Rs,Rp, dRs,dRp,
          dRsa,dRpa)
      sx = real (Rs)
      sy = aimag (Rs)
      px = real (Rp)
      py = aimag (Rp)
      call polar (sx,sy,sr,sa,1) ! |Rs|
      call polar (px,py,pr,pa,1) ! |Rp|
      sr2 = sr*sr
      pr2 = pr*pr
      sr2pr2 = (sr2+pr2)*0.5 ! total
      call scatr (qq,angl, b,a,c)
      psi = b(1)*180.0/pi ! psi
      delta = b(2)*180.0/pi ! delta
      if (mv.eq.0) then
        dpsi = c(1)*180.0/pi ! psi'
      else
ddel = c(2) *180.0/pi  ! delta'

else

dpsi = 0.0  ! initialize
ddel = 0.0

do k=1,kv

  iv = ipty(k)
  iv2 = iv+iv
  iv1 = iv2-1
  
  dpsi = dpsi + a(iv1)
  ddel = ddel + a(iv2)

end do

dpsi = dpsi *180.0/pi
ddel = ddel *180.0/pi

end if

if (delta.lt.0.0) then  ! [0,360)
delta = delta + 360.0
end if

if (delta.ne.0.0) then  ! phase shift
delta = 360.0 - delta
end if

-----------------------------------------------

if (ichoos.eq.1) then
  write (iplt,433) iangl,phi,
               delta,psi, sr2pr2

  else if (ichoos.eq.2) then
  write (iplt,433) iangl,phi,
               sr,pr,            

  else if (ichoos.eq.3) then
  write (iplt,433) iangl,phi,
               sr,pr,            

  else if (ichoos.eq.4) then
  write (iplt,433) iangl,phi,
               ddel, dpsi
  end if

  else

    stop

  end if

  -----------------------------------------------

  end do  ! angle
  end do  ! repeat
  end do  ! ambient

  do k=1,kv

    i = iptx(k)
    iptw(i) = 0
  end do

  end do

  do k=1,kv

    i = iptx(k)
    iptw(i) = 0
  end do

  write (iout,434)
Output/plot $|g|$ over a grid of model parameters $(v)$, 1D or 2D.

5 continue

call scan2g

return

411 format (" PLTDAT, select the format of the output table."
& "/
& 1, (i,ai, delta,psi, intensity )
& 2, (i,ai, $|R_s|$, $|R_p|$, $|R_s|*2$, $|R_p|*2$)
& 3, (i,ai, $|R_s|$, $|R_p|$, intensity )
& 4, (i,ai, d/d parameter (delta, psi))
& " Enter: option " )

412 format (" option " ', i1 )

413 format (" oops, inconsistent value ascribed unto: ', a6)

421 format (" PLTDAT, select the domain of incident angles."
& "/
& 1, at experiment/measurement points
& 2, grid scan, incident angles (1,89)
& " Enter: option " )
format (', option': ', i1')
format (option:(', i1', ',', i1, ')') ! 12 characters

format (', ... oops, only (0 or 1) parameters may VARY')
format (ix, 5i4, ', mang1,nu, imbien,iwave,sampl', a)
format (ix, i4, ix, f10.4, 1p4e13.4)
format ('/ output written to: x.plot')

end
subroutine scan2
include 'iounit.'
include 'definit.'
include 'filmm.'
include 'scanc.'

logical lbdry, local
tcharacter blank, comma
data blank /'/, comma /','/

Solve the forward scattering problem over a grid of model parameters.

Input grid info for scanning the model parameters

raddeg = 180.0 / 3.14159265 ! degrees ---- radians

write (iout,101)
i = 0
do m=1,mlmnts
   ! (n+ik)
      if (lvary1(m) .eq. 1) then
         i = i+1
         read (idat, *) j, p1, p2, p3
         write (iout,102) i, p1, p2, p3, j, '(n+ik)'
         if (j.ne.m .or. p2.lt.p1 .or. p1.lt.0.0 .or. p2.lt.0.0 .or. p3.lt.0.0) then
            write (iout,103)
            stop
         end if
         if (p3.eq.0.0) p3=p2-p1
         if (p3.eq.0.0) then
            iii2(i) = 1
         else
            iii2(i) = 1 + nint ((p2-p1)/p3)
         end if
         ppp1(i) = p1
         ppp2(i) = p2
         ppp3(i) = (p2-p1) / float (max (1, iii2(i)-1))
         c psav(i) = diefcn(m) ! n+ik
         iptu(i) = m
      end if
   end do

   if (mfilm.ne.0) then
      do m=1,mfilm
         ! z - thickness
            if (lvaryz(m) .eq. 1) then
               i = i+1
               read (idat, *) j, p1, p2, p3
               write (iout,102) i, p1, p2, p3, j, '(z )'
         end if
      end do
if (j.ne.m .or. p2.lt.01 .or. p1.lt.0.0 .or. p2.lt.0.0 .or. p3.lt.0.0) then
write (iout,103)
stop

end if
if (p3.eq.0.0) p3=p2-p1
if (p3.eq.0.0) then
  iii2(i) = i
else
  iii2(i) = 1 + nint ((p2-p1)/p3)
end if
pppl(i) = p1
ppp2(i) = p2
ppp3(i) = (p2-p1) /float (max (1, iii2(i)-1))
psav(i) = widths(m)
iptu(i) = mlmnts+m

end if
end do   ! mfilmm

mvary = i   ! depth of do-loop nest

-------------------------------

Discern breakpoint

read (idat,*,err=11,end=11) mvarii, kt, kts, old
write (iout,121)
write (iout,111) mvarii, kt, kts, old
if (mvarii.ne.mvary .or. kt.lt.1 .or. kts.lt.1 .or.
  old.le.0.0 ) goto 11

do i=1,mvary
  read (idat,*,err=11,end=11) j, iii(i), pppp(i), psav(i)
  write (iout,112) j, iii(i), pppp(i), psav(i)
  if (iiii(i) .lt. 1 .or. j.ne.i .or.
    & iii(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
    stop     ! goto 11
  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,113)
    stop     ! goto 11
  end if
end do
write (iout,114)
iistp = 1

107
itt1 = iftime (i)  ! start clock
goto 2
110
11 continue
write (iout,115)
itt1 = iftime (i)  ! start clock
113
c = Emulate, initialize, activate: the nest of do-loops
do i=1,mvary
    iii(i) = 0  ! initialization: i1-i3
end do

120
kt = 0
121
iistp = 1  ! do-loop increment
122
1 iii = iii+1  ! index of: iii-th nested do.
123
if (iii .gt. mvary) goto 3
124
2 iii(iii) = iii(iii) + iistp  ! update do-loop variable
125
if (iii(iii) .lt. iii2(iii)) then  ! test upper limit
    pppp(iii) = ppp1(iii) + ppp3(iii)*float (iii(iii) -1)
goto 1
else if (iii(iii) .eq. iii2(iii)) then
    pppp(iii) = ppp1(iii)
else
    pppp(iii) = ppp2(iii)
end if
129
goto 1
132
end if
133
iiii(iii) = 0  ! reset inner do, i1-i3
134
iii = iii-1  ! backup one do-level
135
if (iii .eq. 0) goto 4  ! escape nest of do-loops
goto 2
138
3 iii = iii-1  ! level of inner-most nested do.
c = begin processing body

143
kt = kt+1  ! simple counter
do i=1,mvary  ! re-set model parameters
    m = iptu(i)
    if (m .le. mlnmts) then
        diefcn(m) = pppp(i)  ! n+ik
    else
        m = m-mlnmts
        widths(m) = pppp(i)  ! z
end if
end do  ! vary
call zoom2 (ppp1, ppp2, ppp3)  ! iterate " steepest descent
llnorm = .false.
call asml
call norm (meqns,bb,bnorm,i)  ! retain norm of residual
bnorm = bnorm*raddeg  ! degrees
if (kt.eq.1) then
    kts = 1
    old = bnorm
    do i=1,mvary
        m = iptu(i)
        if (m.le.mlnts) then
            psav(i) = diefcn(m)
        else
            m = m-mlnts
            psav(i) = widths(m)
        end if
    end do  ! vary
else if (bnorm.lt.old) then
    kts = 1
    old = bnorm
    do i=1,mvary
        m = iptu(i)
        if (m.le.mlnts) then
            psav(i) = diefcn(m)
        else
            m = m-mlnts
            psav(i) = widths(m)
        end if
    end do  ! vary
else if (bnorm.eq.old) then
    kts = kts+1  ! retain density of states along the minimum
end if
it2 = iftime(i)  ! CPU clock
tim = float (it2-it1) /6.0E4  ! CPU minutes elapsed
if (tim .gt. 15.0) then  ! breakpoint
    it1 = it2
    open (unit=isout, file='x.sout', status='unknown')
    write (isout,111) mvary, kt, kts, old
    do i=1,mvary
        write (isout,112) i, iiii(i), pppp(i), psav(i)
    end do
    call time (bufft)
call date (buffd)
write (isout,116) bufft, buffd ! convenience
close (unit=isout)
end if
--- ! finish processing body
goto 2  ! nested do-loop: iiii
4 continue  ! last line of do-nest, iiii
lbdry = .false.
write (iout,104) kt, kts, old  ! density of states along minimum
write (iout,105)
do i=1,mvary
  local = (psav(i) .le. ppp1(i)+ppp3(i)*0.001) .or.
  & (psav(i) .ge. ppp2(i)-ppp3(i)*0.001)
c*
  local = local .and. (iii2(i).ne.1)
if (local) lbdry=.true.
m = iptu(i)
if (m.le.mlmnts) then
  ! n+ik " diefcn
  diefcn(m) = psav(i)
  if (local) then
    write (iout,106) i, psav(i), ' (n+ik)', blank
  else
    write (iout,106) i, psav(i), ' (n+ik)'
  end if
else
  ! z " widths
  m = m-mlmnts
  widths(m) = psav(i)
  if (local) then
    write (iout,106) i, psav(i), ' (z )', blank
  else
    write (iout,106) i, psav(i), ' (z )'
  end if
end if
end do!
vary
if (lbdry) write (iout,107)
return

101 format (" Scan a grid of model parameters."
  & " Grid info:' , 4x, 'initial', 6x, 'final',
  &  5x, 'increment' )
102 format (3x, i4, ',') , 3f13.4, " for:', i5, ', ', a6)
103 format (" scan, error: inconsistent data input")
104 format (" number of grid points scanned, kt = ', i10
  & " population along the minimum, kts = ', i10
  & " norm of residual, |g| = ', 1pe13.5,
  & " (degrees) ')
105 format (" Model parameter value along the minimum:")
106 format (1x,i4, '), 1p1e15.6, " for:', i5, ', ', a6,
  & a1, " boundary")
107 format (" Note that the minimum point is near a boundary.")
111 format (1x, i4,1x, 2i10, 1p1e15.6, ' mvary, kt, kts, b(min)')
112 format (1x, 2(i4,1x), 1p2e15.6, ' i, ip, p, p(min)')
113 format (" The attempted restart was NOT acceptable.")
114 format (" The attempted restart was acceptable.")
115 format (" Note: NO attempt was made to restart.")
116 format (" wall clock: time = ', a8, 1x, ' hh:mm:ss 
  & '/ date = ', a9, ' dd-mm-yy' )
121 format (')
end
subroutine scan2g  ! Plot |g|
include 'iounit.'  ! Called by: PLTDAT
include 'defnit.'
include 'filmm.'
include 'scancc.'

logical lbdry, local
character blank, comma
data blank /'/, comma ', '/

c Solve the forward scattering problem over a grid of model parameters.

c Input grid info for scanning the model parameters

raddeg = 180.0 / 3.14159265  ! degrees <--- radians
write (iout,101)
i = 0
do m=1,mlmnts  ! (n+ik)
if (lvaryl(m) .eq. 1) then
  i = i+1
  read (idat, *) j, p1, p2, p3
  write (iout,102) i, p1, p2, p3, j, '(n+ik)' &
  if ( j.ne.m .or. p2.lt.p1 .or. p1.lt.0.0 .or. p2.lt.0.0 .or. p3.lt.0.0) then
    write (iout,103)
    stop
  end if
  if (p3.eq.0.0) p3=p2-p1
  if (p3.eq.0.0) then
    iii2(i) = 1
  else
    iii2(i) = 1 + nint ((p2-p1)/p3)
  end if
  ppp1(i) = p1
  ppp2(i) = p2
  ppp3(i) = (p2-p1) /float (max (1, iii2(i)-1))
c psv(i) = diefcn(m)  ! n+ik
  iptu(i) = m
end if
do  ! mlnnts
if (mfilm.mne.0) then
  do m=1,mfilm  ! z " thickness
    if (lvaryz(m) .eq. 1) then
      i = i+1
      read (idat, *) j, p1, p2, p3
      write (iout,102) i, p1, p2, p3, j, '(z )'
end if
end do
if (j.ne.m .or. p2.lt.p1 .or. p1.lt.0.0 .or. p2.lt.0.0 .or. p3.lt.0.0) then
write (iout,103)
stop
end if
if (p3.eq.0.0) p3=p2-p1
if (p3.eq.0.0) then
  iii2(i) = 1
else
  iii2(i) = 1 + nint ((p2-p1)/p3)
end if
ppp1(i) = p1
ppp2(i) = p2
ppp3(i) = (p2-p1) /float (max (1, iii2(i)-1))
psav(i) = widths(m)
iptu(i) = mlmnts+m
end if
end do  ! mfilmm
end if
mvary = i ! depth of do-loop nest
--------------------------------------------------------------------------
c Test and Ensure that we consider only: 1D or 2D plots
--------------------------------------------------------------------------
if (mvary.eq.0 .or. mvary.gt.2) then
  write (iout,211)
  stop
end if
--------------------------------------------------------------------------
c Discern breakpoint
--------------------------------------------------------------------------
read (idat,*,err=11,end=11) mvarii, kt, kts, old
write (iout,121)
write (iout,111) mvarii, kt, kts, old
if (mvarii.ne.mvvar .or. kt.lt.1 .or. kts.lt.1 .or. old.1e.0.0 ) goto 11
do i=1,mvary
  read (idat,*,err=11,end=11) j, iii(i), pppp(i), psav(i)
  write (iout,112) j, iii(i), pppp(i), psav(i)
if (iii(i) .lt. 1 .or. j.ne.i .or. iii(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,113)
    stop  ! goto 11
  end if
endif
if (iii(i).lt.iii2(i) .or. iii2(i).eq.1) then
ppppi = ppp1(i)+ppp3(i)*float(iii(i) -1)  ! [,)
else
ppppi = ppp2(i)

end if

if (abs(pppp(i)-ppppi) .gt. ppp3(i)*0.01) then
write (iout,113)
stop ! goto 11
end if

end do
write (iout,114)
iiistp = 1
ii = mvary
i1 = iftime(i) ! start clock
goto 2 ! start at interior, resume

11 continue ! start at beginning
write (iout,115)
i1 = iftime(i) ! start clock

------------------------------------
c Formulate header card for the plot utility
c Plot format: nx,ny,nu / (i,x,y,u) =====> index backwards
c
m = 1 ! (v,g)
kt = mvary+1
if (mvary .eq. 1) then
write (iplt,212) (iiii2(j), j=1,mvary), m ! header card
else
write (iplt,213) (iiii2(j), j=1,mvary), m ! header card
end if

=====================================
c Emulate, initialize, activate: the nest of do-loops
c
do i=1,mvary
  iii(i) = 0 ! initialization: i1-i3
end do
kt = 0
iiistp = 1 ! do-loop increment
ii = 0 ! index of: nested do-variables.
1 iii = iii+1 ! index of: iii-th nested do.
if (iiii .gt. mvary) goto 3
2 iii(iii) = iii(iii) + iiistp ! update do-loop variable
if (iiii(iii) .lt. ii2(iii)) then ! test upper limit
  pppp(iii) = ppp1(iii) + ppp3(iii)*float (iiii(iii) -1)
goto 1
else if (iiii(iii) .eq. ii2(iii)) then
  if (ii2(iii) .eq. 1) then
  pppp(iii) = ppp1(iii)
  else
  pppp(iii) = ppp2(iii)
  end if
  goto 1
end if
if (iiii .eq. 0) goto 4 ! escape nest of do-loops

122
157  goto 2
158  3 iii = iii-1
159  c
160  -----------------------------------------
161  ! level of inner-most nested do.
162  ! begin processing body
163  c
164  if (m.le.mlnts) then
165      diefcn(m) = pppp(i)
166  else
167      m = m-mlnts
168      widths(m) = pppp(i)
169  end if
170  end do
171  c
172  call zoom2 (ppp1, ppp2, ppp3)
173  -----------------------------------
174  ! iterate - steepest descent
175  c
176  call asmb1
177  call norm (meqns,bb,bnorm,1)        
178  bnorm = bnorm*raddeg
179  c
180  m = mvary+1
181  write (iplt, 214) kt, (pppp(m-j), j=1,mvary), bnorm
182  c
183  if (kt.eq.1) then
184    kts = 1
185    old = bnorm
186  do i=1,mvary
187      m = iptu(i)
188      if (m.le.mlnts) then
189         psav(i) = diefcn(m)
190      else
191         m = m-mlnts
192         psav(i) = widths(m)
193      end if
194    end do
195  else if (bnorm.lt.old) then
196    kts = 1
197    old = bnorm
198  do i=1,mvary
199      m = iptu(i)
200      if (m.le.mlnts) then
201         psav(i) = diefcn(m)
202      else
203         m = m-mlnts
204         psav(i) = widths(m)
205      end if
206    end do
207  else if (bnorm.eq.old) then
208  do i=1,mvary
209      m = iptu(i)
210      if (m.le.mlnts) then
211         psav(i) = diefcn(m)
212      else
213         m = m-mlnts
214         psav(i) = widths(m)
215      end if
216  end do
217  c
218  if (kt.eq.1) then
219    kts = kts+1
220  else if (bnorm.eq.old) then
221  do i=1,mvary
222      m = iptu(i)
223      if (m.le.mlnts) then
224         psav(i) = diefcn(m)
225      else
226         m = m-mlnts
227         psav(i) = widths(m)
228      end if
229    end do
end if

it2 = iftime (i) ! CPU clock

tim = float (it2-it1) /6.0E4 ! CPU minutes elapsed

if (tim .gt. 15.0) then ! breakpoint
  it1 = it2
  open (unit=isout, file='x.sout', status='unknown')
  write (isout,111) mvary, kt, kts, old
  do i=1,mvary
    write (isout,112) i, iii(i), pppp(i), psav(i)
  end do
  call time (bufft)
  call date (buffd)
  write (isout,116) bufft, buffd ! convenience
  close (unit=isout)
end if

------------------------- ! finish processing body
4 goto 2 ! nested do-loop: iii

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

lbdry = .false.

write (iout,104) kt, kts, old ! density of states along minimum
write (iout,105) do i=1,mvary
  local = (psav(i) .le. pppp1(i)+ppp3(i)*0.001) .or. 
    (psav(i) .ge. pppp2(i)-ppp3(i)*0.001)
    & local = local .and. (iii2(i).ne.1)
  if (local) lbdry=.true.
  m = iptu(i)
  if (m.le.mlmlnts) then ! n+ik  " diefcn
    diefcn(m) = psav(i)
    if (local) then
      write (iout,106) i, psav(i), m, '(n+ik)', blank
    else
      write (iout,106) i, psav(i), m, '(n+ik)'
    end if
    else ! z  " widths
      m = m-mlmlnts
      widths(m) = psav(i)
      if (local) then
        write (iout,106) i, psav(i), m, '(z )', blank
      else
        write (iout,106) i, psav(i), m, '(z )'
      end if
    end if
  end do  ! vary
if (lbdry) write (iout,107)
write (iout,108)
return

101 format (/' Scan a grid of model parameters.  (scan2g)'
Grid info:', 4x, 'initial:', 6x, 'final:', 5x, 'increment')

102 format (3x, i4, '), 3f13.4,
& ' for:', i5, '), a6)

103 format ('scan2g, error: inconsistent data input')
104 format ('number of grid points scanned, kt = ', i10
& /' population along the minimum, kts = ', i10
& '/' norm of residual, |g| = ', 1pe13.5,
& ' (degrees) ')
105 format ('Model parameter value along the minimum:')
106 format (ix,i4, ''), 1pe15.6, ' for:', i5, ', ', a6,
& a1, ' boundary')
107 format ('Note that the minimum point is near a boundary.')
108 format ('output written to: x.plot')

111 format (ix, i4,ix, 2i10, 1pe15.6, ' mvary, kt, kts, b(min)')
112 format (ix, 2(i4,ix), 1pe15.6, ' i, ip, p, p(min)')
113 format ('The attempted restart was NOT acceptable.')
114 format ('The attempted restart was acceptable.')
115 format ('Note: NO attempt was made to restart.')
116 format (' wall clock: time = ', a8, ix, ' hh:mm:ss'
& ' date = ', a9, ' dd-mmm-yy')

121 format (' ')
subroutine scan3
include 'iounit.'
include 'definit.'
include 'filmm.'
include 'filmss.'
include 'scancc.'

integer iipt(niii), jjpt(niii)
integer jjjj(niii), jjj2(niii)
real qqqq(niii), qqq1(niii), qqq2(niii), qqq3(niii)
real qsav(niii)

logical lbdry, local, lcoupl
character blimk, comma

! Solve the forward scattering problem over a grid of model parameters.
! Note: the model parameters involve partitions of:
! [ froz / froz (grid) / vary (grid, continuous) ]
! [froz ---> --- vary]

raddeg = 180.0 / 3.14159265  ! degrees <-- radians

=== Input grid info for scanning the model parameters ===

write (iout,iii)  
ki = 0  ! simple counter 
k = 0  ! froz, grid
i = 0  ! vary, grid
do m=1,mlmnts  ! (n+ik)
    if (lvaryl(m).ne.0) then  ! grid
        ki = ki+1
        read (idat, *) j, p1, p2, p3
        write (iout,112) ki, p1, p2, p3, j, '(n+ik)'
        if (j.ne.m .or. p2.lt.p1 .or. p1.lt.0.0 .or. p2.lt.0.0 .or. p3.lt.0.0) then
            write (iout,113)
            stop
        end if
        if (lvaryl(m).eq.1) then  ! vary
            i = i+1
            j = niii+1-i  ! backward
        else
            k = k+1
            j = k
        end if
    end if
if (p3.eq.0.0) p3=p2-p1
if (p3.eq.0.0) then
    iii2(j) = 1
else
    iii2(j) = 1 + nint ((p2-p1)/p3)
end if
ppp1(j) = p1
ppp2(j) = p2
ppp3(j) = (p2-p1) / float (max (1, iii2(j)-1))
psav(j) = diefcn(m) ! n+ik
iipt(j) = m
jjpt(m) = j
end if
end do ! mlnmuts
if (mfilmm.ne.0) then
do m=1,mfilmm ! z " thickness
  if (1varyz(m) .ne. 0) then
    ki = ki+1
    read (idat, *) j, p1, p2, p3
    write (iout,112) ki, p1, p2, p3, j, '(z )'
    write (iout,113)
  &
  if ( (j.ne.m .or. p2.lt.p1 .or.
  p1.lt.0.0 .or. p2.lt.0.0 .or. p3.lt.0.0) then
    write (iout,113)
  stop
  end if
  if (1varyz(m) .eq. 1) then ! vary
    i = i+1 .
    j = iii+1-i ! backwards
  else
    k = k+1
    j = k
  end if
  if (p3.eq.0.0) p3=p2-p1
  if (p3.eq.0.0) then
    iii2(j) = 1
  else
    iii2(j) = 1 + nint ((p2-p1)/p3)
  end if
  ppp1(j) = p1
  ppp2(j) = p2
  ppp3(j) = (p2-p1) / float (max (1, iii2(j)-1))
  psav(j) = widths(m)
  iipt(j) = m+mlnmuts
  jjpt(m+mlnmuts) = j
end if
end do ! mfilmm
end if
mfrozz = k ! depth of do-loop nest/grid: froz
mvaryy = i ! depth of do-loop nest/grid: vary
muv = mfrozz + mvaryy

Output range of do-loops for the grid: froz, vary

if (mfrozz .eq. 0) then
  write (iout,114) mfrozz
  stop
else
  write (iout,121) 'froz'
  do i=1,mfrozz
    m = ipt(i)
    if (m.le.mlmnts) then
      write (iout,122) i, iii2(i), m, '(n+ik)'
    else
      m = m-mlmnts
      write (iout,122) i, iii2(i), m, '(z )'
    end if
  end do
end if

if (mvaryy .eq. 0) then
  write (iout,115) mvaryy
  stop
else
  write (iout,121) 'vary'
  do i=1,mvaryy
    j = iii+1-i ! backwards
    m = ipt(j)
    if (m.le.mlmnts) then
      write (iout,122) i, iii2(j), m, '(n+ik)'
    else
      m = m-mlmnts
      write (iout,122) i, iii2(j), m, '(z )'
    end if
  end do
end if

Discern whether: VARY model parameters are coupled between samples.
Note here that the couplings:
1) between samples are suppressed, while those
2) within the sample are fully accounted.

