AD

TECHNICAL REPORT ARCCB-TR-01015

IN SITU MEASUREMENT OF GROWING THIN FILMS BY ENERGY DISPERSIVE X-RAY REFLECTIVITY: THEORY AND EXPERIMENTAL DESIGN

JASON SUMMERS
TOH-MING LU
DONALD WINDOVER
SABRINA LEE

JULY 2001

US ARMY ARMAMENT RESEARCH,
DEVELOPMENT AND ENGINEERING CENTER
CLOSE COMBAT ARMAMENTS CENTER
BENÉT LABORATORIES
WATERVLIET, N.Y. 12189-4050

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The use of trade name(s) and/or manufacturer(s) does not constitute an official endorsement or approval.

DESTRUCTION NOTICE

For classified documents, follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19, or DoD 5200.1-R, Information Security Program Regulation, Chapter IX.

For unclassified, limited documents, destroy by any method that will prevent disclosure of contents or reconstruction of the document.

For unclassified, unlimited documents, destroy when the report is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 2001	Final

4. TITLE AND SUBTITLE
IN SITU MEASUREMENT OF GROWING THIN FILMS BY ENERGY DISPERSIVE X-RAY REFLECTIVITY: THEORY AND EXPERIMENTAL DESIGN

5. FUNDING NUMBERS
AMCMS No. 7780.45.E251.2

6. AUTHOR(S)
Jason Summers (RPI, Troy, NY), Toh-Ming Lu (RPI),
Donald Windover (Benet and RPI), and Sabrina Lee

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC
Benet Laboratories, AMSTA-AR-CCB-O
Watervliet, NY 12189-4050

8. PERFORMING ORGANIZATION REPORT NUMBER
ARCCB-TR-01015

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC
Close Combat Armaments Center
Picatinny Arsenal, NJ 07806-5000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A method for the in-situ measurement of thin-film growth by x-ray reflectivity is described. Description of the underlying theory is given. Real-time characterization of thickness and roughness variation during growth is possible via this technique. A deposition chamber for the implementation of these measurements has been designed and constructed. This apparatus is described, along with software developed for the purpose of data acquisition and analysis.

14. SUBJECT TERMS
In-Situ, Real-Time, Thickness, Roughness, Growth, Thin Films, X-Ray Reflectivity

15. NUMBER OF PAGES
58

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Introduction

In situ measurement of thin film growth is of great interest to both scientists and engineers. For the scientist, real-time measurement of film properties within the deposition chamber enables the study of evolution and growth processes. For the engineer, such techniques offer greater process control and, ultimately, improve devices. Among those methods which can adapted for *in situ* measurement, x-ray reflectivity is particularly well suited to the characterization of thin films. X-ray reflectivity can recover density, thickness, and roughness information from layers and multilayers through bulk reflection and interference effects. It is equally capable of measuring metals and dielectrics, unlike ellipsometry, and the short wavelength of the probing radiation enables the measurement of films having thicknesses as low as a few nanometers [1]. In addition, x-rays are insensitive to environment, making x-ray reflectivity suitable for use in sputtering chambers, unlike electron-beam techniques [2].

Conventionally, x-ray reflectivity measurements have probed k-space by varying the incident angle of a monochromatic beam in an angular dispersive technique. This methodology has been difficult to integrate into deposition systems by virtue of the expense and time delay inherent to high precision goniometer scans. The system developed as part of this study circumvents these concerns by utilizing an energy dispersive technique. A fixed beam geometry is employed and k-space probed by photons having a range of incident energies. Recent work by Chason [3] and Kellerman [4] has shown the suitability of this technique for use *in situ*. At present, work done on the measurement of growing films has concentrated on the use of single energies and fixed angles to produce "deposition curves"- variations in reflectivity with film growth[5,6,7]. Energy dispersive technique expands on these methods to produce a standard reflectivity curve which evolves in time.

In order to study the use of energy dispersive x-ray reflectivity for *in situ* characterization, a specialized deposition and measurement system has been developed. The system makes use of an existing x-ray diffractometer through a demountable vacuum deposition chamber, computer based acquisition card, and custom analysis software.

Theory of X-ray Reflectivity

Optical means of material characterization differ from one another in two aspects: the first is the energy of the incident photons used and the second is the geometry chosen for the positioning of source and detector. In the technique of x-ray reflectivity, information is gathered by photons having energies in the range of 1 to 100 KeV that are directed at the material at small grazing angle and detected in the region of specular reflection. The interaction of materials with radiation in the x-ray region of the spectrum is quite different from that in the visual region. Due to their high frequency, x-rays do not interact with nuclei but instead are sensitive to electron density [8]. Moreover, the high energies involved allow for absorption due to electronic ionization. Both of these properties are taken into account by writing the index of a material as $n = 1 - \delta - i\beta$. In this expression, δ determines the phase change of the wave and is directly related to electron density, while β determines the x-ray absorption. The two constants are related to the material properties by

$$\delta = \frac{\lambda^2 \rho r_0}{2\pi} \text{ and } \beta = \frac{\lambda}{4\pi}\mu(\lambda)$$

where ρ is the electron density, r_0 is the classical electron radius, and $\mu(\lambda)$ is the linear absorption coefficient

Both δ and β are small positive values. Thus, for most materials, the index of refraction for x-rays will be slightly less than one. This would appear problematic due to the suggestion of a wave speed greater than the speed of light in vacuum, c. However, while the index determines the phase velocity of the wave, the group velocity (the speed at which energy is transported) remains less than c. [8] Because the refractive index is less than one, x-rays passing from air into a material can experience total external reflection over a range of incident grazing angles between 0 and some critical angle, θ_c. The critical angle can be calculated from Snell's law, which is written as

$$n_1 \cos\theta_1 = n_2 \cos\theta_2$$

for angles measured from the material surface. In general, β is smaller than δ by an order of magnitude. Therefore, in calculating the critical angle, the index of a material can be

taken as $n = 1 - \delta$. Using the Taylor expansion for the cosine, the critical angle can be related to the electron density of the material.

$$\cos\theta_c = 1-\delta$$
$$1-\frac{\theta_c^2}{2} \cong 1-\delta$$
$$\theta_c = \sqrt{2\delta} \propto \sqrt{2\rho}$$

From this expression it is clear that denser materials will have a higher critical angle. For incident angles less than θ_c the incident and reflected intensities will be equal. At angles greater than θ_c some portion of the light will enter into the material. This relation between the incident and reflected intensity is known as the reflectivity

$$R = \frac{I_{reflected}}{I_{incident}}.$$

In the simple case described here, the reflectivity plot for a material would be a step function: equal to one at angles below θ_c and dropping rapidly at angles larger than θ_c. In actual materials, absorption smoothes this transition causing a gradual drop in reflected intensity in the region of the critical angle. Beyond this region, the reflectivity can be expressed in terms of the incident and critical angles by

$$R \cong \left(\frac{\sin\theta}{\sin\theta_c}\right)^{-4}.$$

To express the reflectivity of a material in greater detail, Fresnel theory must be used. Under the assumption that the atomic granularity of materials relative to x-ray wavelengths can be neglected due to the relatively large penetration depth [9], Fresnel theory for homogenous media can be used to relate the electric field amplitudes of the incoming and reflected waves. [10] The scenario is depicted in *Figure 1*. For the s polarization case the Fresnel reflection coefficient is written as

$$r = \frac{E_R}{E_0} = \frac{n_0 \sin\theta_0 - n_1 \sin\theta_1}{n_0 \sin\theta_0 + n_1 \sin\theta_1} = \frac{k_0^{\perp} - k_1^{\perp}}{k_0^{\perp} + k_1^{\perp}}$$

where $k^{\perp}$ is the perpendicular component of the wave vector as defined by

$$k_j^{\perp} = \frac{2\pi}{\lambda_0} n_j \sin\theta_j = \frac{2\pi}{\lambda_j} \sin\theta_j.$$

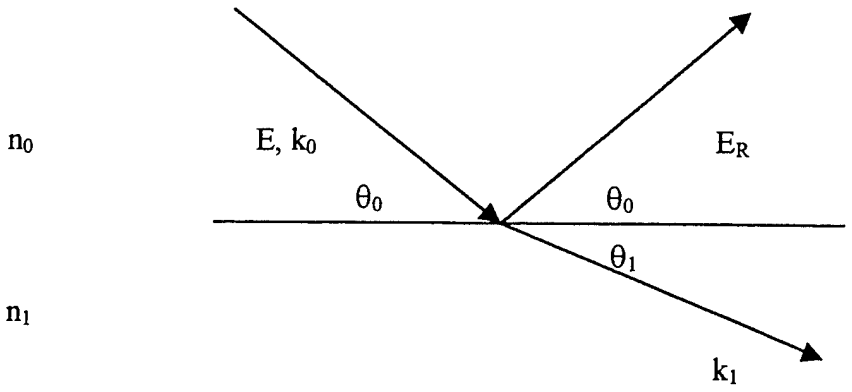

Figure 1. A schematic representation of x-rays incident on a single layer of index n_1.

The reflectivity is calculated from the Fresnel coefficient by taking the product of the coefficient with its complex conjugate

$$R = rr^* .$$

Formal calculation in the case of unpolarized light requires that the reflectivity be evaluated as the average of the reflectivities of s and p polarization. However, for calculations involving short wavelengths and small angles, the effects of polarization are negligible and the s polarization equations will suffice for all cases. [11]

For the condition in which x-rays are incident from air or vacuum, the reflectivity from a single layer can be rewritten in terms of the incident angle and the real and complex parts of the index [12]. The first step in the this reduction is the application of a trigonometric identity

$$n_1 \sin\theta_1 = n_1\sqrt{1-\cos^2\theta_1} .$$

This becomes

$$= \sqrt{n_1^2 - n_0^2\cos^2\theta_0}$$

by application of Snell's law. Expanding the cosine and neglecting terms second order or higher in δ and β yields

$$= \sqrt{\theta_0^2 - 2\delta_1 - 2i\beta_1} .$$

Using this expression and replacing $\sin\theta_0$ with θ_0 by the small angle approximation, the reflection coefficient is now written as

$$r_{0,1} = \frac{E_R^0}{E_T^0} = \frac{\theta_0 - \sqrt{\theta_0^2 - 2\delta_1 - 2i\beta_1}}{\theta_0 + \sqrt{\theta_0^2 - 2\delta_1 - 2i\beta_1}}$$

From this form of the reflection coefficient, the reflectivity can be plotted as a function of the ratio β / δ, as shown in *Figure 2*.

Multilayer Reflectivity

In films which are sufficiently thin, interference fringes can be produced in the reflectivity curve by the interaction of the reflections from the air/film interface and the film/substrate interface, as first observed by Kiessig [13].

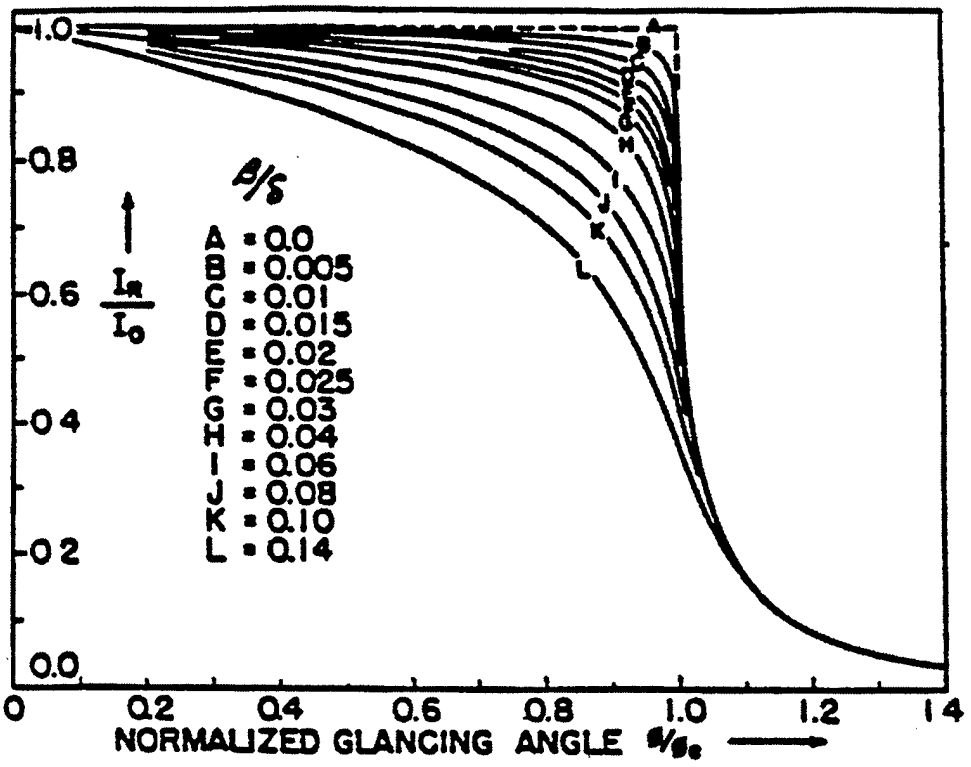

Figure 2. A plot of reflectivity versus the normalized glancing angle, θ/θ_c, as a function of β/δ. As the ratio increases, the drop in reflectivity in the region of critical angle becomes progressively more gradual.

