Fall 11-24-2015

Multiwavelength X-Ray Diffraction

Maziar Saleh Ziabari
DePaul University, Chicago, helasraizam@gmail.com

MULTIWAVELENGTH X-RAY DIFFRACTION

A thesis presented in partial fullfilment
for the degree of Master of Science

by
Maziar Saleh Ziabari

August 2015

Physics Department
College of Science and Health
DePaul University
Chicago, IL

Table of Contents

List of Figures

List of Tables

Abstract

The central problem of diffraction theory is determining the structure of an object given only the intensity of its diffraction pattern. In general, both the amplitude and phase must be known to uniquely solve a structure; however a coherent radiation source is usually required in order to measure the diffracted phases (e.g. in holography). In many important cases, coherent sources are unavailable and other assumptions or methods must be employed to substitute for the missing phase information. One method that has been successfully applied to the specific problem of determining atomic structure from X-ray diffraction intensity patterns has been the use of multiple-wavelength measurements in the vicinity of an x-ray absorption edge. The large dispersion of the atomic form factors near resonance can be used to constrain the inversion of the diffraction pattern. This thesis explores the extension of this approach to near-perfect crystals where dynamical diffraction theory is required to describe the X-ray diffraction lineshapes. In particular, this thesis presents a study of the depth-dependent transient strain that exists in an otherwise perfect Germanium crystal following the absorption of an intense ultrafast light pulse. This analysis shows that additional structural information may be obtained by the use of multiple-wavelength techniques.

Chapter 1

Introduction

Light is a solution of Maxwell's Equations in which the magnetic and electric fields are perpendicular. This is observable in the effect of light on a moving or stationary electron. As essential a part as light plays in the world, we do not come across it except through its interactions with matter—dominantly, the vibration of electrons caused by the vibration of other electrons. Examining the interactions of light and matter have contributed to countless advances including wireless communication, Cosmology, and the storage systems in most digital electronics.

Matter is made up of atoms containing many electrons, each of which reacts to the light incident upon it. This causes more general, macro-scale interactions dubbed transmission, reflection, and absorption. For example, a coherent laser light—a plane wave—travelling through air is shaking air's electrons, which causes them to radiate in all directions. However, only the light in the forward direction radiates *in phase*, or additively, while the light directed outside of the direction of the laser light is *incoherent*, and cancels itself for the most part. Light can also be transmitted through an interface, or a material with different properties than air; it is then refracted in a direction given by Snell's Law. Finally, light going through a material is absorbed, and its energy is converted to another form such as heat in vibrating the material's atoms or molecules. If the light is of sufficient energy, it might turn into the extra kinetic energy of an electron which is in turn excited to a higher energy state or kicked entirely out of the atom with some velocity in the Photoelectric Effect. If there is some light left over, it is called a Compton scatter. Finally, light hitting light or a particle can annihilate to create an electron-positron pair in pair production.

One application of these processes can be used to measure the shapes of molecules, crystals, and proteins using x-ray diffraction methods. X-rays, while not visible, constitute light of higher frequency than ultraviolet and lower frequency than gamma rays, and their small wavelength allows them to measure up to nanometer imaging resolution and to femtometer resolution using X-ray diffraction. To facilitate the growing demand for high energy x-ray sources, there exist very large and expensive facilities called synchrotrons, in the shape of a donut over a kilometer in circumference[3]. The donut shapes are the accommodate very fast electrons made to turn with the circle every so often—the turns are in fact decelerations which create x-ray radiation in the tangential direction, where they are harnessed so that scientists in the fields including biology, physics, and chemistry carry out their experiments.

We use this approach to find how a germanium wafer reacts to being hit with a laser. When the laser hits the germanium—which can be treated as layers of atoms due to its crystalline structure—a compression wave of heat is sent into the lattice. If the laser energy is above germanium's band gap, it will kick electrons which previously held the layers together in the valence band to the conduction band through Compton scattering or the photoelectric effect, collapsing layers and causing an elastic response in the crystal. By positioning the laser and x-ray probe so they can be treated as transverse to the crystal surface, we assure that the strain is essentially in one dimension, the depth of the crystal.

1.1 Motivation

Germanium and silicon are two of many elements and compounds with diamond structures. Silicon is used in integrated circuits in countless electronics, and like germanium, it is also a semiconductor. The demand for these is so high that scientific advances have allowed the mass production of perfect silicon crystals for use in electronics on a grand scale. It comes as no surprise, then, that understanding the reaction of these elements to light—even moreso, to heat, which light absorption also generates—is pivotal to fast electronics, which generate so much heat that we customize heat sinks and fans to cool them down in computers.

One-dimensional strain in germanium and silicon structures can occur if one side of the crystal is met with a surge of heat. Undestanding the reaction of the lattice structure to such surges has applications in electronics where heat is a byproduct, or as an infrared detector where the heat is intended. Optoelectronic devices used in fiber optics and photovoltaics also result in heat generation following light absorption. Germanium is further a very sensitive infrared detector due to its transparency in that domain. Calculating the strain of germanium after excitation helps to calculate the speed of the electrons ejected through the conduction band as a result, which is pivotal for detectors.

When the laser pulse meets the germanium, its energy transfers to heat vibrations over varying time scales[14]. Processes include the production of electron-hole pairs, the conversion of photons into thermal vibrations, and free carrier absorption. Electrons excited to the conduction band in the penetration depth of the laser (less than 1μm) propagate into the material at supersonic speeds[14].

If we could measure both the intensity and the phase of incoming waves, we could determine three-dimensional structures from X-ray diffraction. Because we only measure intensity, obtaining three-dimensional structures requires clever methods. One such method is multi-wavelength anomalous diffraction, known as MAD phasing. This technique allows for the determination of the structure of select unstrained, imperfect crystals, via atomic replacement in the unit cell. This is to achieve an accessible K-edge to find the positions of atoms within the (very large) unit cell. The method does not apply for perfect crystals, however, where dynamical diffraction theory is required, the unit cells are small, and atomic replacement does not occur. The method used in this thesis is collecting data at multiple wavelengths (hence penetration depths) to obtain a depth-dependent strain profile. If two rocking curves are obtained for the same crystal; one strained and one unstrained, then the measured Darwin curve offset of each wavelength corresponds roughly to a change in cumulative average lattice spacing at that penetration depth. This provides data for the average interplanar spacing as a function of depth, which can yield information about the strain profile.

Chapter 2

Theory

2.1 Overview

In order to describe the obtained rocking curves—data of reflective intensity over a range of angles—for our perfect crystals, the dynamical theory of diffraction is required. Unlike the kinematical approximation, the dynamical method accounts for inner reflections that end up in the same direction as the original incident xray beam, which can reflect again to contribute to the reflected intensity; hence the dynamical approach.

The crystal is spatially periodic in all three dimensions, and if viewed from certain angles, one will observe that the crystal is made up of atomic layers that are identical up to translation in their own planes. In this regard, the simplest theory of diffraction is **kinematical diffraction**. Kinematical diffraction treats the layers as identical and applies Bragg's law to find that the reflected intensity is equal to the incident at some critical angle θ_C at which the reflections of the incident wave from each layer are in phase. Given by $m\lambda = 2d \sin \theta_C$, this angle depends on the interplanar separation d and the wavelength of the incident beam λ. As the x-ray penetration depth is orders higher than the interplanar spacing d, there are enough reflections from each atomic plane to cancel completely at non-integer m; the result is a rocking curve consisting of a single, thin intensity equal to the incident at θ_C.

One thing the kinematical derivation doesn't account for is internal reflections. As the x-ray beam is reflected out of the material from some layer, it is also partly reflected back into itself by the previous; the theory that takes this into account is **dynamical diffraction**. This partial internal reflection itself then contributes both to the next layer's incident beam and to the present layer's beam—this is explained in more detail in the next sections (see , Figure 2.9, Figure 2.12), and gives rise to the need for a kinematical calculation, in which each beam splits into a reflected and transmitted beam at each the bottom and top interfaces of each layer infinitely many times. The result is that the peak is not only widened, but also shifted by some amount derived as Equation 2.33.

However, there is still one element missing, and that is absorption at each interface. The beam loses intensity as it enters the material, and we thus turn to **dynamical diffraction with absorption** to analyze our perfect crystal's rocking curves. Model rocking curves with this more complete theory are not only wide and shifted (by the same amount as without absorption), but are asymmetric; this is because changing the angle towards the normal of the plane increases the extinction depth, whereas changing it in the other direction decreases it, and this changes how much is absorbed in the layers.

We have already summarized kinematical diffraction, and will expand upon this and core concepts in Section 2.2. The theory of dynamical diffraction, which takes internal reflections and transmissions into account, is derived in Section 2.3, and its expansion to many layers in Section 2.3.6 is reconciled with the results of refractive kinematical diffraction in Section 2.3.5. Finally, absorption is factored into our derivations with a simple but powerful expansion to dynamical diffraction with absorption in Section 2.3.8.

2.2 Kinematical Diffraction

The kinematical approximation is the assumption that the x-ray scatters only once within the material; it is not used for perfect crystals in which this effect is significant. This approximation is illustrated in Figure 2.1.

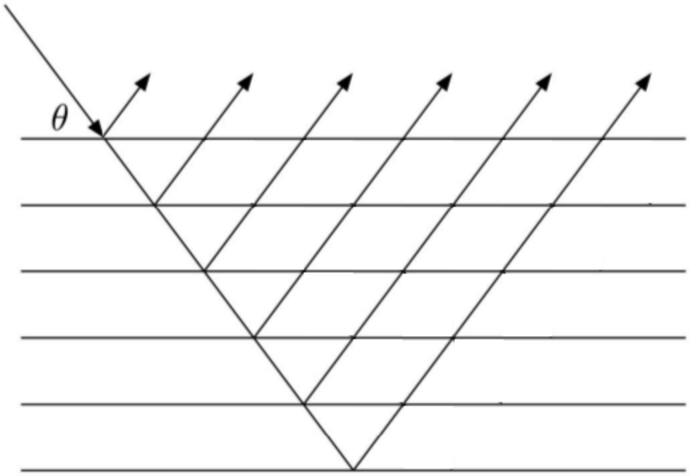


Figure 2.1: An illustration of the path of rays of light in kinematical diffraction. Horizontal lines are reflecting planes, such as the periodic atomic planes of a crystal. The rays interact constructively according to Bragg's Law.

2.2.1 Thomson Scattering Length (r_0)

The electrostatic potential energy of an electron is given through work by

$$\begin{aligned} U_E(r) = -W &= -\int_{\infty}^{r} q\boldsymbol{E} \cdot d\boldsymbol{s} \\ &= -\int_{\infty}^{r} q\left(\frac{q}{4\pi\epsilon_0 r^2}\right) d\boldsymbol{r} \\ &= -\frac{q^2}{4\pi\epsilon_0}\int_{\infty}^{r} r^{-2} d\boldsymbol{r} \\ &= \frac{q^2}{4\pi\epsilon_0 r} \end{aligned}$$

But the energy is also known to be $U = mc^2$, so that

$$mc^2 = U = \frac{q^2}{4\pi\epsilon_0 r} \implies r_0 = \frac{q^2}{4\pi\epsilon_0 mc^2},$$

where we have reserved the assumptions that $\boldsymbol{\nabla} \times \boldsymbol{E} = 0$ (conservative $\boldsymbol{E}-$field) and $\epsilon = \epsilon_0$ (free space). If it is an electron we are referring to, we can express the *Thomson Scattering Length* (r_o) of the electron:

$$r_0 = \frac{e^2}{4\pi\epsilon_0 m_e c^2} \tag{2.1}$$

2.2.2 Scattering Vector

The principle aim of this chapter is to understand how a perfect crystal will affect light incident upon it. Any observable effects in the light can be attributed to the charge distribution of the material. To this end, we consider the effect of some charge distribution on an incoming plane wave. A plane wave is a solution of Maxwell's equations with the perpendicular sinusoidal $\boldsymbol{E}$ and $\boldsymbol{B}$ fields propagating in the direction given by $\boldsymbol{E} \times \boldsymbol{B}$. The sinusoidal electric field in a plane wave can be expressed

$$\boldsymbol{E}(\boldsymbol{r}, t) = \boldsymbol{E}_0 \cos(\boldsymbol{k} \cdot \boldsymbol{r} - \omega t) = \text{Re}\left[\boldsymbol{E}_0 e^{i(\boldsymbol{k}\cdot\boldsymbol{r}-\omega t)}\right]$$

(and the $\boldsymbol{B}$ field similarly), and it is clear that this is the equation of a sinusoidal wave pulling in the $\pm\hat{E}_0$ direction along the $\hat{k}$ axis with wavelength $\lambda = \frac{2\pi}{|k|}$ moving to the right with angular velocity ω and with amplitude $|\boldsymbol{E}_0|$. The complex part of this and subsequent equations is to be ignored; we only make use of the complex plane because it offers mathematical simplifications, such as the multiplication of $e^{i\phi}$ to apply a phase shift (this is more cumbersome in the real domain). The incoming plane wave initially makes its way from the source to the distribution, undergoes optical effects, and then makes its way out of the distribution, where it can be observed. Our focus is on light that has been shining long enough for observable effects to have stabilized (ie, the source having been turned on and eventually being turned off are beyond our interests). In our region of interest, it then suffices to observe the waves frozen in space, where they are characterized uniquely by their wavelength and direction, which are combined in the $\boldsymbol{k}$ vector. We then represent the incoming wave as $\boldsymbol{k}$ with the understanding that this describes the wavevector of the electric (and magnetic) field propagating along that direction. The only information we lose in this notation is the direction that the electric (and magnetic) fields pull, $\hat{E}_0$ ($\hat{B}_0$), which is safely left ambiguous in our initial theoretical developments.

To begin, we analyze the effect on a pair of electrons. In this and further analyses, we restrict both the directions of the incident wave $\boldsymbol{k}$ and the scattered wave $\boldsymbol{k}'$. This is akin to shining an x-ray beam on a sample and placing a detector of finite size at some chosen angle for a reading. The results will be functions of $\boldsymbol{k}'$, so that they still hold the information for scattering at any angle. For example, the incident wave in Figure 2.2 is pointing to the right, and we've chosen the detection angle to be 45° in the plane of the page in denoting $\boldsymbol{k}'$.

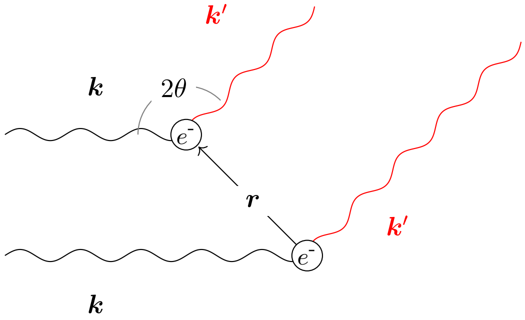


Figure 2.2: Demonstrating the usefulness of the scattering vector. One electron is at the origin at the lower right, and another is located at $\boldsymbol{r}$. To find the phase shift, we sum the shift from the delay of the incident vector k hitting the electron at r with the shift of the scattered vector from the origin reaching that of the other. This figure also defines θ as half the angle between the incident and measured angles.

The *Scattering Vector* ($\boldsymbol{Q} = \boldsymbol{k} - \boldsymbol{k}'$) is very useful in finding the phase difference between scattering waves of two scattering objects located a vector $\boldsymbol{r}$ apart. Take one such object, an electron, to be at the origin, another at $\boldsymbol{r}$. Since our detector defines k', it remains the same direction. The phase shift between the electrons is given by a retarding shift from the radians along k between them (ie, $\boldsymbol{k}\cdot\boldsymbol{r}$) and the backwards shift from the electron at $\boldsymbol{r}$, since it is closer to the detector. The former is a shift of $\boldsymbol{k}\cdot\boldsymbol{r} = \frac{2\pi}{\lambda}\hat{k}\cdot\boldsymbol{r}$, and the latter $\boldsymbol{k}'\cdot\boldsymbol{r}$. See Figure 2.2 for details. The phase shift of the electron at $\boldsymbol{r}$ with respect to that at the origin (denoted by the crosshairs) is then the sum of the delay of the incident wave in reaching it ($\boldsymbol{k}\cdot\boldsymbol{r}$), and the negated shift in distance reaching it from the electron at the origin ($-\boldsymbol{k}' \cdot \boldsymbol{r}$):

$$\Delta\phi = (\boldsymbol{k} - \boldsymbol{k}') \cdot \boldsymbol{r} = \boldsymbol{Q} \cdot r$$

The *Scattering Vector* $\boldsymbol{Q} = \boldsymbol{k} - \boldsymbol{k}'$ is particularly useful in describing an inelastic Bragg's law–assuming that the scattered wave has a higher wavelength (less energy) than the incident, $m\lambda = 2d\sin\theta$ will no longer do, since the λ are different, and we now have $\boldsymbol{Q} \cdot \boldsymbol{r} = 2\pi$, where $\boldsymbol{r}$ is the vector from the first to the second scatterer.[1] In the case of elastic absorption where $|k| = |k'|$, we have:

$$|\boldsymbol{Q}| = 2k\sin\theta = \frac{4\pi}{\lambda}\sin\theta \tag{2.2}$$

where θ is defined in Figure 2.2.