mm = mlmnts + mfilmm

do m=1,mm
  iptw(m) = 0 ! initialize template
  iptv(m) = 0 ! cumulative
end do

iws = 0
mws = 0

do is=1,msample
\[ mwave = \text{nwave}(is) \]
\[ mfilm = \text{mfilm}(is) \]
\[ mfilms = mfilm + 1 \]
\[ \text{do } iwave = 1, mwave \]
\[ \quad \text{iws} = iws + 1 \]
\[ \quad \text{iwave} = \text{iwave}(iws) \]
\[ \quad \text{mbien} = \text{mbient}(iws) \]
\[ \quad \text{do } m = 1, mfilms \]
\[ \quad \quad \text{if } (m \neq \text{mfilm}) \text{ then} \]
\[ \quad \quad \quad \text{mws} = mws + 1 \]
\[ \quad \quad \quad \text{iz} = \text{ifilm}(mws) \]
\[ \quad \quad \quad \text{if } (\text{lvaryz}(iz) \text{.eq. } 1) \text{ then} \]
\[ \quad \quad \quad \quad j = \text{mlmnts} + iz \]
\[ \quad \quad \quad \quad \text{iptv}(j) = 1 \quad \text{! vary} \]
\[ \quad \quad \text{end if} \]
\[ \quad \text{end if} \]
\[ \text{do } \text{ink} = 1, 2 \]
\[ \quad \text{mws} = mws + 1 \]
\[ \quad \text{nk} = \text{ifilm}(mws) \]
\[ \quad \text{if } (\text{lvaryl}(nk) \text{.eq. } 1) \text{ then} \]
\[ \quad \quad \text{iptv}(nk) = 1 \quad \text{! vary} \]
\[ \quad \text{end if} \]
\[ \quad \text{end do} \quad \text{! n+ik} \]
\[ \text{end do} \quad \text{! films} \]
\[ \text{end do} \quad \text{! waves} \]
\[ \text{do } m = 1, mm \]
\[ \quad \text{iptw}(m) = \text{iptw}(m) + \text{iptv}(m) \quad \text{! overlap} \]
\[ \quad \text{iptv}(m) = 0 \quad \text{! cumulative} \]
\[ \quad \text{end do} \]
\[ \text{end do} \quad \text{! samples} \]
\[ k = 0 \quad \text{! count couplings} \]
\[ \text{do } m = 1, mm \]
\[ \quad \text{if } (\text{iptw}(m) \text{.gt.} 1) \quad k = k + 1 \]
\[ \quad \text{end do} \]
\[ \text{if } (k \text{.eq. } 0) \text{ then} \]
\[ \quad \text{lcoupl} = \text{.false.} \quad \text{! no vary couplings} \]
\[ \text{else} \]
\[ \quad \text{lcoupl} = \text{.true.} \]
\[ \quad \text{write } (iout,131) k \]
\[ \quad \text{do } m = 1, mm \]
\[ \quad \quad \text{if } (\text{iptw}(m) \text{.gt.} 1) \text{ then} \]
\[ \quad \quad \quad \text{if } (m \text{.le. } \text{mlmnts}) \text{ then} \]
\[ \quad \quad \quad \quad \text{write } (iout,132) m, '(n+ik)' \]
\[ \quad \quad \quad \text{else} \]
\[ \quad \quad \quad \quad j = m-mlmnts \]
\[ \quad \quad \quad \quad \text{write } (iout,132) j, '(z)' \]
\[ \quad \quad \text{end if} \]
\[ \quad \text{end if} \]
\[ \text{end do} \]
\[ \text{write } (iout,133) \]

129
c* stop
end if

Discern breakpoint, [froz ---> <<< vary]

write (iout,141)
read (idat,*,err=11,end=11) kk, ii, ktu, bmin
write (iout,142) kk, ii, ktu, bmin
if (kk.ne.mfrozz .or. ii.ne.mvarya .or.
& ktu.lt.1 .or. bmin.lt.0.0 ) goto 11

do i=1,mfrozz
  j = i
  read (idat,*,err=11,end=11) ii, iii(i),pppp(i),psav(i),im
  write (iout,143) ii, iii(i),pppp(i),psav(i),im
  m = iipt(i)
  if (m.gt.mlnts) m=m-mlnts

  if ( ii.ne. i .or. 
& im.ne. m .or. 
& iii(i).lt.1 .or. 
& iii(i).ge.ii2(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,146)
    stop ! goto 11
  end if

  if ( iii(i).lt.ii2(i) .or. iii2(i).eq.1 then
    ppppi = ppp1(i) + ppp3(i)*float (iii(i) -1) ! [,)
  else
    ppppi = ppp2(i)
  end if
  if (abs(pppp(i)-ppppi) .gt. ppp3(i)*0.001) then
    write (iout,146)
    stop ! goto 11
  end if
end do ! mfrozz

if (mvarya.ne.0) then
  do i=1,mvarya
    j = niii+1-i ! backwards
    read (idat,*,err=11,end=11) ii, psav(j), im
    write (iout,144) ii, psav(j), im
    m = iipt(j)
    if (m.gt.mlnts) m=m-mlnts

    if ( ii.ne. i+mfrozz .or. 
& im.ne. m .or. 
& psav(j).lt. ppp1(i)-ppp3(i)*0.001 .or. 

& psav(j) .gt. ppp2(j)+ppp3(j)*0.001  
      then
      write (iout,146)
      stop    ! goto 11
   end if
end do
write (iout,147)
iiistp = 1
iii = mfrozz
it1 = iftime (i)       ! start clock
      goto 22       ! re-start
   continue
write (iout,148)       ! start from beginning
it1 = iftime (i)       ! start clock

============================================================================
      c Emulate, initialize, activate: nest of do-loops (froz)
      c
      ktu = 0               ! index/counter
      do i=1,mfrozz
         iiii(i) = 0        ! initialization: i1-i3
      end do
      iiistp = 1          ! do-loop increment
      iii = 0             ! index of: nested do-variables.
      21 iii = iii+1       ! index of: iii-th nested do.
      if (iii .gt. mfrozz) goto 23
      22 iiii(iii) = iiii(iii) + iiistp       ! update do-loop variable
      if (iiii(iii) .lt. iii2(iii)) then      ! test upper limit
         pppp(iii) = ppp1(iii) + ppp3(iii)*float (iiii(iii) -1)
      goto 21
      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 21
      end if
      iii(iii) = 0           ! reset inner do, ii-i3
      iii = iii-1            ! backup one do-level
      if (iii .eq. 0) goto 24  ! escape nest of do-loops
      goto 22
      23 iii = iii-1       ! level of inner-most nested do.
============================================================================
      c ! begin processing body
      ktu = ktu+1               ! simple counter
      do i=1,mfrozz            ! re-set model parameters
         m = iipt(i)
         if (m.le.mlmts) then
            diefcn(m) = pppp(i)        ! n+ik
         else
      end do


```fortran
  m = m-mlmnts
  widths(m) = pppp(i) ! z
  end if
  end do
  isum = 0 ! sum: meqns
  sumi = 0.0 ! sum: |g|
  Output model parameters associated with grid: froz
  write (iout,151)
  write (iout,152) ktu
  do i=i,mfroz
    m = iipt(i)
    if (m.le.mlmnts) then
      write (iout,153) i, iii(i), pppp(i), m, '(n+ik)'
    else
      m = m-mlmnts
      write (iout,153) i, iii(i), pppp(i), m, '(z )'
    end if
  end do
  lbdry = .false.
  ngl = 0
  iws = 0
  mws = 0
  iaws = 0
  iraws = 0
  do is=1,msampl
    isx = is ! save
    nglx = ngl
    iwsx = iws
    mwsx = mws
    iawsx = iaws
    irawsx = iraws
    do m=1,mm ! localize: vary
      iptw(m) = 0
    end do
    mwave = nnwave(is)
    mfilm = nnfilm(is)
    mfilms = mfilm+1
    do iw=1,mwave
      iws = iws+1
      iwave = iiwave(iws)
      mbien = nnbent(iws)
      do m=1,mfilm
        if (m.ne.mfilms) then
```
mws = mws+1
iz = iifilm(mws)
zzz(m) = widths(iz)
if (lvaryz(iz) .eq. 1) then
  iptw(nlmmts+iz) = 1
end if
end if
do ink=1,2
  mws = mws+1
  nk = iifilm(mws)
  if (lvary1(nk) .eq. 1) then
    iptw(nk) = 1
  end if
end do
do mbn=1,mBien
  ! pointers
  iaws = iaws+1
  mrpeat = nnrepeat(iaws)
do irrepeat=1,mrrepeat
    iaws = iaws+1
    mangle = mangle(iaws)
    ngl = ngl+mangle ! multiple angles
  end do
end do ! repeat
end do ! ambient
c
--------------- ! arrang, localized
i = 0
k = 0
do m=1,mm
  if (iptw(m) .eq. 1) then ! vary, local
    i = i+1
    iptu(i) = m
    iptv(m) = i
    iptw(m) = 0
  else ! froz, otherwise
    k = k+1
    km = mm+1-k
    iptu(km) = m
    iptv(m) = km
  end if
end do
mvary = i
mfroz = k
---------------
if (mvary.eq.0) then ! everything is frozen
  llnorm = .false.
call asmb13 (isx, ivsx, mwsx, iawsx, irawsx, nglx)
call norm (meqns, bb, old,1) ! retain norm of residual
old = old+raddeg ! degrees
goto 34
write (iout,161) mvary, is
stop
end if

------------------------------- do-nest: local

do i=1,mvary
   m = iptu(i)           ! n+ik, ix+mlmnts <--
   j = jjpt(m)
   jjj2(i) = iii2(j)
   qqq1(i) = ppp1(j)
   qqq2(i) = ppp2(j)
   qqq3(i) = ppp3(j)
end do

================================================

c Emulate, initialize, activate: nest of do-loops (vary)

ktv = 0           ! index/counter
do i=1,mvary
   j jjj(i) = 0
end do

iiistp = 1
jjj = 0          ! index of: nested do-variables.
jjj = j jjj + 1   ! index of: iii-th nested do.
if (jjj .gt. mvary) goto 33
if (jjj(j jjj) .eq. jjj2(j jjj)) then
   qqqq(j jjj) = qqq1(j jjj) + iiistp ! update do-loop variable
   if (jjj(j jjj) .le. jjj2(j jjj)) then ! test upper limit
      qqqq(j jjj) = qqq1(j jjj) + qqq3(j jjj)*float (jjj(j jjj) -1)
      goto 31
   else
      qqqq(j jjj) = qqq2(j jjj)
   end if
   goto 31
end if
jjj(j jjj) = 0       ! reset inner do, ii-i3
jjj = jjj - 1       ! backup one do-level
if (jjj .eq. 0) goto 34 ! escape nest of do-loops
goto 32
jjj = jjj - 1       ! level of inner-most nested do.

-----------------------------------------------
  ! begin processing body

ktv = ktv + 1       ! simple counter
do i=1,mvary
   m = iptu(i)
   if (m.le.mlmnts) then
      di excuses(m) = qqqq(i)
   else
      m = m - mlmnts
      widths(m) = qqqq(i)
   end if
end do
call zoom3 (qqq1, qqq2, qqq3, ! damped least squares
    isx, iwsx, mwsx, iawsx, irawsx, nglx)
llnorm = .false.
call asml3 (isx, iwsx, mwsx, iawsx, irawsx, nglx)
call norm (meqns, bb, bnorm, 1) ! retain norm of residual
bnorm = bnorm*raddeg ! degrees

if (ktv.eq.1) then ! initial
    ktvmin = 1
    old = bnorm
    do i=1,mvary
        m = iptu(i)
        j = jjpt(m)
        if (m.le.mlmnts) then
            qsav(j) = diefcn(m)
        else
            m = m-mlmnts
            qsav(j) = widths(m)
        end if
    end do
else if (bnorm.lt.old) then
    ktvmin = 1
    old = bnorm
    do i=1,mvary
        m = iptu(i)
        j = jjpt(m)
        if (m.le.mlmnts) then
            qsav(j) = diefcn(m)
        else
            m = m-mlmnts
            qsav(j) = widths(m)
        end if
    end do
else if (bnorm.eq.old) then ! along minimum
    ktvmin = ktvmin+1 ! population
end if

----------------------------------------------------------------------------------- ! finish processing body
goto 32 ! nested do-loop: jjjj
continue ! last line of do-nest

Output results: VARY model parameters of the sample.

if (is.eq.1) then
    write (iout,171)
end if
write (iout,172) is, old

if (mvary.ne.0) then
    do i=1,mvary
        m = iptu(i)
        j = jjpt(m)
        local = (qsav(j) .le. ppp1(j)+ppp3(j)*0.001) .or.
&
  (qsav(j) .ge. ppp2(j)-ppp3(j)*0.001)
  local = local .and. (iii2(j).ne.1)
  if (local) lbdry=.true.

  if (m.le.mlmnts) then
    if (local) then
      write (iout,173) qsav(j), m, '(n+ik)', blank
    else
      write (iout,173) qsav(j), m, '(n+ik)'
    end if
  else
    m = m-mlmnts
    if (local) then
      write (iout,173) qsav(j), m, '(z )', blank
    else
      write (iout,173) qsav(j), m, '(z )'
    end if
  end if
  end do

  isum = isum + meqns ! meqns
  sumi = sumi + old*old*float (meqns) ! |g|

  end do ! samples

  Note: summing |b(s)| from each sample ==> NO coupling between samples by VARY parameters.

  sumi = sqrt (sumi/float (isum)) ! |g| without coupling
  if (lbdry) then
    write (iout,174) sumi, comma
  else
    write (iout,174) sumi
  end if

  If coupling exists between samples via VARY model parameters, the calculation of the cumulative residual is meaningless, because the coupled VARY parameter is NOT necessarily held common.

  Discern the cumulative residual along the minimum.
  Restore the VARY model parameters associated with the minimum.

  if (nvaryy.ne.0) then
    do i=1,nvaryy
      j = niii+1-i ! backwards
      m = iipt(j)
      if (m.le.mlmnts) then
        diefcn(m) = qsav(j)
      else
        m = m-mlmnts
        widths(m) = qsav(j)
end if
end do
end if

-------------------------------------------------------------
llnorm = .false.
call asbl
call norm (meqns,bb,bnorm,1) ! retain norm of residual
bnorm = bnorm*raddeg ! degrees

if (ktu.eq.1) then ! density of states
bmin = bnorm
do i=1,muv ! retain model parameters
  if (i.le.mfrozz) then
    j = i
  else
    j = niit+1 - (i-mfrozz)
  end if
  m = iipt(j)
  if (m.le.mlmnts) then
    psav(j) = diefcn(m)
  else
    m = m-mlmnts
    psav(j) = widths(m)
  end if
end do
else if (bnorm.lt.bmin) then
ktumin = 1
bmin = bnorm ! retain model parameters
do i=1,muv
  if (i.le.mfrozz) then
    j = i
  else
    j = niit+1 - (i-mfrozz)
  end if
  m = iipt(j)
  if (m.le.mlmnts) then
    psav(j) = diefcn(m)
  else
    m = m-mlmnts
    psav(j) = widths(m)
  end if
end do
else if (bnorm.eq.bmin) then ! retain density of states
ktumin = ktumin+1 ! along the minimum
end if

-------------------------------------------------------------

 Breakpoint

 c Breakpoint

  it2 = iftime (i) ! CPU clock
  tim = float (it2-it1) /6.0E4 ! CPU minutes elapsed

  137
if (tim.ge.15.0) then               ! breakpoint
  if1 = it2

  close (unit=iout)
  open (unit=iout, file='x.out', status='old', access='append')

  open (unit=iout, file='x.sout', status='unknown')
  write (isout,142) mfrozz, mvaryy, ktu, bmin
  do i=1,mfrozz
      m = iipt(i)
      if (m.le.mlmnts) then
          write (isout,143) i, iii(i), pppp(i), psav(i), m,
                            ', (n+ik), froz'
      else
          m = m-mlmnts
          write (isout,143) i, iii(i), pppp(i), psav(i), m,
                            ', (z ), froz'
      end if
  end do

  if (mvaryy.ne.0) then
    do i=1,mvaryy
      ii = i+mfrozz            ! convenience counter
      j = iii(i)+1-i           ! backwards
      m = iipt(j)
      if (m.le.mlmnts) then
          write (isout,144) ii, psav(j), m,
                            ', (n+ik), vary'
      else
          m = m-mlmnts
          write (isout,144) ii, psav(j), m,
                            ', (z ), vary'
      end if
    end do
  end if

  call time (bufft)
  call date (buffd)
  write (isout,200) buffet, buffd        ! convenience
  close (unit=isout)
  end if

  c-------------------------------------- ! finish processing body
  goto 22                                  ! nested do-loop: iii
  continue                                ! last line of do-nest
  c--------------------------------------

  write (iout,151)
  write (iout,181) ktu, ktumin, bmin
  lbdry = .false.

  write (iout,182) 'froz'
  do i=1,mfrozz
    j = i
    m = iipt(i)
local = (psav(j) .le. ppp1(j)+ppp3(j)*0.001) .or. 
        (psav(j) .ge. ppp2(j)-ppp3(j)*0.001)
* local = local .and. (iii2(j).ne.1)
if (local)  lbdry=.true.
if (m.le.mlmnts) then
  diefcn(m) = psav(j)
  if (local) then
    write (iout,183) psav(j), m, '(n+ik)', blank
  else
    write (iout,183) psav(j), m, '(n+ik)'
  end if
else
  m = m-mlmnts
  widths(m) = psav(j)
  if (local) then
    write (iout,183) psav(j), m, '(z )', blank
  else
    write (iout,183) psav(j), m, '(z )'
  end if
end if
end do
if (mvary .ne. 0) then
  write (iout,182) 'vary'
do i=1,mvary
  j = niit+1-i
  m = iipt(j)
  local = (psav(j) .le. ppp1(j)+ppp3(j)*0.001) .or. 
        (psav(j) .ge. ppp2(j)-ppp3(j)*0.001)
* local = local .and. (iii2(j).ne.1)
if (local)  lbdry=.true.
if (m.le.mlmnts) then
  diefcn(m) = psav(j)
  if (local) then
    write (iout,183) psav(j), m, '(n+ik)', blank
  else
    write (iout,183) psav(j), m, '(n+ik)'
  end if
else
  m = m-mlmnts
  widths(m) = psav(j)
  if (local) then
    write (iout,183) psav(j), m, '(z )', blank
  else
    write (iout,183) psav(j), m, '(z )'
  end if
end if
end do
if (lbdry ) write (iout,184)
if (lcoupl) write (iout,133)
return

111 format (' Scan a grid of model parameters.'
 & '/ Grid info:', 4x, 'initial', 6x, 'final',
 & 5x, 'increment')

112 format (1x, i4, '), 3x, 3f13.4,
 & ' for:', i5, ', ', a6)

113 format ('/ oops, inconsistent data input')

114 format ('/ oops, number of model parameters in the grid:
 & '/' froz ', i4
 & '/' use alternate option: ichoic 3')

115 format ('/ oops, number of model parameters in the grid:
 & '/' vary ', i4 )

121 format ('/ Range of do-loops: ', a4)

122 format (1x, i4, '), i7, 3x, 'grid points for:', i6, ', ', a6)

131 format ('/ Couplings between samples involve: ', i3,
 & ' distinct model parameters')

132 format (9x, i4, ', ', 3x, a6)

133 format ('/ Note: The existence of coupling between samples',
 & '/' due to the VARY model parameters -- ',
 & '/ induces an unusual interpretation unto the residual |g|',
 & '/ because, while providing coupling, they are NOT ',
 & '/ necessarily of similar value on different samples. /)

141 format ('/ Attempt to read breakpoint information')

142 format (1x, 3i5, 1p1e15.6, 26x, ' mfroz, mvary, ',
 & ' ktu, bmin')

143 format (1x, 2i5, 5x, 1p2e15.6, i5, 5x, ' i, ip, p, psav, m',
 & a16 )

144 format (1x, i5, 10x, 1p1e15.6, i6, 5x, ' i, psav, m',
 & a16 )

146 format ('/ The attempted restart was NOT acceptable.')

147 format ('/ The attempted restart was acceptable.')

148 format ('/ Note: NO attempt was made to restart.')

161 format (1x, 17('====') )

162 format (4x, '#', i7, 19x, ' grid: froz')

153 format (1x, i4, '), 2x, i4, 1x, 1p1e15.6,
 & ' model parameter for:', i5, ', ', a6)

161 format ('/ oops, mvary =', i4, '<------- Note, local'
 & ' sample =', i4 )

171 format (5x, 16('-----')
 & '/ 8x, 'sample', 17x, ' grid: vary')

172 format (8x, i4, 1x, 1p1e15.6, 3x, '|g|')

173 format (13x, 1p1e15.6,
 & ' for:', i5, ', ', a6, a1, 8x, ' boundary')

174 format (13x, 1p1e15.6, '|g| summed over samples',
 & a1, ' boundary')
181 format (" number of FROZ grid points scanned, ktu = ", i10
& "/" population along the minimum, ktum = ", i10
& "/" norm of residual, |g| = ",
& eps13.6, ",(degrees)"
182 format (" Model parameter value along the minimum: ", 4x, a4)
183 format ( 13x, 1pe15.6, " for: ", i5, ", ", a6,
& a1, " boundary")
184 format (" Note that the minimum point is near a boundary.")
200 format (" wall clock: time = ", a8, ix, " hh:mm:ss ",
& "/" date = ", a9, " dd-mmm-yy ")
end
subroutine zoom  ! unconstrained optimization
include 'iounit.'
include 'defnlt.'
include 'filmnn.'
include 'wstack.'
include 'cgnxxl.'
data  small  /  1.0E-5 /
raddeg = 180.0 / 3.14159265
llnorm = .false.  ! ASMBL
call asml  ! ia, ja, aa, bb
call norm (meqns, bb, bn, 1)  ! residual
call scaljj (meqns, mvary, ia, ja, aa, xx, aats, w, 2)  ! scale columns, A
if (bn .eq. 0.0) return
bn = bn*raddeg  ! degrees <-- radians
bn1 = bn  ! initial
bn2 = bn  ! last
loop = 0
niter = mvary*4  ! number of iterations
write (iout,101)
write (iout,102) loop, bn
1  itry = 0  ! initialize
loop = loop+1
do i=1,mvary  ! initialize
  xx(i) = 0.0  ! Newton step
end do
call cgnl (meqns, mvary, ia, ja, aa, bb, xx, &
  niter, u, v, w, xx, 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)*0.2  ! step length
  s = se(i)*0.2  ! estimated std dev
  if (j.le.mlmnts) then  ! n+ik
    f = diefcn(j)
    g = uncerl(j)
c* write (iout,121) i, j, f, h
    ha = amax1 (s, 0.001, abs(f*0.2))
    ha = amax1 (g, abs(h), ha)
    if (h.lt.0.0) ha=-ha
    aat(i) = f  ! retain
    diefcn(j) = amax1 (0.0, f+ha)
else
  j = j-mlmnts
  f = widths(j)
  g = uncerz(j)
  write (iout,122) i,j,f,h
  ha = amax1 (s, 0.01, f*0.2)
  ha = amini (g, abs(h), ha)
  if (h.lt.0.0) ha=-ha
  aat(i) = f
  widths(j) = amax1 (0.0, f+ha)
end if

end do

call asmlbl
! ia,ja,aa,bb
call norm (meqns, bb, bn, i)
! residual
call scaljj (meqns,mvary,ia,ja,aa,xx, aats,w,2) ! scale columns, A
bn = bn*raddeg
! degrees <--- radians
! total reduction
relat = bn /bn2
! relative reduction

if (total .le. 1.0E-5) then
  bn2 = bn
  goto 4
end if

if (relat .le. 0.999) then
  bn2 = bn
  goto 3
end if

do i=1,mvary
  j = iptu(i)
  if (j.le.mlmnts) then
    diefcn(j) = aat(i)
  else
    j = j-mlmnts
    widths(j) = aat(i)
  end if
end do

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

if (loop .le. 3) then
  write (iout,104)
if (relat .le. 1.0) then
  write (iout,105)
else
  convergence
end if

if (loop .le. 3) then
  write (iout,104)
if (relat .le. 1.0) then
  write (iout,105)
else
  divergence
end if
write (iout,106)
end if
end if
goto 5  ! escape

3 write (iout,103) loop, relat, total, bn
goto 1
4 write (iout,103) loop, relat, total, bn
5 continue

Output results of iterations involving least squares.

write (iout,111)  ! results
do i=1,mvay
  j = iptu(i)
  if (j.le.mlmnts) then  ! n+ik
    write (iout,112) i, diefcn(j), se(i), j
  else  ! z
    j = j-mlmnts
    write (iout,113) i, widths(j), se(i), j
  end if
end do
write (iout,114) bn1, bn2  ! compare
return

101 format ('/ zoom:  loop, ratio of reduction, |g|' & '/ (rel) (total) ')
102 format (8x, i5, 24x, 1pe12.3, ' (degrees)' )
103 format (8x, i5, 1pe12.3)
104 format (8x, 'stepsize reduction attempted.')
105 format (8x, '... slow convergence.')
106 format (8x, '... divergence.')