An example of this is shown in *Figure 3*. The spacing of the fringes is directly related to the thickness of the film layer. This is expressed in terms of the Bragg condition,

$$2d\theta = \Delta\lambda = \frac{2\pi}{\Delta k^{\perp}}\theta ,$$

which yields,

$$d = \frac{\pi}{\Delta k^{\perp}} .$$

For the simple three layer interface of air, film, and substrate depicted in *Figure 4,* one can define two sets of Fresnel coefficients. One for the air/film interface and a second for the film/substrate interface. Considering s polarization only, one has for the first layer

$$r_{0,1} = \left(\frac{E_R^0}{E_T^0}\right)_s = \frac{n_0 \sin\theta_0 - n_1 \sin\theta_1}{n_0 \sin\theta_0 + n_1 \sin\theta_1} = \frac{k_0^{\perp} - k_1^{\perp}}{k_0^{\perp} + k_1^{\perp}},$$

$$t_{0,1} = \left(\frac{E_T^1}{E_T^0}\right)_s = \frac{2n_0 \sin\theta_0}{n_0 \sin\theta_0 + n_1 \sin\theta_1} = \frac{2k_0^{\perp}}{k_0^{\perp} + k_1^{\perp}},$$

and for the second layer

$$r_{1,2} = \left(\frac{E_T^1}{E_R^1}\right)_s = -r_{0,1} = \frac{k_2^{\perp} - k_1^{\perp}}{k_2^{\perp} + k_1^{\perp}},$$

$$t_{1,2} = \left(\frac{E_R^0}{E_R^1}\right)_s = \frac{n_2 \sin\theta_2}{n_1 \sin\theta_1} t = \frac{k_2^{\perp}}{k_1^{\perp}} t .$$

The reflected field for the entire film will be a superposition of the reflections from both boundary layers with the field from the film/surface interface being phase shifted by an amount

$$\phi = \frac{2\pi}{\lambda_0} n_1 d \sin\theta_1 = k_1^{\perp} d .$$

The electric field amplitudes are then written as

$$E_R^0 = r_{0,1} E_T^0 + e^{-i\phi} t_{1,2} E_R^1 \text{ and}$$

$$e^{-i\phi} E_T^1 = t_{0,1} E_T^0 + e^{i\phi} r_{1,2} E_R^1 .$$

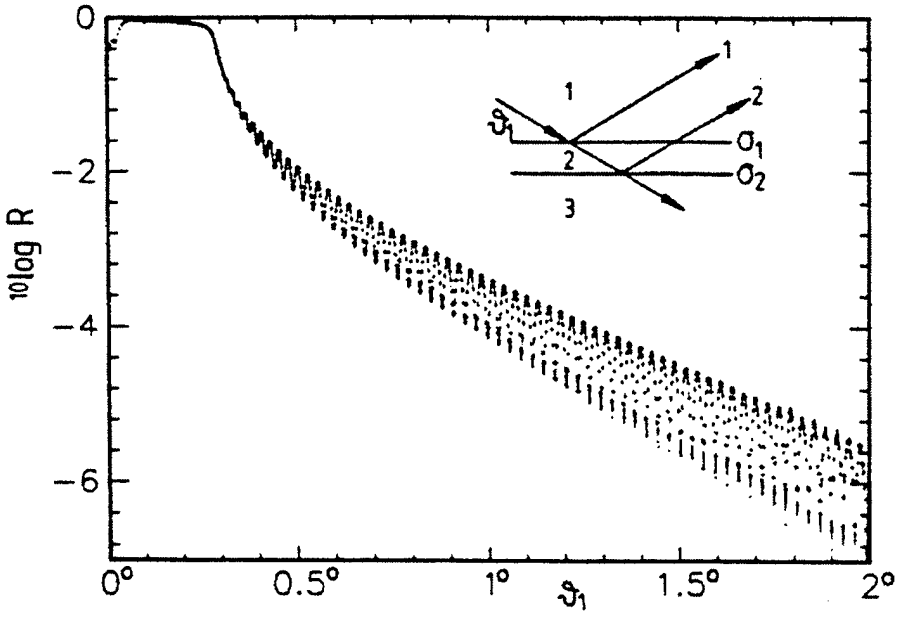

Figure 3. A plot of reflectivity versus glancing angle illustrating thickness oscillations.

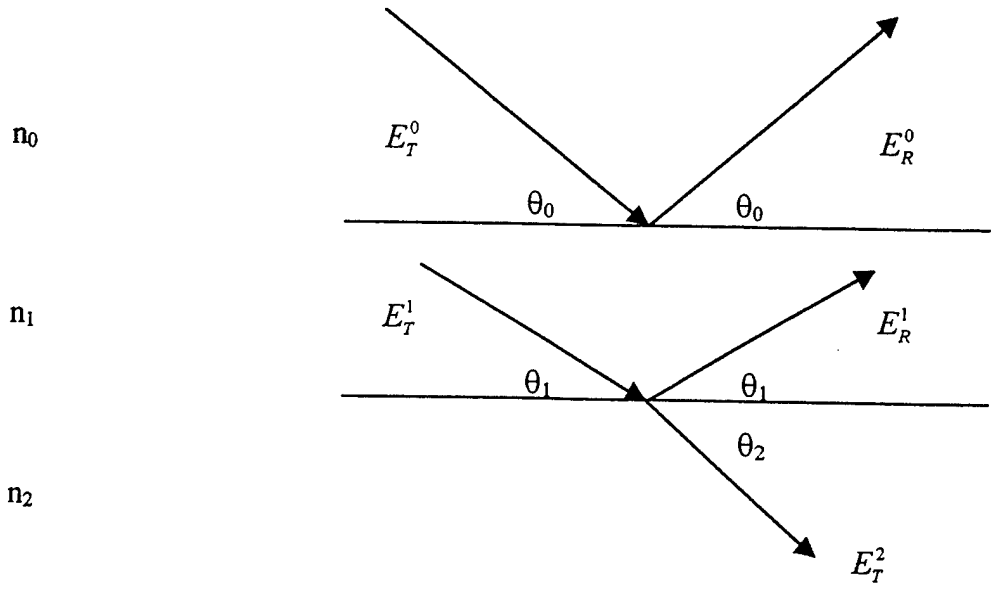

Figure 4. A three layer interface. n_0 is air, n_1 the film, and n_2 the substrate.

By rearranging these expressions the total reflectivity of the layers can be expressed in terms of a transfer matrix, of the type commonly used for multilayer optical coatings. [14]

$$\begin{bmatrix} E_T^{j-1} \\ E_R^{j-1} \end{bmatrix} = M_{j-1,j} \begin{bmatrix} E_T^{j} \\ E_R^{j} \end{bmatrix}$$

The matrix of transmission and reflection coefficients may be expressed most generally in terms of the perpendicular component of the incident wave vector, $k^{\perp}$. The transfer matrix is written as the product of the Fresnel coefficient matrix, F, and the propagation matrix, P.

$$M = FP,$$

where

$$F = \frac{1}{2} \begin{bmatrix} 1 + \frac{k_{j-1}^{\perp}}{k_j^{\perp}} & 1 - \frac{k_{j-1}^{\perp}}{k_j^{\perp}} \\ 1 - \frac{k_{j-1}^{\perp}}{k_j^{\perp}} & 1 + \frac{k_{j-1}^{\perp}}{k_j^{\perp}} \end{bmatrix},$$

and

$$P = \begin{bmatrix} \exp\left(ik_j^{\perp} d\right) & 0 \\ 0 & \exp\left(-ik_j^{\perp} d\right) \end{bmatrix}.$$

$k^{\perp}$ in each of the layers can be calculated from the incident wave vector and the index in that layer according to

$$k_j^{\perp} = \frac{2\pi}{\lambda_0} n_j \sin\theta_j \cong \frac{2\pi}{\lambda_0} \sqrt{\theta_0^2 - 2\delta_j - 2i\beta_j}\,.$$

Films of graded density can be modeled as a series of layers of varying index, as shown in *Figure 5*. Multilayer reflectivity is calculated using the product of the transfer matrices for each of the of the layers

$$\begin{bmatrix} E_T^0 \\ E_R^0 \end{bmatrix} = \prod_{j=1}^{N+1} M_{j-1,j} \begin{bmatrix} E_T^{N+1} \\ E_R^{N+1} \end{bmatrix}.$$

A solution is found by setting E_R^{N+1} to zero under the assumption that the thickness of the substrate will prevent a reflection from the lower-most interface. It is then convenient to define reflection and transmission coefficients for the entire system

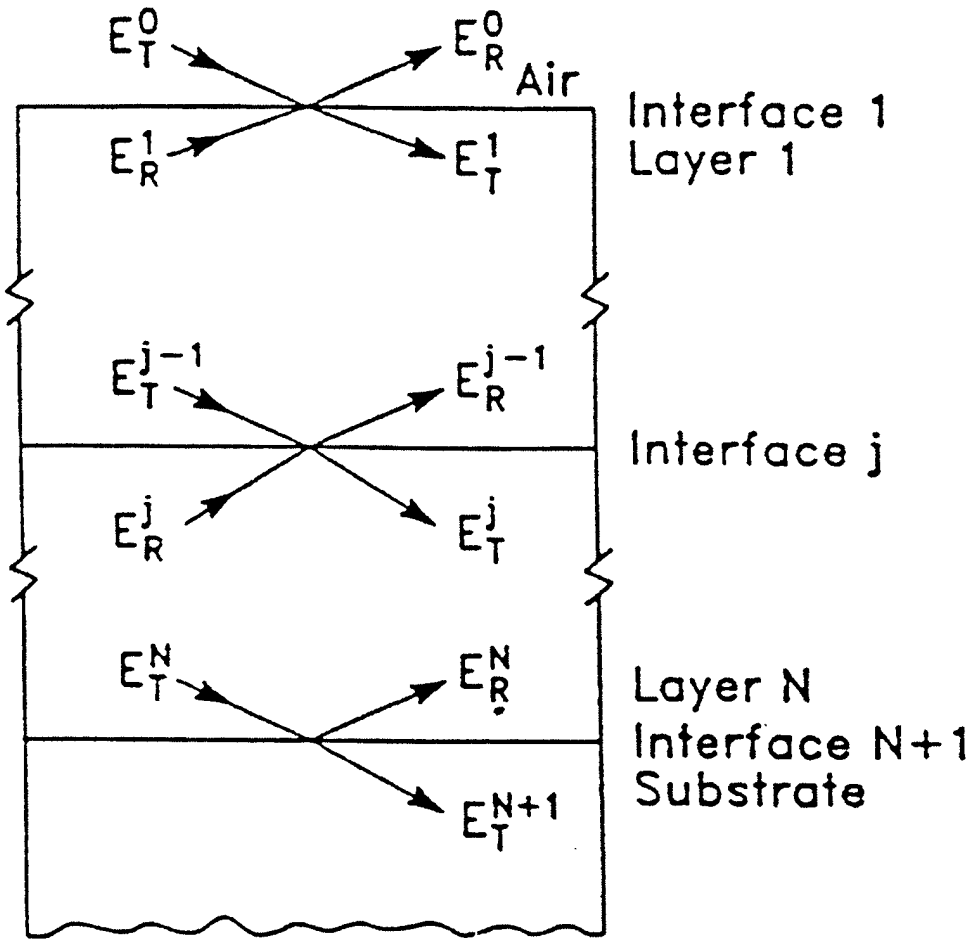

Figure 5. A figure depicting a film composed of N layers on a substrate. Air is the 0 layer and the substrate is the N+1 layer.

$$r = \frac{E_R^0}{E_T^0} \quad \text{and} \quad t = \frac{E_T^{N+1}}{E_T^0}.$$

Using these the matrix expression can be rewritten as

$$\begin{bmatrix} 1 \\ r \end{bmatrix} = \Gamma \begin{bmatrix} t \\ 0 \end{bmatrix},$$

where

$$\Gamma = \prod_{j=1}^{N+1} M_{j-1,j}.$$

The reflection and transmission for the layer is written in terms of the elements of the transfer matrix for the system,

$$r = \frac{\Gamma_{21}}{\Gamma_{11}}, \; R = rr^*,$$

and

$$t = \frac{1}{\Gamma_{11}}, \; T = tt^*.$$

Reflectivity from Imperfect Interfaces

In the preceding analysis, ideal interfaces between the various media have been assumed. However, in actual materials, the transition between two regions having different densities will occur gradually, as shown in *Figure 6*. The variation in index across an interface may be expressed in terms of the error function

$$n(z) = n_1 + (n_2 - n_1) Erf(z, \sigma), \; [15]$$

where the error function is defined by

$$Erf(z, \sigma) = \left(\sigma\sqrt{2\pi}\right)^{-1} \int_{-\infty}^{z} \exp\left(-\xi^2 / 2\sigma^2\right) d\xi.$$

σ in the above expressions represents the standard deviation of the Gaussian distribution and can be viewed as the rms interface width of the transition between n_1 and n_2, as *Figure 6* illustrates.