2.2.3 Atomic Form Factor

The scattering amplitude at Q of a plane wave incident on an isolated atom is given by its *Atomic Form Factor* $f^0(\boldsymbol{Q})$, where $\boldsymbol{Q} = \boldsymbol{k} - \boldsymbol{k}'$, $\boldsymbol{k}$ is set by the incident plane wave, and $\boldsymbol{k}'$ is given by where one places the detector. This can be expressed as the sum of contributions from the electron density $\rho(\boldsymbol{r})$ with phase shift given by $e^{i\boldsymbol{Q}\cdot\boldsymbol{r}}$ over the volume:

$$f^0(\boldsymbol{Q}) = \int \rho(\boldsymbol{r}) e^{i\boldsymbol{Q}\cdot\boldsymbol{r}} d\boldsymbol{r} \tag{2.3}$$

where $d\boldsymbol{r}$ is an infinitesimal volume.

The scattering is lessened by the attraction of the electron to the nucleus of the atom; this diminishment is denoted by $f'(\hbar\omega)$, the bound reduction. At higher energies, when the electron is less bound to the atom, the bound reduction becomes negligible.

Finally, the scattered wave is further diminished by f'', the *phase lag dissipation* from the phase lag between the driving plane wave and the electron response. It also accounts for attenuation due to damping by the bound electron, and is dependent on the strength of the $\boldsymbol{E}$-field on the electron, which is given by the energy $\hbar\omega$. Together, f' and f'' are known as the dispersion corrections. Summing these corrections to f^0 gives the corrected form factor, which carries both angular and energy dependence:

$$f(\boldsymbol{Q}, \hbar\omega) = f^0(\boldsymbol{Q}) + f'(\hbar\omega) + if''(\hbar\omega) \tag{2.4}$$

We can calculate $f^0(\boldsymbol{Q})$ by finding the electron distribution $\rho(\boldsymbol{r})$ of the atom using computational methods such as relativistic Hartree-Fock and Dirac-Slater methods and evaluating Equation 2.3 for various

$\boldsymbol{Q}$. This has been done in the Crystallographic Tables[6], where a method is derived for accurate interpolation of these values; given coefficients $\{a_i, b_i\}_{i=1}^4$ and c, a sample of which are given in Table 3.3, one can find the interpolated f^0 by evaluating

$$f^0(\boldsymbol{Q}) = \sum_{j=1}^{4} a_j e^{-b_j Q^2} + c \tag{2.5}$$

with the scattering length given by $-r_0 f_0$, negative because of the double integral in getting from $F = qE = q\cos(kx - \omega t)$ to $x = \iint \frac{F}{m} dtdt$. The International Tables of Crystallography also provide values of f' and f'' for various energies[5], samples of which are given in Table 3.4)—however, this is not sufficient for our needs, as the Crystallographic Tables do not cover the understandably eratic behavior near the band edges. Instead, we use more accurate data given around the K-edge from another source[7], made available online[9]. Interpolation methods are then used to obtain their values at energies between those given—this paper uses simple linear interpolation.

Crystal structures are categorized by their symmetries into 14 Bravais lattices in three dimensions, ranging from rhombohedral to hexagonal in shape. The materials we are interested in are all cubic lattice systems, meaning that their conventional *unit cell*—an arrangement of atoms that repeats throughout the crystal in the Cartesian directions—is a cube. We restrict our derivations to cubic lattices for the remainder of our derivations. Cloning this cube at each of its faces recursively then creates the crystal structure. To exploit this, we express the vector from our origin to each atom in the entire crystal as $R_i + r_j$, where R_i points to the origin of the ith unit cell, and r_j to the jth atom within that unit cell. Iterating over j gives us insight into the properties of a single unit cell, which simplifies calculations over complex symmetries of atoms to calculations over simple cubic symmetries of unit cells.

Explicitly, we can split the scattering amplitude of the entire crystal into

$$F^{\text{crystal}}(\boldsymbol{Q}) = \left(\sum_n e^{i\boldsymbol{Q}\cdot\boldsymbol{R}_n}\right)\left(\sum_j f_j(\boldsymbol{Q}) e^{i\boldsymbol{Q}\cdot\boldsymbol{r}_j}\right) = F^{\text{u.c.}}(\boldsymbol{Q}) \sum_n e^{i\boldsymbol{Q}\cdot\boldsymbol{R}_n}.$$

where we've defined the form factor of a unit cell, the *Unit Cell Structure Factor*, as

$$F^{\text{u.c.}}(\boldsymbol{Q}) = \sum_j f_j(\boldsymbol{Q}) e^{i\boldsymbol{Q}\cdot\boldsymbol{r}_j}. \tag{2.6}$$

This method is explained more thoroughly in §2.2.5 and is applied dynamically to fully exploit symmetry in §2.3.

Of the terms in the product, $F^{\text{u.c.}}(\boldsymbol{Q})$ has been discussed and will be simplified in the following sections. The real challenge becomes summing over every unit cell in the lattice—effectively infinite, we will eventually exploit absorption through the crystal layers to get meaningful results.

2.2.4 Reciprocal Lattice

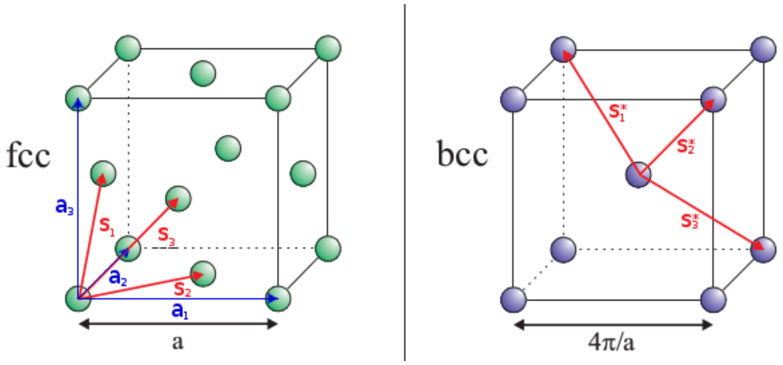


Figure 2.3: Face-centered cubic crystal lattice (left) and its reciprocal lattice (right). Note that the reciprocal lattice of the face-centered cubic (fcc) lattice is body-centered cubic (bcc).

Whenever we write a vector as a column matrix, it is in Cartesian coordinates:

$$\boldsymbol{r} = x\hat{x} + y\hat{y} + z\hat{z} = x\begin{bmatrix}1\\0\\0\end{bmatrix} + y\begin{bmatrix}0\\1\\0\end{bmatrix} + z\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}x\\y\\z\end{bmatrix}, \qquad x, y, z \in \mathbb{R}$$

If we want to denote the position of each *unit cell* in the lattice, we seek to describe only the position of the corner we take to be the origin in each unit cell (the corner from which the arrows in Figure 2.3 originate), and for this task we scale the Cartesian basis by the lattice parameter a in creating the new basis $\{\boldsymbol{a}_i\}$ where

$$\boldsymbol{a}_1 = a\hat{x} \quad \boldsymbol{a}_2 = a\hat{y} \quad \boldsymbol{a}_3 = a\hat{z},$$

and this has the benefit that any unit cell can be specified as an *integer* linear combination of these basis vectors:

$$\boldsymbol{R}_i = x\boldsymbol{a}_1 + y\boldsymbol{a}_2 + z\boldsymbol{a}_3 \qquad x, y, z \in \mathbb{Z}.$$

If we now seek to describe the positions of the atoms within the unit cell, a more fitting basis is the one drawn in Figure 2.3 consisting of $\{\boldsymbol{s}_i\}$ where similarly, the position of any atom within the unit cell

$$r_j = x\boldsymbol{s}_1 + y\boldsymbol{s}_2 + z\boldsymbol{s}_3, \qquad x, y, z \in \mathbb{Z}$$

can be expressed as an *integer* linear combination of these basis vectors.

The "crystal basis" which is just a scaled Cartesian basis $\{\boldsymbol{a}_j\}$ is then useful for describing terms having to do with the unit cell within the crystal, whereas the "unit cell basis" $\{\boldsymbol{s}_j\}$ is useful for describing the positions of atoms relative to each unit cell.

A given basis $\{\boldsymbol{q}_j\}$ has a **reciprocal space**, which is actually the same space but spanned by a reciprocal basis, defined by

$$\boldsymbol{q}_i \cdot \boldsymbol{q}_j = 2\pi\delta_{ij}. \tag{2.7}$$

Both the crystal basis and the unit cell basis have corresponding reciprocal bases. In the crystal lattice $\{\boldsymbol{a}_i\}$, it is clear that the condition $\boldsymbol{a}_i \cdot \boldsymbol{a}_j^* = 2\pi\delta_{ij}$ is satisfied by $\boldsymbol{a}_i^* = \frac{2\pi}{a^2}\boldsymbol{a}_i$.

In the crystal reciprocal basis, we can now show that a vector given by

$$\boldsymbol{G} = h\boldsymbol{a}_1^* + k\boldsymbol{a}_2^* + l\boldsymbol{a}_3^*, \qquad h, k, l \in \mathbb{Z} \tag{2.8}$$

is perpendicular to a plane crossing in the Cartesian basis the points $(\frac{a}{h}, 0, 0)$, $(0, \frac{a}{k}, 0)$ and $(0, 0, \frac{a}{l})$ and the family of planes parallel to it (ie, the family of Miller planes given by $(h\,k\,l)$) by noting that the equation of such a plane is given by $xh + yk + zl = a$ and its normal vector is then

$$\boldsymbol{n} = h\hat{x} + k\hat{y} + l\hat{z}$$

whereas the equation for $\boldsymbol{G}$ can be rewritten in the $\{\boldsymbol{a}_i\}$ basis of the simple cubic lattice to take the form

$$\boldsymbol{G} = \frac{2\pi}{a^2}\left[h\boldsymbol{a}_1 + k\boldsymbol{a}_2 + l\boldsymbol{a}_3\right] \tag{2.9}$$

which is parallel to $\boldsymbol{n}$ and scaled by a factor of $\frac{2\pi}{a}$, so that $\boldsymbol{G}$ is perpendicular to the plane. Hence $\boldsymbol{G}$ is perpendicular to the family of Miller planes $(h\,k\,l)$. The interplanar distance of the Miller planes is given by definition as the distance from the origin to the closes plane (which crosses the x, y, and z axes at the aforementioned points), and this is given by taking the projection of any vector extending from the origin to the plane (eg, $\frac{\boldsymbol{a}_1}{h}$) onto $\boldsymbol{G}$, given by

$$d_{hkl} = \frac{\boldsymbol{a}_1}{h} \cdot \frac{\boldsymbol{G}}{|\boldsymbol{G}|} = \frac{2\pi}{|\boldsymbol{G}|} \tag{2.10}$$

with the subscript *hkl* emphasizing the dependence of the interplanar distance on which family of Miller planes we choose. Plugging Equation 2.9 into this yields for the simple cubic lattice

$$d_{hkl} = \frac{a}{\sqrt{h^2 + k^2 + l^2}}. \tag{2.11}$$

To find the unit cell reciprocal lattice basis vectors $\left\{\boldsymbol{s}_j^*\right\}_{j=1}^3$, we solve uniquely for the equation $\boldsymbol{s}_j \cdot \boldsymbol{s}_j^* = 2\pi\delta_{ij}$, where represents the Kronecker delta. Denoting the volume of the unit cell

$$v_c = \boldsymbol{s}_1 \cdot (\boldsymbol{s}_2 \times \boldsymbol{s}_3) = \begin{bmatrix} 0 \\ \frac{a}{2} \\ \frac{a}{2} \end{bmatrix} \cdot \left(\begin{bmatrix} \frac{a}{2} \\ 0 \\ \frac{a}{2} \end{bmatrix} \times \begin{bmatrix} \frac{a}{2} \\ \frac{a}{2} \\ 0 \end{bmatrix} \right) = \left(\frac{a}{2}\right)^3 \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right) = \frac{a^3}{8} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} = \frac{a^3}{4},$$

we note that in order for the reciprocal lattice $\left\{\boldsymbol{s}_j^*\right\}$ to satisfy its definition $\boldsymbol{s}_i \cdot \boldsymbol{s}_j^* = 2\pi\delta_{ij}$, (1) $\boldsymbol{s}_j^*$ must be orthogonal to $\boldsymbol{s}_k^*$, $k \neq j$, and (2) $\boldsymbol{s}_j^* \cdot \boldsymbol{s}_j$ must have magnitude 2π. The first condition is achieved if we take s_j^* to be the cross-product of the other two vectors (in cyclic order), since the cross product of two vectors is orthogonal to each. This yields for $i \neq j \neq k$ with $\boldsymbol{s}_i \times \boldsymbol{s}_j = \boldsymbol{s}_k$ that

$$\boldsymbol{s}_q \cdot \boldsymbol{s}_i^* = \boldsymbol{s}_q \cdot (\boldsymbol{s}_j \times \boldsymbol{s}_k) = \delta_{qi} \boldsymbol{s}_i \cdot (\boldsymbol{s}_j \times \boldsymbol{s}_k),$$

so the second condition is satisfied by multiplying our $\boldsymbol{s}_i^*$ by the factor

$$\frac{2\pi}{\boldsymbol{s}_i \cdot (\boldsymbol{s}_j \times \boldsymbol{s}_k)} = \frac{2\pi}{v_c} = \frac{8\pi}{a^3}$$

where a is the form factor.

$$\begin{aligned}
\boldsymbol{s}_1^* &= \frac{8\pi}{a^3} \boldsymbol{s}_2 \times \boldsymbol{s}_3 \\
&= \frac{8\pi}{a^3} \left(\frac{1}{2} (\boldsymbol{a}_3 + \boldsymbol{a}_1) \right) \times \left(\frac{1}{2} (\boldsymbol{a}_1 + \boldsymbol{a}_2) \right) \\
&= \frac{2\pi}{a^3} \left(\boldsymbol{a}_3 \times \boldsymbol{a}_1 + \boldsymbol{a}_3 \times \boldsymbol{a}_2 + \cancel{\boldsymbol{a}_1 \times \boldsymbol{a}_1} + \boldsymbol{a}_1 \times \boldsymbol{a}_2 \right) \\
&= \frac{2\pi}{a} \left(\hat{a}_3 \times \hat{a}_2 + \hat{a}_3 \times \hat{a}_1 + \hat{a}_1 \times \hat{a}_2 \right) \\
&= \frac{2\pi}{a} \left(-\hat{a}_1 + \hat{a}_2 + \hat{a}_3 \right)
\end{aligned}$$

Likewise,

$$\boldsymbol{s}_2^* = \frac{2\pi}{a} \left(\hat{a}_1 - \hat{a}_2 + \hat{a}_3 \right) \qquad \boldsymbol{s}_3^* = \frac{2\pi}{a} \left(\hat{a}_1 + \hat{a}_2 - \hat{a}_3 \right)$$

Hence we come to find the result that is Figure 2.3, a cubic reciprocal lattice.

2.2.5 Lattice Sum

Equation 2.6 defines F^{crystal} as the product of the unit cell structure factor and the *lattice sum*, the latter consisting of a sum over a large order of magnitude (the number of unit cells in a sample). The claim is made that unless all contributions from the lattice sum are in phase, the sum is of order unity (which, as our form factor is in units of r_0, is very small)[1]; this corresponds to $\boldsymbol{Q} \cdot \boldsymbol{R}_n = 2\pi m$, $m \in \mathbb{Z}$. The solution is integer coordinates in reciprocal space, defined in Equation 2.8, which yields:

$$\boldsymbol{G} \cdot \boldsymbol{R}_n = (h\boldsymbol{a}_1^* + k\boldsymbol{a}_2^* + l\boldsymbol{a}_3^*) \cdot (x\boldsymbol{a}_1 + y\boldsymbol{a}_2 + z\boldsymbol{a}_3) = 2\pi(hx + ky + lz) = 2\pi m, \qquad m \in \mathbb{Z}$$

Since $h, k, l, x, y, z \in \mathbb{Z}$. This clarifies the role of $\boldsymbol{G}$; **$\boldsymbol{G}$ is the $\boldsymbol{Q}$ at which constructive interference occurs** in our derivation—remember that $\boldsymbol{Q} = \boldsymbol{k} - \boldsymbol{k}'$, so $\boldsymbol{Q}$ is an expression of the angle between the incident and reflected beam (neglecting absorption, for now).