111 format ('/ model parameter value along the minimum: ')
112 format (1x, i4, ' '), f15.4, f13.5, ' for:',
       & '/' initial |g| =', 1pe13.5, ' (degrees)' & '/ final |g| =', e13.5)
113 format (1x, i4, ' '), f15.4, f13.5, ' for:',
       & i4, ' (z ), estimated uncertainty')
114 format (20x, i5, ' '), i5, 1pe15.5, ',', (n+ik), d(n+ik)'
115 format (20x, i5, ' '), i5, 1pe15.5, ',', (z ), d(z )')

end
subroutine zoom2 (ppp1, ppp2, ppp3)  ! constrained optimization
   real         ppp1(1), ppp2(1), ppp3(1)  ! used by:  SCM2

include 'iounit.'
include 'definit.'
include 'film3m.'
include 'wstack.'
include 'cgnx1.'
data     small / 1.0E-5 /

raddeg = 180.0 / 3.14159265  ! degrees <--- radians
llnorm = .false.  ! ASMBL
call asmb1  ! ia, ja, aa, bb
call norm (meqns, bb, bn, 1)  ! residual
call scaljj (meqns, mvary, ia, ja, aa, xx, aats, w, 2) ! scale columns, A

if (bn .eq. 0.0) return  ! degrees <--- radians
bn = bn*raddeg  ! initial
bn1 = bn  ! last
bn2 = bn
loop = 0
niter = mvary*4  ! number of iterations
c write (iout,101)
c write (iout,102)  loop, bn

1 itry = 0  ! initialize
loop = loop+1
2 do i=1,mvary  ! initialize
   xx(i) = 0.0  ! Newton step
end do
call cgnl (meqns, mvary, ia, ja, aa, bb, xx, &
    niter,    u, v, w, xx, 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)*0.2  ! step length
   s = se(i)*0.2  ! estimated std dev
   if (j .le. mlmnts) then  ! n+ik
      f = diefcn(j)
      g = uncencr(j)
   c* write (iout,121) i,j,f,h
      ha = amax1 (s, 0.001, abs(f+h))
      ha = amini (g, abs(h), ha, ppp3(i))
      if (h.lt.0.0) ha=-ha
end do
aat(i) = f ! retain

diefcn(j) = amax1 (0.0, f+ha)

if (diefcn(j) .lt. ppp1(i)) then ! ensure compactness
    diefcn(j) = ppp1(i)
else if (diefcn(j) .gt. ppp2(i)) then
    diefcn(j) = ppp2(i)
end if

else ! z
    j = j-mlmnts
    f = widths(j)
    g = uncerz(j)
    write (iout,122) i,j,f,h
    ha = amax1 (s, 0.01, f*0.2)
    ha = amini (g, abs(h), ha, ppp3(i))
    if (h.lt.0.0) ha=-ha
    aat(i) = f ! retain
    widths(j) = amax1 (0.0, f+ha)

    if (widths(j) .lt. ppp1(i)) then ! ensure compactness
        widths(j) = ppp1(i)
    else if (widths(j) .gt. ppp2(i)) then
        widths(j) = ppp2(i)
    end if
end if
end do

call asmbl ! ia,ja,aa,bb
call norm (meqns, bb, bn, 1) ! residual
call scaljj (meqns,mvary,ia,ja,aa,xx, aats,v,2) ! scale columns, Anb = bn*raddeg ! degrees

total = bn /bn1 ! total reduction
relat = bn /bn2 ! relative reduction

if (total .le. 1.0E-5) then ! convergence
    bn2 = bn ! retain
    goto 4 ! escape
end if

if (relat .le. 0.999) then ! converging
    bn2 = bn ! retain
    goto 3 ! iterate
end if

do i=1,mvary ! reset
    j = iptu(i)
    if (j.le.mlmnts) then
        diefcn(j) = aat(i)
    else
        j = j-mlmnts
        widths(j) = aat(i)
    end if
end do
if (itr.y .lt. 3) then
  itry = itry+1
  do i=1,mvary
    xx(i) = xx(i)*0.5
  end do
  goto 2
end if

if (loop .le. 3) then
  ! convenience
  cwrite (iout,104)
  if (relat .le. 1.0) then
    ! marginal
    cwrite (iout,105)
  else
    ! divergence
    cwrite (iout,106)
  end if
end if

goto 5
! escape

3 continue
write (iout,103) loop, relat, total, bn
goto 1
4 continue
write (iout,103) loop, relat, total, bn
5 continue
write (iout,114) bn1, bn2
! compare
return

101 format ('zoom2: loop, ratio of reduction, |g|' & '/' rel (total) ')
102 format ( 8x, i5, 24x, ip1e12.3, ' (degrees)' )
103 format ( 8x, i5, ip3e12.3)
104 format ( 8x, 'stepsize reduction attempted.' )
105 format ( 8x, '... slow convergence.' )
106 format ( 8x, '... divergence.' )

111 format ('model parameter value along the minimum: '
114 format ('initial |g| =', ip1e13.5, ' (degrees)'
118 format (20x, 2i5, 1p2e15.5, ', diefcn')
122 format (20x, 2i5, 1p2e15.5, ', widths')

end
subroutine zoom3 (ppp1, ppp2, ppp3,
& isx, iwsx, mwsx, iawsx, irawsx, nglx)
real ppp1(1), ppp2(1), ppp3(1)
integer isx, iwsx, mwsx, iawsx, irawsx, nglx
include 'icunit.'
include 'definit.'
include 'filmimn.'
include 'wstack.'
include 'cgxxi.'
data small / 1.0E-5 /
c
This routine is called/used by: SCAN
raddeg = 180.0 / 3.14159265 ! degrees <- radians
llnorm = .false. ! ASMBL
call asml3 (isx, iwsx, mwsx, iawsx, irawsx, nglx) ! ia,ja,aa,bb
call norm (meqns, bb, bn, 1) ! residual
call scaljj (meqns,mvary,ia,ja,aa,bb,xx,aats,u,v,w,2) ! scale columns, A
if (bn .eq. 0.0) r-- rm
bn = bn*raddeg ! degrees <- radians
bn1 = bn ! initial
bn2 = bn ! last
loop = 0
niter = mvary*4 ! number of iterations
c write (iout,101)
c write (iout,102) loop, bn
itry = 0 ! initialize
loop = loop+1
do i=1,mvary
xx(i) = 0.0 ! Newton step
end do
call cgnl (meqns,mvary, ia,ja,aa,bb,xx,
& u,v,w, xw,se)
do i=1,mvary
xx(i) = xx(i)/aats(i) ! account for scaling
se(i) = se(i)/aats(i)
end do
2 do i=1,mvary
j = iptu(i)
h = xx(i)*0.2 ! step length
s = se(i)*0.2 ! estimated std dev
if (j .le. mlmnts) then ! n+ik
  f = diefcn(j)
end if
148
g = uncerl(j)
write (iout,121) i,j,f,h
ha = amax1 (s, 0.001, abs(f*0.2))
ha = amini (g, abs(h), ha, ppp3(i))
if (h.lt.0.0) ha=-ha
aat(i) = f ! retain
diefcn(j) = amax1 (0.0, f+ha)

if (diefcn(j) .lt. ppp1(i)) then ! ensure compactness
diefcn(j) = ppp1(i)
else if (diefcn(j) .gt. ppp2(i)) then
diefcn(j) = ppp2(i)
end if
else ! z

j = j-mlmnts
f = widths(j)
g = uncerz(j)
write (iout,122) i,j,f,h
ha = amax1 (s, 0.01, f*0.2)
ha = amini (g, abs(h), ha, ppp3(i))
if (h.lt.0.0) ha=-ha
aat(i) = f ! retain
widths(j) = amax1 (0.0, f+ha)

if (widths(j) .lt. ppp1(i)) then ! ensure compactness
widths(j) = ppp1(i)
else if (widths(j) .gt. ppp2(i)) then
widths(j) = ppp2(i)
end if
end if
end do
call asmbl3 (isx, iwsx, mwsx, iawsx, irawsx, nglx) ! ia,ja,aa,bb
call norm (meqns, bb, bn, 1) ! residual
call scaljj (meqns, mvar7, ia, ja, aa, xx, aats.o, 2) ! scale columns, A
bn = bn*raddeg ! degrees
total = bn /bn1 ! total reduction
relat = bn /bn2 ! relative reduction
if (total .le. 1.0E-5) then ! convergence
bn2 = bn ! retain
goto 4 ! escape
end if
if (relat .le. 0.999) then ! converging
bn2 = bn ! retain
goto 3 ! iterate
end if
do i=1,mvar
  j = iptu(i)
  if (j.le.mlmnts) then
diefcn(j) = aat(i)
do
else
    j = j-mlmnts
widths(j) = aat(i)
end if
end do

if (i try.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,104)
c
    if (relat .le. 1.0) then  ! marginal
        write (iout,105)
c
    else  ! divergence
        write (iout,106)
c
    end if
    goto 5  ! escape
end if

3 continue
write (iout,103) loop, relat, total, bn
goto 1
4 continue
write (iout,103) loop, relat, total, bn
5 continue
write (iout,114) bn1, bn2  ! compare
return

101 format (/' zoom3: loop, ratio of reduction, |g|'
& '/ (rel) (total)' )
102 format ( 8x, i5, 24x, 1p1e12.3, ' (degrees)' )
103 format ( 8x, i5, 1p3e12.3)
104 format ( 8x, 'stepszie reduction attempted.' )
105 format ( 8x, '... slow convergence.' )
106 format ( 8x, '... divergence.' )

111 format (/' model parameter value along the minimum: ')
114 format ('/ initial |g| =', 1p1e13.5, ' (degrees)' )
& '/ final |g| =', e13.5)
121 format (20x, 2i5, 1p2e15.5, ', diefcn')
122 format (20x, 2i5, 1p2e15.5, ', widths')
end
subroutine asmbl
include 'iounit.'
include 'defnit.'
include 'filmm.'
include 'filmss.'
include 'seamx1.'

logical firstv
real a(nrows*2), b(2), c(2)

The sparse matrix format for the model parameters is of the form:

\[ [(n/k)_{(1)}, \ldots, (n/k)_{(mlmnts)}] \text{ diefcn (w)} \]

\[ [ z_{(1)}, \ldots, z_{(mfilms)}] \text{ widths} \]

call arrang
mm = mmlnts+mfilm
ii = 1
ia(1) = 1
jj = 0
iws = 0
mws = 0
iawss = 0
ngl = 0

do is=1,msamp
  mfilm = nnfilm(is)
  mwave = nnwave(is)
  mfilms = mfilm+1
  nrow = mfilm*3+2
  do iw=1,mwave
    iws = iws+1
    iwave = iwave(iws)
    mbien = nnbent(iws)
    qq = waveqq(iwave)
    do i=1,nrow
      iptx(i) = 0
      ipty(i) = 0
    end do
  end do

  iv = 0
  kv = 0
  mv = 0

  do m=1,mfilms
    if (m.ne.mfilms) then
      mws = mws+1
      iz = iifilm(mws)
      zzzz(m) = widths(iz)
    iv = iv+1
  end do
if (lvarys(iz).eq.1) then ! vary
  j = mlmnts+iz
  i = iptv(j) ! nonlocal, compress
  kv = kv+1 ! local, non-unique
  iptx(kv) = i ! local * nonlocal
  ipty(kv) = iv ! local
  if (iptw(i).eq.0) then ! compress, unique
    mv = mv+1 ! local counter
    iptw(i) = mv ! local
    jj = jj+1 ! within row A
    ja(jj) = i ! column
  end if
end if
do ink=1,2 ! n+ik
  mws = mws+1
  nk = iiifilm(mws)
  c(ink) = diefcn(nk) ! n,k
  iv = iv+1 ! local, full
if (lvaryl(nk).eq.1) then ! vary
  i = iptv(nk) ! nonlocal, compress
  kv = kv+1 ! local, non-unique
  iptx(kv) = i ! local * nonlocal
  ipty(kv) = iv ! local
  if (iptw(i).eq.0) then ! compress, unique
    mv = mv+1 ! local counter
    iptw(i) = mv ! local
    jj = jj+1 ! within row A
    ja(jj) = i ! column
  end if
end if
end do
die(m) = cmplx (c(1), c(2)) **2 ! FILMSS
end do ! mfilms
if (mv.eq.0) then
  write (iout,100) is,iw,iwave
  stop
end if
firstv = .true.
do mbn=1,mbien ! ambients
  iaws = iaws+1
  imbien = iiibent(iaws)
  mrpeat = nnpeat(iaws)
  air = ambient(imbien) ! FILMSS
  do irpeat=1,mrpeat
    iraws = iraws+1
    mangl = mangle(iraws)
do iangl=1,mangl ! incident angles
    ngl = ngl+1 ! measured data
      angl = angles(ngl)
end do

end program
call scatr (qq,angl, b,a,c)

write (iout,101) ngl, psiiis(ngl), deltas(ngl), b

bb(ii ) = psiiis(ngl) - b(1)  ! psi
bb(ii+1) = deltas(ngl) - b(2)  ! delta

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

if (mv.ne.0) then
  if (firstv) then ! psi completed already
    firstv = .false.
    do j=1,mv ! unique
      jj = jj+1
      ja(jj) = ja(jj-mv) ! delta
    end do
  else
    do j=1,mv ! unique
      jj = jj+1
      ja(jj ) = ja(jj-mv) ! psi
      ja(jj+mv) = ja(jj ) ! delta
    end do
    jj = jj+mv
  end if

j1 = ia(ii )
j2 = ia(ii+1)-1

do j=j1,j2 ! compress
  aa(j ) = 0.0 ! psi
  aa(j+mv) = 0.0 ! delta
end do

do k=1,kv ! local, vary, non-unique
  iv = ipty(k) ! local, full
  iv2 = iv+iv ! delta " a " local,full
  iv1 = iv2-1 ! psi " aa
  i  = iptx(k) ! nonlocal, compress
  imv = iptw(i) ! local, compress --> ja,aa
  jj1 = ji-1+imv ! psi " aa
  jj2 = jj1 + mv ! delta " aa
  aa(jj1) = aa(jj1) + a(iv1) ! psi'
  aa(jj2) = aa(jj2) + a(iv2) ! delta'
end do ! two rows in A
end if

Renormalize rows in the matrix, least square.
if (llnorm) then
  bb(ii ) = bb(ii )/psiiiu(ngl)  ! psi
  bb(ii+1) = bb(ii+1)/deltau(ngl)  ! delta
end if
if (mv.ne.0) then
  do j=1,mv
    aa(j1-1+j) = aa(j1-1+j) /psiiu(ngl)  psi
    aa(j2 +j) = aa(j2 +j) /deltau(ngl)  delta
  end do
  end if

ii = ii+2
end do  ! angle
end do  ! repeat
end do  ! ambient

if (mv.ne.0) then
  do k=1,kv
    i = iptx(k)
    iptw(i) = 0
  end do
  end if

end do  ! wave
end do  ! sample
msqns = ii-1

if (jj .ne. ia(ii)-1) then
  write (iout,102) ii, jj, ia(ii)
  stop
end if
if (jj.gt.nnjaaa) then
  write (iout,104)
  stop
end if
return

100 format (' oops, there is NO varying model parameter'
  &/' for the case involving:  sample = ' , i3
  &/' where the ' , i3, ' -th wave = ' , i3)
101 format (' asmbl, ' , i5, ip2e14.4, ' 
  &/' ' , 5x, 2e14.4)
102 format (' asmbl, inconsistent format of sparse matrix, '
  &/' ii = ' , i10
  &/' jj = ' , i10, ' /= ' , i10, ' = ia(ii)-1 ')
103 format (' asmbl, array allocation for the sparse matrix '
  &/' has been exceeded. '
  &/' aa,ja <=--- nnjaaa (DEFNIT.)' )
end
13 called by: ZOOM3

subroutine asmbl3 (isx, iwsx, mwsx, iawsx, irawsx, nglx)

include 'iounit.'
include 'defnit.'
include 'filmmm.'
include 'filmss.'
include 'seamxl.'

logical firstv
real a(nrows*2), b(2), c(2)

The sparse matrix format for the model parameters is of the form:
[(n/k)_1, ..., (n/k)_(mmlnts)]  diefcn (w)
[ z_(1), ..., z_(mfilms)]  widths

! model parameters
! IA
! JA

! measured data
! FILMSS
! films/substrate
! initialize local pointers
! varies --> unique
! varies --> full
! local, full, non-unique
! local, vary, non-unique
! local, vary, compress
! films/substrate

do is=1,msampl
mfilm = nfilm(is)
mwave = nnwave(is)
mfilms = mfilm+1
nrow = mfilm+3+2
do iw=1,mwave
  iws = iws+1
  iwave = iwave(iws)
  mbien = nnbent(iws)
  qq = waveqq(iwave)
do i=1,nrow
    iptx(i) = 0
    ipty(i) = 0
end do
iv = 0
kv = 0
mv = 0
do m=1,mfilms
if (m.ne.mfilms) then          ! films > z
  mws = mws+1
  iz = ifilm(mws)
  zzzz(m) = widths(iz)       ! FILMSS
  iv = iv+1                  ! local, full
if (lvaryz(iz).eq.1) then     ! vary
  j = mlmnts+iz
  i = iptv(j)                ! nonlocal, compress
  kv = kv+1                  ! local, non-unique
  iptx(kv) = i               ! local, nonlocal
  ipty(kv) = iv               ! local
if (iptw(i).eq.0) then         ! compress, unique
  mv = mv+1                  ! local counter
  iptw(i) = mv                ! local
  jj = jj+1                  ! within row A
  ja(jj) = i                 ! column
  end if
end if

do ink=1,2
  mws = mws+1
  nk = ifilm(mws)
  c(ink) = dieific(nk)       ! n,k
  iv = iv+1                  ! local, full
if (lvaryy(nk).eq.1) then      ! vary
  i = iptv(nk)               ! nonlocal, compress
  kv = kv+1                  ! local, non-unique
  iptx(kv) = i               ! local, nonlocal
  ipty(kv) = iv               ! local
if (iptw(i).eq.0) then         ! compress, unique
  mv = mv+1                  ! local counter
  iptw(i) = mv                ! local
  jj = jj+1                  ! within row A
  ja(jj) = i                 ! column
  end if
end if

end do

die(m) = cmplx (c(1), c(2)) **2     ! FILMSS
end do          ! mfilms

if (mv.eq.0) then
  write (iout,100) is, iw, iwave
  stop
end if

firstv = .true.
do mbn=1,mbien
  iaws = iaws+1
  imbien = iibent(iaws)
  mrpeat = nnpeat(iaws)
  air = ambient(imbien)       ! FILMSS
  do irpeat=1,mrpeat

156
IRAWS = IRAWS + 1
MANGL = MANGLE(IRAWS)
DO IANGL=1,MANGL
   NGL = NGL + 1
   ANGL = ANGLES(NGL)
   CALL SCATR (QQ, ANGL, B, A, C)
   WRITE (IOUT,101) NGL, PSIIS(NGL), DELTAS(NGL), B

   BB(I) = PSIIS(NGL) - B(1) ! PSI
   BB(I+1) = DELTAS(NGL) - B(2) ! DELTA

   IA(I+1) = IA(I) + MV ! PSI
   IA(I+2) = IA(I+1) + MV ! DELTA

   IF (MV.NE.0) THEN
      IF (FIRSTV) THEN ! PSI COMPLETED ALREADY
         FIRSTV = .FALSE.
         DO J=1,MV
            JJ = JJ+1
            JA(JJ) = JA(JJ-MV) ! DELTA
         END DO
      ELSE
         DO J=1,MV
            JJ = JJ+1
            JA(JJ ) = JA(JJ-MV) ! PSI
            JA(JJ+MV) = JA(JJ ) ! DELTA
         END DO
         JJ = JJ+MV
      END IF

      J1 = IA(I) 
      J2 = IA(I+1)-1
      DO J=J1,J2 ! COMPRESS
         AA(J ) = 0.0 ! PSI
         AA(J+MV) = 0.0 ! DELTA
      END DO

      DO K=1,KV ! LOCAL, VARY, NON-UNIQUE
         IV = IPTY(K) ! LOCAL, FULL
         IV2 = IV+IV ! DELTA = A * LOCAL,FULL
         IV1 = IV2-1 ! PSI = A * LOCAL,FULL
         I = IPTX(K) ! NONLOCAL, COMPRESS
         IMV = IPTW(I) ! LOCAL, COMPRESS --> JA, AA
         JJ1 = J1-I+IMV ! PSI = AA
         JJ2 = JJ1 + MV ! DELTA = AA

         AA(JJ1) = AA(JJ1) + A(IV) ! PSI'
         AA(JJ2) = AA(JJ2) + A(IV2) ! DELTA'
      END DO
   END IF ! <====================
Renormalize rows in the matrix, least square.

if (l1norm) then

bb(ii ) = bb(ii ) /psiinn(ngl) ! psi
bb(ii+1) = bb(ii+1)/deltau(ngl) ! delta

if (mv.ne.0) then ! --------------
do j=1,mv

aa(j1-1+j) = aa(j1-1+j) /psiinn(ngl) ! psi
aa(j2 +j) = aa(j2 +j)/deltau(ngl) ! delta
end do
! angle
! repeat
! ambient

if (kv.ne.0) then

do k=1,kv ! re-initialize

i = iptx(k) ! unique-ness
iptw(i) = 0 ! indicator
end do
! wave
! sample

meqns = ii-1

if (jj.ne.ia(ii)-1) then

write (iout,102) ii, jj, ia(ii)
stop
end if

if (jj.gt.nnjaaa) then

write (iout,104)
stop
end if

return

100 format ("oops, there is NO varying model parameter"
& /* for the case involving: sample = , i3
& /* where the , i3, 'th wave = , i3)
101 format (' asml, , i5, 1p2e14.4, " ngl, psi, delta'
& /* , 5x, 2e14.4)
102 format (' asml, inconsistent format of sparse matrix, '
& /* ii = , ii0
& /* jj = , ii0, ' /= ' , ii0, ' = ia(ii)-1 ')
104 format (' asml, array allocation for the sparse matrix '
& /* has been exceeded. '
& /* aa,ja <=-- nnjaaa (DEFINIT.)")
end
subroutine asmblx                        ! called by:  SEAMA
include 'iounit.'
include 'definit.'
include 'filmms.'
include 'filmss.'
include 'seamxl.'
logical firstv, firstu
real a(nrows*2), b(2), c(2)

Construct sparse matrix associated with model experiment.
Configuration:  angle, repeat, ambient, wave, sample.
____________________________________________________________
Ordering of indices in IPTU:  [vary-->  <--froz]
call arrang
mm = mlnm+mfilm                        ! model parameters

ii = 1
ia(1) = 1 ! vary
jj = 0

iaa(1) = 1 ! frozen
jjj = 0

iws = 0 ! wave,sampl
mws = 0 ! nfilm,wave,sampl
iaw = 0 ! ambient,wave,sampl
iraw = 0 ! repeat,ambient,wave,sampl
ngl = 0 ! ndeg,repeat,ambient,wave,sampl

do is=1,msampl
  mfilm = nnfilm(is)                        ! FILMSS
  mwave = nnwave(is)
  nfilm = mfilm+1
  nrow = mfilm*3+2
  do iws=mwave
    iw = iws+1
    iw = iws+1(iw)
    mbien = nmbent(iws)
    qq = waveqq(iw)
    ! FILMSS
  do ii=1,nrow
    iptx(i) = 0 ! vary ---> unique
    ipty(i) = 0 ! vary ---> full
    kptx(i) = 0 ! froz ---> unique
    kpty(i) = 0 ! froz ---> full
  end do
  iv = 0 ! local, full, non-unique
  kv = 0 ! local, vary, non-unique
  mv = 0 ! local, vary, compress
  ku = 0 ! local, froz, non-unique
mu = 0

! local, froz, compress

do m=1,mfilms
   ! films/substrate
   if (m.ne.mfilms) then
      ! films "z
      mws = mws+1
      iz = iiifilm(mws)
      zzz(m) = widths(iz)
      iv = iv+1
      j = mlmnts+iz
      ! nonlocal, full
      if (lvaryz(iz).eq.1) then ! vary
         i = iptv(j)
         kv = kv+1
         iptx(kv) = i
         ! local, non-unique
         ipty(kv) = iv
         ! local, full
         if (iptw(i).eq.0) then ! compress, unique
            mv = mv+1
            iptw(i) = mv
            jj = jj+1
            ja(jj) = i
         end if
      else ! frozen
         i = iptv(j)
         ku = ku+1
         kptx(ku) = i
         ! local " nonlocal (backwards)
         kpty(ku) = iv
         ! local, full
         if (iptw(i).eq.0) then ! compress, unique
            mu = mu+1
            iptw(i) = mu
            jja = jja+1
            ja(jja) = i
         end if
      end if
   end if
else ! nonlocal, compress (backwards)
   ! z
   do ink=1,2
      ! n+ik
      mws = mws+1
      nk = iiifilm(mws)
      c(ink) = diefcn(nk)
      ! n,k
      iv = iv+1 ! local, full, non-unique
      if (lvaryl(nk).eq.1) then ! vary
         i = iptv(nk)
         kv = kv+1
         ! local, non-unique
         iptx(kv) = i
         ! local, non-unique
         ipty(kv) = iv
         ! compress, unique
         if (iptw(i).eq.0) then ! local counter
            mv = mv+1
            iptw(i) = mv
            jj = jj+1
            ja(jj) = i
         end if
      else ! frozen
         i = iptv(nk)
         ku = ku+1
         ! local, non-unique
         kptx(ku) = i
         ! (backwards)
kpty(ku) = iv
if (iptw(i).eq.0) then ! compress, unique
  mu = mu+1 ! local counter
  iptw(i) = mu ! local
  jja = jja+1
  jaa(jja) = i ! backwards, mm+1-i=k
end if
end if
end do ! n+ik
die(m) = cmplx(c(1),c(2)) **2 ! FILMSS
end do ! mfilms

if (mv.eq.0) then
  write (iout,100) is,iw,iwave
  stop
end if

firstv = .true.
firstu = .true.
do mbn=1,mbien ! ambient
  iaws = iaws+1
  imbien = iibent(iaws)
  nnpeat = nnpeat(iaws)
  air = ambient(imbien) ! FILMSS
  do irpeat=1,nnpeat ! repeats
    iaws = iaws+1
    mangl = mangle(iaws)
    do iangl=1,mangl ! angles of incidence
      ngl = ngl+1 ! measurement data
      angl = angles(ngl)
      angu = angleu(ngl)
      call scatr(qq,angl,b,a,c)
      bb(ii ) = psiiis(ngl) - b(1) ! deviation
      bb(ii+1) = deltas(ngl) - b(2)
      cc(ii ) = c(1) ! d/d incident angle
      cc(ii+1) = c(2)
    end do
  end do
  else
    do j=1,mv
      jj = jj+1
      ja(jj) = ja(jj-mv)
    end do
  end if
  if (firstv) then
    firstv = .false.
    do j=1,mv
      jj = jj+1
    end do
  else
    do j=1,mv
      jj = jj+1
  end do