As a result of these imperfect interfaces, scattering of the incident and reflected x-rays will occur. The result of this scattering at buried interface layers is a reduction in the

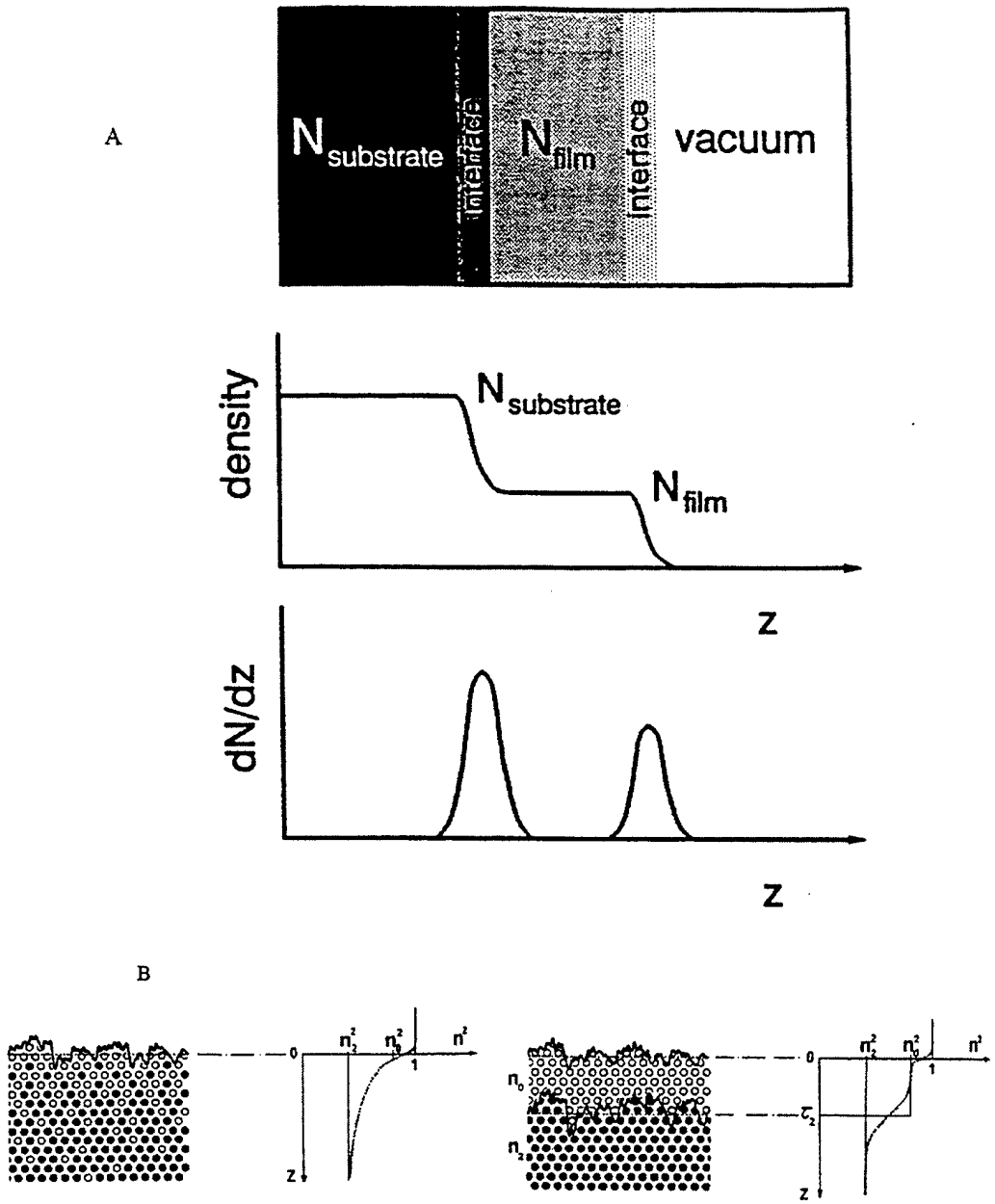

Figure 6. (A) Schematic representation of the density variation in a simple three-layer model, and (B) illustration of the translation between a real interface or region of graded density and the error function models of roughness used.

amplitude of the interference fringes. Scattering from the surface of the film increases the rate of fall off of the reflectivity curve, in addition to damping the interference fringes. This is shown in *Figure 7*. [16]

Two models for the scattering at interfaces can be considered. For both cases the magnitude of the reflected component is reduced due to the imperfect interface. In the first model this is assumed to produce a global energy loss due to scattering. The second model assumes that no energy is lost and that, rather, the transmission through each given interface is correspondingly increased [15].

The loss of reflectivity is incorporated in the model through a modification of the Fresnel reflection coefficients. Considered in terms of scattering formalism [17], the correction is given by the Fourier transform of the density gradient across the interface.

$$\rho'(z) = \left(\sigma\sqrt{2\pi}\right)^{-1} \exp\left(-z^2/2\sigma^2\right)$$

$$\rho'(k) = \int_{-\infty}^{\infty} p'(z) e^{2ikz} dz = \exp\left(-4\sigma^2 k^2/2\right)$$

which is equivalent to the well known expression for the Debye-Waller factor

$$D_{i,j} = \left(-16\pi^2\sigma^2 \sin^2\theta_i/\lambda_i^2\right).$$

Due to assumptions implicit in scattering formalism, this expression is only valid for $\theta >> \theta_c$. Nevot *et. al.* [17] have developed more general roughness correction factors for interfaces having predominately high spatial frequencies,

$$D_{i,j} = \exp\left(-2\sigma^2 k_i^{\perp} k_j^{\perp}\right),$$

and for those having predominately low spatial frequencies,

$$D_{i,j} = \exp\left(-2\sigma^2 \left(k_i^{\perp}\right)^2\right).$$

For this study, the factor for high spatial frequencies is used. It is included in the multilayer model by modifying the form of the Fresnel portion of the transfer matrix such that the roughness factor multiplies the off-diagonal terms.

$$F = \frac{1}{2}\begin{bmatrix} 1+\frac{k_{j-1}^{\perp}}{k_j^{\perp}} & 1-\frac{k_{j-1}^{\perp}}{k_j^{\perp}}\exp\left(-2\sigma^2 k_{j-1}^{\perp} k_j^{\perp}\right) \\ 1-\frac{k_{j-1}^{\perp}}{k_j^{\perp}}\exp\left(-2\sigma^2 k_{j-1}^{\perp} k_j^{\perp}\right) & 1+\frac{k_{j-1}^{\perp}}{k_j^{\perp}} \end{bmatrix}$$

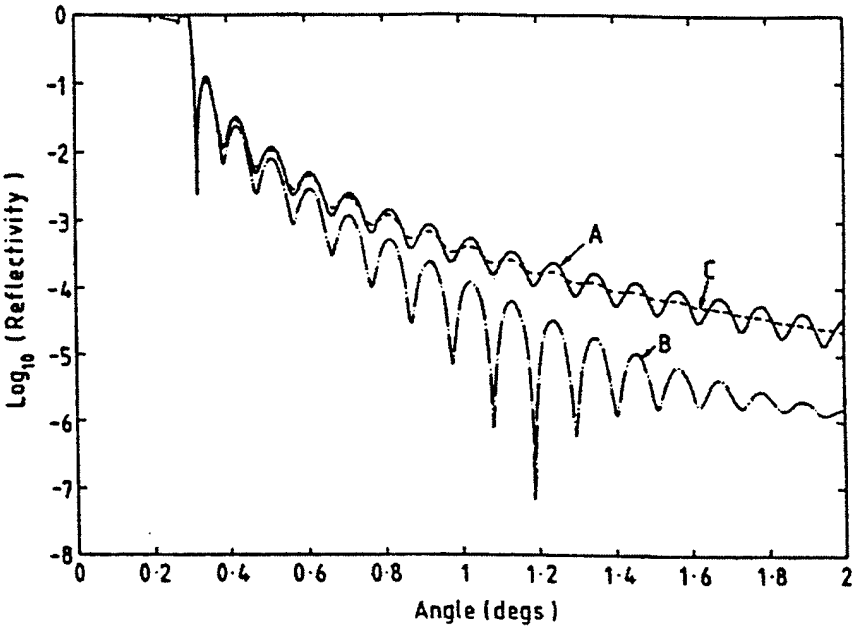

Figure 7. A plot of reflectivity for a series of similar 40 nm films showing the effect of surface and interface roughness. Curve A has no surface or interface roughness. Curve B has an interface width on the surface of 1 nm. Curve C has an interface width of 1 nm at the film/substrate interface.

Assuming the roughness model which allows for global energy loss, this is the only correction necessary.

Energy Dispersive Reflectivity

In the previous analysis, the reflectivity of a multilayer has been expressed as a function of the component of the propagation vector perpendicular to the film surface, $k^{\perp}$. As is evident from the expression defined earlier,

$$k_j^{\perp} = \frac{2\pi}{\lambda_0} n_j \sin\theta_j \cong \frac{2\pi}{\lambda_0}\sqrt{\theta_0^2 - 2\delta_j - 2i\beta_j},$$

$k^{\perp}$ is a function of both the wavelength of the incident beam, λ_0, and the angle of incidence, θ_0. Conventional practice has followed after the initial work of Kiessig [13] and Parratt [9], both of whom measured reflectivity by scanning through θ with a monochromatic beam and plotting the intensity of the specular reflection measured at 2θ. However, it is possible to gather the same information by measuring the intensity as a function of wavelength using a fixed angle, specular geometry. In this method, first used by Bilderback, *et al.* [18] to analyze the critical angle of x-ray mirrors, the incident and outgoing angles are fixed and the energy spectrum of a white source is profiled. Dhez, *et. al.* [19] have illustrated the relationship of these two techniques graphically in *Figure 8*. A comparison of two reflectivity curves of the same sample made by the angular dispersive and energy dispersive techniques is shown in *Figure 9* [1].

While the energy dispersive technique allows for a simplified mechanical apparatus, analysis of energy dispersive data presents additional challenges. The index of materials for x-rays has been expressed in terms of real and imaginary components, δ and β. Each of these has a dependence on the wavelength of the x-rays. The behavior of δ and β can be predicted by the classical dispersion laws

$$\delta = \frac{\lambda^2 \rho r_0}{2\pi} = \frac{e\lambda}{2\pi mc} N_{avo} \rho \sum_a Z + f \Big/ \sum_a A,$$

and

$$\beta = \frac{\lambda}{4\pi}\mu(\lambda) = \frac{e\lambda}{2\pi mc} N_{avo} \rho \sum_a i * f \Big/ \sum_a A,$$

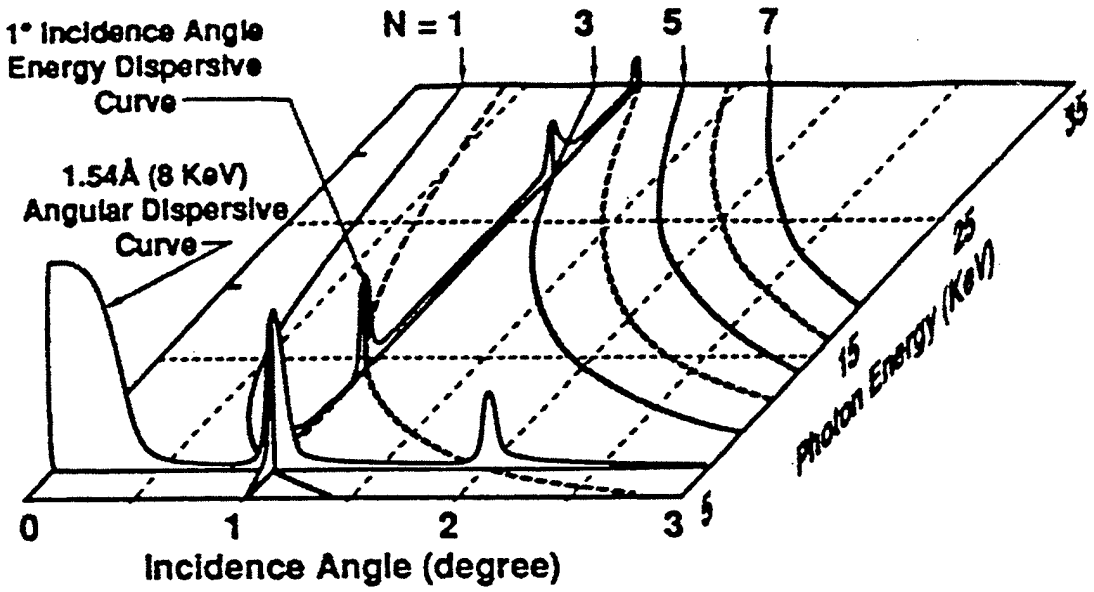

Figure 8. Graph illustrating the relationship between energy dispersive and angular dispersive methods of measurement. The reflectivity of a periodic multilayer is depicted as a function of both angle and energy. The maximum intensity of a given order of Bragg peak will always lie along the line drawn in the horizontal plane.

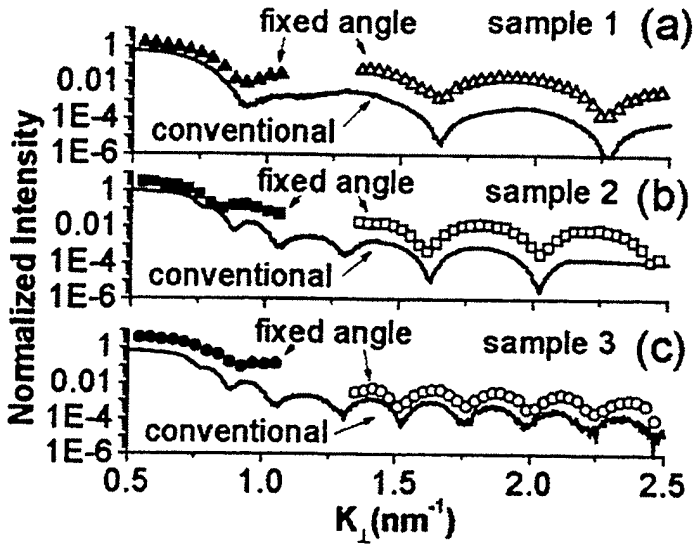

Figure 9. Angular-dispersive reflectivity compared with fixed angle results for the three tantalum on silicon samples. Curve (a) gives normal reflectivity data (solid curve) compared with 0.6° 2θ fixed angle data (closed triangles) and 1.5° 2θ fixed angle data (open triangles) for sample 1. Curve (b) gives the normal reflectivity data (solid curve) compared with 0.6° 2θ fixed angle (closed squares) and 1.5° 2θ fixed angles (open squares) for sample 2. Curve (c) shows the normal reflectivity (solid curve) versus the 0.6° 2θ fixed angle (closed circles) and the 1.5° 2θ fixed angle (open circles) information for sample 3. By varying the incident angle, the limited range of incident energies can be used to profile the entire reflectivity curve. Often, however, only a range of the curve is necessary for analysis.

where f' is the real dispersion correction (due to the scattering cross-section) and f'' is the imaginary dispersion correction. However, extrapolating the index based on these formulas is only effective for small changes in λ. The indices may also may computed from f' and f'' listed in tables; however, current tables do not extend beyond 10 keV. In practice the values are predicted from theory, but fit experimentally as two additional free parameters.