In evaluating Equation 2.6, we carry out our calculations for only the four atoms located at the origin and basis vectors, as show in Figure 2.3; ie, $\boldsymbol{r}_0 = \boldsymbol{0}, \boldsymbol{r}_1 = \boldsymbol{s}_1, \boldsymbol{r}_2 = \boldsymbol{s}_2$, and $\boldsymbol{r}_3 = \boldsymbol{s}_3$. This is justified once we realize that the cubic unit cell is to be repeated at every corner, and cloning these four atoms at every unit cell site R_i yields the fcc structure (normally, one performs calculations on every atom in the cell, taking whatever fraction of the atom is in the cell to avoid redundancy). For given angles of incidence and detection (reflection) corresponding to constructive interference and expressed by $\boldsymbol{G}$, Equation 2.6 can be expressed

$$F^{\text{fcc}}(\boldsymbol{G}) = f(\boldsymbol{G}) \sum_{j=0}^{3} e^{i\boldsymbol{G}\cdot\boldsymbol{r}_j}, \tag{2.6}$$

where F is the form factor for the unit cell and $\boldsymbol{r}_j$ points to each of the four atoms which we choose to represent the unit cell. As these atoms are located at the origin and the three basis vectors, they can be expressed:

$$\boldsymbol{r}_0 = 0 \qquad \boldsymbol{r}_1 = \boldsymbol{s}_1 = \frac{a}{2}(\hat{y} + \hat{z}) \qquad \boldsymbol{r}_2 = \boldsymbol{s}_2 = \frac{a}{2}(\hat{z} + \hat{x}) \qquad \boldsymbol{r}_3 = \boldsymbol{s}_3 = \frac{a}{2}(\hat{x} + \hat{y}) \tag{2.12}$$

Using Equation 2.9, we can express $\boldsymbol{G}$ in the non-reciprocal crystal basis as

$$\boldsymbol{G} = h\boldsymbol{a}_1^* + k\boldsymbol{a}_2^* + l\boldsymbol{a}_3^* = \frac{2\pi}{a^2}\left[h\boldsymbol{a}_1 + k\boldsymbol{a}_2 + l\boldsymbol{a}_3\right] = \frac{2\pi}{a}\left[h\hat{a}_1 + k\hat{a}_2 + l\hat{a}_3\right]$$

Combining this with Equation 2.12, we can solve Equation 2.6:

$$F_{\text{hkl}}^{\text{fcc}} = f(\boldsymbol{G}) \sum_{j=0}^{3} e^{i\boldsymbol{G}\cdot\boldsymbol{r}_j} = f(\boldsymbol{G}) \sum_{j=0}^{3} \left[e^{i0} + e^{i\pi(h+k)} + e^{i\pi(k+l)} + e^{i\pi(h+l)} \right]$$

Ignoring $e^{i0} = 1$ and observing the pairity of h, k, l, if two are odd and one even, the sum of the odds will be even and the other two odd. The same happens if one is odd and two even. This results in the cancellation of the sum, as $e^{2\pi n} = 1$ and $e^{\pi i} = -1$. The sum is then zero, and the reflection said to be *forbidden*, if h, k, l are not all even or all odd. If they are all even or all odd, their pairwise sums are even, and

$$F_{\text{hkl}}^{\text{fcc}} = 4f(\boldsymbol{G}) \qquad \text{(allowed reflections)}$$

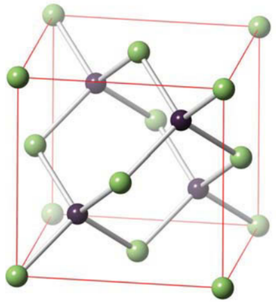


Figure 2.4: Diamond lattice structure. Note that it is a convolution of two monoatomic fcc structures.

Using these results, the trick to expanding the method to silicon blende crystals consisting of two atoms is to treat them as two monoatomic crystals, and add the two with a shift (similar to the shift of each atom within the unit cell), remembering to multiply them by their appropriate atomic form factors. A subtlety that arises in polyatomic crystals is that what were previously forbidden reflections from h, k, l causing the unit cell structure factor to add to zero are no longer necessarily forbidden, due to the shift between the monoatomic crystals and the differing atomic form factors of each. In the diamond structure shown in Figure 2.4, the "shifted" lattice is at $\frac{1}{4}(a, a, a)$, as can be verified in the image. Suppose we choose the origin atom and those in the fcc lattice corresponding to it as Gallium, and the "shifted" atoms Arsenic, arbitrarily. It suffices to add a phase shift for the distance between the two, in the form of $e^{i2\pi(h/4+k/4+l/4)}$; this will add $h/4, k/4$, or $l/4$ to r. Note that $\boldsymbol{G}\cdot\boldsymbol{r}$ adds a factor of $\frac{2\pi}{a}$ to r, which already has a factor of a. The 2π in our phase shift assures that the shift is applied to $\boldsymbol{r}$, not $\boldsymbol{G}$.

2.3 Dynamical Diffraction

The kinematical derivation above assumes that each diffraction process is independent. This doesn't account for the refraction shown in Figure 2.9 for a single slab; in fact, the x-rays are reflected back into the material and refract over and over again throughout the different layers as in Figure 2.5—hence the dynamical theory of diffraction.

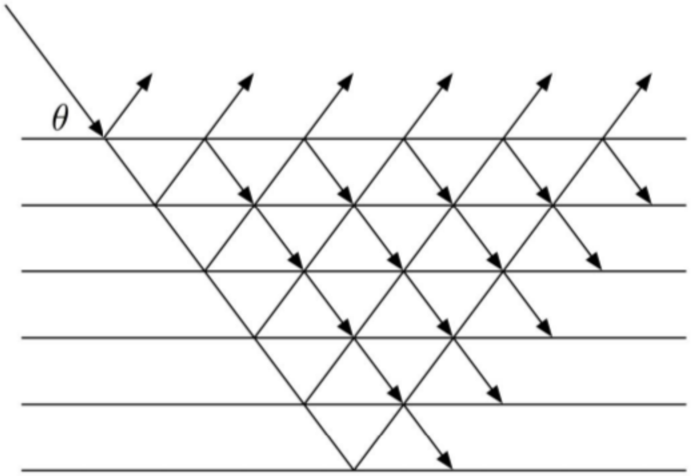


Figure 2.5: Illustration of dynamical derivation. The dynamical theory accounts for internal reflections dynamically through each layer.

2.3.1 The Refractive Index

Suppose we strike a thin medium of thickness Δ with a plane wave E_0 whose propagation is normal to the slab from the left, a distance R_0 away, and record a frame frozen in time after the wave has reached R_0 to the right of the slab (Figure 2.6).

The incident field, frozen in time, can be denoted $E_0 = e^{ikz}$ emanating R_0 away from it. There are two equivalent ways of finding the electric field at R_0 to the right (or R_0, and we will refer to R_0 to the left as $-R_0$); by using refractive principles, or by solving for the collective electron scatter. We compare the two to obtain a relationship between the index of refraction and scattering properties.

Refraction

In the refractive case, the wave travels $R_0 - \frac{\Delta}{2}$ with wavelength λ and wave vector k, then travels Δ with wavelength $\frac{\lambda_1}{n}$ since

$$\frac{\lambda_2}{\lambda} = \frac{v_2/f}{c/f} = \frac{v_2}{c} = \frac{1}{n}$$

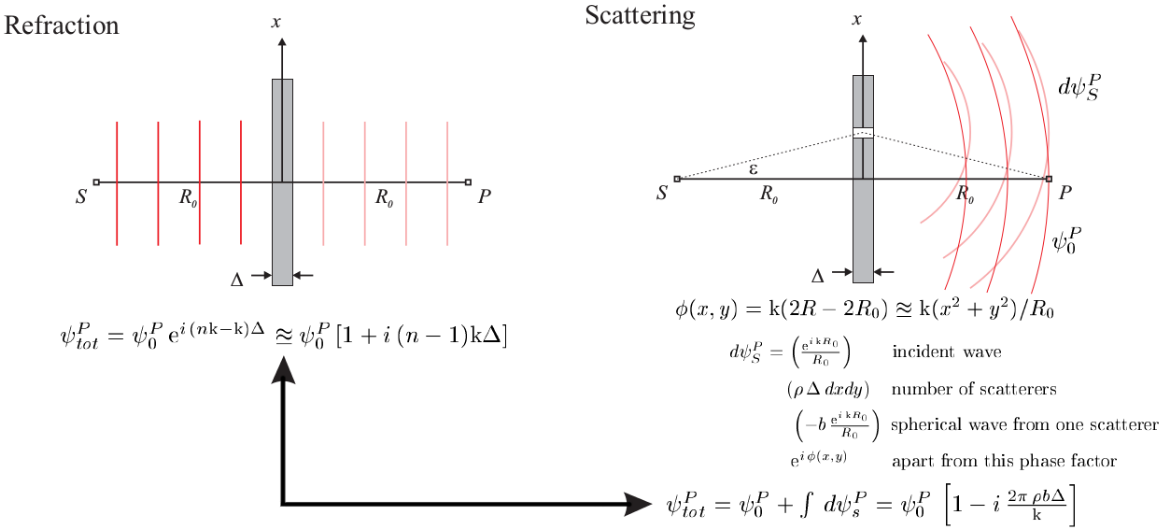

Refraction
x
S
R_0
R_0
P
Δ
$\psi_{tot}^P = \psi_0^P \, \mathrm{e}^{i\,(n\mathrm{k}-\mathrm{k})\Delta} \approxeq \psi_0^P\,[1 + i\,(n-1)\mathrm{k}\Delta]$
Scattering
x
$d\psi_S^P$
ε
S
R_0
R_0
P
ψ_0^P
Δ
$\phi(x,y) = \mathrm{k}(2R - 2R_0) \approxeq \mathrm{k}(x^2+y^2)/R_0$
$d\psi_S^P = \left(\frac{\mathrm{e}^{i\,\mathrm{k}R_0}}{R_0}\right)$ incident wave
$(\rho\,\Delta\,dxdy)$ number of scatterers
$\left(-b\,\frac{\mathrm{e}^{i\,\mathrm{k}R_0}}{R_0}\right)$ spherical wave from one scatterer
$\mathrm{e}^{i\,\phi(x,y)}$ apart from this phase factor
$\psi_{tot}^P = \psi_0^P + \int\, d\psi_s^P = \psi_0^P\,\left[1 - i\,\frac{2\pi\,\rho b\Delta}{\mathrm{k}}\right]$

Figure 2.6: Setup for refraction and scattering derivations—absorption is ignored for now.

and wave vector $k_2 = \frac{2\pi}{\lambda_2} = \frac{2\pi}{\lambda_1}n = nk$. The wave then travels $R_0 - \frac{\Delta}{2}$ again, for a total phase shift from the incident given by:

$$\frac{E_0 e^{2i(R_0 - \frac{\Delta}{2})k + nk\Delta}}{E_0 e^{2iR_0 k}} = e^{i\Delta(n-1)k}$$

Hence we observe a phase shift of $e^{i(n-1)k\Delta}$ to the original incident wave E_0, so that the total wave measured instantaneously at R_0 is

$$E_{tot}(R_0) = E_0 e^{in(k-1)\Delta} \approx E_0\,[1 + i(n-1)k\Delta] \tag{2.13}$$

using the Taylor approximation, where $E_0 = E_0(R_0)$ implicitly.

Scattering

Each electron scatters the wave in phase if $n > 1$ and precisely out of phase when $n < 1$, where n is the material index of refraction. As the latter is most often the case in x-rays, which deviate very slightly from 1, we want to denote

$$n = 1 - \delta. \tag{2.14}$$

Following Figure 2.6, we take an infinitesimal section of the slab of area Δdx at some height x from the source S, with $R_0 \gg x$. The distance that the wave travels before reaching this section is

$$R = \sqrt{R_0^2 + x^2} \approx R_0\left[1 + \frac{1}{2}\left(\frac{x}{R_0}\right)^2\right]$$

using the Maclaurin series. As the wave hits dx and is refracted unto point P, it has travelled

$$\left(2R_0 + \frac{2x^2}{2R_0}\right) = R_0 + \frac{x^2}{R_0}.$$

The extra distance $\frac{x^2}{R_0}$ from the plane wave taking the shortest path accounts for a phase difference of $e^{ik\frac{x^2}{R_0}}$. As our slab is three dimensional (expanding infinitely in the xy–plane and with thickness Δ in the z), we expand to cylindrical coordinates for a phase shift from the volume element $\Delta dxdy$ at (x, y) of $e^{ik\frac{r^2}{R_0}}$.

The spherical wave solution to Maxwell's Equations is of the form $\frac{e^{ikr}}{r}$. Our incident wave is of the form $\frac{e^{ikR_0}}{R_0}$ throughout the origin, and $\frac{e^{ikR_0}}{R_0}e^{ik\frac{r^2}{2R_0}}$ for each point $(r, \theta, 0)$ in our slab (which is very thin, so this approximately holds for $|z| < \Delta$.) This is the wave received by each electron, which scatters it out of phase (assuming $n > 1$) with attenuation r_0 given by the Thomson scattering length. The wave is now of the form $-r_0\frac{e^{ikR_0}}{R_0}e^{ik\frac{r^2}{2R_0}}$ as it returns the length $|\boldsymbol{R}_0 + \boldsymbol{r}|$ to our observation point, contributing another phase shift and attenuation of $\frac{e^{ikR_0}}{R_0}e^{ik\frac{r^2}{2R_0}}$ as measured at the observation point. Thus, from each electron, a scattered contribution of

$$-r_0\frac{e^{2ikR_0}}{R_0^2}e^{ik\frac{r^2}{R_0}}$$

is measured. To integrate over the slab with volume density ρ, we first note that for $a \in \mathbb{R}^+$,

$$\int_0^\infty re^{-ar^2}dr \left| \begin{array}{l} u = ar^2 \\ du = 2ardr \end{array} \right| = \frac{1}{2a}\int_0^\infty e^{-u}du = \frac{1}{2a}.$$

This can be extended to any complex a; we choose $a = \frac{-ik}{R_0}$ to obtain, with the approximation $\Delta^2 \approx 0$,

$$\begin{aligned}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}}\int_0^{2\pi}\int_0^\infty -r_0\frac{e^{2ikR_0}}{R_0^2}e^{ik\frac{r^2}{R_0}}\rho rdrd\theta dz &= -r_0\rho\frac{e^{2ikR_0}}{R_0^2}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}}dz\int_0^{2\pi}d\theta\int_0^\infty re^{-ar^2}rdr \\ &= -2\pi\Delta\rho r_0\frac{e^{2ikR_0}}{R_0^2}\left(\frac{1}{2a}\right) \\ &= -\frac{i\rho r_0\pi\Delta}{k}\frac{e^{2ikR_0}}{R_0}\end{aligned}$$

The wave at our point of observation is then the initial incident wave (ignoring absorption) added to this contribution from the slab. The incident wave upon reaching the observation point is $\frac{e^{ik2R_0}}{R_0}$ so that the total wave at our point of observation is

$$E_{tot} = E_0 + E_S = E_0\left[1 - \frac{2\pi i\rho r_0\Delta}{k}\right],$$

and substituting that by the definition of f^0 as discussed above, the electron density $\rho = \rho_{at} f^0(\boldsymbol{Q})$ with ρ_{at} the atomic number density and $\boldsymbol{Q} = 0$ (as our wave goes directly through the material at $\theta = 0$), we obtain

$$E_{tot} = E_0 \left[1 - i\frac{2\pi\rho_{at} f^0(0) r_0 \Delta}{k}\right] = E_0 \left[1 - ig_0\right] \approx E_0 e^{-ig_0} \tag{2.15}$$

with

$$g_0 = \frac{\lambda\rho_{at} f^0(0) r_0 \Delta}{\sin\theta} \tag{2.16}$$

is the phase shift due to the slab, and Δ was substituted with its equivalent with a change in incident angle by $\frac{\Delta}{\sin\theta}$.

We compare this result to its equivalent in the subsection above to obtain

$$E_0 \left[1 + i(n-1)k\Delta\right] = E_0 \left[1 - i\frac{2\pi\rho_{at} f^0(0) r_0 \Delta}{k}\right]$$

with solution

$$\delta = 1 - n = \frac{2\pi\rho r_0}{k^2} = \frac{2\pi\rho_{at} f^0(0) r_0}{k^2} \tag{2.17}$$

having used Equation 2.14 as the definition of δ.

Absorption

The intensity of a beam diminishes by factor $e^{-\mu z}$, where z is the distance within the material travelled and μ the absorption coefficient of the material. We can include this in our math elegantly by allowing n to be complex. As refractive indices in the x-ray domain are often just less than 1, we want to denote

$$n = 1 - \delta + i\frac{\mu}{k} \tag{2.18}$$

Then, for a distance z, we can combine the phase shift and attenuation into $e^{inkz} = e^{i(1-\delta)kz}e^{-\mu z}$ with δ is given by Equation 2.17. If we set $f'' = -\frac{\mu k}{2\pi\rho_{at} r_0}$, we obtain

$$n = 1 - \frac{2\pi\rho_{at} r_0}{k^2}\left(f^0(0) + if''\right)$$

We submit without derivation that additional dispersion corrections take the form of f' and contribute to corrections to f'' with the result

$$n = 1 - \frac{2\pi\rho_{at} r_0}{k^2}\left(f^0(0) + f' + if''\right). \tag{2.19}$$

These are the components that make up the form factor given in Equation 2.4

2.3.2 Fresnel's Equations

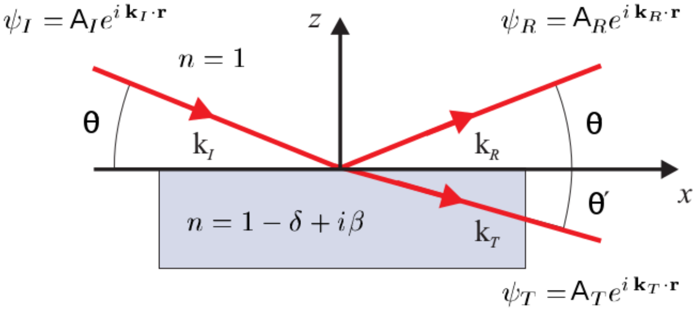

$\psi_I = A_I e^{i\,\mathbf{k}_I\cdot\mathbf{r}}$
z
$\psi_R = A_R e^{i\,\mathbf{k}_R\cdot\mathbf{r}}$
$n = 1$
θ
k_I
k_R
θ
x
θ′
$n = 1 - \delta + i\beta$
k_T
$\psi_T = A_T e^{i\,\mathbf{k}_T\cdot\mathbf{r}}$

Figure 2.7: An incident wave $\psi_I = A_I e^{i\boldsymbol{k}_I\cdot\boldsymbol{r}}$ hits a sharp interface at $z = 0$ with angle θ and is reflected at the same angle as $\psi_R = A_R e^{i\boldsymbol{k}_R\cdot\boldsymbol{r}}$ and transmitted at angle θ' as $\psi_T = A_T e^{i\boldsymbol{k}\cdot\boldsymbol{r}}$. Since $\theta, \beta \ll 1$, it is assumed that θ and θ' are very small.