161
ja(jj) = ja(jj-mv) 
ja(jj+mv) = ja(jj) 
end do
jj = jj+mv
end if

jj1 = iaa(ii) 
j2 = iaa(ii+1)-1
do j=j1,j2      ! compress, initialize
   aa(j) = 0.0  ! psi
   aa(j+mv) = 0.0  ! delta
end do

do k=1,kv      ! local, non-unique
   iv  = ipty(k)  ! local full
   iv2 = iv+iv
   iv1 = iv2-1
   ! psi'' a'' local,full
i = iptx(k)  ! nonlocal, compress
imv = iptv(i)  ! local, compress --> ja,aa
jj1 = jj1+imv
jj2 = jj1 + mv ! delta'' aa

aa(jj1) = aa(jj1) + a(iv1)  ! psi'
aa(jj2) = aa(jj2) + a(iv2)  ! delta'
end do

end if
*************************************************************

iaa(ii+1) = iaa(ii)+mu      ! frozen
iaa(ii+2) = iaa(ii)+mu+mu
if (mu.ne.0) then
   if (firstu) then
      firstu = .false.
      do j=1,mu
         jja = jja+1
         jaa(jja) = jaa(jja-mu)
      end do
   else
      do j=1,mu
         jja = jja+1
         jaa(jja ) = jaa(jja-mu)
         jaa(jja+mu) = jaa(jja )
      end do
   end if

   j1 = iaa(ii) 
j2 = iaa(ii+1)-1
   do j=j1,j2      ! compress, initialize
      aaa(j) = 0.0
      aaa(j+mu) = 0.0
   end do
do k=1,ku
   iv = kpty(k)  ! local Jacobian
   iv2 = iv+iv
   ivl = iv2-1  ! psi
   i = kptx(k)  ! backwards
   imu = iptw(i) ! local, compress
   jj1 = j1-1+imu
   jj2 = jj1 + mu
   aas(jj1) = aas(jj1) + a(ivl)
   aas(jj2) = aas(jj2) + a(iv2)
end do
end if  ! frozen
---------------------------------------------------------------------
if (llnorm) then
   bb(ii ) = bb(ii ) /psiiiu(ngl)
   bb(ii+1) = bb(ii+1) /deltau(ngl)
   cc(ii ) = cc(ii ) /psiiiu(ngl)
   cc(ii+1) = cc(ii+1) /deltau(ngl)
   j1 = ia(ii )
   j2 = ia(ii+1)-1
   do j=j1,j2
      aa(j ) = aa(j ) /psiiiu(ngl)
      aa(j+mv) = aa(j+mv) /deltau(ngl)
   end do
if (mu.ne.0) then  ! frozen
   j1 = iaa(ii )
   j2 = iaa(ii+1)-1
   do j=j1,j2
      aas(j ) = aas(j ) /psiiiu(ngl)
      aas(j+mu) = aas(j+mu)/deltau(ngl)
   end do
end if
if (kv.ne.0) then  ! vary
   do k=1,kv
      i = iptx(k)
      iptw(i) = 0
   end do
end if
if (ku.ne.0) then  ! frozen
   do k=1,ku
      i = kptx(k)
      iptw(i) = 0
   end do
end if
end do
end if

end do ! wave
end do ! sample
meqns = ii-1

if (jj .ne. ia(ii)-1) then
  write (iout,102) ii, jj, ia(ii)
  stop
end if
if (jja .ne. iaa(ii)-1) then
  write (iout,103) ii, jja, iaa(ii)
  stop
end if
if (jj.gt.nnjaaa .or. jja.gt.nnjaaa) then
  write (iout,104)
  stop
end if

return

100 format (' asmlx, there is NO varying model parameter '
  & '/' for the case involving: sample = ', i3
  & '/' where the ', i3, '-th wave = ', i3 )
102 format (' asmlx, inconsistent format of sparse matrix, '
  & '/' 'aa " vary'
  & '/' ii = ', i10
  & '/' jj = ', i10, ' =/=', i10, ' =/= ia(ii)-1')
103 format (' asmlx, inconsistent format of sparse matrix, '
  & '/' 'aaa " froz'
  & '/' ii = ', i10
  & '/' jja= ', i10, ' =/=', i10, ' =/= iaa(ii)-1')
104 format (' asmlx, array allocation for the sparse matrix '
  & '/' has been exceeded.'
  & '/' aa,ja <---- nnjaaa (DEFNIT.)'
  & '/' aaa,jaa <---- nnjaaa (DEFNIT.)' )

end
6.2.15 SCATR.FOR

This subroutine calculates the ellipsometric angles \((\Delta, \psi)\) and the associated partial derivatives.

```fortran
subroutine scatr (qq, angl, b, a, c)
include 'defnit.'
include 'filmss.'
include 'rstack.'

real a(1), b(2), c(2)
data pi / 3.1415926 /

call forwrd (qq, angl, Rs, Rp, dRs, dRp, dRsa, dRpa) ! angl 'radians

nrow = mfilm*3+2 ! (z,n,k) (z,n,k)
sx = real (Rs)
sy = imag (Rs)
px = real (Rp)
py = imag (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
if (delta.lt.0.0) then
  delta = delta+2.0
  goto 1
else if (delta.ge.2.0) then
  delta = delta-2.0
  goto 1
end if
delta = delta*pi

b(1) = psi ! radians
b(2) = delta ! radians
k = 0
```

165
do n=1,nrow
     sx = real(dRs(n))
     sy = imag(dRs(n))
     px = real(dRp(n))
     py = imag(dRp(n))
     ssx = (sx*px+sy*py)/sr ! |Rs|'  
     ppr = (px*px+py*py)/pr ! |Rp|'  
     sdp = (sx*sy-px*py)/sr ! delta(s)'  
     pdep = (px*py-px*py)/pr ! delta(p)'  
     sdel = pdep-sdel ! delta'  
     dpsi = (sr*pprd-pr*ssrd)/(sr*sr+pr*pr) ! psi'  
     k = k+1
     a(k) = dpsi
     k = k+1
     a(k) = ddel
end do