Reflectivity from Growing Films

For a growing film thickness, density, and surface roughness will be evolving in time. Modeling such a system as a single layer bound by atmosphere and substrate layers, as shown in Figure 4, a total reflection coefficient is written as,

$$r = \frac{E_R^0}{E_T^0} = \frac{r_{0,1}D_{0,1} + r_{1,2}D_{1,2}\exp\left(2idk_1^{\perp}\right)}{1 + r_{0,1}r_{1,2}D_{0,1}D_{1,2}\exp\left(2idk_1^{\perp}\right)} \text{ [5].}$$

For regions of the curve far from the critical angle the Fresnel coefficients take on the asymptotic form

$$r_{i,j} \cong \frac{\left|n_i - n_j\right|}{4\sin^2\theta} \text{ [5]}.$$

If the average index of the film is assumed to be constant, the reflectivity of the film can be considered a function of d, σ, and $k^{\perp}$. Reflectivity measured using fixed angle geometry will evolve along with d and σ. As the layer thickens, the reflectivity will vary sinusoidally as the wave reflected from the film surface and the wave reflected from the substrate pass in and out of the constructive interference condition. Assuming that β can be neglected in the expression for the index, the reflectivity can be expressed in terms of the Bragg relation

$$\mathrm{m}\,R(t) = 2T\left[1 - \frac{\delta}{\sin^2\theta}\right] \text{[5],}$$

where T is the spatial period of the oscillations and m is order of interference. As the film thickens it roughens, damping the amplitude of the peaks in the reflectivity plot. This growth in roughness can be modeled as

$$w(d) \cong t^{\beta}\ [20],$$

where β ranges between .2 and 1. Baranov et. al. [5], Lee et. al. [6], and Heilman et. al. [7] have demonstrated this technique for evaporated and sputtered films. An experimental curve for the growth of aluminum by evaporation is shown in *Figure 10.*

A primary concern of using x-ray reflectivity for the measurement of growing thin film is the time averaging which will occur as a result of the finite collection time required to generate a cure. The reflectivity of a sample will vary as it thickens and roughens. For a collected curve to be meaningful, the variation in the reflectivity, ΔR, must be small over the collection time, Δt, relative to the features of the curve which are of interest. For this reason, angular dispersive technique is not feasible. The fixed angle technique described is far more feasible because only x-ray intensity and detector efficiency extend the measurement time. Nonetheless, it is desirable to produce conventional reflectivity curves of the growing film. Using energy dispersive methods, a reflectivity curve could be constructed for each data point on the curve of a fixed angle growth oscillation. However, what must be shown is that ΔR will be sufficiently small over Δt for useful curves to be generated. In recent experiments, a measurement time of ten seconds has been shown to produce a usable curve, though 300 seconds is preferred for maximum resolution as shown in *Figure 11.* [1]. To investigate this a series of models were created using spreadsheet based x-ray reflectivity simulation software. A layer of tantalum on a silicon wafer was considered with an interface width between film and substrate of 5 nm. The average rms percentage errors between the log reflectivities of the initial and final states of various growth conditions were calculated. A typical growth rate for work with growth oscillations is .2 nm min^{-1} [6] . For a growth of 1.5 nm (450 seconds measurement time at this growth rate) and an increase in roughness given by β = 1 of a 4.0 nm film with initial interface width of 3.0 nm the average error was .2%. For a growth of .5 nm (150 seconds) the error reduced to .07%. Because thickness fringes are often a very small effect, a slow growth time coupled with a fast measurement time may be necessary. A .1 nanometer growth during a thirty second measurement time results in an error of only .01% which is largely indistinquishable to the eye in the semilog plot shown in *Figure 12.* Using the fastest collection time of ten seconds, this allows for minimal time smearing effects at growth rates up to .6 nm min^{-1}. In samples

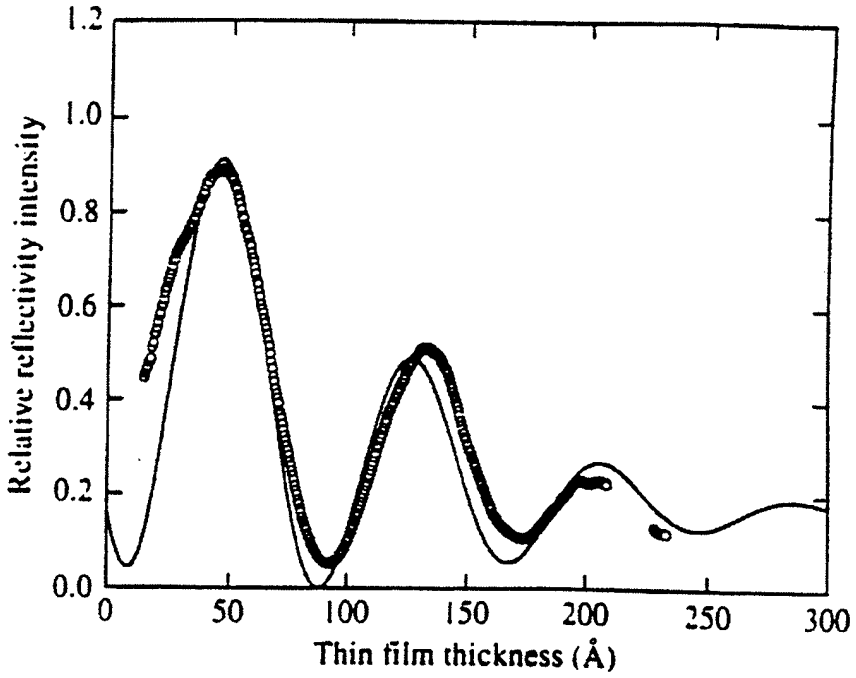

Figure 10. Growth oscillation for evaporated aluminum. The circles indicate the reflectivity measured as a function of the film thickness. The solid line denotes reflectivity calculated from a three-layer model that is assumed to have surface roughness proportional to the film thickness.

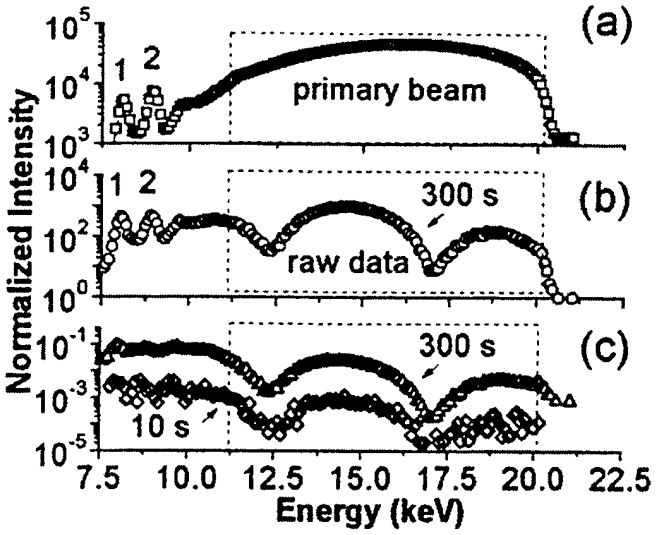

Figure 11. Fixed-angle energy dispersive x-ray reflectivity for a thin tantalum film (~8.5 nm) on silicon. The scan was taken with $2\theta = 1.5°$ with copper source at 20 kV and 20 mA. Curve (a) represents the energy spectrum of the primary beam at $2\theta = 0°$ for 300 s. Curve (b) is the raw energy data taken at $2\theta = 1.5°$ for 300 s. Curve (c) is the normalized data, (b) divided by (a), showing the "Kiessig curves" or thickness oscillations for the tantalum film: triangles showing 300 s of collection time, diamonds 10 s of collection time.

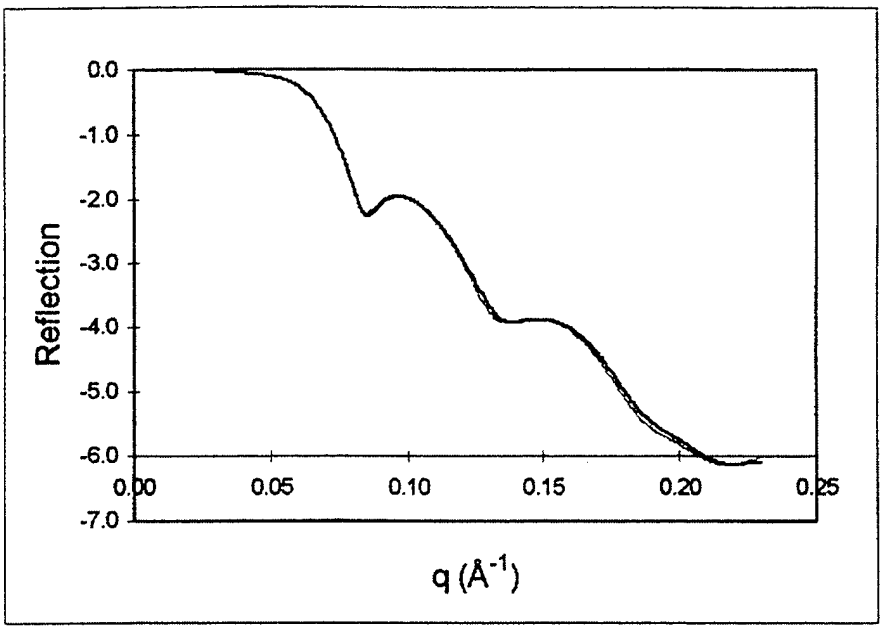

Figure 12. A theoretical plot of the reflectivity curves of two samples differing in thickness and roughness. This is representative of a growth time of 30 seconds for a growth rate of .2 nm min^{-1}. The average rms error between the two curves is .01%.

having less of a difference between the indices of the film and substrate the amplitude of the thickness oscillations will be reduced requiring lower deposition rates.

Data Analysis

Information may be derived from a reflectivity curve by a number of methods. As has been described, the particular features of a curve are related to particular properties of the film which produced that curve. One class of methods uses this fact to rapidly determine a single film property by evaluating a single aspect of the reflectivity curve. A second class of methods evaluates all of the film properties by minimizing the fitting error between a complete model of the reflectivity curve and the data from the experimental curve. Both forms of analysis will be used in this study. For rapid assessment of film thickness a Fourier transform method is used. More complete analysis is made through use of a least-squares fit to the transfer matrix model.

Because layers of different density can give rise to oscillations in the reflectivity curve whose period is related to the thickness of those layers, it can be expected that the Fourier transform can be used to recover thickness information. However, the intensity variations and imperfect periodicity inherent in a reflectivity curve complicate this analysis. Therefore, data must undergo a preliminary transformation to restore its periodicity.

For purposes of illustration, the reflectivity curve may be represented as

$$R = A + B\cos 2d\left(\left(k^{\perp}\right)^2 - \left(k_c^{\perp}\right)^2\right)^{1/2} \quad [21]$$

where A and B are components which vary slowly with $k^{\perp}$. For a transform of the data to yield accurate results the nonperiodic components of the reflectivity must be removed. The plateau before the critical angle and constant slope following, represented by A, can be removed by limiting the region analyzed and dividing by an appropriate factor, respectively. The varying periodicity which results from the inclusion of $k_c^{\perp}$ in the cosine can also be removed by an appropriate choice of multiplying factor.

Scattering theory confirms the relation between the Fourier transform and thickness information. Under this formalism, the reflectivity above the critical angle of a layer having index profile n(z) is given in terms of the scattering vector, Q, by

$$R(Q) = \frac{(4\pi)^2}{Q^4}\left|\int_{-\infty}^{\infty}\frac{dn}{dz}e^{iQz}dz\right|^2, [22]$$

where the scattering vector represents the momentum transfer to the surface and is related to $k^{\perp}$ by

$$Q = k_{in} - k_{out} = 2k^{\perp} = \frac{4\pi\sin\theta}{\lambda}.$$

From this form of the reflectivity it is clear that in the region beyond the critical angle the reflectivity is proportional to $(k^{\perp})^{-4}$. Therefore, the fall-off in intensity can be compensated for by multiplying by $(k^{\perp})^4$. Bridou [22] has shown that a modified form of $k^{\perp}$ can be employed to correct for the imperfect periodicity of the oscillations. This is given by

$$k^{\perp} = \frac{2\pi\sqrt{\sin^2\theta_m - \sin^2\theta_c}}{\lambda},$$

where m is the order of the interference and θ_c is the critical angle as measured from the half maximum point of the reflectivity curve.