We know that an incident wave upon a boundary of changing refractive index and its derivative must be continuous at this boundary. In Figure 2.7, this means that, do to the point where this ray meets the material having been chosen as the origin,

$$\psi_I(0) + \psi_R(0) = \psi_T(0) \qquad \frac{d\psi_I}{dt}(0) + \frac{d\psi_R}{dt}(0) = \frac{d\psi_T}{dt}(0) \tag{2.20}$$

$$A_I e^{i\boldsymbol{k}_I\cdot\mathbf{0}} + A_R e^{i\boldsymbol{k}_R\cdot\mathbf{0}} = A_T e^{i\boldsymbol{k}_T\cdot\mathbf{0}} \qquad A_I\boldsymbol{k}_I + A_R\boldsymbol{k}_R = A_T\boldsymbol{k}_T \tag{2.21}$$

$$A_I + A_R = A_T \tag{2.22}$$

With the interface surface as reference, taking the parallel and perpendicular parts of the rays separately, and remembering that the wavevector is k outside of the interface and nk within it, we obtain:

$$A_I k\cos\theta + A_R\cos\theta = A_T(nk)\cos\theta'$$

for the perpendicular and

$$A_R k\sin\theta + A_T(nk)\sin\theta' = A_I k\sin\theta$$

for the parallel.[1] We derive Snell's Law from the former paired with Equation 2.22:

$$(A_I + A_R)\,k\cos\theta = A_T(nk)\cos\theta' \implies \cos\theta = n\cos\theta'$$

[1] A note about θ: This is not necessarily the angle to the surface of the crystal, but the angle to the Miller planes that we are reflecting off of. Since as we shall see reflection is limited to small glancing angles, we can often restrict our reflections to just one Miller plane. Thus, θ is not an absolute angle, but one relative to reflecting Miller planes.

and recast the latter using Equation 2.22 as

$$(A_R - A_I)\sin\theta = A_T n \sin\theta' = (A_R + A_I)\, n \sin\theta'.$$

Expanding cosines with their Maclaurin equivalents for the critical angle θ_c in Snell's law yields

$$\begin{aligned}
\cos\theta &= n\cos\theta' \\
\left(1 - \frac{\theta^2}{2}\right) &= n\left(1 - \frac{\theta'^2}{2}\right) \\
&= n && \text{(total external reflection)} \\
\theta_c^2 &= 2(1-n) = 2\delta \\
\theta_c &= \sqrt{2\delta} && (2.23)
\end{aligned}$$

where we have ignored the attenuation term $i\beta$ as it is not relevant. This ensures the result

$$\frac{A_R - A_I}{A_R + A_I} = \frac{n\sin\theta'}{\sin\theta} \approx \frac{\theta'}{\theta}$$

having expanded the sines for small angles using the Maclaurin series, and having taken advantage of $n \approx 1$.

We use this equation to derive the Fresnel equations:

$$\begin{aligned}
(A_I - A_R) &= \frac{\theta'}{\theta}(A_I + A_R) & 2A_I &= (A_I + A_R) + (A_I - A_R) \\
A_I\left(\theta - n\theta'\right) &= A_R\left(n\theta' + \theta\right) & &= A_T + \frac{\theta'}{\theta}A_T \\
r = \frac{A_R}{A_I} &= \frac{\theta - \theta'}{\theta + \theta'} & t = \frac{A_T}{A_I} &= \frac{2\theta}{\theta + \theta'}
\end{aligned}$$

where r and t are the relative reflected and transmitted amplitudes. While less intuitive, the equations are more helpful to us using the wavevector transfer $Q = 2k\sin\theta \approx 2k\theta$ (by definition), with $Q_c = 2k\sin\theta_c = 2k\theta_c$. We introduce the dimensionless

$$q = \frac{Q}{Q_c} \approx \frac{2k\theta}{Q_c} \qquad q' = \frac{Q'}{Q_c} \approx \frac{2k\theta'}{Q_c} \tag{2.24}$$

so that $q \propto \theta$ and $q' \propto \theta'$ gives the same ratios in the Fresnel equations

$$r(q) = \frac{q - q'}{q + q'} \quad t(q) = \frac{2q}{q + q'} \quad \Lambda(q) = \frac{1}{Q_c \mathrm{Im}\left(q'\right)} \tag{2.25}$$

where the latter Λ is the penetration depth.

2.3.3 Perfect Crystals, Imperfect Monochromators

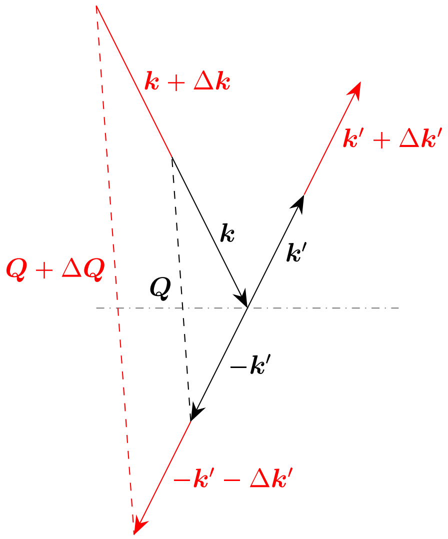

$\boldsymbol{k} + \Delta\boldsymbol{k}$
$\boldsymbol{k}' + \Delta\boldsymbol{k}'$
$\boldsymbol{k}$
$\boldsymbol{k}'$
$\boldsymbol{Q} + \Delta\boldsymbol{Q}$
$\boldsymbol{Q}$
$-\boldsymbol{k}'$
$-\boldsymbol{k}' - \Delta\boldsymbol{k}'$

Figure 2.8: In this symmetric reflection off of the dotted/dashed line as a surface, we see the initial (blue) incident ray $\boldsymbol{k}$ reflected symmetrically but attenuated to form $\boldsymbol{k}'$. We negate the latter to obtain $\boldsymbol{Q} = \boldsymbol{k} - \boldsymbol{k}'$. If we send another ray of a different energy, with wavevector $\boldsymbol{k} + \boldsymbol{\Delta k}$, we negate the reflected $\boldsymbol{k' + \Delta k'}$ to obtain $\boldsymbol{Q + \Delta Q}$. We demonstrate in the text that these should be similar triangles if $k' = ak$ (linear attenuation).

Monochromators make use of Bragg's law to separate a single wavelength for use in x-ray diffraction, but even with perfect crystals as monochromators the beam will have some nonzero wavelength band of energies. Another deviation is in the angle, which we may wish to vary very slightly for a "rocking curve"–a plot of the intensity reflectivity versus angle for a fixed energy. For these reasons, we simultaneously claim and define the relative deviation ζ of all of these with

$$\zeta = \frac{\Delta\lambda}{\lambda} = \frac{\Delta k}{k} = \frac{\Delta Q}{Q} = \frac{\Delta\theta}{\tan\theta} \tag{2.26}$$

A very general interpretation of ζ is as a deviation from $Q = mG$, with m an integer:

$$Q = mG(1 + \zeta). \tag{2.27}$$

In deriving Equation 2.26, we will refer to Figure 2.8. We begin with the definition that $\zeta = \frac{\Delta k}{k}$ as in Figure 2.8 and derive $\zeta = \frac{\Delta\lambda}{\lambda}$. We make use of the notation $\boldsymbol{k}_2 = \boldsymbol{k} + \boldsymbol{\Delta k}$ and likewise for other variables.

As $k = \frac{2\pi}{\lambda}$, our calculation is straightforward:

$$\zeta = \frac{\Delta k}{k} = \frac{k_2 - k}{k} = \frac{\frac{2\pi}{\lambda_2} - \frac{2\pi}{\lambda}}{\frac{2\pi}{\lambda}} = \frac{\frac{1}{\lambda_2}}{\frac{1}{\lambda}} - 1 = \frac{\lambda_2 - \lambda}{\lambda} = \frac{\Delta\lambda}{\lambda}.$$

The next equality is best shown by Figure 2.8. We have negated the reflected k' and $k' + \Delta k'$ vectors in the bottom half of the image to form $\boldsymbol{Q} = \boldsymbol{k} - \boldsymbol{k'}$. We will show that the overlayed blue and red triangles are similar, from which it follows that

$$\frac{k + \Delta k}{k} = \frac{Q + \Delta Q}{Q} \implies 1 + \frac{\Delta k}{k} = 1 + \frac{\Delta Q}{Q} \implies \frac{\Delta k}{k} = \frac{\Delta Q}{Q}.$$

The triangles clearly have one equal angle, and because $k' = ak$ where a is independent of k, we have for $k_2 = k + \Delta k$ that $k'_2 = a(k_2) = a(k + \Delta k)$ so that $\frac{k'_2}{k'} = \frac{a(k+\Delta k)}{ak} = \frac{k+\Delta k}{k} = \frac{k_2}{k}$. The triangles are then congruent and $\frac{\Delta k}{k} = \frac{\Delta Q}{Q}$.

Finally, we use Bragg's law and its derivative to show the angular equality:

$$m\lambda = d\sin\theta$$

$$m\mathrm{d}\lambda = d\cos\theta\mathrm{d}\theta$$

$$\frac{\mathrm{d}\lambda}{\lambda} = \frac{\cos\theta}{\sin\theta}\mathrm{d}\theta = \frac{\mathrm{d}\theta}{\tan\theta}$$

2.3.4 One slab

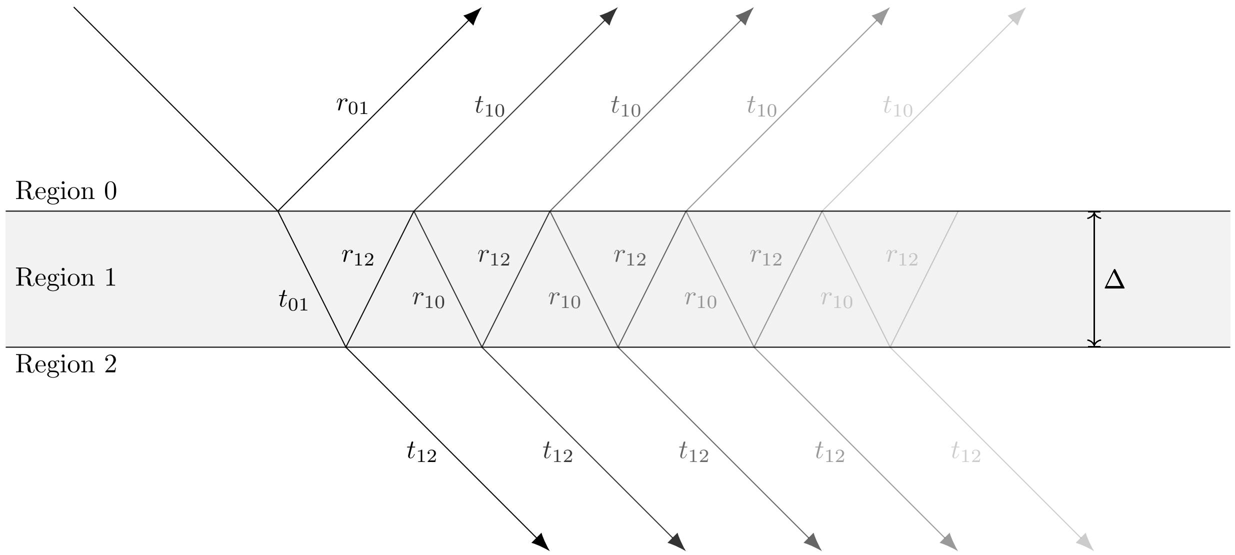


Figure 2.9: A light ray entering a slab of finite thickness Δ reflects internally many times.

A light ray in a medium “0” is incident on a slab of finite width Δ constituting medium “1”, which is followed by medium “2”. The incident light ray reflects immediately a relative reflection r_{01} from medium 0 to medium 1, and transmits t_{01} from medium 0 to medium 1. This transmitted ray is of concern, since it reflects internally r_{12} between medium 1 and medium 2, while also transmitting t_{12} from medium 1 to medium 2. This is shown in Figure 2.9, which the incident ray coming in at an angle for clarity. **For this derivation, however, we assume $2\theta = \pi$ and that we are hitting the material straight on**. For this reason, $Q = 2kr\sin(2\theta) = 2kr$ at glancing angle 2θ, and the total amplitude reflectivity accounting for the phase shift $p = e^{i\boldsymbol{k}\cdot\boldsymbol{r}} = e^{ik\Delta} = e^{iQ\Delta/2}$ for each distance Δ travelled between mediums 0 and 2 is

$$\begin{aligned} r_{slab} &= r_{01} + t_{01}r_{12}p^2\left(t_{10} + r_{10}r_{12}p^2\left(t_{10} + r_{10}r_{12}p^2\left(t_{10} + \ldots\right)\right)\right) \\ &= r_{01} + t_{01}r_{12}t_{10}p^2\left(1 + r_{10}r_{12}p^2\left(1 + r_{10}r_{12}p^2\left(1 + \ldots\right)\right)\right) \end{aligned}$$

which reduces immediately to

$$r_{slab} = r_{01} + t_{01}t_{10}r_{12}p^2 \sum_{m=0}^{\infty} \left(r_{10}r_{12}p^2\right)^m .$$

The geometric series is of the form

$$S_N = \sum_{n=0}^{N-1} k^n.$$

The solution follows from the equations $S_N - S_{N-1} = k^{N-1}$ and $kS_{n-1} + 1 = S_N$. Solving for S_{N-1},

$$\begin{aligned} S_N - k^{N-1} &= S_{N-1} = \frac{1}{k}\left(S_N - 1\right) \\ kS_N &= k^N + S_N - 1 \\ S_N &= \frac{1 - k^N}{1 + k}. \end{aligned} \tag{2.28}$$

The limit is given by

$$\lim_{N\to\infty} S_N = \begin{cases} \frac{1}{1-k}, & |k| < 1 \\ \infty, & |k| \geqslant 1 \end{cases} \tag{2.29}$$

so that

$$r_{slab} = r_{01} + t_{01}t_{10}r_{12}p^2 \frac{1}{1 - r_{10}r_{12}p^2}.$$

Letting Q_0 correspond to reflection and Q_1 to transmission, we put Equation 2.25 to use:

$$r_{01} = \frac{Q_0 - Q_1}{Q_0 + Q_1} \qquad t_{01} = \frac{2Q_0}{Q_0 + Q_1}.$$

This implies that $r_{01} = -r_{10}$ and that $r_{01}^2 + t_{01}t_{10} = 1$ by substitution. Plugging these in, we obtain for r_{slab}:

$$\begin{aligned} r_{slab} &= r_{01} + \frac{t_{01}t_{10}r_{12}p^2}{1 - r_{10}r_{12}p^2} \\ &= \frac{r_{01}\left(1 - r_{10}r_{12}p^2\right) + t_{10}t_{01}r_{12}p^2}{1 - r_{10}r_{12}p^2} \\ &= \frac{r_{01} + r_{12}p^2\left(r_{01}^2 + t_{10}t_{01}\right)}{1 - r_{10}r_{12}p^2} \\ &= \frac{r_{01} + r_{12}p^2}{1 - r_{10}r_{12}p^2} \\ &= \frac{r_{01}\left(1 - p^2\right)}{1 - r_{01}^2p^2} && (r_{01} = -r_{12}) \end{aligned}$$

where in the last step it was assumed $n_2 = n_0$.