sx = real(dRs(n))
sy = imag(dRs(n))
px = real(dRp(n))
py = imag(dRp(n))
ssx = (sx*px+sy*py)/sr ! |Rs|'  
ppr = (px*px+py*py)/pr ! |Rp|'  
ssdp = (sx*sy-px*py)/sr ! delta(s)'  
ppdep = (px*py-px*py)/pr ! delta(p)'  
ssd = pdep-sdel ! delta'  
dpsi = (sr*pprd-pr*ssrd)/(sr*sr+pr*pr) ! psi'  
c(1) = dpsi ! d(psi ) /d(angle), Jacobian  
c(2) = ddel ! d(delta) /d(angle), Jacobian  
return  
end
This subroutine calculates the reflection coefficients \((R_s, R_p)\) and the associated partial derivatives.

```
subroutine forwd (qq, angl, Rs, Rp, dRs, dRp, dRsas, dRpas)

real qq, angl ! incident angle ` radians
complex Rs, Rp
complex half, one, two, four, eta0, eta0a, cta0, cta0a
complex top, bot, ss, pp, ssz, ppz, sse, ppa, pp1, pp2, pp3

include 'defnit.' ! nfilms, nrows
include 'filmss.'

complex Rs, Rp, dRs(l), dRp(l), dRsas, dRpas
complex eta0, eta0a, cta0, cta0a
complex top, bot, ss, pp, ssz, ppz, sse, ppa, pp1, pp2, pp3

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

mfilms = nfilms + 1

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

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

Determine the reflection coefficients in air.

Method for TE: \( admittance = Y = - \frac{Hx}{Ey} \), \( Rs \) uses \( E \) field.
47 c
48 c
49 ys(mfilms) = eta(mfilms) ! substrate
50 yp(mfilms) = cta(mfilms)
51 if (mfilm.ne.0) then
52   do i=mfilm,1,-1 ! backwards
53     x = zzzz(i)*qq
54     zq(i) = cmplx (0.0, x) ! izq
55     ss = cmplx (0.0, x+x) * eta(i) ! izqn2
56     x = exp (real (ss))
57     y = aimag (ss)
58     ss = cmplx (x*cos(y), x*sin(y)) ! exp (izqn2)
59     ee(i) = ss
60     ep(i) = one-ss
61     ys(i) = eta(i)*((em(i)*eta(i)+ep(i)*ys(i+l))/
62     & (ep(i)*eta(i)+em(i)*ys(i+l)))
63     yp(i) = cta(i)*((em(i)*cta(i)+ep(i)*yp(i+l))/
64     & (ep(i)*cta(i)+em(i)*yp(i+l)))
65   end do
66 end if
67 Rs = (etaO-ys(l)) /(etaO+ys(l)) ! air
68 Rp = (ctaO-yp(l)) /(ctaO+yp(l)) ! air
69 c* Rp = -Rp ! E ------ H
70 c
71 c
72 do i=mfilms,1,-1 ! backwards
73   if (i.eq.mfilms) then ! source substrate, Y'
74     sse = half/eta(i) ! d(eta) /d(e)
75     ss = sse ! d(eta) /d(-air*sin)**2)
76     ppa = (cmplx (2.0*as2, 0.0) - die(i))
77     & * half / (eta(i)*die(i)*die(i)) ! d(cta)/d(e)
78     ppa = half / (eta(i)*die(i)) ! d(cta)/d(-as**2)
79   else ! source film
80     bot = ep(i)*eta(i) + em(i)*ys(i+1) ! denominator, TE
81     top = em(i)*eta(i) + ep(i)*ys(i+1) ! numerator
82     ss = em(i) + zq(i)*se(i)*(ys(i+1)-eta(i))
83     & + ep(i)*half*(ys(i+1)/eta(i))
84     pp = ep(i)*half - zq(i)*se(i)*(ys(i+1)-eta(i))
85     sse = (ss/bot) - (pp*top)/(bot*bot)
86     ss = (ss + ep(i)*eta(i)*ssa) /bot -
87     & (pp + em(i)*eta(i)*ssa)*top /(bot*bot)
88     ssz = two*se(i)* (eta(i)**2) * ! d/d (izq2)
89     & (ys(i+1)-eta(i)) * (ys(i+1)+eta(i)) /(bot*bot)
90     bot = ep(i)*eta(i) + em(i)*yp(i+1)*die(i) ! denominator, TM
91     top = em(i)*eta(i) + ep(i)*yp(i+1)*die(i) ! numerator
92     pp = (cmplx (2.0*as2, 0.0) - die(i))
93     & * half / (eta(i)*die(i)*die(i)) ! d(cta)/d(e)
94     ppl = pp*top/bot

168
pp2 = \( \text{em}(i) \cdot \text{half/die}(i) + \text{ep}(i) \cdot \text{cta}(i) \cdot \text{yp}(i+1) + \text{zq}(i) \cdot \text{se}(i) \cdot (\text{yp}(i+1) - \text{cta}(i)) \)

pp3 = \( \text{ep}(i) \cdot \text{half/die}(i) + \text{em}(i) \cdot \text{cta}(i) \cdot \text{yp}(i+1) - \text{zq}(i) \cdot \text{se}(i) \cdot (\text{yp}(i+1) - \text{cta}(i)) \)

ppe = \( \text{pp1} + \text{pp2} / \text{bot} - \text{pp3} / \text{top} / (\text{bot} \cdot \text{bot}) \)

pp1 = \( \text{half/} \text{top} / (\text{bot} \cdot \text{eta}(i) \cdot \text{die}(i)) \)

pp2 = \( \text{em}(i) \cdot \text{half/die}(i) + \text{ep}(i) \cdot \text{eta}(i) \cdot \text{ppa} + \text{zq}(i) \cdot \text{se}(i) \cdot (\text{yp}(i+1) - \text{cta}(i)) \)

pp3 = \( \text{ep}(i) \cdot \text{half/die}(i) + \text{em}(i) \cdot \text{eta}(i) \cdot \text{ppa} - \text{zq}(i) \cdot \text{se}(i) \cdot (\text{yp}(i+1) - \text{cta}(i)) \)

ppa = \( \text{pp1} + \text{pp2} / \text{bot} - \text{pp3} / \text{top} / (\text{bot} \cdot \text{bot}) \)

pp = \( \text{eta}(i) - \text{die}(i) \cdot \text{yp}(i+1) \) * \( \text{eta}(i) + \text{die}(i) \cdot \text{yp}(i+1) \)

ppz = \( -\text{two} \cdot \text{se}(i) \cdot \text{eta}(i) \cdot \text{cta}(i) \) * \( \text{pp} / (\text{bot} \cdot \text{bot}) \)

end if

if (i.ne.1) then
    k = i-1
    do j=k,1,-1
        sse = sse*four*se(j)* ((eta(j)**2) /
        &((ep(j)*eta(j)+em(j)*ys(j+1)**2))
        &ppe = ppe*four*se(j)* ((cta(j)**2) /
        &((ep(j)*cta(j)+em(j)*yp(j+1)**2)))
    end do
end if

sse = -sse*two*eta0 /((eta0+ys(1)**2) ! R'(e)
ppe = -ppe*two*cta0 /((cta0+yp(1)**2)

if (i.eq.mfilm) then
    sse = sse*two*eta0 /((eta0+ys(1)**2) ! R'(e)
    ppz = ppz*two*eta0 /((eta0+ys(1)**2)
end if

The format of the Jacobian vector has the form:

air / film#1 / film#2 / ... / mfilm / substrate

\([z,n,k, \ldots z,n,k, n,k]\)

k = 3*(i-1) ! position within Jacobian
if (i.eq.mfilm) then ! substrate, (z,n,k)
k = k-1
else ! film, (z,n,k)
ss = cmplx (0.0, qq*2.0) ! d(izq2)/dz
ppz = -ppz
end if
\[ d\tilde{R}(k+l) = \text{ss} \ast \text{ss} \]
\[ d\tilde{R}(k+1) = \text{pp} \ast \text{ss} \]
end if
\[ \text{ss} = \sqrt{(\text{dies}(i))} \]
\[ \text{pp} = -\text{pp}\]
\[ d\tilde{R}(k+2) = \text{ss} \ast \text{ss} \ast \text{cmplx}(2.0, 0.0) \]
\[ d\tilde{R}(k+3) = \text{ss} \ast \text{ss} \ast \text{cmplx}(2.0, 0.0) \]
\[ d\tilde{R}(k+4) = \text{pp} \ast \text{ss} \ast \text{cmplx}(0.0, 2.0) \]
\[ d\tilde{R}(k+4) = \text{pp} \ast \text{ss} \ast \text{cmplx}(0.0, 2.0) \]
end do

Since the angular partials involved:
\[ d\tilde{R}/d(-\text{air}\ast \text{sin})^{2} \]
and we want partials wrt angle, i.e.:

\[ \text{ss} = \text{cplx}(-\text{air} \ast \text{air} \ast \text{sin}(\text{angl} + \text{angl}), 0.0) \]
\[ \text{ss} = \text{ss} \ast \text{ss} \]
\[ \text{pp} = -\text{pp} \]
\[ \text{ss} = \text{two} \ast (\text{eta0} \ast \text{ys}(1) - \text{eta0} \ast \text{ss}) / ((\text{eta0} + \text{ys}(1))^{2}) \]
\[ \text{pp} = \text{two} \ast (\text{eta0} \ast \text{yp}(1) - \text{eta0} \ast \text{pp}) / ((\text{eta0} + \text{yp}(1))^{2}) \]

Since the TM calculation utilizes H fields, i.e.,
\[ \text{Rp} = \text{H(reflected)}/\text{H(incident)}, \]
and that convention uses E fields, i.e.,
\[ \text{Rp} = \text{E(x, reflected)} / \text{E(x, incident)}, \]
one may induce the minus sign (-) onto Rp above.

return

end
subroutine stat22 (n,b)
real b(n)
icomment 'iounit,'
logcal llzero
data pi / 3.14159265359/

Perform: simple statistics of deviations from the model.
Discern: (mean, standard deviation) of deviations.
Assume: b(1) - d(psi) deviation experiment-model
        b(2) - d(delta) deviation experiment-model

raddeg = 180.0 /pi ! degrees <--- radians
nh = n/2 ! psi, delta
h = float (nh)

write (iout,111)
if (n.le.1 .or. n.ne.nh+nh) then
    write (iout,112) n
    stop
end if

al = 0.0
a2 = 0.0

do i=1,n,2 ! mean deviation
    al = al + b(i ) ! psi
    a2 = a2 + b(i+1) ! delta
end do

al = al /h ! psi
a2 = a2 /h ! delta

llzero = .false. ! zero variance

s1 = 0.0
s2 = 0.0
s3 = 0.0

do i=1,n,2 ! variance of deviations
    s1 = s1 + (b(i ) -al)**2 ! <psi  psi>
    s2 = s2 + (b(i+1)-a2)**2 ! <delta|delta>
    s3 = s3 + (b(i+1)-a2)*(b(i)-al) ! <delta| psi>
end do

s1 = sqrt (s1 /h) ! standard deviation
s2 = sqrt (s2 /h)
s3 = s3 /h ! covariance

if (s1.eq.0.0 .or. s2.eq.0.0) then ! exact fit " no scatter
    llzero = .true.
else
    s3 = s3 /(s1*s2) ! correlation coefficient
end if

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, 16('----'))
& / Statistics of deviations, experiment-model, g,
& / where: g = column array of length 2M,
& / mean (g) = m(g) = <g> = (1/M) sum: g(i),
& / variance (g) = <[g(i) - m(g)]^2 >,
& / covariance = <[g(1) - m(1)][g(2) - m(2)] >,
& / std dev = sqrt (variance),
& / correlation coef = covariance / [std dev (psi) * std dev (delta)]

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, 'corr coef': 4x, f10.3, 6x, f10.3, 1x,
& / 'correlation coefficient <psi|delta>'

114 format ( 26x, 'UNnormalized, because at least one of the',
& / 26x, 'standard deviations vanish.' )

end
6.2.18  CORLAT.FOR

1 subroutine corlat
2 include 'iounit.'
3 include 'defnit.'
4 include 'filmm.'
5 include 'wstack.'
6 include 'cgnxx1.'
7
dimension det(2), inert(3)  ! LINPACK
8
9 raddeg = 180.0 / 3.14159265
10
12 c Discern variance of model parameters, normalized, A(T)*A.
13
llnorm = .false.
14 call asml
15 call stat22 (meqns, bb)  ! mean, std dev.
17 call pltdata (2)  ! plot deviations of fit
18
kv = 0
19 do jv=1,mvare
20   do iv=1,jv
21     sum1 = 0.0
22     sum2 = 0.0
24     do i=1,meqns,2
25       j1 = ia(i )
26       j2 = ia(i+1)-1
27       mv = j2-j1+1
28       s1 = 0.0
29       s2 = 0.0
30       s3 = 0.0
31       s4 = 0.0
32     do j=j1,j2  ! within a row of A
33       jaj = ja(j)
34       if (jaj.eq.iv) then
35         s1 = aa(j )
36         s3 = aa(j+mv)
37       end if
38       if (jaj.eq.jv) then
39         s2 = aa(j )
40         s4 = aa(j+mv)
41       end if
42     end do
43     sum1 = sum1 + s1*s2  ! psi * dot product
44     sum2 = sum2 + s3*s4  ! delta
45   end do
46   sum = sum1+sum2
47   sum = sum /float (meqns)  ! normalization
48   kv = kv+1  ! packed format * storage
49   aat (kv) = sum  ! upper triangle + diagonal
50   if (jv.eq.iv) then  ! diagonal
51     kv = kv+1
52      aat (kv) = sum  ! upper triangle + diagonal
53   end if
\[ xx(jv) = \sqrt{\text{sum}} \]

\[ \text{end if} \]
\[ \text{end do} \]
\[ kv = 0 \]
\[ \text{do } jv=1,mv \text{ vary} \quad ! \text{ renormalize} \]
\[ \quad \text{do } iv=1,jv \]
\[ \quad kv = kv+1 \]
\[ \quad \text{aat(kv)} = \text{aat(kv)} / (xx(jv) * xx(iv)) \]
\[ \text{end do} \]
\[ \text{end do} \]
\[ \text{write (iout,111)} \]
\[ k2 = 0 \]
\[ \text{do } i=1,mv \text{ vary} \]
\[ \quad k1 = k2+1 \]
\[ \quad k2 = k2+i \]
\[ \quad \text{write (iout,114)} i, \text{(aat(k),k=k1,k2)} \]
\[ \text{end do} \]
\[ \text{write (iout,120)} \]
\[ \text{write (iout,121)} (xx(i), i=1,mv \text{ vary}) \quad ! \text{ normalization coefficients} \]
\[ \text{LINPACK, Chapter 5, Solving symmetric indefinite matrices.} \]
\[ \text{call sspco (aat, mv, ipvt, rcond, w)} \quad ! \text{UD*T(U) decomposition} \]
\[ \text{write (iout,116)} \text{ rcond} \]
\[ \text{if (.true.) return} \quad ! \text{------------------------------------------} \]
\[ \text{c* if (1.0 .eq. 1.0+rcond  ) then} \quad ! \text{singular matrix} \]
\[ \text{if (1.0 .eq. 1.0+rcond*0.01) then} \quad ! \text{ill-conditioning} \]
\[ \quad \text{call sspdi (aat, mv, ipvt, det, inert, w, 111)} \]
\[ \quad \text{write (iout,117)} \text{ det, inert} \]
\[ \quad k2 = 0 \]
\[ \quad \text{do } i=1,mv \]
\[ \quad \quad k1 = k2+1 \]
\[ \quad \quad k2 = k2+i \]
\[ \quad \quad \text{write (iout,112)} i, \text{(aat(k), k=k1,k2)} \]
\[ \text{end do} \]
\[ \text{end if} \]
\[ \text{c} \]
\[ \text{Alternate representation of correlation, normalized.} \]
\[ \text{do } jv=1,mv \quad ! \text{mean} \quad <A(:,j)> \]
\[ \quad \text{sum = 0.0} \]
\[ \quad \text{do } i=1,n \text{ eqs,2} \]
\[ \quad \quad j1 = ia(i) \]
\[ \quad \quad j2 = ia(i+1)-1 \]
\[ \quad \quad mv = j2-j1+1 \]
\[ \quad \quad \text{do } j=j1,j2 \quad ! \text{row of A} \]
\[ \quad \quad \quad jaj = ja(j) \quad ! \text{column} \]
\[ \quad \quad \quad \text{if (jaj.eq.jv) then} \]
\[ \quad \quad \quad \quad \text{sum = sum + aa(j)+aa(j+mv)} \]
end if
end do
end do  ! meqns
sum = sum/float (meqns)
v(jv) = sum
end do  ! mvary

kv = 0
!

do jv=1,mvary
  do iv=1,jv
    sum = 0.0
    do i=1,meqns,2
      j1 = ia(i )
      j2 = ia(i+1)-1
      mv = j2-j1+1
      s1 = 0.0
      s2 = 0.0
      s3 = 0.0
      s4 = 0.0
      do j=1,j2
        jaj = ja(j)
        if (jaj.eq.jv) then
          s1 = aa(j )-v(jv)
          s3 = aa(j+mv)-v(jv)
        end if
        if (jaj.eq.iv) then
          s2 = aa(j )-v(iv)
          s4 = aa(j+mv)-v(iv)
        end if
      end do
      sum = sum + s1*s2 + s3*s4
    end do  ! meqns
    sum = sum/float (meqns-l)
    kv = kv+1
    aat(kv) = sum
    if (iv.eq.jv) then
      xx(jv) = sqrt (sum)
    end if
  end do  ! mvary

end do

kv = 0
!

do jv=1,mvary
  do iv=1,jv
    kv = kv+1
    if (xx(iv).eq.0.0 .or. xx(jv).eq.0.0) then
      aat(kv) = -2.0
    else
      aat(kv) = aat(kv) / (xx(iv)*xx(jv))
    end if
  end do  ! correlation
end do

write (iout,110)  ! Print out results
\texttt{k2 = 0}
\texttt{do i=1,mvary}
\texttt{\hspace{1em} k1 = k2+1}
\texttt{\hspace{1em} k2 = k2+i}
\texttt{\hspace{1em} write (iout,114) i, (aat(k),k=k1,k2)}
\texttt{end do}
\texttt{write (iout,120)}
\texttt{write (iout,121) (xx(i), i=1,mvary) ! normalization coefficients}
\texttt{return}
\texttt{111 format (' J(T)*J: (renormalized for correlation)' )}
\texttt{112 format (ix, i4, '), 1x, 1p10e10.2, : /(7x, 10e10.2))}
\texttt{114 format (ix, i4, '), 1x, 10f10.5, : /(7x, 10f10.5))}
\texttt{116 format (' rcond=', 1pe12.3, ', condition number')}
\texttt{117 format (' J(T)*J: (renormalized for correlation)' )}
\texttt{& '/ Determinant: ', f8.4, ' E ', f8.4}
\texttt{& '/ Inertia: (' , 3i4,}
\texttt{& '/ number of (+,-,0) eigenvalues'}
\texttt{& '/ Inverse: upper+diagonal matrix')}
\texttt{118 format (' J(T)*J: ' (x-<x>)*(y-<y>)' )}
\texttt{120 format (' Normalization coefficients: sqrt [J(T)*J](i,i)')}
\texttt{121 format ( ix, 1p10e10.2)}
\texttt{end}
subroutine seama
include 'icunit.'
include 'definit.'
include 'filmm.'
include 'seamx1.'
include 'wstack.'
include 'cgnxxl.'
real det(2), ss(4)
integer inert(3)
data pi / 3.14159268 /

Sensitivity analysis for multiple: angle,ambient,wave,sample.
Ordering of indices in IPTU: [vary--> <--froz]
The matrix equation is:  g = AV*v + AU*u + AT*t
where:  v " vary,
u " frozen,
t " d(\phi) " angle of incidence
 g " deviations " experiment - model

raddeg = pi/180.0  ! radians <-- degrees
mm = mlmnts+mfilmn  ! model parameters
llnorm = .false.
call asmblx

c, Renormalize the columns of AA, and retain factors in AATS.
call scalj(j,meqns,mvary,ia,ja,aa,xx,aats,w,2) ! aa,aats

c, Formulate the "Normal" equations.
c, Note that:  \( J(v,T)J(v) \) " symmetric matrix.
c, Construct: upper triangle, \( i \leq j \).
do jv=1,mvary
do iv=1,jv  ! upper triangle
  s = 0.0
do i=1,meqns,2
  j1 = ia(i )
j2 = ia(i+1)-1
  mv = j2-j1+1
  s1 = 0.0
  s2 = 0.0
  s3 = 0.0
  s4 = 0.0
do j=j1,j2
    jaj = ja(j)  ! column
    if (jaj.eq.iv) then
      s1 = aa(j )  ! psi
s3 = aa(j+mv)  ! delta
end if
if (jaj.eq.jv) then
s2 = aa(j)  ! psi
s4 = aa(j+mv)  ! delta
end if
end do
s = s + s1*s2 + s3*s4
end do
ij = iindex (3,mvarya, iv,jv)  ! column-wise
ant(ij) = s  ! upper triangle
end do  ! rows
end do  ! columns

gggpsi = 0.0  ! maximum magnitude
ggdel = 0.0
ggang = 0.0
do ii=1,meqns,2
ngl = (ii+1)/2
gggpsi = amaxi (gggpsi, abs ( bb(ii )))
ggdel = amaxi (ggd del, abs ( bb(ii+1)))
ggang = amaxi (ggang, abs (angleu(ngl)))  ! radians
end do
gg psi = 0.0  ! estimate variances
ggd el = 0.0
ggang = 0.0
do ii=1,meqns,2  ! rescale
ngl = (ii+1)/2  ! index uncertainties
gg psi = gggpsi + ( bb(ii )/gggpsi)**2  ! psi deviations
ggd el = ggd el + ( bb(ii+1)/ggd del)**2  ! delta deviations
ggang = ggang + (angleu(ngl)/ggang)**2  ! angle radians
end do
gg psi = gggpsi*gggpsi**2
ggd el = ggd el*ggd del**2
ggang = ggang*ggang**2
ggvar = (gggpsi+ggd el) /float (meqns-mvarya)! <gg> variance
ggang = ggang /float (meqns/2)! <aa>
ggvars = sqrt (ggvar)! standard deviation
ggangs = sqrt (ggang)
write (iout,111)
write (iout,126) ggvar,ggvars, ggang,ggangs
write (iout,112)
write (iout,121)
c write (iout,112)
call sspco (aat,mvarya,ipvt,rcond, xx)  ! UD*U(T)
c if (1.0 .eq. 1.0+rcond ) then
if (1.0 .eq. 1.0+rcond*0.01) then  ! ill-conditioned
write (iout,101)
return
end if

call sspdi (aat,mvary,ipvt,det,inert,xx,1) ! inverse

cc1xxx = 0.0          ! psi, max magnitude
cc2xxx = 0.0          ! delta, max magnitude

do ii=1,meqns,2
    cc1xxx = amax1 (cc1xxx, abs (cc(ii )))
    cc2xxx = amax1 (cc2xxx, abs (cc(ii+1)))
end do

do jv=1,mvary
    bbb1xx = 0.0          ! column
    bbb2xx = 0.0          ! [A(T)*A]**-1*A(T)
    do ii=1,meqns,2
        j1 = ia(ii )
        j2 = ia(ii+1)-1
        mv = j2-j1+1
        s1 = 0.0
        s2 = 0.0
        do jj=j1,j2
            jaj = ja(jj)
            k = iindex (3,mvary, jv,jaj)
            s1 = s1 + aat(k)*aa(jj ) ! psi
            s2 = s2 + aat(k)*aa(jj+mv) ! delta
        end do
        bbb(ii ) = s1 ! psi
        bbb(ii+1) = s2 ! delta
    bbb1xx = amax1 (bb1xx, abs (s1)) ! maximum
    bbb2xx = amax1 (bb2xx, abs (s2))
end do

write (iout,112)

do iv=1,jv
    bbb1xx = 0.0 ! row
    bbb2xx = 0.0
    do ii=1,meqns,2
        j1 = ia(ii )
        j2 = ia(ii+1)-1
        mv = j2-j1+1
        s1 = 0.0
        s2 = 0.0
        do jj=j1,j2
            jaj = ja(jj)
            k = iindex (3,mvary, iv,jaj)
            s1 = s1 + aat(k)*aa(jj ) ! psi
            s2 = s2 + aat(k)*aa(jj+mv) ! delta
        end do
        bb(ii ) = s1 ! psi  Note:
        bb(ii+1) = s2 ! delta overwriting
    bbb1xx = amax1 (bb1xx, abs (s1))
    bbb2xx = amax1 (bb2xx, abs (s2))
do j=1,4
   ss(j) = 0.0
end do
do ii=1,meqns,2
   ss(1) = ss(1) + (bbb(ii )/bbb1xx)
   &   ( bb(ii )/bbb1xxx)
   ss(2) = ss(2) + (bbb(ii+1)/bbb2xx)
   &   ( bb(ii+1)/bb2xxx)
   ss(3) = ss(3) + (bbb(ii )/bbb1xx)
   &   ( bb(ii )/bb1xxx)
   &   ( cc(ii )/cc1xxx)**2
   ss(4) = ss(4) + (bbb(ii+1)/bbb2xx)
   &   ( bb(ii+1)/bb2xxx)
   &   ( cc(ii+1)/cc2xxx)**2
end do

ss(1) = ss(1)*bbb1xx*bb1xxx
ss(2) = ss(2)*bbb2xx*bb2xxx
ss(3) = ss(3)*bb1xx*bb1xxx* cc1xxx**2
ss(4) = ss(4)*bbb2xx*bb2xxx* cc2xxx**2

ss(1) = (ss(1) + ss(2))*ggvar
   &   EggB
ss(3) = (ss(3) + ss(4))*ggang
   &   BJaaJB

ss(1) = ss(1)/(aats(jv)*aats(iv))
   &   scale
ss(3) = ss(3)/(aats(jv)*aats(iv))

ss(2) = sqrt(abs(ss(1)))
   &   " std dev
ss(4) = sqrt(abs(ss(3)))

write (iout,131) jv,iv, ss(1),ss(3), ss(2),ss(4)
end do  ! iv

random = sqrt(ss(1) + ss(3))
   &   diagonal, (j,j)
system = 0.0
if (mfroz.ne.0) then
   &   systematic errors
   do k=1,mfroz
      xxx(k) = 0.0
   &   initialize
   end do
   do ii=1,meqns,2
      j1 = iaa(ii )
      j2 = iaa(ii+1)-1
      if (j1.le.j2) then
         mu = j2-j1+1
         do j=j1,j2
            km = jaa(j)
            k = mm+1-km
            xxx(k) = xxx(k)
            &   + bbb(ii )*aaa(j )
         &   + bbb(ii+1)*aaa(j+mu)
      end do
end do
end if  ! row
end do  ! meqns

do k=1,mfroz
  km = mm+i-k  ! backwards
  m = iptu(km)
  if (m.le.mlmlnts) then
    uncert = uncerl(m)
  else
    m = m-mlmlnts
    uncert = uncerz(m)
  end if
  xxx(k) = abs (xxx(k)*uncert)  ! |BJu|
  xxx(k) = xxx(k)/aats(jv)  ! scale
  system = system + xxx(k)
end do
end if  ! frozen

Output results ------------------------
write (iout,113)
total = random + system  ! diagonal
m = iptu(jv)  ! unique,nonlocal,full
if (m.le.mlmlnts) then
  s1 = diefcn(m)
  s2 = uncerl(m)
  write (iout,122) jv, total, s2, s1, m, '(n+ik)'
  write (iout,123) random, ss(2), ss(4)
else
  m = m-mlmlnts
  s1 = widths(m)
  s2 = uncerz(m)
  write (iout,122) jv, total, s2, s1, m, '(z )'
  write (iout,123) random, ss(2), ss(4)
end if

if (mfroz.ne.0) then  ! systematic errors
write (iout,124) system
write (iout,110)
do k=1,mfroz
  km = mm+i-k
  m = iptu(km)
  if (m.le.mlmlnts) then
    write (iout,125) k, xxx(k), m, '(n+ik)'
  else
    m = m-mlmlnts
    write (iout,125) k, xxx(k), m, '(z )'
  end if
end do
end if
end do  ! vary
writs

return

100 format ('/seama, insufficient allocation for array, AAT')
101 format ('/seama, singular or ill-conditioned matrix ')
102 format ('/seama, uncertainties')

110 format ('')
111 format (1x, 17('=='))
112 format (1x, 17('-----'))
113 format (13x, 14(' - - '))

121 format ('Discern: Uncertainty in model parameters ')
& /* where: |v| = sqrt(BB(T)<gg> + (BJ)(BJ)(T)<aa>)'
& /* + |BJ||u|,
& // B = [J(T)*J]**-1 *J(T), (nonsquare J)'
& // vary', 6x, 'total ', 4x, 'initial', 4x, 'parameter')

122 format (3x, i4, ''), 2x, 1p2e11.2, e15.6,
& ', for:', i4, '', '', a)
123 format (10x, 1p3e11.2, 4x, 'random, |Bg|, |BJa|')
124 format (21x, 1p1e11.2, 4x, 'systematic ' total ')
125 format (10x, 6x, i4, ')
& 1p1e11.2, 4x, 'systematic '' |BJu| '',i4,'',',a)
126 format (' <gg> '', 1p2e12.3, 4x, '(variance, std dev)'
& /* <aa> '', 2e12.3, 4x, '(variance, std dev)')
127 format (10x, 2i5, 1p2e11.2, 4x, '(j,i), |BgB|, |BJaJB|'
& / 20x, 2e11.2, 4x, '' |Bg|, |BJa| ')
Note: This routine should be compared to: SEAMA

Here: Methods 2,3 are not available.

There are two estimates of errors, but NO use is made of that retained in: xw

subroutine seamax
include 'iounit.'
include 'definit.'
include 'filmm.'
include 'seamx1.'
include 'seamx2.'
include 'cgmx1.'
include 'wstack.'

logical square, lsout
integer inert(3), ks(mrowss*2+nrowss)
real det(2), ss(4), sss(4,2), sk(mrowss*2+nrowss)

data pi / 3.14159265 /

Sensitivity/error analysis for multiple: angle, ambient, wave, sample.
Ordering of indices in IPTU: [vary->] <-froz]
call arran
raddeg = pi/180.0 ! radians <- degrees
mm = mlmnts+mfilmm ! model parameters
call seam2 ! construct matrix table
lsout = maraws.le.2 ! single/double angle
if (lsout) then ! output header cards
    open (unit=lsout, file='x.sout', status='unknown')
    write (lsout,104) msampl, mvary, ndegr
    write (lsout,104) (nnwave(is), is=l,msampl)
    iws = 0
    iaws = 0
    iraws = 0
    do is=l,msampl
        mwave = nnwave(is)
        do iw=1,mwave
            iws = iws+1
            mbien = nnbent(iws)
            write (lsout,104) mbien
            do mbn=1,mbien
                iaws = iaws+1
                mrrepeat = nnrepeat(iaws)
                do irrepeat=1,mrrepeat
                    iraws = iraws+1
                    istep = isteps(iraws)
mangle = mangle(iras)
write (isout,104) istep, mangl
end do ! repeat
end do ! ambient
end do ! wave
end do ! sample
end if

! initialize

call poplat (xx, 1)

!

===================================================================

Provide: do-loop nesting to the necessary depth.
Utilize: goto, if, and three vectors.
(angle | ambient, wave, sample), i.e., without repeats.
For each distinct set of (ambient, wave, sample),
scan a set of multiple angles,
where each set of multiple angles are of the form:
do i(1)=1+(manglm-1)*step, ndegr , step

do i(2)=1+(manglm-2)*step, i(1)-step, step

do i(3)=1+(manglm-3)*step, i(2)-step, step
...
do i(manglm)=1, i(manglm-1)-step, step

!

kt = 0 ! index counter
kts = 0 ! index singular events

!

j2 = 0 ! (angle, ambient, wave, sample)
k = 0 ! ( ambient, wave, sample)

!

1 k = k+1
if (k.gt.mraws) goto 4 ! work
mangl = mangle(k) ! multiple angle
istep = isteps(k) ! do-loop increment
j1 = j2+1
j2 = j2+mangl
iiil(j1) = ndegr ! limit outer do-variable

!

iiii(j) = iiii(j) - istep ! initialize do-variable

!

end do

!

j = j1-1 ! index of: nested do-variables.
2 j = j+1 ! index of: j-th nested do.
if (j.gt.j2) goto 1

!

3 iiii(j) = iiii(j) + istep ! update do-loop variable

!

if (iiii(j) .le. iiii(j)) then ! test upper limit
if (j.ne.j2) iiii(j+1)=iiii(j)-istep

go to 2
end if

!

iiii(j) = iiii(j) - istep ! reset inner do
j = j-1 ! backup one do-level
if (j.ge.j1) goto 3

!

if (k.eq.1) goto 6 ! escape do-loop nest
k = k-1 ! backup
mangl = mangle(k)
104 istep = isteps(k)
105 j2 = j1-1
106 j1 = j1-mang1
107 goto 3
108 4 continue

! inner-most nested do.

110 j1save = j1
111 j2save = j2
112 if (j.ne.maraws+1) then
113 write (iout,107) maraws, j
114 stop
115 end if
116
117 write (iout,102) kt, (iii(i), i=1,maraws)
118 c if (.true.) goto 5
119 c---------------------------------------------------------------
120 c The matrix equation is:
121 c \[ \mathbf{b} = \mathbf{A}^t \mathbf{v} + \mathbf{A} \mathbf{u} + \mathbf{A} \mathbf{T} \mathbf{t} \]
122 c where:
123 c \[ \mathbf{v} \] vary,
124 c \[ \mathbf{u} \] frozen,
125 c \[ \mathbf{t} \] d(phi ' angle of incidence)
126 c \[ \mathbf{b} \] deviations ' experimental uncertainty
127 c The method of solution for \(|\mathbf{v}|\),
128 c depends on whether the matrix \([\mathbf{A}]\) is square.
129
c call seam3 ! model experiment, table
130 square = meqns.eq.mvary
131
132 if (.not.square) then ! renormalize columns
133 call scaljj (meqns,mvary, ia,ja,aa,xx,aats,w,2) ! aa,aats
134 end if
135
136 if (square) then ! non-symmetric matrix
137 mvaryy = mvary*mvary ! full, square
138 if (mvaryy .gt. naat) then ! WSTACK, allocation
139 write (iout,100)
140 stop
141 end if
142 do i=1,mvaryy ! square matrix
143 aat(i) = 0.0 ! initialize
144 end do
145 do i=1,meqns,2 ! LINPACK format
146 j1 = ia(i )
147 j2 = ia(i+1)-1
148 mv = j2-j1+1
149 do j=j1,j2 ! row
150 jaj = ja(j) ! column
151 kpsi = iindex (6,mvary, i ,jaj) ! stored column-wise
152 kdel = iindex (6,mvary, i+1,jaj)
153 aat(kpsi) = aa(j ) ! psi
154 aat(kdel) = aa(j+mv) ! delta
155 end do
156 end do
call sgeco (aat,mvary,mvary,ipvt,rcond, xx)  ! LU decomposition
if (1.0 .eq. 1.0+rcond ) then  ! singular matrix
  kts = kts+1
  do i=i,mvary
    xx(i) = -1.0  ! truncate
  end do
else
  call sgedi (aat,mvary,mvary,ipvt,det, xx,1)  ! inverse
  do i=i,mvary  ! row
    do j=1,2
      do kj=1,4
        sss(kj,j) = 0.0  ! initialize
      end do
    end do
    do j=1,meqns,2  ! (psi,delta)
      uang = 0.002*raddeg  ! radians, SEAM2
      kpsi = iindex (5,mvary, i,j )
      kdel = iindex (5,mvary, i,j+1)
      ss(1) = aat(kpsi)*bb(j )  ! psi
      ss(2) = aat(kdel)*bb(j+1)  ! delta
      ss(3) = aat(kpsi)*cc(j )*uang  ! psi'
      ss(4) = aat(kdel)*cc(j+1)*uang  ! delta'
      do kj=1,4
        sss(kj,1) = sss(kj,1) + ss(kj)
        sss(kj,2) = sss(kj,2) + ss(kj)**2
      end do
    ss(1) = abs (sss(1,1) + sss(2,1))  ! |Dg |
    &  + abs (sss(3,1) + sss(4,1))  ! |Djt|
    ss(2) = sqrt (sss(1,2) + sss(2,2))  ! rms(Dg )
    &  + sqrt (sss(3,2) + sss(4,2))  ! rms(Djt)
  end do
if (mfroz.ne.0) then  ! frozen
  do k=1,mfroz
    xxx(k) = 0.0  ! initialize
  end do
  do ii=1,meqns,2  ! even
    j1 = iaa(ii )
    j2 = iaa(ii+1)-1
    if (j1.le.j2) then
      mu = j2-j1+1
      do j=j1,j2  ! row
        km = jaa(j)  ! backwards
        k = mm+1-km  ! column
        kpsi = iindex (5,mvary, i,ii )
        kdel = iindex (5,mvary, i,ii+1)
        xxx(k) = xxx(k)
        + aat(kpsi)*aaa(j )
        + aat(kdel)*aaa(j+mu)
      end do
    end if
  end do

end do
end if

do k=1,mfroz
  km = mm+1-k
  m = iptu(km)
  if (m.le.mlmts) then
    uncert = uncer1(m)
  else
    m = mlmts-m
    uncert = uncerz(m)
  end if
  uncert = uncert * abs(xxx(k)) ! |DJ|/u|
  ss(1) = ss(1) + uncert
  ss(2) = ss(2) + uncert
end do

end if ! frozen

xx(i) = ss(1)
xv(i) = ss(2)
end do ! vary

end if

if (method.eq.1) then ! "Normal" equations
  do jv=1,mvary
    do iv=1,jv
      s = 0.0
      do i=1,meqns,2
        j1 = ia(i )
j2 = ia(i+1)-1
        mv = j2-j1+1
        s1 = 0.0
        s2 = 0.0
        s3 = 0.0
        s4 = 0.0
        do j=j1,j2
          jaj = ja(j) ! column
          if (jaj.eq.iv) then
            s1 = aa(j )
s3 = aa(j+mv)
          end if
          if (jaj.eq.jv) then
            s2 = aa(j )
s4 = aa(j+mv)
          end if
        end do
        s = s + s1*s2 + s3*s4
      end do
      ij = iindex (3,mvary, iv,jv) ! column-wise
      aat(ij) = s ! upper triangle
    end do
  end do
  ! square matrix
  --------------------------------------------------------
end do

call sspco (aat,mvary,ipvt,rcond, xx) ! UD*U(T)

if (1.0 .eq. 1.0+rcond*0.01) then ! ill-conditioned
write (iout,101) (iiii(i), i=1,maraws)
kts = kts+1
do i=1,mvary
xx(i) = -1.0 ! truncate
end do
else

call sspdi (aat,mvary,ipvt,det,inert,xx,1) ! inverse
do i=1,mvary

   do ii=1,meqns,2 ! [A(T)*A]**-1 *A(T)
      j1 = ia(ii )
      j2 = ia(ii+1)-1
      mv = j2-j1+1
      s1 = 0.0
      s2 = 0.0
      do jj=j1,j2 ! row
         jaj = ja(jj)
         k = iindex (3,mvary, i,jaj)
         s1 = s1 + aat(k)*aa(jj ) ! psi
         s2 = s2 + aat(k)*aa(jj+mv) ! delta
      end do
      bbb(ii ) = s1 ! psi
      bbb(ii+1) = s2 ! delta
   end do
   do jj=1,2
      do kj=1,4
         sss(kj,jj) = 0.0 ! initialize
      end do
   end do
   do ii=1,meqns,2
      uang = 0.002+raddeg ! radians, SEAM2
      ss(1) = bbb(ii )*bb(ii )
      ss(2) = bbb(ii+1)*bb(ii+1)
      ss(3) = bbb(ii )*cc(ii )*uang
      ss(4) = bbb(ii+1)*cc(ii+1)*uang
      do kj=1,4
         sss(kj,1) = sss(kj,1) + ss(kj)
         sss(kj,2) = sss(kj,2) + ss(kj)**2
      end do
   end do
   ss(1) = abs (sss(1,1) + sss(2,1)) ! |Dg |
   & + abs (sss(3,1) + sss(4,1)) ! |DJt|
   ss(2) = sqrt (sss(1,2) + sss(2,2)) ! rms(Dg )
   & + sqrt (sss(3,2) + sss(4,2)) ! rms(DJt)
   if (mfroz.ne.0) then
      do k=1,mfroz
         xxx(k) = 0.0
      end do
   do ii=1,meqns,2
j1 = iaa(ii)
j2 = iaa(ii+1)-i
if (j1.le.j2) then
  mu = j2-j1+1
  do j=j1,j2
    km = jaa(j)
    k = mm+1-km
    xxx(k) = xxx(k)
    + bbb(ii)*aaa(j)
    + bbb(ii+1)*aaa(j+mu)
  end do
end if
end do

do k=1,mfroz
  km = mm+1-k
  m = iptu(km)
  if (m.le.mlmnts) then
    uncert = uncert(m)
  else
    m = m-mlmnts
    uncert = uncert(z)
  end if
  uncert = uncert * abs(xxx(k))  ! |DJ|/|u|
  ss(1) = ss(1) + uncert
  ss(2) = ss(2) + uncert
  end do
end if  ! frozen