The optical model presented in this paper allows for the reflectivity of a given film to be accurately predicted from a knowledge of the structure. However, the direct inversion of this model to yield structure information from the measured reflectivity is not possible due to the absence of phase information. Instead, film parameters can be determined by fitting a multilayer optical model to the measured curve and extracting the film parameters from the model. This fit is made by minimizing the mean square of the error between the model and data, defined as

$$\chi^2 = \sum\left[(R_{data} - R_{model})^2/\partial^2\right](N-P-1)^{-1},$$

where N is the number of data points, P is the number of free parameters, and ∂ is the experimental uncertainty [17]. If models with equal numbers of free parameters are to be used, the term (N-P-1) can be replaced with N. The minimization of this quantity is effected by calculating the reflectivity from the model based on an initial guess of the parameters and comparing this with the measured reflectivity. A systematic search of parameter space is then performed according to a particular algorithm and the reflectivity again calculated and compared with the measured reflectivity. This cycle continues until

a specified degree of convergence is reached. Great care must be exercised in the design of such a routine as least-square fits to reflectivity models are known to produce solutions which are not unique. [11] Moreover, the initial values chosen for the parameter search may affect final set of parameters which the fit converges upon. Because of this, it has been found that the accuracy of a fit is improved by providing initial values for the parameters space search which are near to the actual values. [23]

Experimental

To enable the measurement of reflectivity curves from growing films a specialized system was constructed. A schematic representation of this system is shown in *Figure 13*. The current state of the system is a manufactured chamber undergoing testing, functioning acquisition software, and prototyped analysis software. A second x-ray system, tailored to energy dispersive measurement, is being designed and constructed. It will not initially be part of the investigation, however, and will not be detailed.

Deposition Chamber

The first phases of this investigation will be performed with a conventional Bragg-Brentano geometry diffractometer having typical source-to-sample distances on the order of tens of centimeters. Because of this, very particular restrictions were placed on the design of the deposition chamber. Designed to use the existing θ / 2θ stage, the chamber is small, rapidly demountable, and reasonably light-weight. It consists of two independent substructures as shown in *Figure 14*. The chamber itself, constructed of 300 series stainless steel, is tubular in design. It consists of a long flanged section in which the sample is housed and two end caps which house all of the feedthroughs for the chamber. Copper-gasketed conflat flanges are used at each of these access points. The three sections of the chamber bolt together and are sealed by Viton o-rings. Kapton film is used to seal the x-ray windows which run along the sides of the chamber.

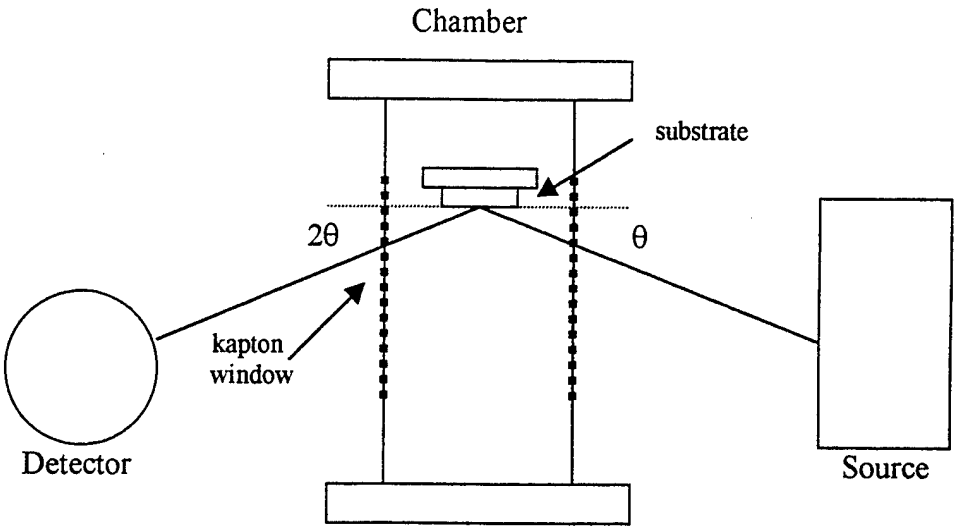

Figure 13. The basic layout of chamber, source and detector showing Kapton windows, position of substrate and incident and reflected angles.

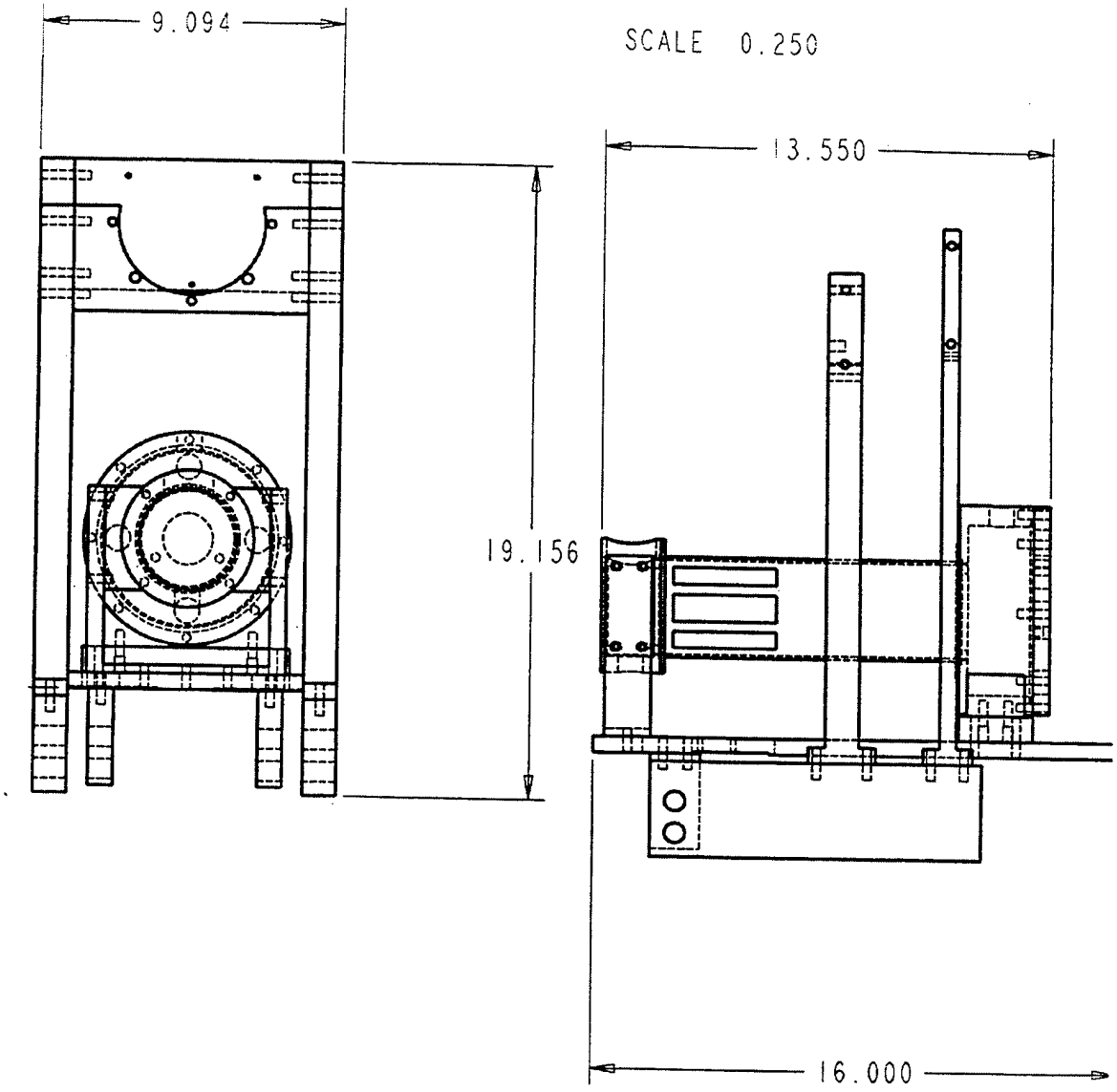

Figure 14. Drawings of the deposition chamber that illustrate the layout and scale of the assembly.

A second, aluminum construction attaches the chamber to the goniometer and provides support and vibration isolation for the turbomolecular pump. The chamber bolts to an aluminum plate which in turn mounts to the goniometer by means of two support brackets. The pump is bolted into a mounting cradle above the chamber. For future use, the deposition chamber may be adapted to another system simply by redesign of this mounting assembly.

The sample stage rides on two linear bearing stages which traverse the length of the chamber and allow micrometer adjustment of the sample stage over a range of two inches. This travel, in addition to the length of the Kapton windows, allows for a wide range of incident angles and substrate thicknesses. In order to provide controlled growth, the substrate holder itself is of copper block construction and incorporates temperature control by means of a Peltier heating/cooling element and thermocouple in addition to water cooling.

Deposition is presently performed by evaporation, making use of a heated tungsten filament and thin foil sources. However, the chamber is designed to allow for replacement of the evaporation source by a small diameter sputtering source without retooling.

X-ray System

For *in situ* measurement, an x-ray configuration similar to that used in the current system for energy dispersive measurements is anticipated. This configuration is described by Windover, et. al. [1]

> [We used a] 20 kV, 20 mA (400 Watt) 0.4 mm × 12 mm copper anode sealed source X-ray tube on a commercial Scintag X-ray diffractometer. The source-to-sample and detector-to sample distance was 286 mm. Source and scatter 0.05 mm collimator slits were used on the source side, providing 0.043° beam divergence in the reflection plane. One detector 0.05 mm collimator slit was used providing a fixed angle with 0.01° width. A nominal 600-μm thick, Si (100) single side polished wafer was used in the primary beam path to filter the Cu kα and kβ lines from the incident beam spectrum.

Through the use of the silicon wafer as a filter, the intensity of the kα and kβ peaks relative to the broad spectrum *brehmstrahlung* is reduced. This enables higher beam

intensity and increased count rates across the spectrum without detector saturation. The more even spectrum also improves the accuracy of the intensity normalization. *Figure 11* shows the spectrum of the incident beam and the impact of normailzation on the reflectivity data.

Previous work with fixed-angle, energy dispersive X-ray reflectivity has involved using a high voltage source. The approach adopted for this work will be to use lower voltage sources, copper or chromium operating at 20 kV to produce only the region of the reflectivity spectrum needed to recover the thickness or density information. Semiconductors are prone to device degradation by X-ray exposure due to the creation of charge traps caused by X-ray generated defects [1]. By using a lower power X-ray source and minimizing the exposure time, this measurement method can be made suitable for routine semiconductor device characterization.

Data Acquisition

Reflected intensity is measured by a Peltier cooled, Silicon Kevex detector. This device has an energy resolution of 180 eV FWHM at 5.6 keV, which is the equivalent of a scan resolution of a few hundredths of a degree. The relation between the two resolutions is determined from the resulting resolutions in $k^{\perp}$.

$$\Delta k^{\perp} = 2\pi \frac{\Delta E}{hc}\theta$$

$$\Updownarrow$$

$$\Delta k^{\perp} = 2\pi \frac{E}{hc}\Delta\theta$$

A Nucleus PCA II multichannel analyzer set to 256-channel mode is currently used for capturing spectral data. For the range of energies typically considered, this allows for a resolution in $k^{\perp}$ greater than that imposed by the detector. In previous energy dispersive measurements the linearity and calibration was set using 7 data points: Cr kα and kβ, Cu kα and kβ, and Mo kα and kβ, and the 20 kV edge in the data.

Data from the card is stored to a file using a routine written in C. (See appendix 1.)

Computational Analysis

After a sufficient number of counts have accumulated on the MCA, the value from the card is read to a file. This data is then transferred into an analysis routine. For rapid monitoring of thickness, a Fourier transform may be used. This is implemented within LabView using a windowed FFT and a peak-fitting function. The pretransform and normalization can both be performed directly within LabView.

For more in-depth study, a fitting routine based on the optical transfer matrix is utilized. (The most current version of this software and is detailed in appendix 2.) In this routine, each successive data set is represented as another layer in the model, having unique parameters. A new multilayer model having a number of layers equal to the number of data sets which have been recorded is calculated for each data set. The parameters of the new layer are determined from a fit of this model to the experimental data. Multilayer fits and the challenges associated with them have been discussed. Of particular concern is the convergence of the fit to inaccurate results. However, developing the model by sequential fits made during the film growth minimizes the non-uniqueness problem by reducing the number of free parameters. Only six parameters are required for each fit: thickness, density, surface roughness, interface roughness, δ and β. Further, the chance of false convergence is minimized by taking initial values for the parameter search from the fitted values of the previous layer. A flow chart of this process is shown in *Figure 15*.

The Levenberg-Marquardt algorithm is used to fit the data. Reflectivity data does not have a linear dependence on its parameters as given by

$$y(x) = \sum_{k=1}^{m} a_k f_k(x). \text{ [24]}$$

Therefore, a general nonlinear fitting algorithm is used. The Levenberg-Marquardt has the advantage of being able to converge on the correct minimum from far away while also being able to rapidly converge to the answer when the initial guess is close to the actual value. [24] The code for this procedure is borrowed largely from Press, *et. al.* [25].