At sufficiently large angles, $|r_{01}| \ll 1$ so that $r_{01}^2 \approx 0$, and the angle of refraction approaches the incident angle so that $\frac{Q_0}{Q_c} \approx \frac{Q_1}{Q_c}$ and $r \approx \frac{1}{(2q)^2} = \frac{Q_c^2}{4Q^2}$ (see Equations 2.25 and 2.24). We can then reduce r_{slab} to

$$r_{slab} = \frac{r_{01}\left(1 - p^2\right)}{1 - r_{01}^2p^2} \approx r_{01}\left(1 - p^2\right) \approx \left(\frac{Q_c}{2Q}\right)^2 \left(1 - e^{iQ\Delta}\right)$$

Plugging in for $Q_c = 2k\sin\theta_c \approx 2k\theta_c$ with θ_c given by Equation 2.17 plugged into Equation 2.23, we obtain $Q_c = \sqrt{16\pi\rho r_0}$ so that

$$\begin{aligned} r_{slab} &= \frac{Q_c^2}{4Q^2}\left(1 - e^{iQ\Delta}\right) \\ &= \frac{16\pi\rho r_0}{4Q^2}\left(-e^{iQ\Delta/2}\left[e^{iQ\Delta/2} - e^{-iQ\Delta/2}\right]\right) \\ &= \left[\frac{16\pi\rho r_0}{4Q}\frac{\Delta}{Q(\Delta/2)}\frac{i}{2i}\right]\left(-e^{iQ\Delta/2}\left[e^{iQ\Delta/2} - e^{-iQ\Delta/2}\right]\right) \\ &= \frac{4\pi\rho r_0\Delta}{Q}\frac{e^{iQ\Delta/2}}{Q\Delta/2}\left(-i\left[\frac{e^{iQ\Delta/2} - e^{-iQ\Delta/2}}{2i}\right]\right) \\ &= -i\frac{4\pi\rho r_0\Delta}{Q}\frac{\sin\left(Q\Delta/2\right)}{Q\Delta/2}e^{iQ\Delta/2} \end{aligned}$$

Assuming a slab thin enough that $Q\Delta \ll 1$, since $\lim_{\theta\to 0}\frac{\sin\theta}{\theta} = 1$, we drop the phase term to obtain reflectivity

$$r_{\text{thin slab}} \approx -i\frac{4\pi\rho r_0\Delta}{Q} = -i\frac{\lambda\rho r_0\Delta}{\sin\theta} = -ig \tag{2.30}$$

making use of Equation 2.2. This equation is valid in calculating intensity reflectivity at small glancing angles where the reflectivity is weak; such angles surpass the critical angle θ_c. Note that this defines the variable

$$g = \frac{\lambda \rho r_0 \Delta}{\sin\theta} = \frac{(2d\sin\theta/m)\left(|F|/v_c\right) r_0 d}{\sin\theta} = \frac{2d^2 r_0}{m v_c}\left|F\right| \tag{2.31}$$

where m is the Bragg index, Δ is rewritten as d, the distance between reflecting planes, and ρ has been generalized to $|F|/v_c$, the reflected amplitude (form factor) per unit volume. Hence one thin slab will reflect $-ig$ while transmitting $1 - ig_0$ of the original amplitude. We can rewrite Equation 2.16:

$$g_0 = \frac{|F_0|}{|F|} g. \tag{2.32}$$

2.3.5 Many Layers: Kinematical Approximation

This approach varies from that of Section 2.2 in that we are now including refractive effects. In the kinematical approximation, the product of the number of layers N and the reflectivity per layer g (with a phase factor of 180°, or factor of $-i$) is small ($Ng \ll 1$). The amplitude reflectivity for these layers is just

$$r_N = -ig \sum_{j=0}^{N-1} e^{i(Qd-2g_0)j}$$

where e^{-ig_0} is the reflectivity of a single slab neglecting dynamical effects as derived in Equation 2.15 and factors doubly per layer since the wave is transmitted then reflected through each layer, and $e^{iQd/2} = e^{ikd}$ is the phase shift from travelling through one layer, and it has also been squared to account for travelling as a transmitted wave down the layer, then travelling the same distance d up the layer as a reflected wave. The reader should keep in mind that we are still working with glancing angle $2\theta = \pi$, so that our x-ray beam is parallel to the surface normal of the reflecting planes.

Plugging in Equation 2.27 and Equation 2.11 into the previous, we obtain

$$\begin{aligned} r_N = -ig \sum_{j=0}^{N-1} e^{i(Qd-2g_0)j} &= -ig \sum_{j=0}^{N-1} e^{i\left([mG(1+\zeta)]\left[\frac{2\pi}{G}\right]-2g_0\right)j} \\ &= -ig \sum_{j=0}^{N-1} e^{i2\pi(\cancel{m}+m\zeta-g_0/\pi)j} \\ &= -ig \sum_{j=0}^{N-1} e^{i2\pi(m\zeta-g_0/\pi)j} \end{aligned}$$

which reduces by use of Equation 2.28 to

$$\begin{aligned} r_N &= -ig\frac{1-e^{i2\pi N[m\zeta-g_0/\pi]}}{1-e^{i2\pi[m\zeta-g_0/\pi]}} \\ &= -ig\frac{e^{i\pi N[m\zeta-g_0/\pi]}\left(e^{-i\pi N[m\zeta-g_0/\pi]}-e^{i\pi N[m\zeta-g_0/\pi]}\right)}{e^{i\pi[m\zeta-g_0/\pi]}\left(e^{-i\pi[m\zeta-g_0/\pi]}-e^{i\pi[m\zeta-g_0/\pi]}\right)} \\ &= -ige^{i\pi(N-1)[m\zeta-g_0/\pi]}\frac{\sin(N\pi\left[m\zeta-g_0/\pi\right])}{\sin(\pi\left[m\zeta-g_0/\pi\right])}. \end{aligned}$$

Neglecting the phase terms and denoting the Bragg peak displacement as

$$\zeta_0 = \frac{g_0}{\pi} = \frac{2d^2\left|F_0\right|r_0}{\pi m v_c}, \tag{2.33}$$

we obtain

$$r_N(\zeta) = g\left|\frac{\sin\left(N\pi\left[m(\zeta-\zeta_0/m)\right]\right)}{\sin\left(\pi\left[m(\zeta-\zeta_0/m)\right]\right)}\right|. \tag{2.34}$$

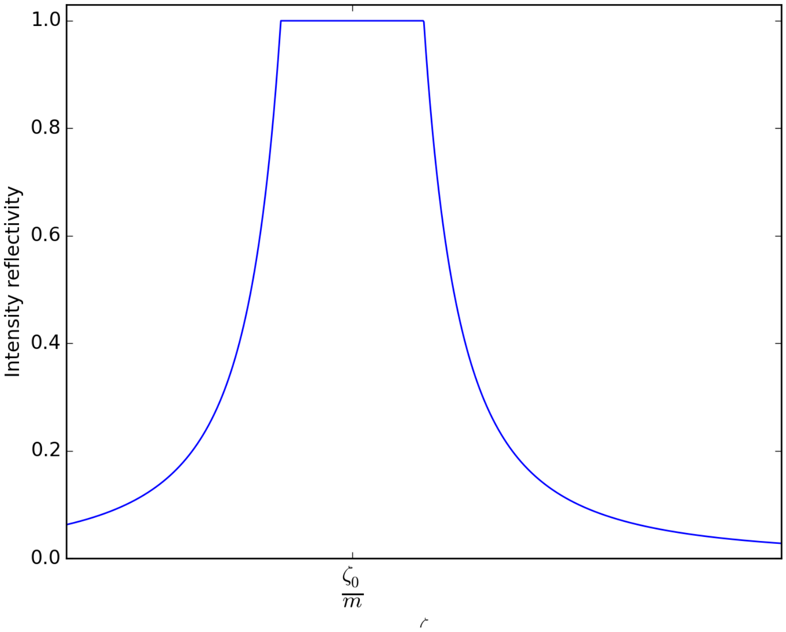


Figure 2.10: Kinematical result for many layers as given by Equation 2.35, with diverging parts replaced by $|r_N|^2 = 1$.

Hence, even in the kinematical approximation, the reflection peak is not centered where we would expect it to be from Bragg's law, but is displaced ζ_0/m. To see the behavior of the intensity reflectivity $|r_N(\zeta)|^2$ as $N \to \infty$ (which is realistic, since the x-ray penetrates into several orders of magnitude more atomic planes than its limit in penetration depth, and plausible, since $Ng \ll 1$ in the kinematical approximation), we note that in this limit the square sine term in the numerator of $|r_N|^2$ oscillates infinitely quickly as $N \to \infty$ in justifying its replacement by the factor of 1/2:

$$|r_N(\zeta)|^2 = \frac{g^2}{2\sin^2(\pi m(\zeta-\zeta_0/m))} \approx \frac{g^2}{2\pi^2 m^2(\zeta-\zeta_0/m)^2}. \tag{2.35}$$

It is clear then that the intensity reflectivity diverges as $\left|r_N^2\right|$ reaches its maximum at $\zeta = \zeta_0/m$, but by its definition the reflectivity can't be greater than 1, so we cap it off at 1 in Figure 2.10. It is clear from the divergence of the intensity reflectivity that the kinematical derivation does not suffice—however, we expect the theory of dynamical diffraction to match at angles farther away from ζ_0/m, where the kinematical result does not diverge, and we expect the reflectivity to be shifted an amount ζ_0/m in the final derivation as well.

2.3.6 Many Layers: Dynamical Derivation

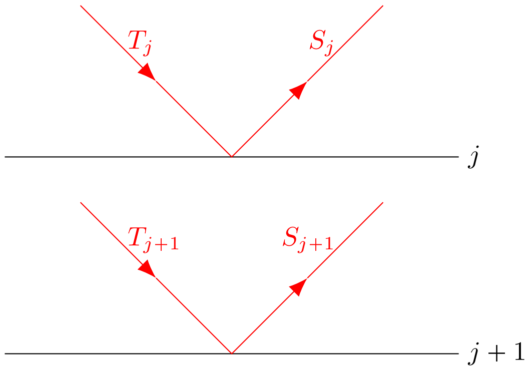


Figure 2.11: Reflection and transmission through two layers j and $j+1$. T_j and S_j, where j corresponds to the layer depth down the crystal, are representative of the T and S fields just above layer j. They include all contributions from previous layers, and have their own phase information.

Let's focus on the T (in the transmitted direction) and S (in the reflected direction) wavefields. These are not the same as the transmitted and reflected portions of an incident wave calculated with the transmittivity and reflectivity; these include all contributions in their respective directions, for their layer—ie, T_j is all contributions in the transmitted direction *just above* layer j, as shown in Figure 2.11.

Let's denote the phase shift ϕ between reflected rays in layer $j+1$ and layer j—which is $2\pi m$ with $m \in \mathbb{Z}$ if they are constructive—by ϕ, and examine as we have been the deviation from constructive interference with the variable Δ, which is defined below along its relationship to ζ:

$$\phi = m\pi + \Delta = m\pi(1+\zeta) \tag{2.36}$$

so that

$$\Delta = m\pi\zeta. \tag{2.37}$$

Let's solve dynamically for T_j and S_j. Figure 2.12 illustrates the contributions to S_j and to T_{j+1}. We have derived in Equation 2.15 that each transmitted wave undergoes a phase shift $(1-ig_0)$ and each reflection $-ig$. The contributions to S_j are the reflected T_j and the transmitted S_{j+1} with an extra phase term $e^{i\phi}$ per Equation 2.36, while the contributions to T_{j+1} are the transmitted T_j and the reflection back into the material of S_{j+1}. The former carries phase term $e^{i\phi}$ due to the extra distance it travels before becoming T_{j+1} (which is just above layer $j+1$), and the latter carries the same doubly—once for the extra phase in reaching layer j, and once after reflection going to layer $j+1$. We thus obtain

$$S_j = -igT_j + (1-ig_0)S_{j+1}e^{i\phi} \qquad T_{j+1}e^{-i\phi} = (1-ig_0)T_j - igS_{j+1}e^{i\phi}$$

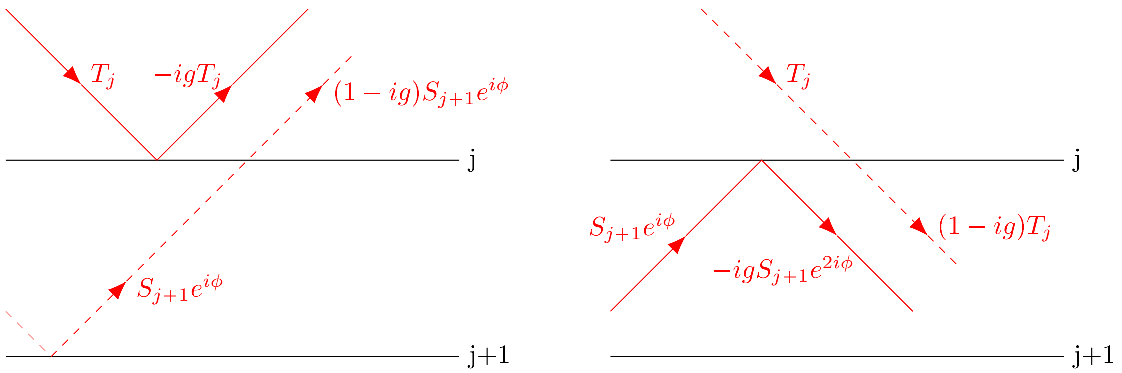


Figure 2.12: Contributions to layer wavefields S_j (left) and T_{j+1} (right). S_j is made up of the refection of T_j combined with the transmission of S_{j+1}, while T_{j+1} is made up of the reflection back into the material of S_{j+1} and the transmitted T_j from the previous layer, and phases due to reflection $(-ig)$, transmission $(1-ig_0)$, and travelling the extra phase ϕ $(e^{i\phi})$ are accounted for.

Rearranging the T_{j+1} equation yields,

$$igS_{j+1} = (1-ig_0)e^{-i\phi}T_j - e^{-2i\phi}T_{j-1}$$

and shifting the indices by -1 yields

$$igS_j = (1-ig_0)e^{-i\phi}T_{j-1} - e^{-2i\phi}T_j.$$

We substitute these into the S_j equation to obtain the second order recursive function

$$(1-ig_0)e^{-i\phi}\left[T_{j+1}+T_{j-1}\right] = \left[g^2+(1-ig_0)^2+e^{-2i\phi}\right]T_j.$$

Skipping a proper derivation, we note the validity of the trial solution $T_{j+1} = e^{-\eta}e^{im\pi}T_j$ with $\eta \in \mathbb{C}$ and $0 < \mathrm{Re}\,\eta \ll 1$ for small attenuations. Plugging the trial solution into the previous and expanding to second order yields

$$\eta^2 = g^2 - (\Delta - g_0)^2$$

with solution

$$i\eta = \pm\sqrt{(\Delta-g_0)^2-g^2}. \tag{2.38}$$

The same solution is found likewise with $S_{j+1} = e^{-\eta}e^{im\pi}S_j$ so that $S_1 = e^{-\eta}e^{-im\pi}S_0$ can be plugged into our original S_j equation to obtain

$$S_0 = -igT_0 + (1-ig_0)S_0e^{-\eta}e^{im\pi}e^{im\pi}e^{i\Delta},$$

or

$$\frac{S_0}{T_0} = \frac{-ig}{1-(1-ig_0)(1-\eta)(1-i\Delta)} \approx \frac{-ig}{ig_0+\eta-i\Delta} = \frac{g}{i\eta+(\Delta-g_0)}$$

Plugging in for η using Equation 2.38 and defining

$$\epsilon = \Delta - g_0 = m\pi\zeta - \pi\zeta_0, \tag{2.39}$$

we obtain

$$r = \frac{S_0}{T_0} = \frac{g}{i\eta+\epsilon} = \frac{g}{\epsilon \pm \sqrt{\epsilon^2-g^2}}, \tag{2.40}$$

where the ambiguous sign of the square root matches that of ϵ at $\epsilon \gg g$ so that the tails tamper off, and at $\epsilon < g$, the intensity reflectivity is

$$r = \left(\frac{S_0}{T_0}\right)\left(\frac{S_0}{T_0}\right)^* = \left(\frac{g}{\epsilon+\sqrt{\epsilon^2-g^2}}\right)\left(\frac{g}{\epsilon-\sqrt{\epsilon^2-g^2}}\right) = \frac{g^2}{\epsilon^2-\epsilon^2+g^2} = 1$$

regardless of the sign.

2.3.7 The Darwin Reflectivity Curve

Multiplying the right hand side of Equation 2.40 by g gives

$$r = \frac{g}{\epsilon \pm \sqrt{\epsilon^2-g^2}}\left(\frac{1/g}{1/g}\right) = \frac{1}{\epsilon/g \pm \sqrt{(\epsilon/g)^2-1}}\left(\frac{\frac{\epsilon}{g} \mp \sqrt{(\epsilon/g)^2-1}}{\epsilon/g \mp \sqrt{(\epsilon/g)^2-1}}\right) = 1 \mp \sqrt{x^2-1}.$$

Introducing the shorthand variable

$$x = \frac{\epsilon}{g} = m\pi\frac{\zeta}{g} - \frac{g_0}{g}, \tag{2.41}$$

we can rewrite the amplitude reflectivity more clearly as

$$r(x) = \begin{cases} x - \sqrt{x^2-1} & x \geqslant 1 \\ x - i\sqrt{1-x^2} & |x| \leqslant 1 \\ x + \sqrt{x^2-1} & x \leqslant 1 \end{cases} \tag{2.42}$$

having swapped the sign in the square root for $|x| \leqslant 1$ to show explicitly that $\sqrt{x^2-1}$ is imaginary in that range. The intensity reflectivity is the square of each piece and its plot is shown in Figure 2.13.

We can solve for intensity reflectivity $R = r^2 = .5$ for $x < 1$ and $x > 1$ to obtain a full width half max

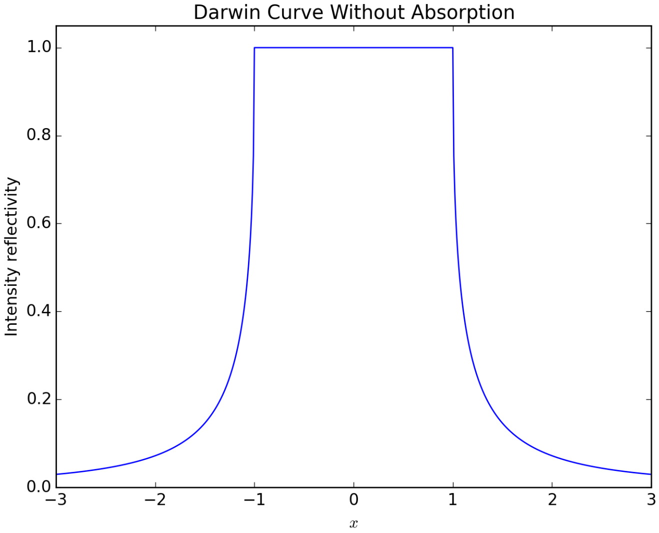


Figure 2.13: Darwin curve without absorption effects—the reflectivity is 100% in the middle, but tapers off to the sides. The intensity reflectivity is the square of the amplitude reflectivity given in Equation 2.42 when $|x| \geqslant 1$ and equals 1 (the product of the complex number and its conjugate in Equation 2.42) when $|x| \leqslant 1$.

of

$$\mathrm{FWHM}_x = \frac{3}{\sqrt{2}}$$

in x, or

$$\mathrm{FWHM}_\zeta = \frac{3}{\sqrt{2}} \frac{g}{m\pi}$$

in ζ (making use of Equation 2.41). This can be translated to angular FWHM using Equation 2.26:

$$\mathrm{FWHM}_\theta = \mathrm{FWHM}_\zeta \tan\theta = \frac{3}{\sqrt{2}} \frac{g}{m\pi} \tan\theta$$

where θ is the Bragg angle. Note that the latter definition depends on energy while the others do not: FWHM_θ varies the angle of a perfectly monochromatic beam while FWHM_ζ varies the wavelength (see the discussion following Equation 2.26).