xx(i) = ss(1)
xw(i) = ss(2)
end do  ! vary

if (method.eq.2) then  ! Brute force, forward problem
  if (.true.) stop  ! Note: xw ' not available
  do i=1,mvary
    xx(i) = 0.0  ! stored uncertainties
  end do
end if

Simulate do-loops to scan possible sign-flips.
There is a sign-flip for each component in: b,t,u.
For the case of systematic error in the angle of incidence,
i.e., d(phi) ' t, then 't' is scalar, not vector.

Simulate:  do i1=1,2
do i2=1,2
do i3=1,2
...
do i(meqns+1+mfroz)
This involves: \(2^{\text{meqns}+\text{mfroz}}\) sign permutations.

Unfortunately, this becomes unwieldy rather quickly.

\[
\text{ns} = \text{meqns}+\text{mfroz} \quad \text{! depth of do-nest}
\]
\[
\text{do } i=1,\text{ns}
\]
\[
\text{ks}(i) = 0 \quad \text{! } i=1, \text{ do-variable, initialize}
\]
\[
\text{end do}
\]
\[
\text{ms} = 0
\]
\[
\text{ms} = \text{ms}+1
\]
\[
\text{if } (\text{ms} \geq \text{ns}) \text{ goto 23}
\]
\[
\text{ks}(\text{ms}) = \text{ks}(\text{ms}) + 1 \quad \text{! update do-variable}
\]
\[
\text{if } (\text{ks}(\text{ms}) \leq 2) \text{ then}
\]
\[
\text{ks}(\text{ms}) = 1.0
\]
\[
\text{else}
\]
\[
\text{ks}(\text{ms}) = -1.0
\]
\[
\text{end if}
\]
\[
\text{goto 21}
\]
\[
\text{end if}
\]
\[
\text{ks}(\text{ms}) = 0 \quad \text{! reset inner do-loop}
\]
\[
\text{ms} = \text{ms}+1 \quad \text{! backup one level of do-s}
\]
\[
\text{if } (\text{ms} \geq 0) \text{ goto 24}
\]
\[
\text{goto 22}
\]
\[
\text{ms} = \text{ms}+1 \quad \text{! level of deepest do}
\]
\[
\text{-----------------------------------}
\]
\[
\text{uangsk} = \text{uang} \star \text{sk(meqns+1)} \quad \text{! systematic error}
\]
\[
\text{do } i=1,\text{meqns},2 \quad \text{! form 'b' vector}
\]
\[
\text{bbb}(i ) = \text{bb}(i )\star\text{sk}(i) - \text{cc}(i )\star\text{uangsk} \quad \text{! psi}
\]
\[
\text{bbb}(i+1) = \text{bb}(i+1)\star\text{sk}(i+1) - \text{cc}(i+1)\star\text{uangsk} \quad \text{! delta}
\]
\[
\text{j1} = \text{iia}(i ) \quad \text{! frozen}
\]
\[
\text{j2} = \text{iia}(i+1)-1
\]
\[
\text{if } (\text{j1} \leq \text{j2}) \text{ then}
\]
\[
\text{mu} = \text{j2}-\text{j1}+1
\]
\[
\text{s1} = 0.0
\]
\[
\text{s2} = 0.0
\]
\[
\text{do } j=\text{j1},\text{j2}
\]
\[
\text{km} = \text{iia}(j) \quad \text{! backwards}
\]
\[
\text{k} = \text{mm}+1-\text{km}
\]
\[
\text{m} = \text{iptu}(\text{km})
\]
\[
\text{if } (\text{m} \leq \text{mlmnts}) \text{ then}
\]
\[
\text{uncert} = \text{uncerl}(\text{m})
\]
\[
\text{else}
\]
\[
\text{m} = \text{m}+\text{mlmnts}
\]
\[
\text{uncert} = \text{uncerz}(\text{m})
\]
\[
\text{end if}
\]
\[
\text{uncert} = \text{uncert} \star \text{sk(meqns+1-k)}
\]
\[
\text{s1} = \text{s1} + \text{aaa}(j ) \star \text{uncert}
\]
\[
\text{s2} = \text{s2} + \text{aaa}(j+\text{mu}) \star \text{uncert}
\]
\[
\text{end do}
\]
\[
\text{bbb}(i ) = \text{bbb}(i ) - \text{s1}
\]
\[
\text{bbb}(i+1) = \text{bbb}(i+1) - \text{s2}
\]
end if
end do  ! meqns
do i=1,mvary
xxx(i) = 0.0  ! initialize, CGNL
end do

niter = mvary*4
call cgnl (meqns,mvary, ia, ja, aa, bbb, xxx, &
        niter, u,v,w, xx,se)
do i=1,mvary  ! retain maximum
    xx(i) = amax1 (xx(i), abs (xxx(i)))
end do
goto 22
24 continue
goto 5
goto 5
end if

-----------------------------------------------
if (method.eq.3) then  ! singular value decomposition
    write (iout,108)
    stop
goto 5
c* end if
c*-----------------------------------------------
5 continue
if (.not.square) then
    if (xx(i) .ne. -1.0) then  ! not singular
        do i=1,mvary  ! account for scaling
            xx(i) = xx(i)/aats(i)
            xw(i) = xw(i)/aats(i)
        end do
    end if
end if

-----------------------------------------------
NOTE:  xx is NOT used ! <======================================

kt = kt+1
if (lsout) then
    write (isout,102) kt, (iiii(k), k=1,maraws)
    write (isout,103) ( xx(k), k=1,mvary )
end if
call poplat (xx, 2)  ! update

k = maraws  ! inner-most level: ( ambient, wave, sample)
j = maraws  ! inner-most level: ( angle, ambient, wave, sample)
j1 = jisave
j2 = j2save
goto 3  ! loop back * nested do, iiii
6 continue  ! last line of nested-do, iiii
c*===============================================
write (iout,111) kts, kt

if (lsout) close (unit=lsout)
call poplat (xx, 3) ! plot
call poplat (xx, 4) ! correlation at minima

write (iout,112)
do i=1,mvaryl
   m = iptu(i) ! unique,nonlocal,full
   if (m.le.mlnts) then
      s1 = diefcn(m)
      s2 = uncerl(m)
      write (iout,113) i, xx(i), s2, s1, m, '(n+ik)'
   else
      m = m-mlnts
      s1 = widths(m)
      s2 = uncerz(m)
      write (iout,113) i, xx(i), s2, s1, m, '(z )'
   end if
end do
return

100 format ('/ seesmax, insufficient allocation for array, AAT')
101 format (' singular mtx, iiiii: ', 20i4, :/(21x, 20i4))
102 format (ix, i10, ', ', 20i4, :/(12x, 20i4))
103 format (ix, ip10e12.4)
104 format ( ix, 20i4)
105 format ( ix, 3i4, 5x, f3.1)
106 format ('/ seesmax, maraws = ', i3, ', ', j=' , i3
       & '/ should be equal.')
107 format ('/ seesmax, method=3, singular value decomposition, '
       & '/ not available, ... yet. )
108 format ('/ seesmax, kts = ', i10, ' singular events'
       & '/ kt = ', i10, ' total events')
109 format ('/ Discern: uncertainty '
       & '/ when: not square, D = [J(T)*J]**-1 *J(T)'
       & '/ square, D = J **-1 '
       & '/ case: 1, |Dg| + |DJa| + |DJ||u|'
       & '/ 2, ||Dg|| + ||DJa|| + |DJ||u|'
       & '/ vary', 6x, 'case 1', 5x, 'case 2',
       & '/ 5x, 'initial', 4x, 'parameter' )
110 format (3x, i4, '), 2x, ip2e11.2, 11x, e15.6,
       & ', for: ', i4, ', ', a)
111 format (3x, i4, '), 2x, ip2e11.2, 11x, e15.6,
subroutine seam2 ! construct table <---- assembly
include 'iounit. '
include 'defnit. '
include 'filmm. '
include 'filmss. '
include 'seamx1. '
include 'seamx2. '
real a(nrows*2), b(2), c(2)

Solve the forward scattering problem
on a grid of distinct incident angles
for each distinct wavelength incident on the sample.
Construct matrix of possible scatterings.

mws = 0 !ofilms,wave,sampl
iws = 0 !wave,sampl
iaws = 0 !ambient,wave,sampl
iraws = 0 !repeat,ambient,wave,sampl
maraws = 0 !mangl,repeat,ambient,wave,sampl
ngl = 0 !nangl,repeat,ambient,wave,sampl
do is=1,misampl
  mfilm = nnfilm(is)
mwave = nnwave(is)
mfilms = mfilm+1
nrow = mfilm*3+2
do iw=1,mwave
    iws = iws+1
    iwave = iwave(iws)
    mbien = nnbien(iws)
    qq = waveqq(iwave)
do i=1,nrow
      iptx(i) = 0 ! vary ---> unique
      ipty(i) = 0 ! vary ---> full
      kptx(i) = 0 ! froz ---> unique
      kpty(i) = 0 ! froz ---> full
end do
iv = 0 !local, full, non-unique
kv = 0 !local, vary, non-unique
mv = 0 !local, vary, compress
ku = 0 !local, froz, non-unique
mu = 0 !local, froz, compress
do m=1,mfilms
  if (m.ne.mfilms) then ! films/z
    mws = mws+1
    iz = iiifilm(mws)
    zzzz(m) = widths(iz)
    iv = iv+1 !local, full
    j = mlnnts+iz !nonlocal, full
    if (1varyz(iz).eq.1) then ! vary
  end if
end do
i = iptv(j) ! nonlocal, compress
kv = kv+1 ! local, non-unique
iptx(kv) = i ! local " nonlocal
ipty(kv) = iv ! local, full
if (iptw(i).eq.0) then ! compress, unique
mv = mv+1 ! local counter
iptw(i) = mv ! local
end if
else ! frozen
i = iptv(j) ! nonlocal, compress
ku = ku+1 ! local, non-unique
kptx(ku) = i ! local " nonlocal
kpty(ku) = iv ! local, full
if (iptw(i).eq.0) then ! compress, unique
mu = mu+1 ! local counter
iptw(i) = mu ! local
end if
end if
end if
end do ! z
do ink=1,2 ! n+ik
mws = mws+1
nk = iiifilm(mws)
c(ink) = diefcn(nk) ! n,k
iv = iv+1 ! local, full, non-unique
if (lvaryl(nk).eq.1) then ! vary
i = iptv(nk) ! nonlocal, compress
kv = kv+1 ! local, non-unique
iptx(kv) = i
ipty(kv) = iv
if (iptw(i).eq.0) then ! compress, unique
mv = mv+1 ! local counter
iptw(i) = mv ! local
end if
else ! frozen
i = iptv(nk) ! nonlocal, compress
ku = ku+1 ! local, non-unique
kptx(ku) = i
kpty(ku) = iv
if (iptw(i).eq.0) then ! compress, unique
mu = mu+1 ! local counter
iptw(i) = mu ! local
end if
end if
end do ! n+ik
die(m) = cmplx (c(1),c(2))**2 ! FILMSS
end do ! mfilms
if (mv.eq.0) then
write (iout,100) is, iw, iwave
stop
end if
mvmu = mv+mu ! compress " (vary+froz| unique,local)
do mbm=1,mbien
   iaws = iaws+1
   imbien = iibent(iaws)
   mrpeat = nmpeat(iaws)
   air = ambient(imbien)
   do irpeat=1,mrpeat
      iraws = iraws+1
      mangl = mangle(iraws)
      maraws = maraws+mangl
      do iangl=1,ndegr
         ngl = ngl+i
         angl = float(iangl)
         angl = angl*raddeg
         call scatr(qq,angl,b,a,c)
         call tform(angl,b,a,c)
         uang = 0.002*raddeg
         b(1) = 0.050*raddeg
         b(2) = 0.050*raddeg
         psii(ngl) = b(1)
         dell(ngl) = b(2)
         psia(ngl) = c(1)
         dela(ngl) = c(2)
      enddo
      enddo
   enddo
   if (kv.ne.0) then
      do k=1,kv
         iv = ipty(k)
         iv2 = iv+iv
         iv1 = iv2-1
         i = iptx(k)
         imv = iptw(i)
         psid(imv,ngl) = psid(imv,ngl) + a(iv1)
         deld(imv,ngl) = deld(imv,ngl) + a(iv2)
      enddo
   enddo
   if (ku.ne.0) then
      do k=1,ku
         iv = kpty(k)
         iv2 = iv+iv
         iv1 = iv2-1
         i = kptx(k)
         imu = iptw(i)
      enddo
imu = imu+mv ! append "(vary+froz)
psid(imu,ngl) = psid(imu,ngl) + a(iv1)
deld(imu,ngl) = deld(imu,ngl) + a(iv2)

end do ! row
end if ! frozen
end do ! angle
c*
end do ! repeat
end do ! ambient

if (kv.ne.0) then ! vary
do k=1,kv
   i = iptx(k)
   iptw(i) = 0 ! indicator
end do
end if
if (ku.ne.0) then ! frozen
do k=1,ku
   i = kptx(k)
   iptw(i) = 0 ! indicator
end do
end if
end do ! wave
end do ! sample

mraws = iraws
return

100 format (// oops, there is NO varying model parameter ' 
& // for the case involving: sample = ', i3, 
& // where the ', i3, '-th wave = ', i3) 
end
subroutine seam3  ! fetch table <--- asmblx
include 'iounit.'
include 'deinit.'
include 'filmread.'
include 'seamx1.'
include 'seamx2.'
logical firstv, firstu

c Construct sparse matrix associated with model experiment.
i = 1
ia(1) = 1  ! vary
jj = 0
iia(1) = 1  ! frozen
jja = 0

iw = 0  ! wave,sampl
mws = 0  ! nfilms,wave,sampl
iaws = 0  ! ambient,wave,sampl
iawv = 0  ! repeat,ambient,wave,sampl
iaws = 0
mgl = 0  ! ndegr,repeat,ambient,wave,sampl

do is=1,msamp
mfilm = nfilm(is)  ! FILMSS
mwave = nnwave(is)
mfilms = mfilm+1
nrow = mfilm*3+2
do iw=1,mwave
iw = iw+1
iwave = iwave(iws)
bien = nbnent(iws)
do i=1,nrow
icx(i) = 0  ! vary ---> unique
ipty(i) = 0  ! vary ---> full
kptx(i) = 0  ! froz ---> unique
kpty(i) = 0  ! froz ---> full
end do
iv = 0
kv = 0
mv = 0
kx = 0
ku = 0
mu = 0

do m=1,mfilms
if (m.ne.mfilms) then
mws = mws+1
iz = iiifilm(mws)
iv = iv+1
j = mlnmts+iz

end do
if (lvaryz(iz).eq.1) then ! vary
  i = iptv(j) ! nonlocal, compress
  kv = kv+1 ! local, non-unique
  iptx(kv) = i ! local "nonlocal
  ipty(kv) = iv ! local, full
  if (iptx(i).eq.0) then ! compress, unique
    mv = mv+1 ! local counter
    iptw(i) = mv ! local
    jj = jj+1
    ja(jj) = i
  end if
else ! frozen
  i = iptv(j) ! nonlocal, compress
  ku = ku+1 ! local, non-unique
  kptx(ku) = i ! local "nonlocal
  kpty(ku) = iv ! local, full
  if (iptx(i).eq.0) then ! compress, unique
    mu = mu+1 ! local counter
    iptw(i) = mu ! local
    jja = jja+1
    jaa(jja) = i ! backwards, mm+1-i=k
  end if
  end if
end if
end if ! z
do ink=1,2 ! n+ik
  mws = mws+1
  nk = iifilm(mws)
  iv = iv+1 ! local, full, non-unique
  if (lvaryl(nk).eq.1) then ! vary
    i = iptv(nk) ! nonlocal, compress
    kv = kv+1 ! local, non-unique
    iptx(kv) = i
    ipty(kv) = iv
    if (iptw(i).eq.0) then ! compress, unique
      mv = mv+1 ! local counter
      iptw(i) = mv ! local
      jj = jj+1
      ja(jj) = i
    end if
  else ! frozen
    i = iptv(nk) ! nonlocal, compress
    ku = ku+1 ! local, non-unique
    kptx(ku) = i
    kpty(ku) = iv
    if (iptw(i).eq.0) then ! compress, unique
      mu = mu+1 ! local counter
      iptw(i) = mu ! local
      jja = jja+1
      jaa(jja) = i ! backwards, mm+1-i=k
    end if
  end if
end do ! n+ik
end do ! mfilms
firstv = .true.
firstu = .true.

do mbn=1,mbien ! ambients
  iaws = iaws+1
  imbien = iiibent(iaws)
ends
mrpeat = mnppeat(iaws)

do irpeat=1,mrpeat ! repeats
  iaraws = iaraws+1
  mangle = mangle(iaraws)
ends
iawg1=1,mangle ! angles
  iaraws = iaraws+1
  jjjj = iii(iaraws) ! angle
  ngll = ng1+jjjj ! pointer

  ia(ii+1) = ia(ii)+mv ! vary
  ia(ii+2) = ia(ii)+mv+mv
if (firstv) then
  firstv = .false.
  do j=1,mv
    jj = jj+1
    ja(jj) = ja(jj-mv)
  end do
else
  do j=1,mv
    jj = jj+1
    ja(jj ) = ja(jj-mv)
    ja(jj+mv) = ja(jj )
  end do
jj = jj+mv
end if
j1 = ia(ii )
j2 = ia(ii+1)-1
do j=j1,j2
  ja(j) = ja(j)
  inv = iptw(ja(j)
  aa(j ) = psid(inv,ngll)
  aa(j+mv) = deld(inv,ngll)
end do

 Extract info from the tabulation.

bb(ii ) = psii(ngll)
bb(ii+1) = deli(ngll)
cc(ii ) = psia(ngll)
cc(ii+1) = dela(ngll)

iaa(ii+1) = iaa(ii)+mu ! frozen
iaa(ii+2) = iaa(ii)+mu+mu
if (mu.ne.0) then
  if (firstu) then
    firstu = .false.
do j=1,mu
  jja = jja+1
  jaa(jja) = jaa(jja-mu)
end do

else
  do j=1,mu
    jja = jja+1
    jaa(jja) = jaa(jja-mu)
    jaa(jja+mu) = jaa(jja)  
  end do
  jja = jja+mu
end if

jji = iaa(ii )
jj2 = iaa(ii+1)-1
  do j=jji,jj2
    jaj = jaa(j)  ! backwards
    imu = iptv(jaj)
    imu = imu+mv  ! append
    aaa(j ) = psid(imu,ngll)
    aaa(j+mu) = deld(imu,ngll)
  end do
end if  ! frozen
ii = ii+2
end do  ! angles
ngl = ngl+ndegr  ! pointer
end do  ! repeat
end do  ! ambient

if (kv.ne.0) then  ! vary
  do k=1,kv
    i = iptx(k)
    ! unique-ness
  end do
end if

if (ku.ne.0) then
  ! frozen
  do k=1,ku
    i = kptx(k)
    ! unique-ness
  end do
end if

end do  ! wave
end do  ! sample

meqns = ii-1

if (jj .ne. ia(ii)-1) then
  write (iout,102) ii,jj,ia(ii)
  stop
end if

if (jja .ne. iaa(ii)-1) then
  write (iout,103) ii,jja,iaa(ii)-1
  stop
end if

if (jj.gt.nnjaaa .or. jja.gt.nnjaaa) then
write (iout,104)
stop
end if
return

102 format (/' asmlx, inconsistent format of sparse matrix, ',
& /' ii = ', i10
& /' jj = ', i10, ' /= ', i10, ' /= ia(ii)-1')

103 format (/' asmlx, inconsistent format of sparse matrix, '
& /' aaa ' froz'
& /' ii = ', i10
& /' jj = ', i10, ' /= ', i10, ' /= ia(ii)-1')

104 format (/' asmlx, array allocation for the sparse matrix '
& /' has been exceeded.'
& /' aa,ja <= nnjaaa (DEFNIT.)'
& /' aaa,jaa <= nnjaaa (DEFNIT.)')

end
subroutine poplat (u, job)  
dimension u(1)  
byte labx(64), laby(64), labg(64) ! graphics 
real tlab(41)  
include 'iounit.'  
include 'definit.'  
include 'filmn.'  
include 'filmss.'  
include 'seamxl.'  
include 'seamx2.'  
include 'wstack.'  

parameter (ndos = 91)  
integer kdos(ndos,nrows)  
real udos(ndos,nrows), xdos(ndos) ! graphics  
real umin(nrows), wsav(nrows), xx2(2), yy2(2)  
logical first(nrows)  

parameter (keep=10)  
integer kkkk(keep*nseams, nrows), kk(nrows)  

common / dossav / udos, xdos, umin, wsav,  
& kkkk, kk, kfull, first  
& equivalence (kdos(1,1), udos(1,1))  

goto (1,2,3,4), job  

1 kfull = (keep*nseams)/maraws ! limit/restrict storage  
kfull = maraws*min (kfull,keep) ! retain minima, truncate  
do m=1,mvary  
first(m) = .true.  
kk(m) = 0 ! pointer  
do i=1,ndos  
kdos(i,m) = 0 ! initialize  
end do  
end do  
return  

2 do m=1,mvary  
w = u(m) ! graphics, positioning  
if ((w.gt.1.0E4) .or. (w.lt.0.0)) then ! truncate, (-5,4)  
w = 4.0  
else if (w .gt. 1.0E-5) then ! map  
w = alog10 (w)  
else  
w = -5.0 ! truncate  
end if
wi = (w+5.0)/9.0 ! (0,1)
i = mint (wi+float (ndos)) ! index/position DOS
i = max (1, min (i, ndos)) ! assurance
kdos(i,m) = kdos(i,m)+1 ! update
if (first(m)) then ! discern minimum
  first(m) = .false.
  umin(m) = u(m)
  wsav(m) = w
  kk(m) = maraws
do j=1,maraws
    kkkk(j,m) = iii(i)
  end do
else if (w.lt.wsav(m)) then ! update
  umin(m) = u(m)
  wsav(m) = w
  kk(m) = maraws
do j=1,maraws
  kkkk(j,m) = iii(i)
end do
else if (w.eq.wsav(m)) then ! multiple minima
  if (kk(m).lt.kfull) then
    k = kk(m)
do j=1,maraws
      k = k+1
    kkkk(k,m) = iii(i)
  end do
  kk(m) = k
else
  kk(m) = kk(m)+maraws ! no retention, but
end if
end do
return

3 do i=1,ndos
  xdos(i) = - 5.0 * (float(ndos-i)/float(ndos-1))
  & + 4.0 * (float(i-1)/float(ndos-1))
end do
xx2(1) = xdos(1 ) ! min
xx2(2) = xdos(ndos) ! max
yy2(1) = 0.0 ! min
yy2(2) = 1.0 ! max
do m=1,mvarya
  write (iout,15) m, umin(m), wsav(m)
write (iplt,19) ndos, m
kmax = 0.0 ! Population, DOS
3 do i=1,ndos
  kmax = max (kmax, kdos(i,m))
end do

umax = 0.94/alog (float (kmax))  ! scale " 0.99 - .05

do i=1,ndos  ! rescale

if (kdos(i,m) .eq. 0) then
  udos(i,m) = 0.0
else
  udos(i,m) = umax*alog (float (kdos(i,m))) + 0.05
end if

write (iplt,20) i, xdos(i), udos(i,m)

end do

encode (12, 21, labx)  ! capital letters only
encode (32, 22, laby)  ! terminated by a
encode (26, 23, labg) m  ! dollar ($) sign.

linlog = 1  ! (linear, linear) " (x,u)
call displa (2, 0, linlog)
call agsetf ('GRAPH/RIGHT.', 0.8)

call aggeti ('LINE/MAXI.', mlln)
call agsetf ('LINE/END.', tcln)
call agcyl (min0(80,mlln), tcln, labg, tlab,ncdum)
call agsetf ('LABE/NAME.', 4H T)
call agseti ('LINE/NUMB.', 100)
call agsetp ('LINE(TEXT.', tlab, 1)

call anotat (labx,laby,1,1, 0,0)
call agstup (xx2,1,0,2,1, yy2,1,0,2,1)
call agback

call agcurv (xdos,1,udos(1,m),1,ndos,1)  ! line solid
call flush
call frame  !----------------------------------------------
end do

do m=1,mvary

  k = kk(m)/maraws  ! population of minima
  write (iout,31) m,k
  k = min (k,keep)  ! truncation
  k2 = 0
  do j=1,k  ! distinct sets of angles
    k1 = k2+1
    k2 = k2+maraws
    write (iout,32) j, (kkkk(i,m), i=k1,k2)
  end do

end do

return

Discern correlation among model parameters at minima.

4 do m=1,mvary

  kt = kk(m)/maraws  ! distinct minima
  write (iout,30)
write (iout,31) m, kt

kt = min (kt,keep) ! truncate
ik = 0

do iki=1,kt
do j=1,maraws ! extract angles at minima
   ik = ik+1
   iiii(j) = kkkk(ik,m)
end do
write (iout,203) (iii(j), j=1,maraws)
call seam3 ! model experiment
kv = 0
do jv=1,mvary ! A(T)*A
do iv=1,jv ! diagonal, upper triangle
   ss = 0.0
   do i=1,meqns,2
      j1 = ia(i )
      j2 = ia(i+1)-1
      mv = j2-j1+1
      s1 = 0.0
      s2 = 0.0
      s3 = 0.0
      s4 = 0.0
      do j=j1,j2 ! row
         jaj = ja(j) ! column
         if (jaj.eq.iv) then
            s1 = aa(j )
            s3 = aa(j+mv)
         end if
         if (jaj.eq.jv) then
            s2 = aa(j )
            s4 = aa(j+mv)
         end if
      end do
      ss = ss + s1*s2 + s3*s4
   end do ! meqns
   ss = ss /float (meqns)
kv = kv+1
aat(kv) = ss
if (iv.eq.jv) then
   u(jv) = sqrt (ss)
end if
end do
kv = 0
do jv=1,mvary ! renormalize
do iv=1,jv
   kv = kv+1
   aat(kv) = aat(kv)/(u(iv)*u(jv))
end do
end do
write (iout,204)
k2 = 0
do i=1,mvary
  k1 = k2+1
  k2 = k2+i
  write (iout,205) i, (aat(k), k=k1,k2)
end do
write (iout,207)
write (iout,208) (u(i), i=1,mvary)
if (iki.ne.kt) write (iout,29)
end do ! minima
end do ! vary
write (iout,30)
return

15 format ('/ vary, i =', i4, ', Umin = ', 1pe11.4, &
           ', ws = ', e11.4)
19 format (1x, i5, ' 1', i5, ' ndos, nu, ivary')
20 format (1x, i5, 5x, 1p2e13.5)
21 format ('UNCERTAINT$') ! 12
22 format (' POPULATION, (LOG, NORMALIZED)$') ! 32
23 format (' VARIATION PARAMETER, I=', i2, '$') ! 26
29 format (1x, 15('-----'))
30 format (1x, 15('===='))
31 format ('/ vary =', i4, ', population at minima =', i10)
32 format (7x, i4, '20i4, :/(12x, 20i4))
203 format (10x, 'iiii:', 20i4, :/(15x, 20i4))
204 format ('/ A(T)*A: (correlation)')
205 format (1x, i4, '), 1x, 10f10.5, :/(7x, 10f10.5))
207 format ('/ normalization coefficients:')
208 format (1x, 1p10e10.2)
end
Plot the data results from: SEAMAX

Sensitivity/Error Analysis for Multiple:
angle, ambient, wave, sample.

program plte
byte labx(64), laby(64), labg(64)
real tlab(41), s(0)
character*64 filenm

include 'defnit.'
integer nnwave(nsampl)
integer nnbent(nbient*nwaves*nsampl)
integer mangle(nbient*nwaves*nsampl)
integer isteps(nbient*nwaves*nsampl)
integer iii(nseams), iil(nseams), ii2(nseams), kkk(nseams)
real u(nrowss), umin, wsv
logical first

parameter (keep=10)
integer mmm(keep*nseams), mm

parameter (nx = 89) ! degrees
real ww(nx*nx), xx(nx), yy(nx) ! graphics
integer info(2*nx*nx)

parameter (ndos = 91)
integer kdos(ndos)
real udos(ndos), xdos(ndos)
equivalence (kdos(1), udos(1))
data in,iout,idat / 5,6,7 /

read ( in,301, err=102,end=102) filenm
close (unit=in)
write (iout,303) filenm
open (unit=idat, file=filenm, status='old',
& readonly, shared, err=103)

loop = 0
11 loop = loop+1 ! ivary
 rewind (unit=idat)
 read (idat,*) msampl, mvary, mdegr
 read (idat,*) (nnwave(is), is=1,msampl)
iws = 0
iaws = 0
iraws = 0
maraw = 0
do is=1,msampl
    mwave = nnwave(is)
do iws=1,mwave
   iws = iws+1
   read (idat,*), mbien
   numbent(iws) = mbien
   do mbn=1,mbien
      iws = iws+1
   c*
      read (idat,*), mrpeat
      nnpeat(iaws) = mrpeat
      c*
      do irpeat=1,mrpeat
         iaws = iaws+1
      end do
      c*
      isteps(iaws) = istep
      mangle(iaws) = mangl
      maraws = maraws+mangl
   end do
   do istep = istep,mangl
      mraws = iaws
   end do
   mx = mdegr
   kfull = (keep*nseams)/maraws ! prevent exceeding storage
   kfull = maraws*min (kfull,keep) ! retain minima, truncation
   first = .true.
   do i=1,ndos
      kdos(i) = 0 ! initialize
   end do
   m = loop ! ivary
   kt = 0
   mm = 0
   do i=1,ndos
      if (k.gt.mraws) goto 4
      istep = isteps(k)
      mangl = mangle(k)
      j1 = j2+1
      j2 = j2+mangl
      ii2(j1) = mdegr ! limit outer-most do
      do j=j1,j2
         iii(j) = 1+(j2-j)*istep ! multiple angle
         iii(j) = iii(j) - istep
      end do
      j = j1-1
      2 j = j+1
      if (j.gt.j2) goto 1
      3 iii(j) = iii(j)+istep
      if (iii(j) .le. ii2(j)) then ! test upper limit
         if (j.ne.j2) ii2(j+1)=iii(j)-istep
         goto 2
      end if
\begin{verbatim}
104  iii(j) = iii(j)-istep               ! reset inner do
105  j = j-1                            ! backup one do-level
106  if (j.ge.j1) goto 3
107
108  if (k.eq.1) goto 6                 ! escape do-nest
109  k = k-1
110  istep = isteps(k)
111  mangl = mangle(k)
112  j2 = j1-1
113  j1 = j1-mangl
114  goto 3
115  continue

117  j1save = j1
118  j2save = j2
119  if (j.ne.maraws+1) then
120        write (iout,107) maraws,j
121        stop
122  end if

123  kt = kt+1
124  read (idat,*) kkkk, (kkk(k),k=1,maraws)
125  read (idat,*) ( u(k),k=1,mvary )
126
127  if (kkkk .ne. kt) then               ! test consistency
128        write (iout,212) kkkk,kt
129        goto 101
130  end if

131  do i=1,maraws                       ! consistency check
132    if (kkk(i) .ne. iii(i)) then
133        write (iout,212) kkkk, kt
134        write (iout,213) (kkk(k), k=1,maraws)
135        write (iout,214) (iii(k), k=1,maraws)
136        goto 101
137  end if
138
139  end do

140  w = u(m)                             ! graphics, positioning
141
142  if ((w.gt.1.0E4) .or. (w.lt.0.0)) then ! (-5,4)
143     w = 4.0
144  else if (w.gt.1.0E-5) then
145     w = alog10 (w)
146  else
147     w = -5.0
148  end if

149  wi = (w+5.0)/9.0                     ! (0,1)
150  i = nint (wi*float (ndos))           ! Density of States
151  i = max (1, min (i,ndos))            ! update
152  kdos(i) = kdos(i)+1
153
154  if (first) then
155     first = .false.
156
157  continue
\end{verbatim}
um = u(m) ! retain minimum
ws = w
do i=1,maraws
   mmm(i) = iii(i) ! retain angles
end do
mm = maraws ! pointer/counter
do if (w.lt.ws) then ! minimum update
   um = u(m)
   ws = w
   do i=1,maraws
      mmm(i) = iii(i) ! retain angles
   end do
   mm = maraws
end if
else if (w.eq.ws) then ! multiple minima
   if (mm.lt.kfull) then ! prevent exceeding storage
      do i=1,maraws
         mm = mm+1
         mmm(mm) = iii(i) ! pointer/counter
      end do
   else
      mm = mm+maraws
   end if
end if

Note that all plots involve two dimensions. So, for cases
involving multiple: angles, ambients, waves, and samples,
... it is NOT clear what should (or is able to) be plotted.
For the special case involving: 1 or 2 multiple angles,
1 ambient, 1 wave, and 1 sample,
we may consider plotting the following arrays:

if (maraws .eq. 1) then
   i = iii(1) ! >0
   ww(i) = float(i) ! x = angle
   ww(i+mx) = w ! u = uncertainty
else if (maraws .eq. 2) then
   i = iii(1) + mx*(iii(2)-1) ! (i1,i2) symmetric
   j = iii(2) + mx*(iii(1)-1) ! (i2,i1)
   ww(i) = w
   ww(j) = w
end if

---
!

k = mraws
j = maraws
j1 = j1save
j2 = j2save
goto 3
6 continue ! escape do-loops
!

k = mm/maraws
write (iout,215) m, k, um, ws ! minimum
k = min (k,keep) ! truncate
! Angular locations
k1 = k2+1
k2 = k2+maraws
write (iout, 216) j, (mmm(i), i=k1,k2)
end do

do i=1, ndos
xdos(i) = - 5.0 * (float(ndos-i)/float(ndos-1))
+ 4.0 * (float(i-1)/float(ndos-1))
end do

xx(1) = xdos(1) ! min
xx(2) = xdos(ndos) ! max
yy(1) = 0.0 ! min
yy(2) = 1.0 ! max

kmax = 0.0 ! Population, DOS

! Population, DOS
do i=1, ndos
  kmax = max (kmax, kdos(i))
end do

umax = 0.94 / alog (float (kmax)) ! scale = 0.99 - 0.05

! rescale
if (kdos(i) .eq. 0) then
  udos(i) = 0.0
else
  udos(i) = umax * alog (float (xdos(i))) + 0.05
end if

end do

encode (12, 221, labx) ! capital letters only
call displa (2, 0, linlog)
call agsetf ('GRAPH/RIGHT.', 0.8)
call aggeti ('LINE/MAJ.', mlln)
call agsetf ('LINE/END.', tcln)
call agcpyl (min0(80, mlln), tcln, labg, tlab, ncdum)
call agsetf ('LAE/NAME.', 4H T)
call agseti ('LINE/NUMB.', 100)
call agsetp ('LINE/TEXT.', tlab, 1)
call anotat (labx, laby, 1, 1, 0, 0)
call aagstup (xx, 1, 0, 2, 1, yy, 1, 0, 2, 1)
call agback
call agcurv (xdos, 1, udos, 1, ndos, 1) ! line solid
call flush
call frame !-----------------------------

c c Plot: uncertainty
if (maraws.eq.1) then
    mxs = mx
    if (istep.ne.1) then
        j = 0
        do i=1,mx,istep
            j = j+1
            ww(j) = ww(i)
            ww(j+mx) = ww(i+mx)
        end do
        mxs = j
    end if
    xx(1) = 0.0
    xx(2) = ww(mxs)
    yy(1) = -5.0
    yy(2) = 4.0
    if (xx(2) .lt. 45.0) then
        xx(2) = 45.0
    else
        xx(2) = 90.0
    end if
    encode (32, 224, labx)
    encode (12, 221, laby)
    encode (26, 223, labg) m
    linlog = 1
    call displa (2, 0, linlog)
    call agsetf ('GRAPH/RIGHT.', 0.8)
    call aggeti ('LINE/MAII.', mlln)
    call aggetf ('LINE/END.', tcln)
    call agcpyl (min0(80,mlln), tcln, labg, tlab, ndum)
    call agsetf ('LABE/NAME.', 4H T)
    call agseti ('LINE/NUMB.', 100)
    call agsetp ('LINE/TEXT.', tlab, 1)
    call anotat (labx,laby,1,1, 0,0)
    call agstup (xx,1,0,2,1, yy,1,0,2,1)
    call agback
    call agcurv (ww(1),1, ww(1+mx),1, mxs,1) ! line solid
    call flush
    call frame !----------------------------------------
end if

if (maraws.eq.2) then
    span = 9.0
    do i=1,mx,istep
        j = j + (i-1)*mx
        ww(j) = 4.0
    end do
    call agstup (xx,1,0,2,1, yy,1,0,2,1)
    call agback
    call agcurv (ww(1),1, ww(1+mx),1, mxs,1) ! line solid
    call flush
    call frame !----------------------------------------
end if
xy = (float(i-1)/float(mx-1)) * 2.0*span  ! scale plots
xx(i) = xy  ! domain " plot routine
yy(i) = xy  ! domain " plot routine
end do

mxs = mx  ! assume: istep=1
if (istep.ne.1) then  ! compress matrix
  mxs = 0  ! compress index
  k = 0
  do j=1,mx,istep
    jmx = (j-1)*mx
    do i=1,mx,istep
      k = k+1
      ww(k) = ww(i+jmx)
    end do
    mxs = mxs+j
  end do
  xx(mxs) = xx(j)
  yy(mxs) = yy(j)
end if

xn = 1.0  ! viewing vector
yn = 1.0
zn = 1.0

s " defines the line of sight of viewer and object.
c the viewer's eye is at : s(1-3) " (x,y,z) " NCAR
c the point looked at is : s(4-6) " NCAR
c effective radius of obj: s(7-9) " convenience

s(4) = span  ! (xmax+xmin)*0.5  center of object or
s(5) = span  ! (ymax+ymin)*0.5  point being viewed
s(6) = 0.0  ! (umax+umin)*0.5
s(7) = span  ! (xmax-xmin)*0.5  radius of object
s(8) = span  ! (ymax-ymin)*0.5
s(9) = 0.0  ! (umin-umax)*0.5
radius = amax1 (s(7), s(8), s(9))  ! effective radius
dist = radius*5.0  ! convenient distance from
ss = xn*xn+yn*yn+zn*zn  ! which to view the object
dist = dist/sqrt(ss)  ! induce unit vector
s(1) = s(4) + xn*dist
s(2) = s(5) + yn*dist
s(3) = s(6) + zn*dist
call seti (10,10)
call surfce (xx,yy,ww,info, mxs,mxs,mxs, s,0.0)
call ezcntr (ww, mxs,mxs)
end if
if (loop.ne.mvary) goto 11
101 close (unit=idat)
stop
102 write (iout,302) in
stop
103 write (iout,302) idat
stop

107 format (' oops, maraws,j: ', 2i5)
212 format (' oops: ... input data is out of sync.'
& '/'
card index, kkkk = ', i10
& '/'
card count, kt = ', i10)
213 format ('/ card data, kkk:', 10i4, :/(16x, 10i4))
214 format ('/ do-loop, iii:', 10i4, :/(16x, 10i4))
215 format ('/ vary, i =',i4,', population of minima =',i10
& '/ 20x, 'Umin =', 1pe11.4, 6x, 'w=', e11.4)
216 format (6x, i4, '), 20i4, :/(10x, 20i4))

221 format ('UNCERTAINTY$') ! 12
222 format ('POPULATION, (LOG, NORMALIZED)$') ! 32
223 format ('VARIATION PARAMETER, I=', I2, '$') ! 26
224 format ('ANGLE OF INCIDENCE, (DEGREES)$') ! 32

301 format ( a64)
302 format (' Unable to open IO unit = ', i3)
303 format ( ix, a64)
end
6.3 General Utilities

6.3.1 DOT.FOR