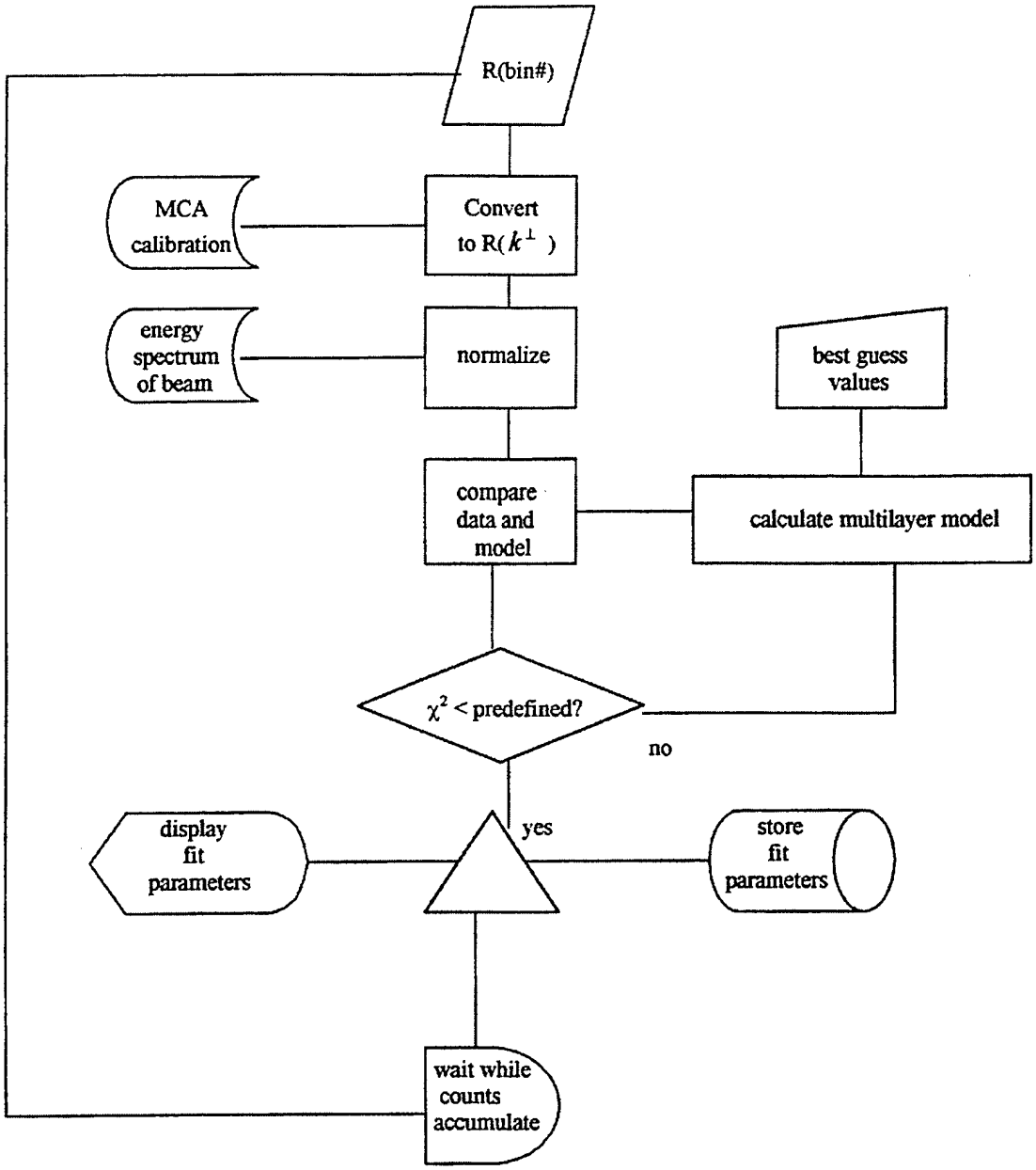

Figure 15. A flow chart illustrating the operation of the measurement software.

Summary

Through theoretical exposition, the feasibility of energy dispersive x-ray reflectivity for the measurement of growing thin films has been shown. A model for the reflectivity based on Fresnel theory has been detailed. This model accounts for the effects of the thickness, roughness, and density of layers and multilayers on the reflectivity of a sample. Methods for the extraction of film parameters from the reflectivity based on optical and scattering formalisms have been detailed. In order to study film growth by energy dispersive reflectivity method a system has been designed and constructed. Software has been written in LabView and C++ to acquire and analyze data using both the Fourier transform method and the least-squares fit to the optical model. Preliminary testing of this system is now underway.

Acknowledgements

To the glory of God. The author would like to thank his advisor, Dr. Toh-Ming Lu, and collaborator, Donald Windover, for their encouragement and assistance in this project. Additionally, the author is indebted to Dr. Sabrina Lee of ARDEC, director of the x-ray diffraction laboratory at Benét Laboratories, for use of their facilities. The author would also like to acknowledge the partial support of this work by the SRC Center for Advanced Interconnect Science and Technology (CAIST).

References

[1] D. Windover, E. Barnat, J. Summers, T.-M. Lu, A. Kumar, H. Bakhru, and S.L. Lee, submitted to *Thin Solid Films.*

[2] D. J. Miller, R. P. Chiarello, H. K. Kim, T. Roberts, H. You, R.T. Kampwirth, K. E. Gray, J. Q. Zheng, S. Williams, J. B. Ketterson, and R. P. Chang, *Appl. Phys. Lett.* **59**, 3174 (1991).

[3] E. Chason and D.T. Warwick, in Advances in Surface and Thin Film Deposition, edited by T.C. Huang, P.I. Cohen, and D.J. Eaglesham, MRS Symposia Proceedings No. 208 (Materials Research Society, Pittsburgh, 1991), p. 351.

[4] B.K. Kellerman, E. Chason, D.P. Adams, T.M. Mayer, and J.M. White, *Surf. Sci.*, **375**, 331 (1997).

[5] A.M. Baranov and I.F. Mikhailov, *Thin Solid Films* **324**, 63 (1998).

[6] C.H. Lee and S.Y. Tseng, *J. Appl. Cryst.* **31**, 181 (1998).

[7] R. K. Heilman and Robert M. Sutter, *Phys Rev. B* **59(4)**, 3075, (1999).

[8] V. M. Pantojas, Ph.D. Thesis, Rensselaer Polytechnic Institute, 1993.

[9] L. G. Parratt, *Phys. Rev.* **95** (1954) 359.

[10] J.D. Jackson, *Classical Electrodynamics*, 3rd ed. (John Wiley & Sons, New York, 1999).

[11] E. Spiller, *Rev. Phys. Appl.* **23**, (1988) 1687

[12] R.W. James, *The Optical Principles of the Diffraction of X-rays* (Cornell University Press, Ithaca, New York, 1965).

[13] H. Kiessig, *Ann. der Physik*, **10**, 715, 769 (1931).

[14] F.L. Pendrotti and L.S. Pendrotti, *Introduction to Optics*, 2nd ed. (Prentice Hall, Upper Saddle River, New Jersey, 1993).

[15] L. Nevot, B. Pardo, and J. Corno, *Rev. Phys. Appl.* **23**, 1675 (1988).

[16] B. K. Tanner, S. J. Miles, D. K. Bowen, L. Hart, and N. Loxely, in Advances in Surface and Thin Film Deposition, edited by T.C. Huang, P.I. Cohen, and D.J. Eaglesham, MRS Symposia Proceedings No. 208 (Materials Research Society, Pittsburgh, 1991), p. 345.

[17] E. Chason and T.M. Mayer, Crit. Rev. in Solid State and Mat. Sci., **22**, 1 (1997).

[18] D.H. Bilderbak and S. Hubbard, Nucl. Instrum. Methods Phys. Res. **195**, 85/91 (1982).

[19] P. Dhez, H. Duval, and J.C. Malaurent, *J. X-ray Sci. and Tech.* **3**, 176 (1992).

[20] H. N. Yang, G.C. Wang, and T.M. Lu, (1994) *Diffraction from Rough Surfaces and Dynamic Growth Fronts*, pp. 136-168, Singapore: World Scientific.

[21] D.T. Brower, R.E. Revay, and T.C. Huang, *Powder Diff.* **11(2)**, 114 (1996).

[22] F. Bridou and B. Pardo, *J. X-ray Sci. Tech.* **4**, 200 (1994).

[23] T.C. Huang and W. Parrish, Advances, edited by C.S. Barret, *X-ray Analysis* **35**, 137 (1992) (Plenum Press, New York, 1992).

[24] P.R. Bevington and D.K. Robinson, *Data Reduction and Error Analysis for the Physical Sciences*, 2nd ed. (McGraw Hill, New York, 1992).

[25] WH. Press, B.P. Flannery, S.A. Teukolsk, and W.T. Vetterling, *Numerical Reciepes in C: The Art of Scientific Computing* (Cambridge University Press, New York, 1987).

[26] B. Lengeler, edited by C.S. Barret, *X-ray Analysis* **35**, 127 (1992) (Plenum Press, New York, 1992).

Appendix 1: C++ code to store the data on the PCA II card to a file

```
                    /* PROGRAM:     Cdemo.c > Cdemo.exe      */
                    /* COMPILER:    Borland Turbo C ver 2.0  */
                        /* DATE:        Fall 1988            */
                      /* COMPANY:      Nucleus Inc.          */
                    /* PURPOSE:      read PCA Board RAM,     */
                        /*            then and write an ascii  */
                        /*            disk file called ram.dat. */
                      /* HARDWARE:     PCA-II                */
    /*                                                                  */
                      /*       modified for MS visual C++      */
                         /*          June 30, 1999            */
                         /*          Jason Summers            */

/* ----------------- Supporting Include Files ----------------- */
#include <process.h>
#include <memory.h>
#include <stdio.h>
#include <windows.h>
#include <conio.h>
/* ----------------- Supporting Include Files ----------------- */

/* ---------------------- Macros ---------------------------- */
#define  TRUE  1
#define  FALSE 0
#define  NIL   0

#define NUMB_CHAN      1024        /* number of channels to process       */
#define PORTNUMB       0x01e0      /* for default dip switch setting      */
#define DATA_FILE_NAME  "ram.dat"   /* name of data (disk) file to write to */
#define SOURCEDESC     2           /* Global Descriptor Table for: source 24 bit address
*/
#define TARGETDESC     3           /* Global Descriptor Table for: target 24 bit address
*/

#define begfunct  {
#define endfunct  }
#define begfor    {
#define endfor    }
#define begwhile  {
#define endwhile  }
#define begif     {
#define endif     }
```

```
#define begelse   {
#define endelse   }
#define begelseif {
#define endelseif }
#define begdo     {
#define enddo     }
#define begswitch {
#define endswitch }

#define BOOL   unsigned char
#define SHORT  int
#define USHORT unsigned short
#define LONG   long
#define ULONG  unsigned long
#define CHAR   char
#define UCHAR  unsigned char
/* ---------------------- Macros ---------------------------- */

/* ========================== DATA TYPES
========================================== */
/* ---------------------- CONTROL STRUCT ---------------------------- */
typedef struct
     {
     USHORT        cardport;       /* number of lowest used PCA-II I/O port */
     USHORT        cardtopport;     /* highest port address used by PCA-II (always:
cardport + 2) */
     } CONTROL;
/* ---------------------- CONTROL STRUCT ---------------------------- */

/* ---------------------- GDT STRUCT ---------------------------------- */
typedef struct {
   unsigned int seg_limit;          /* segment size limit of move        */
   unsigned int base_lo_word;       /* lower 16 bits of 24 bit address   */
   unsigned char base_hi_byte;      /* upper  8 bits of 24 bit address   */
   unsigned char data_access_rights; /* access rights = 0x93 under DOS 3.0 */
   unsigned int data_reserved;      /* reserved for use by interrupt     */
} GDT;
/* ---------------------- GDT STRUCT ---------------------------------- */
/* ========================== DATA TYPES
========================================== */

/* ---------------------- Global Variables ---------------------- */
ULONG  larray[NUMB_CHAN]; /* count array   */
UCHAR  roi[NUMB_CHAN];    /* roi array     */
CONTROL control;          /* data structure */
/* -------------------------------------------------------------- */
```

```
/* ********************** GLOBAL DECRIPTOR TABLE
************************** */
GDT gdt[]= {
      {  0,  0,  0,  0,  0  },  /* dummy table for interrupt    */
      {  0,  0,  0,  0,  0  },  /* dummy table for interrupt    */
      {  0,  0,  0,0x93,  0  },  /* table for the source of move  */
      {  0,  0,  0,0x93,  0  },  /* table for the target of move  */
      {  0,  0,  0,  0,  0  },  /* dummy table for interrupt    */
      {  0,  0,  0,  0,  0  }   /* dummy table for interrupt    */
   };
/* ********************** GLOBAL DECRIPTOR TABLE
************************** */
/* ---------------------- Global Variables ---------------------- */

/* ---------------------- Function Prototypes ------------------- */
void   WINAPI   putportbyte(register SHORT address, UCHAR value);
UCHAR  WINAPI   getportbyte(register SHORT address);

void            get_data_PCAII(void);


/* ---------------------- Function Prototypes ------------------- */

/* ------------ Main Program Start: ---------------------------------------- */
void main(void)
 {
 register SHORT i;        /* channel loop index     */
 FILE        *fp;        /* file pointer           */
 CHAR         ch;        /* temp char              */
 SHORT        ret = 0;   /* var for function return code */


  memset( &larray[0], NIL, (NUMB_CHAN * sizeof(ULONG)) );  /* init the long int
array to zero */



  control.cardport    = PORTNUMB;      /* save for CONTROL */
  control.cardtopport = PORTNUMB + 2;   /* save for CONTROL */

  ch = getportbyte(0);  /* get data on card. (first byte) */
  ret = ch;             /* save so we can restore later on */
  ch++;                 /* add one to the read value      */
  putportbyte(0,ch);    /* write out the incremented data  */

  if (getportbyte(0) != ch)  /* PCA-II card not present */
```

```
 begif
   puts("No PCA-II Card was Found at port 01e0h.");

 endif
else     /* a PCA-II Card was present. so use PCA-II card */
 begelse
   putportbyte(0, (UCHAR) ret);  /* restore the changed data      */
   get_data_PCAII();             /* read data from the PCA-II card */
 endelse


/* NOTE: Now we have both channel counts and ROI data copied */
/*      into easy to use arrays. So now you can do your     */
/*      calculations on the data in the card.               */
/* channel counts in: larray[]  (a long int array)          */
/* ROI data in:       roi[]     (a char array)              */

/* NOTE: (if an ROI byte is not zero then that channel is part of an ROI */

fp = fopen(DATA_FILE_NAME,"w");   /* note: data file name. write mode  */

if (fp != NULL)  /* there was no problem opening the disk file */
  begif

   fprintf(fp,"Channel  Counts ROI \n"); /* some header text */
   fprintf(fp,"-------------------\n");

   /* -------------------- Output Loop to Disk File ------------------- */
   for (i = 0; i < NUMB_CHAN; i++)
    begfor

     if ( (i % 512) == 0) printf("writing record: %d\n",i);

     ret = fprintf(fp,"%4d %8ld %4d\n",i,larray[i],roi[i]);

     if (ret == -1)   /* test for i/o (disk full) error */
      begif
        puts("ERROR: Disk error. Press any key to continue.");
        getchar();       /* pause for a key */
        exit(-9);        /* exit to DOS */
      endif

    endfor
   /* -------------------- Output Loop to Disk File ------------------- */
   i--;  /* adjust loop index */
   printf("writing record: %d\n",i);
```

```
      fclose(fp);  /* close the disk file */
      printf("Write of ascii file: %s complete.\n",DATA_FILE_NAME);
     endif
    else
     begelse
       puts("ERROR:  Opening Disk File."); /* print error message */
     endelse

endfunct  /* end of main program */
/* ---------- Main Program End: ---------- */


/* needed subroutines ('functions' in C Language) */

/* This function: writes data to PCA-II I/O mapped MCA boards */
/* ========================================================== */
void  WINAPI   putportbyte(SHORT address, UCHAR value)
 begfunct
  USHORT port = control.cardtopport; /* port number plus two */

   _outp(port--,  address ); /* low  address byte  */
   address = (address >> 8);   /* move the hi byte into the lo position */
   _outp(port--,  address ); /* high address byte  */
   _outp(port, value );      /* put the data byte  */
 endfunct
/*
========================================================================
*/

/* This function: reads data from PCA-II I/O mapped MCA boards */
/* ========================================================== */
UCHAR static WINAPI getportbyte(SHORT address)
 begfunct
  USHORT port = control.cardtopport; /* port number plus two */

   _outp(port--,  (UCHAR) address);  /* low  address byte  */
   _outp(port--,  address >> 8);     /* high address byte  */
  return( _inp(port) );              /* get the data byte  */
 endfunct
/*
========================================================================
*/
```

```
/* This function:  gets count and ROI data from a PCA-II card */
/* ================================================================
*/
void get_data_PCAII(void)
begfunct
 register SHORT i;          /* loop index           */
 register SHORT cardbyte;   /* byte loop index      */
 CHAR          *byteptr;    /* pointer to count buffer */

  cardbyte = 0;  /* init RAM byte index */
  /* ---------------------- Loop to fill the long integer array ------------- */
  for (i = 0; i < NUMB_CHAN; i++)
   begfor
    byteptr = (UCHAR /*far*/ *) (&larray[i]); /* set byte ptr to the long int array element
*/

    *byteptr++ = getportbyte(cardbyte++); /* get low byte of 24 bit channel value */
    *byteptr++ = getportbyte(cardbyte++); /* get mid byte of 24 bit channel value */
    *byteptr   = getportbyte(cardbyte++); /* get hi byte of 24 bit channel value */
    roi[i]     = getportbyte(cardbyte++); /* get ROI byte */
   endfor
  /* ---------------------- Loop to fill the long integer array ------------- */
endfunct
/*
================================================================
*/


/* ------------------------------------------------------------------ */
```