We can likewise switch between representations of our independent variable as x, ζ, or θ, using the equations compiled below for convenience:

$$\begin{aligned} x(\zeta) &= \frac{g}{m\pi}\zeta - \frac{g_0}{g} \\ \theta(\zeta) &= \zeta \tan\theta \\ \theta(x) &= \zeta(x) \tan\theta = \frac{m\pi}{g}\left(x + \frac{g_0}{g}\right) \tan\theta \end{aligned} \tag{2.43}$$

Using these, a sample plot is given with the independent variable in millidegrees in

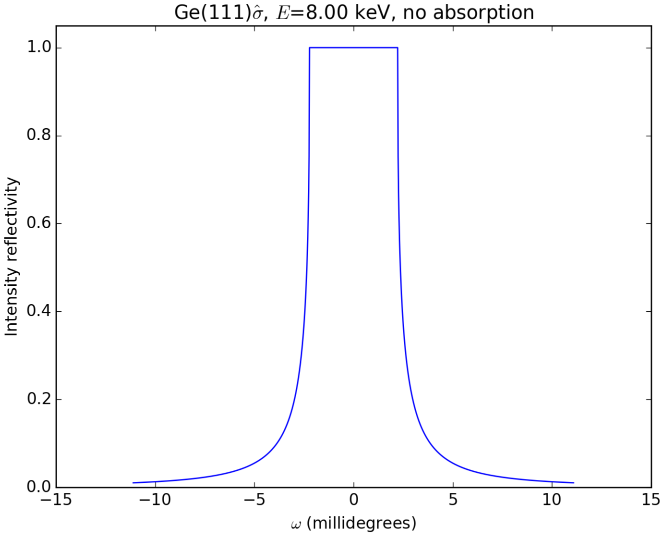


Figure 2.14: Darwin reflectivity curve for Ge(111) probed with 8 keV at $\hat{\sigma}$ orientation.

2.3.8 Darwin Curve with Absorption

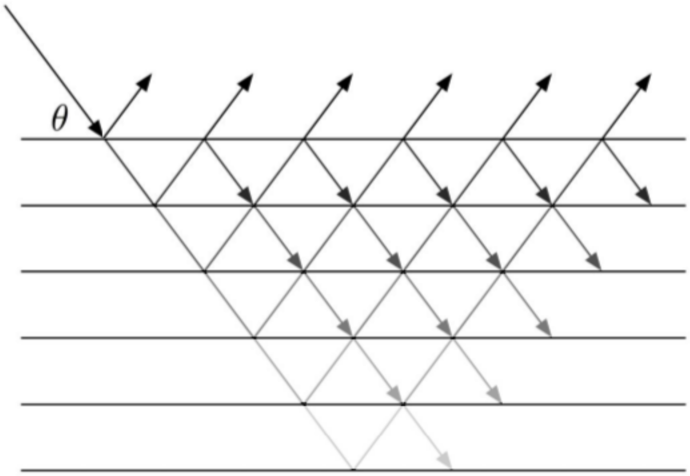


Figure 2.15: Illustration of the dynamical derivation with absorption effects. Similar to Figure 2.5, but the rays diminish as they are transmitted and reflected within the crystal.

Absorption (see Figure 2.15) comes with making x complex. This follows from g_0 being allowed to be complex and represented by

$$g_0 = \left(\frac{2d^2 r_0}{m v_c}\right) F_0 \tag{2.44}$$

where

$$F_0 = \sum_j \left(f_j^0(0) + f_j' + i f_j'' \right) = \sum_j \left(Z_j + f_j' + i f_j'' \right) \tag{2.45}$$

with Z_j the jth atom's atomic number. We still have

$$g = \left(\frac{2d^2 r_0}{m v_c} \right) F \tag{2.46}$$

where

$$F = \sum_j \left(f_j^0(\boldsymbol{Q}) + f_j' + i f_j'' \right) e^{i\boldsymbol{Q}\cdot\boldsymbol{r}_j}. \tag{2.47}$$

We compute rocking curves by the combination of these equations. Figure 2.16 shows the asymmetric results, to be contrasted with Figure 2.14.

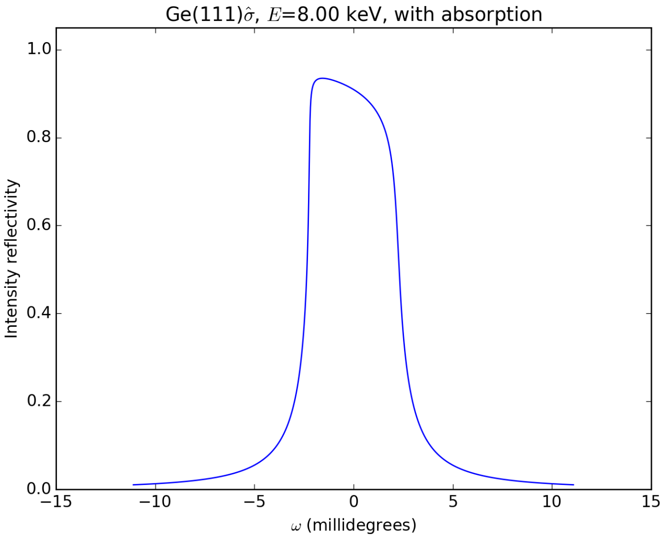


Figure 2.16: Darwin reflectivity curve for Ge(111) probed with 8 keV at $\hat{\sigma}$ orientation accounting for absorption effects.

Chapter 3

Method and Data

3.1 Method

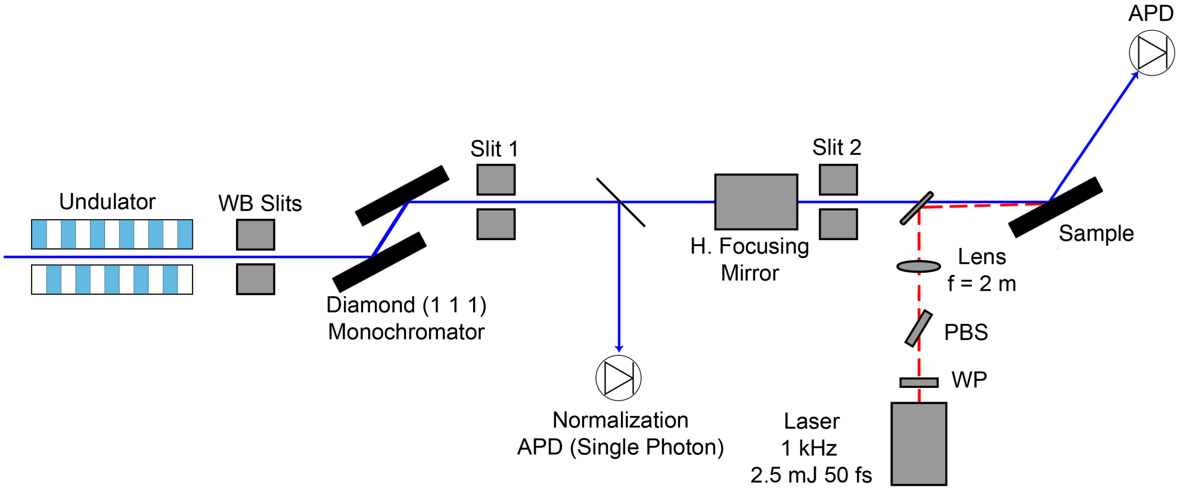


Figure 3.1: Layout of pump-probe experiment at APS 71D.[14] The x-ray beam travels through an undulator before being led through the white beam (WB) slits. The beam is reduced to a single wavelength by the diamond (111) monochromator and slit, after which it is split into an avalanche photodiode (APD) for normalization, and a horizontal focusing mirror and slit. Finally, the beam meets the sample of Ge(100) and is measured by a second APD. The laser beam's polarization is focused by a half-wave plate (WP) and polarizing beam splitter (PBS) and is transmitted through a lens before exciting the sample.

The data analyzed in this thesis was taken using the Time-Resolved X-Ray Diffraction instrument at Sector 7ID of Argonne National Laboratory's Advanced Photon Source (APS). This apparatus has been described in detail elsewhere[14] but is briefly reviewed here. The instrument is shown schematically in Figure 3.1. The 7 GeV, 100 mA electron beam inside the APS storage ring consists of 24 bunches of 100 ps duration full-width at half-maximum which rotate with a 3.68 microsecond period such that the effective x-ray repetition rate is 6.5 MHz. The x-rays at beamline 7ID are produced by an undulator which has its peak brightess at wavelengths near one Angstrom, close to the x-ray energy range used in this experiment. The "white beam" produced from the undulator has a 10 % bandwidth, or about 1 keV/10 keV. The center of this spectrum may be tuned by adjusting the undulator magnet gap, which in turn changes the magnetic field strength and the relativistic doppler shift. A silicon crystal monochromator is used to further filter the x-rays down to a 0.1% relative bandwidth, or approximately 1 eV/10 keV. This is the energy resolution used in our experiments. The monochromator consists of two Silicon (111) crystals which are adjusted such that the Bragg angle is at the desired energy.

A series of slits reduces the spot size and limits the angular resolution in the vertical (diffraction) direction to approximately 1 mdeg. The measured darwin curve widths in our experiments is a result of the combined angular and energy resolution of the x-ray beam following these optics.

The x-rays and laser beam are focused such that the x-rays only probe the very center of the few mm laser spot size on the sample, thereby insuring a one dimensional geometry. The x-ray intensity in each x-ray bunch is measured upstream of the sample using an Avalanche Photodiode (APD). The APD can count single x-ray photons and has a 10 ns rise time, much faster than the 153 ns (6.5 MHz) bunch separation. This signal is used to normalize the counts from the detector placed after the sample, which records a count rate that changes depending on both the angle of the sample (the rocking curve) and the time delay after the laser pulse strikes the sample (the strain profile).

The laser is delivered from an adjacent laboratory. The amplified ultrafast laser pulses arrive at 1000 Hz, with each laser pulse less than 50 fs in duration and centered at a wavelength of 800 nm. They are synchronized to the selected x-ray pulse with a fixed, controllable delay time. At a fixed delay time, the same transient strain field will exist in the sample for each rocking curve datapoint.

3.2 Data

To account for the different properties of different materials, such as unit cell dimensions and response to radiation, it is necessary to seek tabulated data. Table 3.1 consists of lattice constants, which is the size of our cubic unit cells, and Table 3.2 presents some physical constants. Table 3.3 is used for form factors outside of absorption range in Equation 2.5, and NIST values are used near the absorption regime—particularly the K-edge of germanium. Finally, cubic interpolation through the values in Table 3.4 yields f' and f'' as a function of energy, again restricted outside of absorption regimes, for which NIST values were used. Finally, a plot of NIST values is given in Figure 3.2 with cubic interpolation.

3.2.1 Form Factors, Structure Factors, Constants

In order to carry out the calculations in the equations derived above—particularly Equation 2.44–Equation 2.47—one needs to know certain properties of the crystal and x-ray source, and how they interact at given angles and energies. The available data has been tabulated below for reference, and elementary results are given in Tables 4.1 and 4.2 to check reproducability of results.

Table 3.1 gives the lattice parameters for several zinc blende (diamond) structures. These structures form a cubic unit cell as discussed in Section 2.2.3, and the lattice constant is the length of any side of this unit cell. This is used, eg, in Equation 2.11 to find the interplanar spacing of atoms for a given Miller plane. Physical constants are also required to obtain results, and those used were gathered from the National Institute of Standards and Technology (NIST).

Element	Lattice Constant (Å)
Ge	5.64613
Si	5.43095
GaAs	5.6533
InSb	6.4794
C	3.56683

Table 3.1: Selected zinc blende structures and their lattice constants[2]

Parameter	**Value**	**Unit**	**Description**	**Source**
h	$4.135667516(91)\times10^{-15}$	eV·s	Planck's Constant	NIST[11]
c	299792458	m/s	Speed of light in vacuum	NIST[10]
r_0	$2.8179403267(27)\times10^{-5}$	Å	Classical electron radius	NIST[12]

Table 3.2: Physical constants, gathered from NIST.

The constants which are discussed in Section 2.2.3 and necessary to calculate the form factor of Equation 2.5 are obtained with a subscription to the Crystallographic Tables and are found in Ch. 6.1, Table 6.1.1.4, and a parsing algorithm was developed to convert these to a useable form in Appendix B.6; selected elements are outlined for reference in Table 3.3.

Crystallographic Tables Interpolation Factors For Selected Elements									
	a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_4	c
Ge	16.0816	2.8509	6.3747	0.2516	3.7068	11.4468	3.683	54.7625	2.1313
Si	6.2915	2.4386	3.0353	32.3337	1.9891	0.6785	1.541	81.6937	1.1407
Ga	15.2354	3.0669	6.7006	0.2412	4.3591	10.7805	2.9623	61.4135	1.7189
As	16.6723	2.6345	6.0701	0.2647	3.4313	12.9479	4.2779	47.7972	2.531
In	19.1624	0.5476	18.5596	6.3776	4.2948	25.8499	2.0396	92.8029	4.9391
Sb	19.6418	5.3034	19.0455	0.4607	5.0371	27.9074	2.6827	75.2825	4.5909
C	2.31	20.8439	1.02	10.2075	1.5886	0.5687	0.865	51.6512	0.2156

Table 3.3: Nine-factor interpolation constants from Crystallographic Tables[6]

Finally, as per the interpolation discussed in Section 2.2.3, a dispersion corrections as collected from the Crystallographic Tables[5] are given for selected elements and energies in Table 3.4. These are, as mentioned in Section 2.2.3, not suitable for calculations near the band edges, and for this we use extensive tabulated data provided in the Journal of Physica nd Chemical Reference Data[7] which have been made available online by NIST[9], for which code has also been written to minimize load on the server in Appendix B.6. The dispersion corrections are necessary to calculate the atomic form factor in Equation 2.4 after the form factor has been found by Equation 2.5. These are included in Table 3.5, and Figure 3.2 clarifies the advantage of the NIST data around the band edges. This clarifies the choice of NIST for the dispersion correction data, while the Crystallographic Tables are used to find f^0.

(Truncated) Forward-Scattering Dispersion Corrections to Form Factors											
λ (Å)		2.749	2.29	1.936	1.789	1.541	0.709	0.559	0.216	0.209	0.18
Ge	f’	-0.155	-0.386	-0.641	-0.778	-1.089	0.155	0.302	0.088	0.08	0.044
	f”	2.445	1.784	1.329	1.156	0.885	1.8	1.19	0.199	0.186	0.139
Si	f’	0.392	0.365	0.321	0.298	0.254	0.082	0.052	-0.002	-0.003	-0.005
	f”	0.962	0.692	0.508	0.438	0.33	0.07	0.043	0.006	0.005	0.004
Ga	f’	-0.252	-0.499	-0.77	-0.92	-1.285	0.231	0.318	0.08	0.072	0.039
	f”	2.152	1.567	1.166	1.014	0.776	1.608	1.059	0.174	0.164	0.122
As	f’	-0.069	-0.287	-0.526	-0.652	-0.93	0.05	0.276	0.096	0.087	0.05
	f”	2.763	2.019	1.507	1.311	1.005	2.006	1.331	0.225	0.211	0.158
In	f’	-5.133	-1.597	-0.416	-0.147	0.082	-0.728	-1.284	0.101	0.101	0.082
	f”	12.631	9.629	7.359	6.467	5.045	1.31	0.854	1.052	0.992	0.759
Sb	f’	-9.214	-3.064	-0.987	-0.519	-0.056	-0.587	-1.055	0.056	0.062	0.061
	f”	12.766	11.103	8.562	7.537	5.895	1.546	1.01	1.22	1.151	0.883
C	f’	0.049	0.036	0.027	0.024	0.018	0.003	0.002	-0.001	-0.001	-0.001
	f”	0.031	0.021	0.015	0.013	0.009	0.002	0.001	0.0001	0.0001	0.0001

Table 3.4: Dispersion corrections for forward scattering, provided by the Crystallographic tables.[5] Correct interpolation yields form factors at a very wide range of energies.