```fortran
subroutine dot (n, x, y, xy, k)
    real x(n), y(n)

    xy = 0.0
    xn = 0.0
    yn = 0.0
    do i=1,n  ! find maximum
        xn = amax1 (xn, abs (x(i)))
        yn = amax1 (yn, abs (y(i)))
    end do
    if (xn.eq.0.0 .or. yn.eq.0.0) return

    xs = 0.0
    ys = 0.0
    do i=1,n
        xx = x(i)/xn  ! scaled vector component
        yy = y(i)/yn
        xs = xs + xx*xx  ! dot product \(|x*x|\)
        ys = ys + yy*yy  ! dot product \(|y*y|\)
        xy = xy + xx*yy  ! dot product \(|x*y|\)
    end do

    h = float (n)
    xs = xs/h  ! mean square value
    ys = ys/h
    xy = xy/h
    xs = sqrt (xs)  ! root mean square value
    ys = sqrt (ys)

    if (k.eq.0) then  ! usual dot product
        xy = xy*xn*yn*h
    else if (k.eq.1) then  ! un-normalized
        xy = xy*xn*yn
    else  ! normalized
        xy = xy/(xs*ys)
    end if

    return
end
```

215
subroutine norm (n,x,xn,k)
real x(1)
xn = 0.0
  do i=1,n
    xn = amax1 (xn, abs (x(i)))
  end do
  if (xn .eq. 0.0) return

xx = 0.0
  do i=1,n
    xx = xx + (x(i)/xn)**2
  end do
  if (k.eq.1) then
    xx = xx/float (n)
  end if
  xn = xn*sqrt (xx)
return
end
6.3.3 APROD.FOR

```fortran
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 /

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

  x' = x' + y' * A

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
        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
        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
```

217
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)

scale the rows in matrix, $A(m, n)$, $b(m)$.
retain scaling coefficients in $P(m)$.
evaluate $P$ only when $k = 0, 1, 2, 3$.
re-scale $A$ only when $|k| = 2, 3$.
re-scale $B$ only when $|k| = 1, 3$.
matrix $A$ is stored (row-wise) in the Yale Sparse Matrix Format.

ka = iabs (k) ! convenience
if (ka.gt.3) then
   do i=1,m ! out of range
      p(i) = 1.0 ! scan rows
      end do ! default
   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 ! initialize
            ss = 0.0
            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
endif

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
         end if
51 \[ pp = p(i) \]
52 \[ jj = ia(i)-1 \]
53 do \[ j=1,mj \]
54 \[ jj = jj+1 \]
55 \[ a(jj) = a(jj)/pp \]
56 end do
57 end if
58 end do
59 end if

60 if (ka.eq.1 .or. ka.eq.3) then
61 do \[ i=1,m \]
62 \[ b(i) = b(i)/p(i) \]
63 end do
64 end if
65
66 return
67 end
subroutine scaljj(m,n, ia, ja, a, b, p, k)
integer ia(l), ja(l), k, m, n
real a(1), b(1), p(1), w(1)

Scale matrix A so the diagonal of: [Transpose(A)*A] = 1.
Matrix A is stored row-wise in the Yale Sparse Matrix Format.
Scale the columns of matrices, A(m,n), b(n).
Retain the scaling coefficients in P(n).
Use as a dummy work storage array, W(n).

Evaluate P only when k = 0,1,2,3.
Re-scale A only when |k| = 2,3.
Re-scale B only when |k| = 1,3.

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
    w(j) = 0.0 ! initialize
  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
        jj = jj+1
        kk = ja(jj)
        p(kk) = amax1 (p(kk), abs (a(jj))) ! max |A(j,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
        jj = jj+1
        kk = ja(jj)
        w(kk) = w(kk) + (a(jj)/p(kk))**2 ! sums
      end do
    end if
  end do

return
if S3
end do

j=l,n
!
scan columns

p(j) = p(j) sqrt(w(j))
!
retain scale factor
end do

end if

if (ka.eq.2 .or. ka.eq.3) then

j=1,m
!
scan rows, A

mj = ia(i+l)-ia(i)
!
number of columns
if (mj.ne.0)
then

jj = ia(i)-l
!
indexing

do j=1,mj
!
scan columns in row

kk = ja(jj)
!
column

^{jj} > i(jj)/F(l^)
!
rescale column
end do

end if

end do

end do

end if

if (ka.eq.1 .or. ka.eq.3) then

j=1,n
!
scan columns

b(j) = b(j)/p(j)
!
rescale column
end do

end if

return
end
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/

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

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

loop = 0
! initialize

1 loop = loop+1
! update counter

rr = 0.0
! norm of residual

do i=1,ma
   mj = ia(i+1)-ia(i)
   ! number of columns in row
   if (mj .ne. 0) then
      ss = 0.0
      jj = ia(i)-1
      do j=1,mj
         jj = jj+1
         kk = ja(jj)
         ss = ss + aa(jj)*x(kk)
      end do
      ! Ax
      ss = b(i)-ss
      u(i) = ss
      rr = rr+ss*ss
   end if
end do
if (rr .eq. 0.0) return

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

--------------------------- ! Set-up for LSQR
relpr = 1.0E-06 ! relative precision of floating point arithmetic
damp = 1.0E+00 !
atol = 1.0E-06 ! relative error of data in A
btol = 1.0E-06 ! relative error of data in B \times \text{rhs}
conlim = 1.0E+04 ! apparent condition number of matrix A-bar
!(upper limit)
itnlim = itmax ! upper limit on number of iterations
nout = -iout ! output index to printer

call lsqr (ma, na, aprod, damp,
& ia, ja, aa,
& u, v, w, xx, se,
& atol, btol, conlim, itnlim, nout,
& istop, anorm, acond, rnorm, arnorm, xnorm)
do i=1,na
  x(i) = x(i) + xx(i) ! update solution
  xx(i) = 0.0 ! reset
end do
goto 1 ! loop back

101 format (' cgnl, singular row= ', i10)
102 format (' cgnl, singular column= ', i10)
103 format (' cgnl, ', i5, 1p1e11.3, ' loop, ratio '
& '(residual reduction)')
104 format (' cgnl, ', i5, 1p5e11.3)
end
6.3.7 LSQR.FOR

The following subroutine is located in the software library,


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


```fortran
SUBROUTINE LSQR(M,N,APROD,DAMP,
           IA,JA,AA,
           U,V,W,X,SE,
           ATOL,BTOL,CONLIM,ITNLIM,NOUT,
           ISTOP,ANORM,ACOND,RNORM,ARNORM,XNORM )

EXTERNAL APROD

INTEGER M,N,ITNLIM, NOUT, ISTOP
INTEGER IA(M), JA(M)
REAL AA(M),
   U(M),V(N),W(N),X(N),SE(N),
   ATOL,BTOL,CONLIM,DAMP,ANORM,ACOND,RNORM,ARNORM,XNORM

LSQR FINDS A SOLUTION X TO THE FOLLOWING PROBLEMS...