Appendix 2: C++ code which makes a sequential fit of a growing film to a multilayer model

```
/* Program to to test the curve fitting routine: testfit.cpp */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

typedef struct FCOMPLEX {float r; float i;} fcomplex;

#define NPT 256 /* number of channels on MCA*/
#define MA 6 /* number of parameters per layer */
#define SPREAD 0.001
#define SQR(a) ((a)*(a))
#define SWAP(a,b) {float temp=(a);(a)=(b); (b)=temp;}


/* ---------------------- Function Prototypes ------------------- */

void covsrt(float **covar, int ma, int *lista, int mfit);
void gaussj(float **a, int n, float **b, int m);

void mrqcof(float *x, float *y, float *sig, int ndata, float *a, int ma,
                        int *lista, int mfit, float **alpha, float *beta,
                        float *chisq, void(*refcurve) (float,float*,float,float*,
                        float, int));

void mrqmin(float *x, float *y, float *sig, int ndata, float *a, int ma,
                        int *lista, int mfit, float **covar, float **alpha,
                        float *chisq, void(*refcurve) (float,float*,float,float*,
                        float, int),float *alamda);

void refcurve(float x, float *a, float y, float *dyda, float theta, int N);

void refmatrix(int N, int layer, float d, float lambda, float delta1, float delta2,
                          float theta1, float theta2, fcomplex oldmatrix[1][1],
                          fcomplex newmatrix[1][1]);

fcomplex Complex(float re, float im);
float Cabs(fcomplex z);
fcomplex Conjug(fcomplex z);
fcomplex Csub(fcomplex a, fcomplex b);
fcomplex Cmul(fcomplex a, fcomplex b);
```

```
fcomplex Cdiv(fcomplex a, fcomplex b);
fcomplex Cadd(fcomplex a, fcomplex b);
fcomplex RCmul(float x, fcomplex a);

float *vector(int nl, int nh);
int *ivector(int nl, int nh);
void free_vector(float *v, int nl, int nh);
void free_ivector(int *v, int nl, int nh);
void nrerror(char *error_text);
float **matrix(int nrl, int nrh, int ncl, int nch);
void free_matrix(float **m, int nrl, int nrh, int ncl, int nch);


/* ---------- Function Prototypes ---------- */


/* ----------Global Variables---------- */

float theta=1;
int N=3;
fcomplex one;
one.r = 1; one.i=0;

/* ----------Global Variables---------- */


int main(void)

{
        FILE *normalization, *data;
        int i, idum=(-911),itst,k,mfit,*lista, bin[NPT+1], bin2[NPT+1];
        unsigned int larray[NPT+1], normarray[NPT+1];
        float alamda,chisq,ochisq,*x,*y,*sig,**covar,
                ** alpha;

        static float a[MA+1]=
        {0.0,5.0,2.0,3.0,2.0,5.0,3.0}; /* a[1]=theta in; a[2]=lambda;
                                                                a[3]=delta;
a[4]=beta; a[5]=thickness;
                                a[6]=top roughness */

        static float gues[MA+1]=
        {0.0,4.5,2.2,2.8,2.5,4.9,2.8};


        /* read the normalization file in */
```

```
        fopen("c:\\norm.dat","r");
        i=0;
        while (i<NPT)
        {
                fscanf(normalization,"%i %u\n",&bin[i],&normarray[i]);
        }
        fclose (normalization);

        /* read the data file in */
        fopen("c:\\ram.dat","r");
        i=0;
        while (i<NPT)
        {
                fscanf(data,"%i %u\n",&bin2[i],&larray[i]);
        }
        fclose (normalization);



        lista=ivector(1,MA);
        x=vector(1,NPT);
        y=vector(1,NPT);
        sig=vector(1,NPT);
        covar=matrix(1,MA,1,MA);
        alpha=matrix(1,MA,1,MA);
        for (i=1;i<=NPT;i++) {
                x[i]=24*i+34;                           //set energy scale
                y[i]=larray[i]/normarray[i]; //perform normalization
                sig[i]=SPREAD*y[i];
        }
        mfit=6;
        for (i=1;i<=mfit;i++) lista[i]=1;
        alamda = -1;
        for (i=1;i<=MA;i++) a[i]=gues[i];
        mrqmin(x,y,sig,NPT,a,MA,lista,mfit,covar, alpha, &chisq, refcurve, &alamda);
        k=1;
        itst=0;
while (itst < 2)
{
        printf("/n");
        k++;
        ochisq=chisq;
        mrqmin(x,y,sig,NPT,a,MA,lista,mfit,covar,alpha,&chisq,refcurve,&alamda);
        if (chisq > ochisq) itst=0;
        else if (fabs(ochisq-chisq) < 0.1) itst++;
}
```

```
alamda=0.0;
mrqmin(x,y,sig,NPT,a,MA,lista,mfit,covar,alpha,&chisq,refcurve,&alamda);
printf("\nUncertainties:\n");
for (i=1;i<=6;i++) printf("%9.4f",sqrt(covar[i] [i]));
printf("\n");
free_matrix(alpha,1,MA,1,MA);
free_matrix(covar,1,MA,1,MA);
free_vector(sig,1,NPT);
free_vector(y,1,NPT);
free_vector(x,1,NPT);
free_ivector(lista,1,MA);
return (0);
}


/* Levenberg-Marquardt minimization of chi-squared for a fit to a
  non-linear function with parameters a[1..ma] */
/*=============================================================================
*/

void mrqmin(float *x, float *y, float *sig, int ndata, float *a, int ma,
                        int *lista, int mfit, float **covar, float **alpha,
                        float *chisq, void(*refcurve) (float,float*,float*,float*,
                        float, int),float *alamda)

{
        int k,kk,j,ihit;
        static float *da,*atry,**oneda,*beta,ochisq;


        if (*alamda < 0.0) {
                oneda=matrix(1,mfit,1,1);
                atry=vector(1,ma);
                da=vector(1,ma);
                beta=vector(1,ma);
                kk=mfit+1;

        /* check to see if the lista values are reasonable */

                for (j=1;j<=ma;j++) {
                        ihit=0;
                        for (k=1;k<=mfit;k++)
                                if (lista[k] == j) ihit++;
                        if (ihit==0)
                                lista[kk++]=j;
                        else if (ihit > 1)
```

```
                        nrerror ("Bad LISTA permutation in MRQMIN-1");
                }

                if (kk != (ma+1)) nrerror ("Bad LISTA permutation in MRQMIN-2");

        /* okay so proceed */

                *alamda=0.001;
                mrqcof(x,y,sig,ndata,a,ma,lista,mfit,alpha,beta,chisq,refcurve);
                ochisq=(*chisq);
        }
        for (j=1; j<=mfit; j++) {
                for (k=1;k<=mfit;k++)
                        covar [j][k]=alpha[j][k];
                covar[j][j]=alpha[j][j]*(1.0+(*alamda));
                oneda[j][1]=beta[j];
        }
        gaussj(covar,mfit,oneda,1);
        for (j=1;j<=mfit;j++) da[j]=oneda[j][1];
        if (*alamda == 0.0)
        {
                covsrt(covar,ma,lista,mfit);
                free_vector(beta,1,ma);
                free_vector(da,1,ma);
                free_vector(atry,1,ma);
                free_matrix(oneda,1,mfit,1,1);
                return;
        }
        for (j=1;j<=ma;j++) atry[j]=a[j];
        for (j=1;j<=mfit;j++) atry[lista[j]]=a[lista[j]]+da[j];
        mrqcof(x,y,sig,ndata,atry,ma,lista,mfit,covar,da,chisq,refcurve);
        if(*chisq < ochisq) {
                *alamda *= 0.1;
                ochisq=(*chisq);
                for (j=1;j<=mfit;j++) {
                        for (k=1;k<=mfit;k++)
                                alpha[j][k]=covar[j][k];
                        beta[j]=da[j];
                        a[lista[j]]=atry[lista[j]];
                }
        } else {
                *alamda *= 10.0;
                *chisq=ochisq;
        }
        return;
}
```

```
/* This performs math duties for the routine mrqmin */
/*-------------------------------------------------------------------*/
void mrqcof (float *x, float *y, float *sig, int ndata, float *a, int ma,
                        int *lista, int mfit, float **alpha, float *beta,
                        float *chisq, void(*refcurve) (float*,float*,float*,float*,
                        float, int))

{
        int k,j,i;
        float ymod,wt,sig2i,dy,*dyda;

        dyda=vector(1,ma);
        for (j=1;j<=mfit;j++) {
                for (k=1;k<=j;k++) alpha[j][k]=0.0;
                beta[j]=0.0;
        }
        *chisq=0.0;
        for (i=1;i<=ndata;i++) {
                //(*refcurve)(x[i],a,&ymod,dyda,theta,N);
                sig2i=1.0/(sig[i]*sig[i]);
                dy=y[i]-ymod;
                for (j=1;j<=mfit;j++) {
                        wt=dyda[lista[j]]*sig2i;
                        for (k=1;k<=j;k++)
                                alpha[j][k] += wt*dyda[lista[k]];
                        beta[j] += dy*wt;
                }
                (*chisq) += dy*dy*sig2i;
        }
        for(j=2;j<=mfit;j++)
                for (k=1;k<=j-1;k++)
                        alpha[k][j] = alpha[j][k];
        free_vector(dyda,1,ma);
}


/* This sorts covariance matrix */
/*-------------------------------------------------------------------*/
void covsrt(float **covar, int ma, int *lista, int mfit)

{
        int i,j;
        float swap;
```

```
        for (j=1;j<ma;j++)
                for (i=j+1;i<=ma;i++) covar[i][j]=0.0;
        for (i=1;i<mfit;i++)
                for(j=i+1;j<=mfit;j++) {
                        if (lista[j] > lista[i])
                                covar[lista[j]] [lista[i]]=
                                        covar[i][j];
                        else
                                covar[lista[i]][lista[j]]=
                                        covar[i][j];
        }
        swap=covar[1][1];
        for (j=1;j<=ma;j++){
                covar [1][j] = covar [j][j];
                covar[j][j]=0.0;
        }
        covar[lista[1]][lista[1]]=swap;
        for (j=2;j<=mfit;j++) covar[lista[j]]
                [lista[j]]=covar[1][j];
        for (j=2;j<=ma;j++)
                for (i=1;i<=j-1;i++) covar[i][j]=
                        covar[j][i];

}


/* Gauss-Jordon sol'n to lin. eq. */
/*-----------------------------------------------------------------*/

void gaussj(float **a, int n, float **b, int m)


{
        int *indxc,*indxr,*ipiv;
        int i,icol,irow,j,k,l,ll;
        float big,dum,pivinv;

        indxc=ivector(1,n);
        indxr=ivector(1,n);
        ipiv=ivector(1,n);
        for (j=1;j<=n;j++) ipiv[j]=0;
        for (i=1;i<=n;i++) {
                big=0.0;
                for (j=1;j<=n;j++)
                        if (ipiv[j] != 1)
```

```
                        for (k=1;k<=n;k++) {
                                if (ipiv[k] == 0){
                                  if (fabs(a[j][k]) >= big) {
                                                big=fabs(a[j][k]);
                                                irow=j;
                                                icol=k;
                                        }
                                }else if (ipiv[k] > 1) nrerror ("GAUSSJ: singular
matrix-1");
                }
                ++(ipiv[icol]);

                if (irow != icol) {
                        for (l=1;l<=n;l++)
                        SWAP (a[irow][l],a[icol][l])
                        for (l=1;l<=m;l++)
                        SWAP (b[irow][l],b[icol][l])
                }
                indxr[i]=irow;
                indxc[i]=icol;
                if (a[icol][icol] == 0.0)
                        nrerror ("GAUSSJ: singular matrix-2");
                pivinv=1.0/a[icol][icol];
                a[icol][icol]=1.0;
                for (l=1;l<=n;l++) a[icol][l] *=pivinv;
                for (l=1;l<=m;l++) b[icol][l] *=pivinv;
                for(ll=1;ll<=n;ll++)
                        if (ll != icol) {
                                dum=a[ll][icol];
                                        a[ll][icol]=0.0;
                                        for (l=1;l<=n;l++)
                                        a[ll][l] -= a[icol][l]*dum;
                                        for (l=1;l<=m;l++)
                                        b[ll][l] -= b[icol][l]*dum;
                        }
        }
        for (l=n;l>=1;l--) {
                if (indxr[l] !=indxc[l])
                        SWAP(a[k][indxr[l]],a[k][indxc[l]]);
        }
        free_ivector(ipiv,1,n);
        free_ivector(indxr,1,n);
        free_ivector(indxc,1,n);
}
```

```
/* This calculates reflectivity for N layers */
/*
==========================================================================
*/

void refcurve (float x,float *a,float y,float *dyda,float theta,int N)

{
        int j;
        float *deltaA, yplus;
        dyda=vector(1,NPT);
        deltaA=vector(1,MA);


        fcomplex oldmatrix[1][1], newmatrix[1][1], r, R;
        float d,lambda, delta1, delta2, theta1, theta2;


        oldmatrix [0][0] = Complex(1,0);
        oldmatrix [1][1] = Complex(1,0);
        oldmatrix [1][0] = Complex(0,0);
        oldmatrix [0][1] = Complex(0,0);

j=1;

for (j=1;j<=N;j++)
{
        layer=j;
        refmatrix (N,layer,d, lambda,delta1,delta2,theta1,theta2,oldmatrix,newmatrix);
}

r=Cdiv(newmatrix[0][1],newmatrix[1][1]);
y=Cmul(r,r); /* the reflectivity */


/* the following takes a numerical derivative of theoretical
  y(lambda) w.r.t a given parameter*/

for (j=1;j<=MA;j++) deltaA[j]=.00001*a[j];

for (j=1;j<=MA;j++)
{
        if (j=1)
        {
                for (j=1;j<=N;j++)
{
```

```
        layer=j;
        refmatrix (N,layer,d, lambda,delta1,delta2,theta1,theta2,oldmatrix,newmatrix);
}

r=Cdiv(newmatrix[0][1],newmatrix[1][1]);
y=Cmul(r,r);
        }
        a[j]=a[j]+deltaA[j];
        for (j=1;j<=N;j++)
{
        layer=j;
        refmatrix (N,layer,d, lambda,delta1,delta2,theta1,theta2,oldmatrix,newmatrix);
}

r=Cdiv(newmatrix[0][1],newmatrix[1][1]);
yplus=Cmul(r,r);
        a[j]=a[j]-deltaA[j];
for (j=1;j<=N;j++)
{
        layer=j;
        refmatrix (N,layer,d, lambda,delta1,delta2,theta1,theta2,oldmatrix,newmatrix);
}

r=Cdiv(newmatrix[0][1],newmatrix[1][1]);
y=Cmul(r,r);
        dyda[j]=((yplus-y)/deltaA[j]);
}

}

/*=============================================================================
=*/


/* Calculates the optical matrix for refcurve */
/*=============================================================================
=*/

void refmatrix (int N, int layer, float d, float lambda, float delta1,
                        float delta2,float theta1, float theta2,
                        fcomplex oldmatrix[1][1],fcomplex newmatrix[1][1])

{

        fcomplex phi, p1, p2, t, ttilda, r, rtilda, n1, n2, A[1][1];
```

```
        float beta1, beta2;

        n1.r = 1-delta1; n2.r = 1-delta2; n1.i=-beta1; n2.i=-beta2;
        theta2 = acos ((n1.r/n2.r)*cos(theta1));

        p1=RCmul(sin(theta1),n1);
        p2=RCmul(sin(theta2),n2);
        r=Cdiv(Csub(p1,p2),Cadd(p1,p2));
        t=Cdiv(RCmul(2,p1),Cadd(p1,p2));
        rtilda=RCmul(-1,r);
        ttilda=Cdiv(Cmul(t,p2),p1);
        phi=RCmul((2*3.1415*d*sin(theta1))/lambda,n1);


        if (layer!=N)
        {

        A[0][0]=Cmul(Csub(Cmul(t,ttilda),Cmul(r,rtilda)),exp(2*i*phi));
        A[0][1]=r;
        A[1][0]=RCmul(-1,rtilda);
        A[1][1]=1;
        }

        else
        {
        A[0][0]=Cdiv(one,t);
        A[0][1]=Cdiv(r,t);
        A[1][0]=Cdiv(r,t);
        A[1][1]=Cdiv(one,t);
        }

        newmatrix[0][0]=Cadd(Cmul(oldmatrix[0][0],A[0][0]),Cmul(oldmatrix[0][1],A[1
][0]));
        newmatrix[0][1]=Cadd(Cmul(oldmatrix[0][0],A[0][1]),Cmul(oldmatrix[0][1],A[1
][1]));
        newmatrix[1][0]=Cadd(Cmul(oldmatrix[1][0],A[0][0]),Cmul(oldmatrix[1][1],A[1
][0]));
        newmatrix[1][1]=Cadd(Cmul(oldmatrix[1][0],A[0][1]),Cmul(oldmatrix[1][1],A[1
][1]));

        }

/*==========================================================
*/
```

```
/* The following allows C++ to handle complex numbers */
/*=============================================================
========*/


fcomplex Complex(float re,float im)

{
        fcomplex c;
        c.r=re;
        c.i=im;
        return c;
}


float Cabs(fcomplex z)

{       float x,y,ans,temp;
        x=fabs(z.r);
        y=fabs(z.i);
        if (x==0.0)
                ans=y;
        else if (y==0.0)
                ans=x;
        else if (x>y) {
                temp=y/x;
                ans=x*sqrt(1.0+temp*temp);
        }else{
                temp=x/y;
                ans=y*sqrt(1.0+temp*temp);
        }
        return ans;}

fcomplex Conjug (fcomplex z)

{
        fcomplex c;
        c.r=z.r;
        c.i=-z.i;
        return c;
}

fcomplex Csub(fcomplex a,fcomplex b)

{
```

```
        fcomplex c;
        c.r=a.r-b.r;
        c.i=a.i-b.i;
        return c;
}

fcomplex Cmul(fcomplex a,fcomplex b)

{
fcomplex c;
c.r=a.r*b.r-a.i*b.i;
c.i=a.i*b.r+a.i*b.i;
return c;
}

fcomplex Cdiv(fcomplex a,fcomplex b)

{
        fcomplex c;
        float r,den;
        if (fabs(b.r) >= fabs(b.i)) {
                r=b.i/b.r;
                den=b.r+r*b.i;
                c.r=(a.r+r*a.i)/den;
                c.i=(a.i-r*a.r)/den;
        }else{
                r=b.r/b.i;
                den=b.i+r*b.r;
                c.r=(a.r*r+a.i)/den;
                c.i=(a.i*r-a.r)/den;
        }
        return c;
}

fcomplex Cadd(fcomplex a,fcomplex b)


{
fcomplex c;
c.r=a.r+b.r;
c.i=a.i+c.i;
return c;
}


fcomplex RCmul(float x,fcomplex a)
```

```
{
        fcomplex c;
        c.r=x*a.r;
        c.i=x*a.i;
        return c;
}

/* The error routine for numerical recipes routines */

void nrerror(char error_text[])

{
        fprintf(stderr,"run-time error...\n");
        fprintf(stderr,"%s\n",error_text);
        fprintf(stderr,"...now exiting to system...\n");
        exit(1);
}


/* These are utility functions for defining vectors and matrices*/

float *vector(int nl,int nh)

{
        float *v;

        v=(float *)malloc((unsigned) (nh-nl+1)*sizeof(float));
        if(!v) nrerror("allocation failure in vector()");
                return v-nl;
}

int *ivector(int nl,int nh)

{
        int *v;

        v=(int *)malloc((unsigned) (nh-nl+1)
                *sizeof(int));
        if (!v) nrerror("allocation failure in ivector()");
                return v-nl;
}

void free_vector (float *v,int nl,int nh)

{
```

```
        free((char*) (v+nl));
}


void free_ivector (int *v,int nl,int nh)

{
        free((char*) (v+nl));
}



float **matrix(int nrl,int nrh,int ncl,int nch)

{
        int i;
        float **m;

        m=(float **) malloc((unsigned) (nrh-nrl+1)
                *sizeof(float*));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m -= nrl;

        for (i=nrl;i<=nrh;i++) {
                m[i]=(float*) malloc((unsigned)
                        (nch-ncl+1)* sizeof(float));
                if (!m[i]) nrerror("allocation failure 2 in matrix()");
                        m[i] -= ncl;
        }
        return m;
}

void free_matrix(float **m,int nrl,int nrh,int ncl,int nch)

{
        int i;

        for (i=nrh;i>=nrl;i++) free ((char*)
                (m[i]+ncl));
        free ((char*) (m+nrl));
}
```