Selected NIST Forward-Scattering Dispersion Corrections to Form Factors											
E (keV)		0.01	0.87	6.92	22.98	39.20	51.19	139.26	181.85	271.39	405.00
Ge	f’	-27.154	-9.828	-0.625	0.440	0.332	0.252	0.077	0.058	0.040	0.032
	f”	0.000	-3.592	-1.147	-1.113	-0.413	-0.248	-0.034	-0.020	-0.009	-0.004
Si	f’	-9.507	-0.992	0.310	0.065	0.029	0.019	0.005	0.004	0.003	0.003
	f”	-0.495	-1.209	-0.435	-0.040	-0.013	-0.007	-0.001	-0.000	-0.000	-0.000
Ga	f’	-27.824	-11.068	-0.773	0.444	0.314	0.236	0.073	0.055	0.040	0.032
	f”	0.000	-3.092	-1.003	-0.989	-0.365	-0.218	-0.029	-0.017	-0.008	-0.003
As	f’	-27.009	-8.917	-0.501	0.385	0.346	0.264	0.077	0.056	0.037	0.028
	f”	0.000	-4.147	-1.305	-1.245	-0.464	-0.281	-0.038	-0.022	-0.010	-0.004
In	f’	-46.889	-12.393	-0.063	-1.103	0.093	0.357	0.195	0.137	0.078	0.044
	f”	0.000	-23.685	-6.411	-0.794	-2.030	-1.280	-0.203	-0.123	-0.058	-0.027
Sb	f’	-45.397	-14.922	-0.391	-0.813	-0.141	0.310	0.213	0.150	0.085	0.047
	f”	0.000	-26.568	-7.473	-0.941	-2.337	-1.479	-0.239	-0.145	-0.069	-0.033
C	f’	-3.897	0.345	0.026	0.004	0.002	0.002	0.001	0.001	0.001	0.001
	f”	-1.739	-0.750	-0.013	-0.001	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000

Table 3.5: Dispersion corrections for forward scattering, provided by NIST.[9] Data is more abundant than the Crystallographic Tables near the band edges, making this the optimal data source for our calculations. Note that NIST provides $f' + f^0$ and f'', and the atomic form factor f^0 was subtracted from the latter in compiling this table.

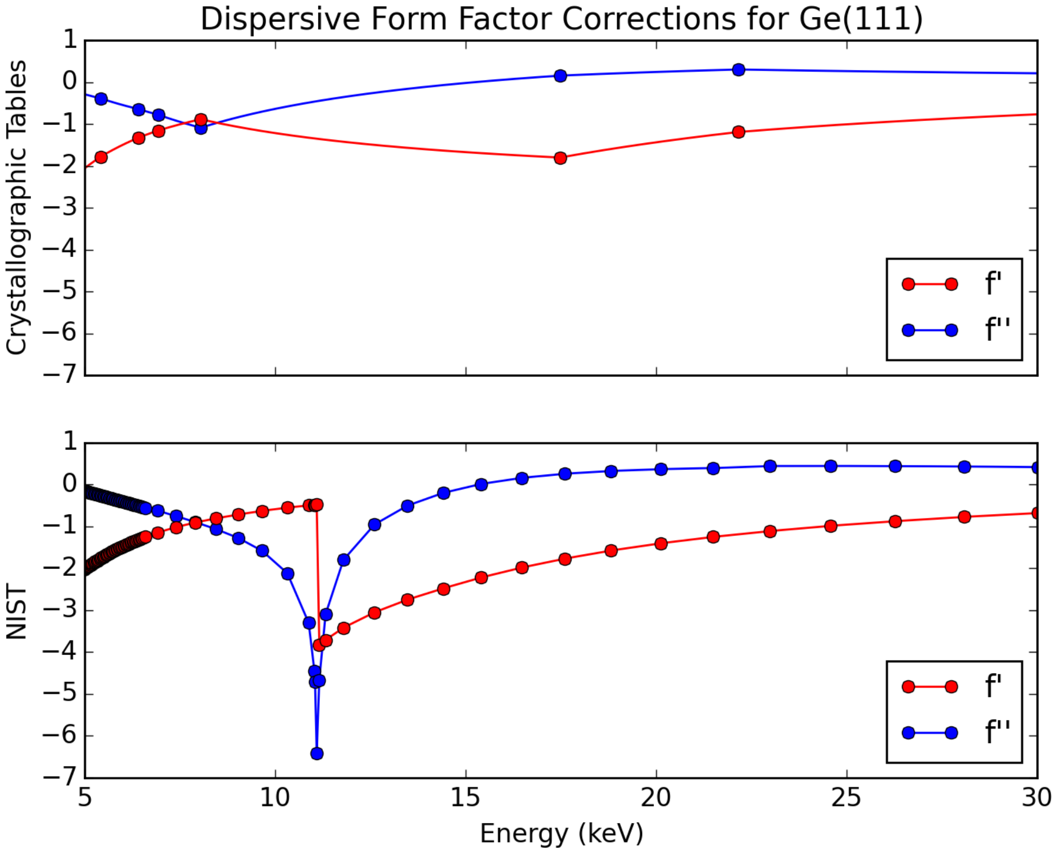
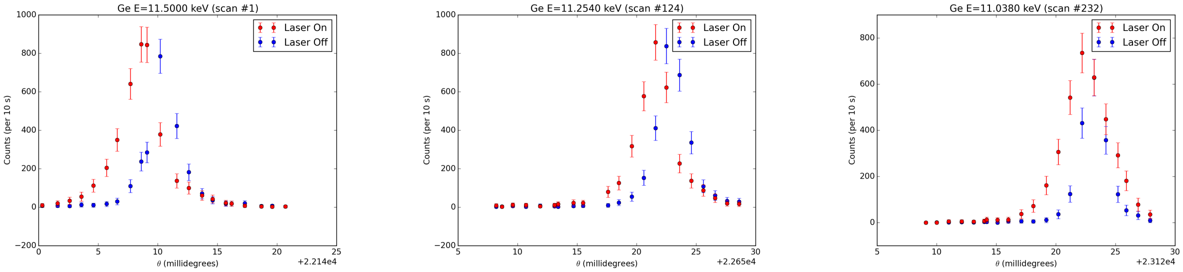


Figure 3.2: Graph of dispersion corrections from the Crystallographic Tables[5] (top) and NIST[9] (bottom). The NIST data is more abundant near the band edges, where the form factors display erratic behavior. The lines are linear interpolation (used in this paper) between the points, which represent energies for which the source provides explicit data.

3.2.2 Collected Data

Data output is in the form of an MDA file, which represents n-dimensional data linearly. A sample excerpt is included in Appendix C.1, and the full data used for this experiment can be found on the APS website.[4] The relevant data in each scan is the energy, labelled detector S16 (47), and detector S15 (51) in the MDA files; these are parsed by the code provided in Appendix B.5, and S16 and S15 hold the number of detector counts before and after the laser strains the material, respectively. Both detectors count the cumulative number of photons over 10 seconds for different energies. This results in rocking curve data for each energy both before and after the laser has strained the material. Selected scans representing our range of energies are illustrated in Figure 3.3.

It is apparent from these scans that the Laser On (red) peak precedes the Laser Off (blue) peak—this is expected, as it implies a larger interplanar spacing for the crystal in its excited state, whereupon it expands. It is from this relative peak position that we will derive the depth profile.



(a) High energy

(b) Middle energy

(c) Low energy

Figure 3.3: Raw data for energies at the extremum and median of data, unnormalized and shifted from Bragg angle.

This strained and unstrained rocking curve data is taken for 232 energies between 11.0380 and 11.5000 keV. When the data is normalized to an angle relative to the Bragg angle—taken as the center of the fit to the unstrained data, to account for any slight movements of the crystal—we begin to observe patterns as in Figure 3.4, which is a superposition of completely raw data at different energies. Note that the differing peak heights, peak widths, and absolute angles are of little relevance here. The peak height depends on the size of the bunch creating our x-ray, so we will normalize it with the height of the unstrained measurement. The peak width is due to uncertainties in the detector (see §5.1). The absolute angle is not as important as the angular difference between the strained and unstrained curves, due to millidegree perturbations of the crystal. This makes clear the importance of the reference unstrained data.

The error bars in Figure 3.3 and Figure 3.4 were obtained by taking the square root of the total count, and since this count was averaged over 10 seconds, we multiply the number of counts by 10 before taking the square root. This implies error increasing monotonically with count; hence the higher error bars near the peaks.

Data on the half-width half-max (HWHM) and the center of each curve is obtained with a Lorentzian fit (see §4.2) through the least squares method. While code and theory have been developed for more accurate fits using Darwin rocking curves, the resolution of the data renders it symmetrical enough that Lorentz curves, which offer data such as the center position and HWHM more readily, are more appropriate.

Figure 3.4 shows rocking curve data taken at different energies. This serves to demonstrates the variance of fluctuations in height, width, and absolute angle between energies, none of which play a major role in data gathering thanks to reference data of the unstrained crystal for each measurement. Of particular focus instead is the relative angle between the strained and unstrained crystal, which gives insight into the interplanar spacing.

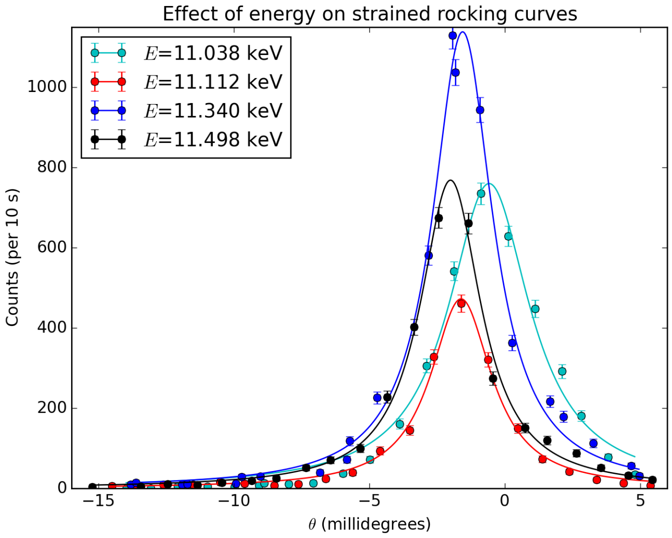


Figure 3.4: Superposition of raw data with Lorentz fits at different energies. Not a lot can be learned from the height, width, and absolute angle, but relative angles give the average interplanar spacing.

Chapter 4

Analysis

4.1 Rocking Curve

In addition to being rich in theory, developing code for finding form factors (Appendices A.2–A.5, B.3) and plotting Darwin curves (Appendices A.6, B.2) will allow us to perform fits on the data more precisely, especially with data at higher resolution. The least squares fitting method is flexible in that it will fit to any number of parameters for any function, making it extendable to fitting a Darwin curve.

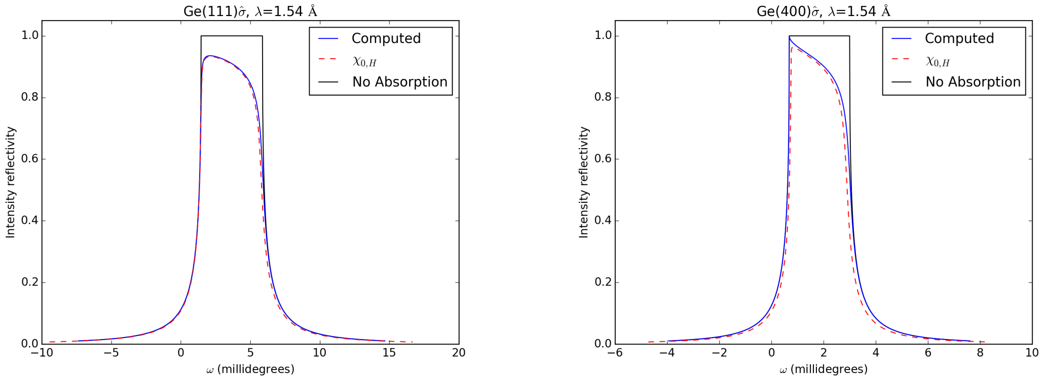
The preliminary part of our analysis should then consist of validating our work with established and well-known databases such as APS's χ_{0h} server.[13]

4.1.1 Checking our Work

The χ_{0h} server, currently hosted on the ANL network, has been online since 1997 and served nearly 2 million requests. It is continually accepting new structures. The application allows the download of generated rocking curves, which were generated manually for comparison before using the rocking curves in our least squares fits.

In Figure 4.1, the curves labelled 'Computed (no absorption)' are computed without accounting absorption effects; those labelled 'computed' are the product of our final code. Finally, the 'X0h' curves are imported directly from the manually downloaded data files. It is concluded that the shifted peak, normalized reflectivity, and width match satisfactorily with the χ_{0h} data for different energies of Ge (see Figure 4.3 for further energies). It is not enough to test different energies, of course, so Figure 4.1 validates our code with χ_{0h} for different Miller planes.

As illustrated in Figure 4.2, dual-element zinc blende structures are just as accurate as single element



(a) Ge(111) at 8.05 keV.

(b) Ge(400) at 8.05 keV.

Figure 4.1: Darwin curves for Ge$\hat{\sigma}$ at different miller indices, compared with X0h.

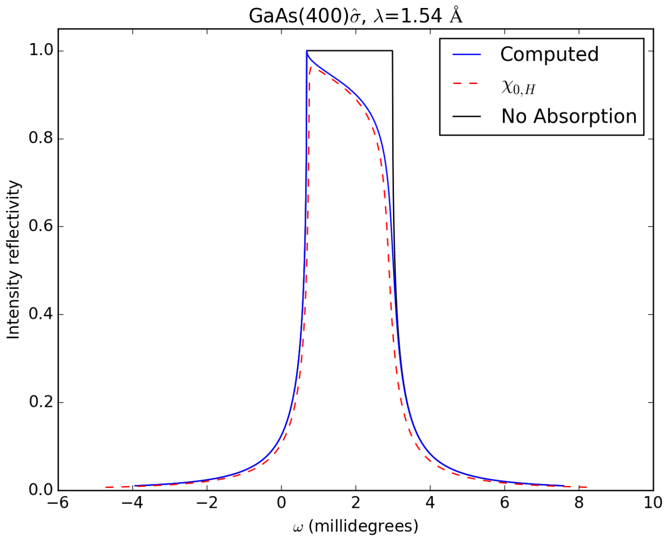
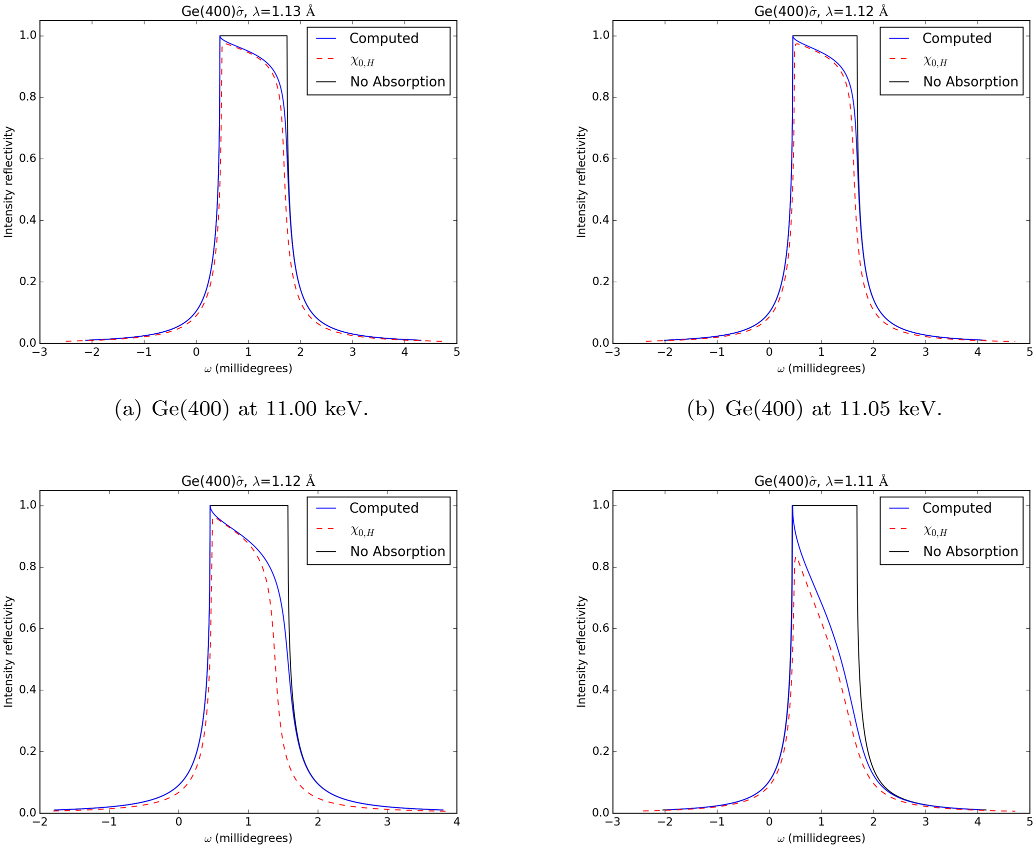


Figure 4.2: Gallium Arsenide at miller index 400.

ones, and this is despite a strange subtlety in taking the conjugate of the form factor outlined in Appendix B.

Near the K-edge, the form factors change drastically. This is why it is particularly unacceptable to interpolate linearly across this region. Furthermore, the International Tables of Crystallography tabulation of the dispersion corrections to the form factors skip from 1.54052 Åto 0.70926 Å, a substantial range which happens to hold the absorption K-edge of germanium, 11.103 keV≈1.12Å—for this reason, we turn to NIST data tabulating at much closer intervals near the K-edge; these are compared in Figure 3.2.

The linear interpolation yields the data of Figure 3.2, which is far from standards in literature due to the effects discussed above. Slight differences between χ_{0h} curves and curves computed in this paper arise from the lack of corrections from effects of temperature and relativity.



(a) Ge(400) at 11.00 keV.

(b) Ge(400) at 11.05 keV.

(c) Ge(400) at 11.10 keV.

(d) Ge(400) at 11.15 keV.

Figure 4.3: General computational agreement of varying Darwin curves with χ_{0h} at different x-ray energies near the K-edge.