1. UNSYMMETRIC EQUATIONS -- SOLVE A*X = B

2. LINEAR LEAST SQUARES -- SOLVE A*X = B
   IN THE LEAST-SQUARES SENSE

3. DAMPED LEAST SQUARES -- SOLVE ( A + DAMP*I )*X = ( B )
   ( DAMP*I ) ( 0 )
   IN THE LEAST-SQUARES SENSE

WHERE A IS A MATRIX WITH M ROWS AND N COLUMNS,
B IS AN M-VECTOR, AND
DAMP IS A SCALAR (ALL QUANTITIES REAL).
THE MATRIX A IS INTENDED TO BE LARGE AND SPARSE.
IT IS ACCESSED BY MEANS OF SUBROUTINE CALLS OF THE FORM
```

224
CALL APROD ( MODE, M, N, X, Y, IA, JA, AA )

Which must perform the following functions...

IF MODE = 1, compute \( Y = Y + A^T X \).
IF MODE = 2, compute \( X = X + (A^T)^T Y \).

The vectors \( X \) and \( Y \) are input parameters in both cases.

If MODE = 1, \( Y \) should be altered without changing \( X \).
If MODE = 2, \( X \) should be altered without changing \( Y \).

The parameters: IA, JA, AA.

May be used for workspace as described below.

The RHS vector \( B \) is input via \( U \), and subsequently overwritten.

Note. LSQR uses an iterative method to approximate the solution.
The number of iterations required to reach a certain accuracy
depends strongly on the scaling of the problem. Poor scaling of
the rows or columns of \( A \) should therefore be avoided where
possible.

For example, in Problem 1 the solution is unaltered by
row-scaling. If a row of \( A \) is very small or large compared to
the other rows of \( A \), the corresponding row of \( (A^T B) \) should
be scaled up or down.

In Problems 1 and 2, the solution \( X \) is easily recovered
following column-scaling. In the absence of better information,
the nonzero columns of \( A \) should be scaled so that they all have
the same Euclidean norm (e.g. 1.0).

In Problem 3, there is no freedom to re-scale if \( \text{DAMP} \) is
nonzero. However, the value of \( \text{DAMP} \) should be assigned only
after attention has been paid to the scaling of \( A \).

The parameter \( \text{DAMP} \) is intended to help regularize
ill-conditioned systems, by preventing the true solution from
being very large. Another aid to regularization is provided by
the parameter \( \text{ACOND} \), which may be used to terminate iterations
before the computed solution becomes very large.

Notation

--------

The following quantities are used in discussing the subroutine
parameters...

\[
\begin{align*}
\text{ABAR} &= \begin{pmatrix} A \end{pmatrix}, \\
\text{BBAR} &= \begin{pmatrix} B \\ \text{DAMP}^T I \end{pmatrix}, \\
\text{O} & \end{align*}
\]
\[ R = B - A \cdot X, \quad R{\text{Bar}} = B{\text{Bar}} - A{\text{Bar}} \cdot X \]

\[ R{\text{Norm}} = \sqrt{\text{Norm}(R)^2 + \text{Damp}^2 \cdot \text{Norm}(X)^2} \]

\[ = \text{Norm}(R{\text{Bar}}) \]

\[ \text{RelPr} = \text{The Relative Precision of Floating-Point Arithmetic} \]

on the machine being used. For example, on the IBM 370, \text{RelPr} is about 1.0e-6 and 1.0d-16 in single and double precision respectively.

LSQR minimizes the function \( \text{RNorm} \) with respect to \( X \).

\[ \text{Parameters} \]

---

\( M \) INPUT THE NUMBER OF ROWS IN \( A \).

\( N \) INPUT THE NUMBER OF COLUMNS IN \( A \).

\( \text{AProd} \) EXTERNAL SEE ABOVE.

\( \text{Damp} \) INPUT THE Damping parameter for Problem 3 above.

\( \text{Damp} \) SHOULD BE 0.0 FOR PROBLEMS 1 AND 2.

IF THE SYSTEM \( A \cdot X = B \) IS INCOMPATIBLE, VALUES

OF \( \text{Damp} \) IN THE RANGE 0 TO \( \text{SQRT} \left( \text{RelPr} \right) \cdot \text{Norm}(A) \)

WILL PROBABLY HAVE A NEGLIGIBLE EFFECT.

LARGER VALUES OF \( \text{Damp} \) WILL TEND TO DECREASE

THE NORM OF \( X \) AND TO REDUCE THE NUMBER OF

ITERATIONS REQUIRED BY LSQR.

THE WORK PER ITERATION AND THE STORAGE NEEDED

BY LSQR ARE THE SAME FOR ALL VALUES OF \( \text{Damp} \).

\( \text{IA} \) INPUT CONTAINS ROW INFORMATION OF \( \text{Array} \) \( A \).

\( \text{JA} \) INPUT CONTAINS COLUMN INFORMATION OF A ROW WITHIN \( A \).

\( \text{AA} \) INPUT THE \( A \) ARRAY.

NOTE. LSQR DOES NOT EXPLICITLY USE THE PREVIOUS FOUR

PARAMETERS, BUT PASSES THEM TO SUBROUTINE APROD FOR

POSSIBLE USE AS WORKSPACE. IF APROD DOES NOT NEED

\( \text{IW} \) OR \( \text{RW} \), THE VALUES \( \text{LENW} = 1 \) OR \( \text{LENW} = 1 \) SHOULD

BE USED, AND THE ACTUAL PARAMETERS CORRESPONDING TO

\( \text{IW} \) OR \( \text{RW} \) MAY BE ANY CONVENIENT ARRAY OF SUITABLE TYPE.

\( U(M) \) INPUT THE RHS VECTOR \( B \). BEWARE THAT \( U \) IS

OVER-WRITTEN BY LSQR.

\( V(N) \) WORKSPACE

\( W(N) \) WORKSPACE

\( X(N) \) OUTPUT RETURNS THE COMPUTED SOLUTION \( X \).
SE(N) OUTPUT

RETURNS STANDARD ERROR ESTIMATES FOR THE

COMPONENTS OF X. FOR EACH I, SE(I) IS SET

TO THE VALUE RNORM * SQRT( SIGMA(I,I) / T ),

WHERE SIGMA(I,I) IS AN ESTIMATE OF THE I-TH

DIAGONAL OF THE INVERSE OF ABAR(TRANSPOSE)*ABAR

AND T = 1 IF M .LE. N,

T = M - N IF M .GT. N AND DAMP = 0,

T = M IF DAMP .NE. 0.

ATOL INPUT

AN ESTIMATE OF THE RELATIVE ERROR IN THE DATA

DEFINING THE MATRIX A. FOR EXAMPLE,

IF A IS Accurate TO ABOUT 6 DIGITS, SET

ATOL = 1.0E-6.

BTOL INPUT

AN ESTIMATE OF THE RELATIVE ERROR IN THE DATA

DEFINING THE RHS VECTOR B. FOR EXAMPLE,

IF B IS Accurate TO ABOUT 6 DIGITS, SET

BTOL = 1.0E-6.

CONLIM INPUT

AN UPPER LIMIT ON COND(ABAR), THE APPARENT

CONDITION NUMBER OF THE MATRIX ABAR.

ITERATIONS WILL BE TERMINATED IF A COMPUTED

ESTIMATE OF COND(ABAR) EXCEEDS CONLIM.

THIS IS INTENDED TO PREVENT CERTAIN SMALL OR

ZERO SINGULAR VALUES OF A OR ABAR FROM

COMING INTO EFFECT AND CAUSING UNWANTED GROWTH

IN THE COMPUTED SOLUTION.

CONLIM AND DAMP MAY BE USED SEPARATELY OR

TOGETHER TO REGULARIZE ILL-CONDITIONED SYSTEMS.

NORMALLY, CONLIM SHOULD BE IN THE RANGE

1000 TO 1/RELPR.

SUGGESTED VALUE --

CONLIM = 1/(100*RELPR) FOR COMPATIBLE SYSTEMS,

CONLIM = 1/(10*SQRT(RELPR)) FOR LEAST SQUARES.

NOTE. IF THE USER IS NOT CONCERNED ABOUT THE PARAMETERS

ATOL, BTOL AND CONLIM, ANY OR ALL OF THEM MAY BE SET

TO ZERO. THE EFFECT WILL BE THE SAME AS THE VALUES

RELPR, RELPR AND 1/RELPR RESPECTIVELY.

ITNLIM INPUT

AN UPPER LIMIT ON THE NUMBER OF ITERATIONS.

SUGGESTED VALUE --

ITNLIM = N/2 FOR WELL CONDITIONED SYSTEMS,

ITNLIM = 4*N OTHERWISE.

NOUT INPUT

FILE NUMBER FOR PRINTER. IF POSITIVE,

A SUMMARY WILL BE PRINTED ON FILE NOUT.

ISTOP OUTPUT

AN INTEGER GIVING THE REASON FOR TERMINATION...

0 X = 0 IS THE EXACT SOLUTION.
189 C NO ITERATIONS WERE PERFORMED.
190 C
191 C 1 THE EQUATIONS A*X = B ARE PROBABLY COMPATIBLE. NORM(A*X - B) IS SUFFICIENTLY SMALL, GIVEN THE VALUES OF ATOL AND BTOL.
192 C
193 C 2 THE SYSTEM A*X = B IS PROBABLY NOT COMPATIBLE. A LEAST-SQUARES SOLUTION HAS BEEN OBTAINED WHICH IS SUFFICIENTLY ACCURATE, GIVEN THE VALUE OF ATOL.
194 C
195 C 3 AN ESTIMATE OF COND(ABAR) HAS EXCEEDED CONLIM. THE SYSTEM A*X = B APPEARS TO BE ILL-CONDITIONED. OTHERWISE, THERE COULD BE AN ERROR IN SUBROUTINE APROD.
196 C
197 C 4 THE EQUATIONS A*X = B ARE PROBABLY COMPATIBLE. NORM(A*X - B) IS AS SMALL AS SEEMS REASONABLE ON THIS MACHINE.
198 C
199 C 5 THE SYSTEM A*X = B IS PROBABLY NOT COMPATIBLE. A LEAST-SQUARES SOLUTION HAS BEEN OBTAINED WHICH IS AS ACCURATE AS SEEMS REASONABLE ON THIS MACHINE.
200 C
201 C 6 COND(ABAR) SEEMS TO BE SO LARGE THAT THERE IS NOT MUCH POINT IN DOING FURTHER ITERATIONS, GIVEN THE PRECISION OF THIS MACHINE.
202 C
203 C 7 THE ITERATION LIMIT ITNLIM WAS REACHED.
204 C
206 C IF DAMP IS SMALL AND IF THE COLUMNS OF A HAVE ALL BEEN SCALED TO HAVE LENGTH 1.0, ANORM SHOULD INCREASE TO ROUGHLY SQRT(N). A RADICALLY DIFFERENT VALUE FOR ANORM MAY INDICATE AN ERROR IN SUBROUTINE APROD (THERE MAY BE AN INCONSISTENCY BETWEEN MODES 1 AND 2).
207 C
208 C ACOND OUTPUT AN ESTIMATE OF COND(ABAR), THE CONDITION NUMBER OF Aabar. A VERY HIGH VALUE OF ACOND MAY AGAIN INDICATE AN ERROR IN APROD.
209 C
210 C RNORM OUTPUT AN ESTIMATE OF THE FINAL VALUE OF NORM(RBAR), THE FUNCTION BEING MINIMIZED (SEE NOTATION ABOVE). THIS WILL BE SMALL IF A*X = B HAS A SOLUTION.
211 C
212 C ARNORM OUTPUT AN ESTIMATE OF THE FINAL VALUE OF NORM( Aabar(TRANSPOSE)*Rbar ), THE NORM OF
THE RESIDUAL FOR THE USUAL NORMAL EQUATIONS.
THIS SHOULD BE SMALL IN ALL CASES. (ARNorm
WILL OFTEN BE SMALLER THAN THE TRUE VALUE
COMPUTED FROM THE OUTPUT VECTOR X.)

\[
KNorm \quad \text{output} \quad \text{an estimate of the norm of the final}
\text{solution vector } x.
\]

SUBROUTINES AND FUNCTIONS USED
-----------------------------

USER    APROD
LSQR    NORMLZ
BLAS    SCOPY,SJRNM2,SSCAL (see Lawson et al. below)
        (SJRNM2 IS USED ONLY IN NORMLZ)
FORTRAN ABS,MOD,SQRT

PRECISION
--------

THE NUMBER OF ITERATIONS REQUIRED BY LSQR WILL USUALLY DECREASE
IF THE COMPUTATION IS PERFORMED IN HIGHER PRECISION. TO CONVERT
LSQR AND NORMLZ BETWEEN SINGLE- AND DOUBLE-PRECISION, CHANGE
THE WORDS
SCOPY, SNRM2, SSCAL
ABS, REAL, SQRT
TO THE APPROPRIATE BLAS AND FORTRAN EQUIVALENTS.

REFERENCES
-------

PAIGE, C.C. AND SAUNDERS, M.A. LSQR\&D AN ALGORITHM FOR SPARSE
LINEAR EQUATIONS AND SPARSE LEAST SQUARES.
ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE 8, 1 (MARCH 1982).

LAWSON, C.L., HANSON, R.J., KINCAID, D.R. AND KROGH, F.T.
BASIC LINEAR ALGEBRA SUBPROGRAMS FOR FORTRAN USAGE.
ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE 5, 3 (SEPT 1979),
308-323 AND 324-325.

LSQR.  THIS VERSION DATED 22 FEBRUARY 1982.

FUNCTIONS AND LOCAL VARIABLES

INTEGER I,ITN,MOD,NCONV,NSTOP
REAL  ABS,SQRT
REAL  ALFA,BBNORM,BETA,BNORM,
      1 CS,CS1,CS2,CTOL,DDMSQ,DDNORM,DELTA,
205  2  GAMMA, GAMBAR, ONE, PHI, PHIBAR, PSI,
206  3  RES1, RES2, RHO, RHOBAR, RHBAR1, RHBAR2, RHS, RTOL,
207  4  SN, SN1, SN2, T, TAU, TEST1, TEST2, TEST3,
208  5  THETA, T1, T2, T3, XINORM, Z, ZBAR, ZERO

209  C
210  C
211  C
212  C
213  C
214  C
215  C
216  C
217  C
218  C
219  C
220  C
221  C
222  C
223  C
224  C
225  C
226  C
227  C
228  C
229  C
230  C
231  C
232  C
233  C
234  C
235  C
236  C
237  C
238  C
239  C
240  C
241  C
242  C
243  C
244  C
245  C
246  C
247  C
248  C
249  C
250  C
251  C
252  C
253  C
254  C
255  C
256  C
257  C
258  C
259  C
260  C
261  C
262  C
263  C
264  C
265  C
266  C
267  C
268  C
269  C
270  C
271  C
272  C
273  C
274  C
275  C
276  C
277  C
278  C
279  C
280  C
281  C
282  C
283  C
284  C
285  C
286  C
287  C
288  C
289  C
290  C
291  C
292  C
293  C
294  C
295  C
296  C
297  C
298  C
299  C
300  C
301  C
302  C
303  C
304  1  WRITE(NOUT, 1000) M,N,DAMP,ATOL,CONLIM,BTOL,ITNLIM
305  ZERO = 0.0
306  ONE = 1.0
307  CTOL = ZERO
308  IF (CONLIM .GT. ZERO) CTOL = ONE/CONLIM
309  DAMPSQ = DAMP**2
310  ANORM = ZERO
311  ACOND = ZERO
312  BBNORM = ZERO
313  DDNORM = ZERO
314  RES2 = ZERO
315  XNORM = ZERO
316  XINORM = ZERO
317  CS2 = -ONE
318  SN2 = ZERO
319  Z = ZERO
320  ITN = 0
321  ISTOP = 0
322  NSTOP = 0
323  DO 10 I = 1, N
324     V(I) = ZERO
325     X(I) = ZERO
326     SE(I) = ZERO
327  10 CONTINUE
328  SET UP THE FIRST VECTORS FOR THE BIDIOAGONALIZATION.
329  THESE SATISFY  BETA*U = B,  ALFA*V = A(TRANSPOSE)*U.
330  CALL NORMLZ( M,U,BETA )
331  CALL APROD ( 2,M,N,V,U, IA,JA,AA )
332  CALL NORMLZ( N,V,ALFA )
333  CALL SCOPT( N,V,1,W,1 )
334  RHOBAR = ALFA
335  PHIBAR = BETA
336  BNORM = BETA
337  RNNORM = BETA
338  ARNNORM = ALFA*BETA
339  IF (ARNORM .LE. ZERO) GO TO 800
340  IF (NOUT .LE. 0 ) GO TO 100
341  IF (DAMPSQ .LE. ZERO) WRITE(NOUT, 1200)
342  IF (DAMPSQ .GT. ZERO) WRITE(NOUT, 1300)
343  TEST1 = ONE
TEST2 = ALFA/BETA
WRITE(OUT, 1500) ITN,X(1),RNORM,TEST1,TEST2
WRITE(OUT, 1600)
C
C MAIN ITERATION LOOP.
C
100 ITN = ITN + 1
C
PERFORM THE NEXT STEP OF THE BIDIAGONALIZATION TO OBTAIN THE
NEXT BETA, U, ALFA, V. THESE SATISFY THE RELATIONS
BETA*U = A*V - ALFA*U,
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 )

USE A PLANE ROTATION TO ELIMINATE THE DAMPING PARAMETER.
THIS ALTERS THE DIAGONAL (RHOBAR) OF THE LOWER-BIDIAGONAL MATRIX.

RHBAR2 = RHOBAR**2 + DAMPSQ
RHBAR1 = SQRT(RHBAR2)
CS1 = RHOBAR/RHBAR1
SN1 = DAMP/RHBAR1
PSI = SN1*PHIBAR
PHIBAR = CS1*PHIBAR

USE A PLANE ROTATION TO ELIMINATE THE SUBDIAGONAL ELEMENT (BETA)
OF THE LOWER-BIDIAGONAL MATRIX, GIVING AN UPPER-BIDIAGONAL MATRIX.

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

UPDATE X, W AND THE STANDARD ERROR ESTIMATES.

T1 = PHI/RHO
T2 = -THETA/RHO
T3 = ONE/RHO
DO 200 I = 1, N
T  = W(I)
X(I) = T1*T + X(I)
W(I) = T2*T + V(I)
T  = (T3*T)**2
SE(I) = T + SE(I)
DDNORM= T + DDNORM

200 CONTINUE

C
C
C
C
C
C
C
DELTA = SN2*RHO
GAMBAR = -CS2*RHO
RHS  = PHI - DELTA*Z
ZBAR  = RHS/GAMBAR
XNORM = SQRT(XNORM + ZBAR**2)
GAMMA = SQRT(GAMBAR**2 + THETA**2)
CS2   = GAMBAR/GAMMA
SN2   = THETA/GAMMA
Z     = RHS/GAMMA
XNORM = XNORM + Z**2

C
C
C
C
C
C
C
TEST FOR CONVERGENCE.
C
C FIRST, ESTIMATE THE NORM AND CONDITION OF THE MATRIX ABAR,
C AND THE NORMS OF RBAR AND ABAR(TRANSPOSE)*RBAR.

C
C
ANORM = SQRT(BBNORM)
ACOND = ANORM*SQRT(DDNORM)
RES1 = PHIBAR**2
RES2 = RES2 + PSI**2
RNORM = SQRT(RES1 + RES2)
ARNORM = ALFA*ABS(TAU)

C
C
C
C
C
C
C
NOW USE THESE NORMS TO ESTIMATE CERTAIN OTHER QUANTITIES,
C SOME OF WHICH WILL BE SMALL NEAR A SOLUTION.

C
C
TEST1 = RNORM/BNORM
TEST2 = ARNORM/(ANORM*RNORM)
TEST3 = ONE/ACOND
T1   = TEST1/(ONE + ANORM*XNORM/BNORM)
RTOL = BTOL + ATOL*ARNORM/XNORM/BNORM

C
C
C
C
C
C
C
THE FOLLOWING TESTS GUARD AGAINST EXTREMELY SMALL VALUES OF
C ATOL, BTOL OR CTOL. (THE USER MAY HAVE SET ANY OR ALL OF
C THE PARAMETERS ATOL, BTOL, CONLIM TO ZERO.)

C
C
C
C
T3 = ONE + TEST3

232
T2 = ONE + TEST2
T1 = ONE + T1
IF (ITN .GE. ITNLIM) ISTOP = 7
IF (T3 .LE. ONE ) ISTOP = 6
IF (T2 .LE. ONE ) ISTOP = 6
IF (T1 .LE. ONE ) ISTOP = 4
C
ALLOW FOR TOLERANCES SET BY THE USER.
C
IF (TEST3 .LE. CTOL) ISTOP = 3
IF (TEST2 .LE. ATOL) ISTOP = 2
IF (TEST1 .LE. RTOL) ISTOP = 1
C
SEE IF IT IS TIME TO PRINT SOMETHING.
C
IF (NOUT .LE. 0) GO TO 600
IF (M.LE.40 .OR. N.LE.40) GO TO 400
IF (ITN .LE. 10) GO TO 400
IF (ITN .GE. ITNLIM-10) GO TO 400
IF (MOD(ITN,10) .EQ. 0) GO TO 400
IF (TEST3 .LE. 2.0*CTOL) GO TO 400
IF (TEST2 .LE. 10.0*ATOL) GO TO 400
IF (TEST1 .LE. 10.0*RTOL) GO TO 400
GO TO 600
C
PRINT A LINE FOR THIS ITERATION.
C
400 WRITE(NOUT, 1500) ITN,X(1),RNORM,TEST1,TEST2,ANORM,ACOND
C
===MOD(ITN,10) .EQ. 0) WRITE(NOUT, 1600)
C
===STOP IF APPROPRIATE.
C
THE CONVERGENCE CRITERIA ARE REQUIRED TO BE MET ON NCONV
CONSECUTIVE ITERATIONS, WHERE NCONV IS SET BELOW.
SUGGESTED VALUE -- NCONV = 1, 2 OR 3.
C
600 IF (ISTOP .EQ. 0) NSTOP = 0
IF (ISTOP .EQ. 0) GO TO 100
NCONV = 1
NSTOP = NSTOP + 1
IF (NSTOP .LT. NCONV .AND. ITN .LT. ITNLIM) ISTOP = 0
IF (ISTOP .EQ. 0) GO TO 100
C
END OF ITERATION LOOP.
C
FINISH OFF THE STANDARD ERROR ESTIMATES.
C
T = ONE
IF (M .GT. N) T = M - N
IF (DAMPSQ .GT. ZERO) T = M
T = RNORM/SQRT(T)

DO 700 I = 1, N
   SE(I) = T*SQRT(SE(I))
700 CONTINUE

PRINT THE STOPPING CONDITION.

800 IF (NOUT .LE. 0) GO TO 900
   WRITE(NOUT, 1900) ITN, ISTOP
   IF (ISTOP .EQ. 0) WRITE(NOUT, 2000)
   IF (ISTOP .EQ. 1) WRITE(NOUT, 2100)
   IF (ISTOP .EQ. 2) WRITE(NOUT, 2200)
   IF (ISTOP .EQ. 3) WRITE(NOUT, 2300)
   IF (ISTOP .EQ. 4) WRITE(NOUT, 2400)
   IF (ISTOP .EQ. 5) WRITE(NOUT, 2500)
   IF (ISTOP .EQ. 6) WRITE(NOUT, 2600)
   IF (ISTOP .EQ. 7) WRITE(NOUT, 2700)
900 RETURN

1000 FORMAT(
   1 // 25X, 46HLSQR -- LEAST-SQUARES SOLUTION OF A*X = B
   2 // 25X, 18H THE MATRIX A HAS, I6, 11H ROWS AND, I6, 6H COLS
   3 / 26X, 36H THE DAMPING PARAMETER IS DAMP =, 1PE10.2
   4 / 26X, 8H ATOL =, 1PE10.2, 10X, 8H CONLIM =, 1PE10.2
   5 / 26X, 8H BTOL =, 1PE10.2, 10X, 8H ITNLIM =, I10)
1200 FORMAT(/ 3X, 8H ITN, 9X, 4H X(1), 14X, 8H FUNCTION, 7X,
   1 46H INCOMPATIBLE NORM(A) COND(A) /
1300 FORMAT(/ 3X, 8H ITN, 9X, 4H X(1), 14X, 8H FUNCTION, 7X,
   1 46H INCOMPATIBLE NORM(A) COND(A) /
1500 FORMAT(I6, 1PE20.10, 1PE19.10, 1P2E13.3, 1P2E11.2)
1600 FORMAT(I4)
1900 FORMAT(/ 20H NO. OF ITERATIONS =, I6,
   1 8X, 21H STOPPING CONDITION =, I3)
2000 FORMAT(/ 52H THE EXACT SOLUTION IS X = 0.
2100 FORMAT(/ 52H A*X - B IS SMALL ENOUGH, GIVEN ATOL, BTOL
2200 FORMAT(/ 52H THE LEAST-SQUARES SOLN IS GOOD ENOUGH, GIVEN ATOL
2300 FORMAT(/ 52H THE ESTIMATE OF COND(ABAR) HAS EXCEEDED CONLIM
2400 FORMAT(/ 52H A*X - B IS SMALL ENOUGH FOR THIS MACHINE
2500 FORMAT(/ 52H THE LEAST-SQUARES SOLN IS GOOD ENOUGH FOR THIS MACHINE
2600 FORMAT(/ 52H COND(ABAR) SEEMS TO BE TOO LARGE FOR THIS MACHINE)
2700 FORMAT(/ 52H THE ITERATION LIMIT HAS BEEN REACHED)
C END OF LSQR
END

SUBROUTINE NORMZ( N,X,BETA )
   INTEGER N
   REAL X(N),BETA
C NORMZ IS REQUIRED BY SUBROUTINE LSQR. IT COMPUTES THE
C EUCLIDEAN NORM OF X AND RETURNS THE VALUE IN BETA.
C IF X IS NONZERO, IT IS SCALING SO THAT NORM(X) = 1.
C FUNCTIONS AND SUBROUTINES

C BLAS SNRM2, SSCAL

C REAL ONE, SNRM2, ZERO

C ZERO = 0.0

C ONE = 1.0

C BETA = SNRM2( N, X, 1 )

C IF (BETA .GT. ZERO) CALL SSCAL( N, (ONE/BETA), X, 1 )

C RETURN

C END OF NORMLZ

C END
6.3.8  SQRRTT.FOR

function sqrtt (z)    ! assure proper branch
complex  sqrtt, z, cmplx
data  pi / 3.1415926E0 /

x = real (z)
y = aimag (z)
call polar (x,y,r,a,l)    ! [0,2)
if (r.eq.0.0) then
  sqrtt = cmplx (0.0, 0.0)
else
  r = sqrt (r)
a = a*pi*0.5    ! [0, pi]
sqrtt = cmplx (r*cos(a), r*sin(a))
end if
return
end

236
subroutine polar (x, y, r, a, icase)

given: x, y
discern: r, a

where: r>0, x=r*cos(a*pi), y=r*sin(a*pi)

case 1: a " [ 0, 2]

case 2: a " (-1, 1]

case 3: a " [ 0, 2*pi) ! radians

case 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
  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
  r = xx * sqrt (a*a+1.0)
  a = atan (a) /pi
  if (x .lt. 0.0) then
    a = a+1.0
  else if (y .lt. 0.0) then
    a = a+2.0
  end if
end if
1 if (icase .eq. 1) return
2   if (icase .eq. 2) then
3     if (a .gt. 1.0) a=a-2.0
4     return
5    end if
6   if (icase .eq. 3) then ! radians
7     a = a*pi
8   else ! degrees
9     a = a*180.0
10    end if
11  return
12 end
Discern the sequential index \((k)\) for the \((i,j)\) matrix element, where the matrix is symmetric, \(a(i,j) = a(j,i) = a(k)\), but where only the upper/lower triangle is stored, \(k = k(i,j)\).

\(N\) "size of the square matrix, i.e., \(A^{'}\) \(a(i,j)\) is \(N\times N\).

\(L\) "indicates the storage format, 1-4.

```plaintext
function iindex (l,n,i,j)
integer iindex, l,n,i,j
if (l.lt.l. or. n.lt.l .or. i.lt.l .or. j.lt.l .or.
  k .lt. l. or. i.gt.n .or. j.gt.n ) then
  call exit (2) ! 2'error, 4'severe error
end if
mn = min (i,j)
mx = max (i,j)
goto (1,2,3,4,5,6), 1 '

Symmetric Matrix --------------------------------
Lower triangular matrix, \(j\leq i\), stored column by column.
1 iindex = mx + ((n+n-mn)*(mn-1))/2
return

Lower triangular matrix, \(j\leq i\), stored row by row.
2 iindex = mn + ((mx-1)*mx)/2
return

Upper triangular matrix, \(i\leq j\), stored column by column.
3 iindex = mn + ((mx-1)*mx)/2
return

Upper triangular matrix, \(i\leq j\), stored row by row.
4 iindex = mx + ((n+n-mn)*(mn-1))/2
return

Asymmetric Matrix ---------------------------------
Full matrix, stored column by column.
5 iindex = i + (j-1)*n
return

Full matrix, stored row by row.
6 iindex = j + (i-1)*n
return
end
```
6.3.11  ISTIME.FOR

```fortran
function istime (i)  ! dummy argument
  integer istime, iftime, i, it
  logical first
  data first / .true. /

  entry iftime (i)
  if (first) then  ! initial
    first = .false.
    call times (-1, it)  ! initialize cpu clock
  end if
  call times (1, it)  ! query elapsed cpu time (centi-seconds)
  istime = it*10  ! milli-seconds
  return
end
```

240
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

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

if (is.eq.0) then
  do i=1,nh
    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
  write (io,8) is
  stop
end if

if (is.lt.0) then
  status = lib$init_timer (handle(-is))
  if (status.eq.ss$_normal) return
  write (io,6) stop
else
  status = lib$stat_timer (2,it,handle(is))
  if (status.eq.ss$_normal) return
  write (io,7)
stop
end if

6 format ('times, error: clock initialization problem')
7 format ('times, error: clock evaluation problem')
8 format ('times, error: attempting use of non-existing' & ', clock #', i3)
end
7. Acknowledgments

It is the author's pleasure to acknowledge Drs. Stanley Ruthberg and George Candela for having suggested and supported this project. On a melancholy note, we mourn the loss of Dr. Ruthberg, whose untimely death occurred during the course of the work which led to the preparation of this manuscript. His vitality and friendship are missed. On a much happier note, 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. The author also appreciates the discussions with M. Carroll Croarkin regarding things statistical.
8. References


**Semiconductor Measurement Technology: A Software Program for Aiding the Analysis of Ellipsometric Measurements, Simple Models**

J. P. Marchiando

**NATIONAL BUREAU OF STANDARDS**
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

**SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS**
National Institute of Standards and Technology
U.S. Department of Commerce
Gaithersburg, MD 20899

**ABSTRACT**
MAIN1 is a software program for aiding the analysis of ellipsometric measurements. MAIN1 consists of a suite of routines written in FORTRAN that are used to invert the standard reflection ellipsometric equations for simple systems. Here, a system is said to be simple if the solid material sample may be adequately 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 dielectric function within each layer/substrate is isotropic, homogeneous, local, and linear. Each layer is characterized in part by a thickness ($b$), while the optical properties for a given material and wavelength are expressed in terms of a refractive index ($n$) and extinction coefficient ($k$). The ellipsometric equations are formulated as a standard damped nonlinear least-squares problem and then solved by an iterative method when possible. Estimates of the uncertainties associated with assigning numerical values to the model parameters are calculated as well.

**KEY WORDS**
- ellipsometry
- FORTRAN
- measurement
- modeling
- program
- sensitivity
- software
- uncertainty

**AVAILABILITY**
- Unlimited
- Order From National Technical Information Service (NTIS), Springfield, VA 22161

**NO. OF PRINTED PAGES**
252

**PRICE**
- Unlimited
- For Official Distribution. Do Not Release to NTIS
- Order From National Technical Information Service (NTIS), Springfield, VA 22161
NIST Technical Publications

Periodicals

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 20036.

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 following NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.


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.