TECHNICAL REPORT INTERNAL DISTRIBUTION LIST

	NO. OF COPIES
TECHNICAL LIBRARY ATTN: AMSTA-AR-CCB-O	5
TECHNICAL PUBLICATIONS & EDITING SECTION ATTN: AMSTA-AR-CCB-O	3
OPERATIONS DIRECTORATE ATTN: SIOWV-ODP-P	1
DIRECTOR, PROCUREMENT & CONTRACTING DIRECTORATE ATTN: SIOWV-PP	1
DIRECTOR, PRODUCT ASSURANCE & TEST DIRECTORATE ATTN: SIOWV-QA	1

NOTE: PLEASE NOTIFY DIRECTOR, BENÉT LABORATORIES, ATTN: AMSTA-AR-CCB-O OF ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST

	NO. OF COPIES
DEFENSE TECHNICAL INFO CENTER ATTN: DTIC-OCA (ACQUISITIONS) 8725 JOHN J. KINGMAN ROAD STE 0944 FT. BELVOIR, VA 22060-6218	2
COMMANDER U.S. ARMY ARDEC ATTN: AMSTA-AR-WEE, BLDG. 3022	1
AMSTA-AR-AET-O, BLDG. 183	1
AMSTA-AR-FSA, BLDG. 61	1
AMSTA-AR-FSX	1
AMSTA-AR-FSA-M, BLDG. 61 SO	1
AMSTA-AR-WEL-TL, BLDG. 59 PICATINNY ARSENAL, NJ 07806-5000	2
DIRECTOR U.S. ARMY RESEARCH LABORATORY ATTN: AMSRL-DD-T, BLDG. 305 ABERDEEN PROVING GROUND, MD 21005-5066	1
DIRECTOR U.S. ARMY RESEARCH LABORATORY ATTN: AMSRL-WM-MB (DR. B. BURNS) ABERDEEN PROVING GROUND, MD 21005-5066	1
COMMANDER U.S. ARMY RESEARCH OFFICE ATTN: TECHNICAL LIBRARIAN P.O. BOX 12211 4300 S. MIAMI BOULEVARD RESEARCH TRIANGLE PARK, NC 27709-2211	1
COMMANDER ROCK ISLAND ARSENAL ATTN: SIORI-SEM-L ROCK ISLAND, IL 61299-5001	1
COMMANDER U.S. ARMY TANK-AUTMV R&D COMMAND ATTN: AMSTA-DDL (TECH LIBRARY) WARREN, MI 48397-5000	1
COMMANDER U.S. MILITARY ACADEMY ATTN: DEPT OF CIVIL & MECH ENGR WEST POINT, NY 10966-1792	1
U.S. ARMY AVIATION AND MISSILE COM REDSTONE SCIENTIFIC INFO CENTER ATTN: AMSAM-RD-OB-R (DOCUMENTS) REDSTONE ARSENAL, AL 35898-5000	2
COMMANDER U.S. ARMY FOREIGN SCI & TECH CENTER ATTN: DRXST-SD 220 7TH STREET, N.E. CHARLOTTESVILLE, VA 22901	1

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, BENÉT LABORATORIES, CCAC, U.S. ARMY TANK-AUTOMOTIVE AND ARMAMENTS COMMAND, AMSTA-AR-CCB-O, WATERVLIET, NY 12189-4050 OF ADDRESS CHANGES.