4.1.2 Generating Data

The computational method can be used to give useful information such as the penetration depth of a material and absorption-corrected peak of a material. Tables 4.1 and 4.2 serve to check the validity of the code in providing intermediate results such as the form factors and dispersion corrections at given angles and energies for silicon and germanium, respectively.

111	λ=1.55 Å	E =8.00 keV	d =3.14 Å	v_c=160.2 Å^3	θ=14.31°
$f^0(0)$=14.00	$f'(0)$ =0.27	$f''(0)$ =-0.33	$f^0(Q)$ =10.54	$f'(Q)$ =0.27	$f''(Q)$ =-0.33
$F_{\text{at}}(0)$=14.27-0.33i		$\|F_{\text{at}}(0)\|$=14.3	$\|F_{\text{at}}(Q)\|$=10.8	$F_{\text{at}}(Q)$=10.81-0.33i	
F_0=114.14-2.67i		$\|F_0\|$=114.2	$\|F(Q)\|$=61.2	$F(Q)$=61.13-1.89i	
g_0=0.000395-0.000009i		$\|g_0\|$=0.000395	$\|g\|$=0.000212	g=0.000211-0.000007i	
ζ_0=0.000126-0.000003i Å		x_0=1.87		θ_0=1.84 m°	
r_0 =0.29-0.00i			$\|r_0\|^2$=0.084		

Table 4.1: Si(111)$\hat{\sigma}$ data.

400	λ=1.13 Å	E =11.0 keV	d =1.41 Å	v_c=180.0 Å^3	θ=23.53°
$f^0(0)$=31.98	$f'(0)$ =-4.13	$f''(0)$ =-0.45	$f^0(Q)$ =20.44	$f'(Q)$ =-4.13	$f''(Q)$ =-0.45
$F_{\text{at}}(0)$=27.85-0.45i		$\|F_{\text{at}}(0)\|$=27.9	$\|F_{\text{at}}(Q)\|$=16.3	$F_{\text{at}}(Q)$=16.31-0.45i	
F_0=222.81-3.64i		$\|F_0\|$=222.8	$\|F(Q)\|$=130.5	$F(Q)$=130.50-3.64i	
g_0=0.000139-0.000002i		$\|g_0\|$=0.000139	$\|g\|$=0.000081	g=0.000081-0.000002i	
ζ_0=0.000044-0.000001i Å		x_0=1.71		θ_0=1.10 m°	
r_0 =0.32-0.00i			$\|r_0\|^2$=0.105		

Table 4.2: Ge(400)$\hat{\sigma}$ data.

We further examine effects of different parameters on the Darwin curves to better understand its variance with several properties.

Figure 4.4 examines the effect of different energies for Si(111)$\hat{\sigma}$ to find that higher energies yield narrower peaks. The peaks shift forward in angle as they spread with decreasing energy. The higher energies also correspond to a lower penetration depth, which will present itself in our data as a more pronounced change in the interplanar spacing at higher energies, due to the x-rays only probing a small depth of the crystal, about where it has been strained. On the other hand, at smaller energies—particularly near the bandgap when the energies becomes very much smaller suddenly—the x-rays penetrate very deep into the crystal.

The more involved effect of changing Miller planes is presented in Figure 4.5 for Ge of constant energy 8.05 keV for some allowed reflection planes. Shape, as well as spread and angle change with changes in the Miller plane.

An analysis wouldn't be complete without examination of different materials, presented in Figure 4.6. This includes diatomic silicon blende structures such as GaAs and InSb. Different materials have different

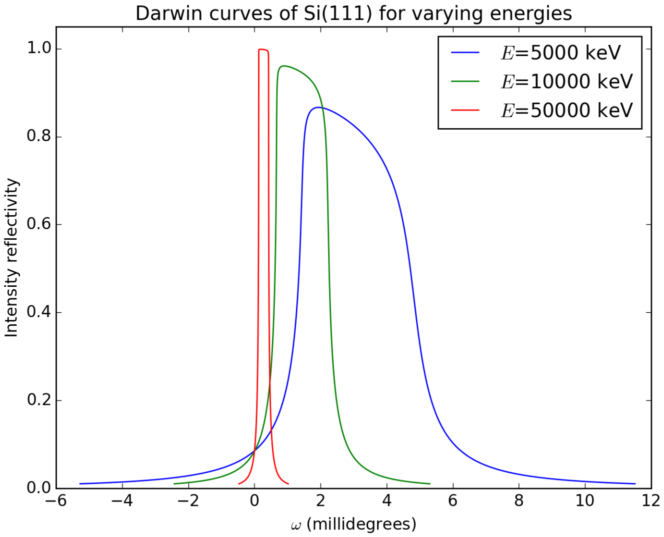


Figure 4.4: The effect of varying energies on Si(111).

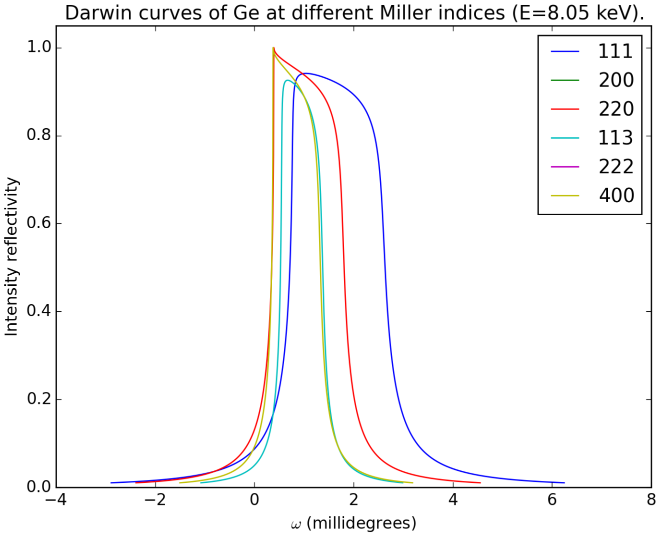


Figure 4.5: Effects of varying Miller planes on the Darwin curve for Ge at 8.05 keV.

natural interplanar spacings due to their differing lattice parameters. This causes variance in their relative Bragg angles, as shown in Figure 4.6. Note the large relative angle of carbon (diamond) compared to the less dense crystals.

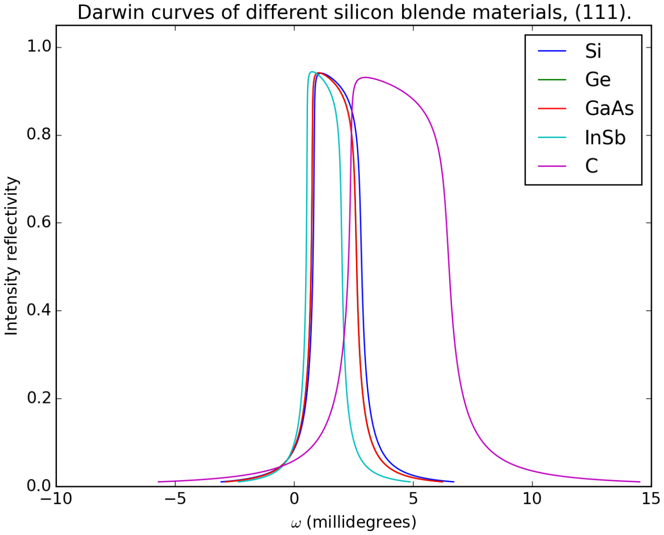


Figure 4.6: This is the effect of different zinc blende materials at Miller index 111. These have not been centered to show the relative offsets (different Bragg angles) due to the variant d-spacings.

4.2 Modeling strain

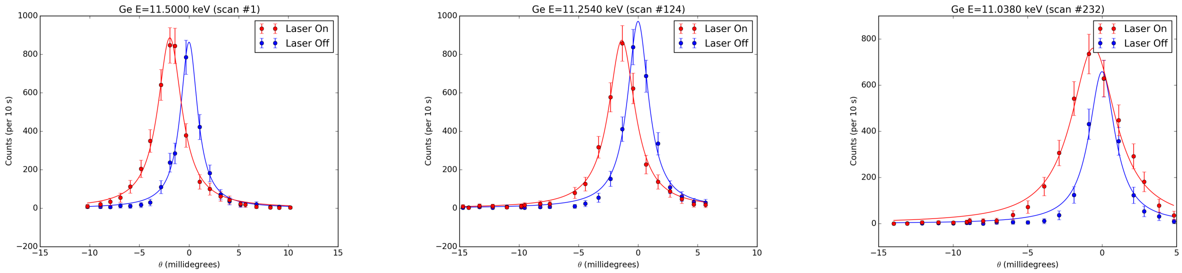
While the most accurate fit to our Darwin curve data is a proper Darwin curve, the graphs of Figure 4.7 show that it is just as fitting to fit with a Lorentzian curve. With data fine enough to contain the characteristic asymmetry of the Darwin curves, a chi-square fit of a Darwin curve yields more accurate data such as the shift in bragg angle (and therefrom, average interplanar spacing) and HWHM. However, Figure 4.7 demonstrates no asymmetry in shape in its resolution, validating the use of a Lorentzian chi-square fit, which more readily yields the center and HWHM. The Lorentzian curve is given by

$$L(\theta) = \frac{L_0 \sigma_{\text{HWHM}}^2}{(\theta - \theta_0)^2 + \sigma_{\text{HWHM}}} \tag{4.1}$$

where L_0 is the peak height of the Lorentzian, σ_{HWHM} is the half-width-half-max, and θ_0 is the x-offset of the Lorentzian's center. These three parameters are fit using the least square method. For finer data, the chi-square method scales very well to general functions, including the Darwin curve.

4.2.1 Patterns in Data

The code of Appendix B.5 parses the data into 232 sets of "Scan" objects, each containing an energy value and the strained and unstrained rocking curve data. Immediately after being read, each data set is normalized according to the unstrained rocking curve—this includes a translation of both curves along the x-axis correcting the unstrained Bragg angle based on the energy to account for tiny movements of the crystal



(a) High energy

(b) Middle energy

(c) Low energy

Figure 4.7: Darwin plots of strained ($\mathrm{B_{on}}$) and unstrained ($\mathrm{B_{off}}$) rocking curves for Ge(400) at decreasing energies (increasing penetration depths), used to model the strain profile. Fits are Lorentzian.

throughout the experiment, and a scaling along the y-axis to the maximum of the unstrained curve, though this is for aesthetic purposes only. The resulting data is of the form of Figure 4.7, which shows three plots of increasing penetration depth, along with their Lorentzian fits. Several trends are observed in at these chosen energies and other energies:

First, the volume of the strained crystal curve is larger than its unstrained counterpart for energies below the K-edge. This is discussed in the next section. Second, the strained ("laser on") peak is always at a lower reflection angle than the unstrained data. This implies by Bragg's Law that the average interplanar spacing is higher after the Ge crystal has been strained by the laser pulse than before at all probed energies. Third, the lesser strained peak angle approaches the unstrained peak angle with increasing penetration depth, at energies above the K-edge. This implies above the K-edge that the average interplanar spacing decreases towards the natural (unstrained) average interplanar spacing of the crystal.

It is more intuitive to analyze the data with respect to penetration depth, rather than the x-ray energies. The penetration depth can be solved from Equation 2.25 to yield

$$\Lambda(x) = \frac{d}{2\mathrm{Re}\,(\eta)} = \frac{d}{2\mathrm{Re}\,\left(g\sqrt{1-x^2}\right)}, \tag{4.2}$$

which is decidedly not linear to E. Figure 4.8 graphs both the (nonlinear) penetration depth and relative average interplanar spacing against energy (top and bottom graphs, respectively). In our aim to plot the relative average interplanar spacing against penetration depth for insight into the one-dimensional strain profile of the crystal, we mention noteable features of the plot. An subtle trait of the relative average interplanar spacing data is that it is always slanted upwards, and this turns out to be quite important. More obviously, we see unmistakeable aberrations near the K-edge of germanium around 11.1 keV in both graphs. The behavior of the penetration depth near the K-edge is expected and has been explained, but that of the

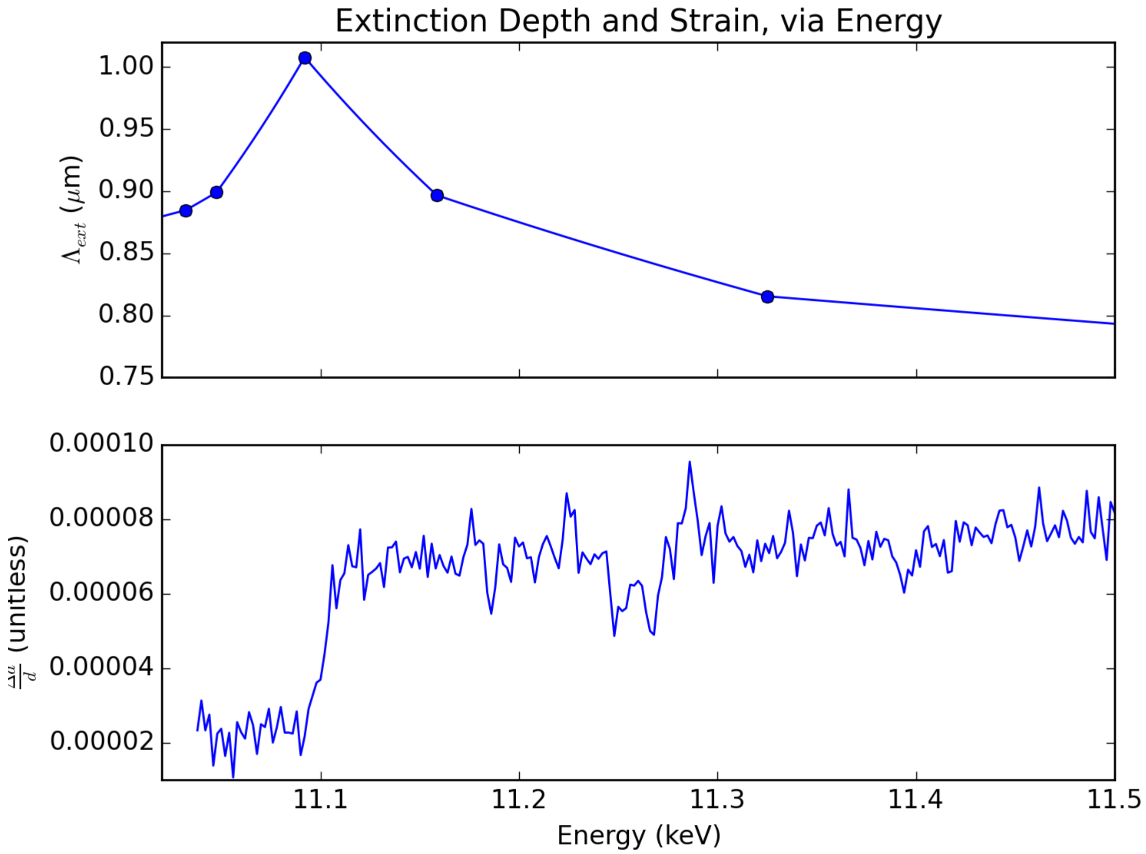


Figure 4.8: Graph of relative difference between strained and unstrained lattice spacing against energy (top) and extinction depth of energies (bottom). Note the abberant behavior near the K-edge of 11.1 keV for Ge (top), and the nonlinearity about it in the bottom graph.

bottom graph leads to two very different results for average interplanar spacing at the same depths.

The discontinuity of the data about the K-edge is not understood by the author. Above the K-edge, x-rays have sufficient energy to expell K-edge electrons, which are then filled by higher shell electrons resulting in the emission of either fluorescent x-ray emission or excite a higher shell auger electron. The only relevant effect observed in the original raw data (of the form of Figure 4.7) is the increased volume of the strained curves *below* the K-edge. Below the K-edge, both the width and height of the strained data surpasses that of the unstrained, and while not the subject of this paper, this is elaborated upon in Section 5.1. The remainder of this chapter focuses on the data above the k-edge, which is plotted using the transformation of Figure 4.8 in Figure 4.9.

4.2.2 Results

By translating the independent axis nonlinearly according to the penetration depth illustrated in Figure 4.8 (top), we can recast the bottom figure of relative average interplanar spacing to be in terms of penetration depth instead of energy, as in Figure 4.9. Before examining this figure, let's work backwards from a strain profile to build an expectation of what the figure should look like given a strain profile.

The lattice constant of germanium is 5.64 Å(see §3.2.1), which places its interplanar (111) spacing on the order of angstroms. With the justification of the (albeit nonlinear) lower resolution of the data presented in Figure 4.9, we make the approximation that our depth profile is continuous. When the crystal is pulsed by the laser, a longitudinal phonon pulse travels through the material in one dimension as outlined in Section 3.1, and this pulse has velocity 5510 $\frac{\text{m}}{\text{s}}$[8]. Our strained data is taken 250 ps after the laser pulse, so the pulse should be located at a depth 1.3 μm, which is beyond the penetration depths reachable by our energies as given in Figure 4.8 by about .3 μm.

The laser pulse has created a phonon pulse which travels down the crystal and is at an unreachable depth of 1.3 μm at the time of our depth-varying snapshots. When the laser first excites the material at its top boundary, the pulse is created and travels downwards. This provokes a restoring force in the layers above which, stretched, pull the layers above them, creating a second, lesser phonon pulse travelling down the material. The depth profile is then expected to exhibit periodicity with an amplitude increasing with depth up to the original phonon pulse of maximum compression. The cumulative average over such a profile is likewise periodic and with diminishing amplitude.

While our actual result in Figure 4.9 doesn't match our predictions, the figure holds important information. First, note that if the depth profile did match our theoretical one, the amplitude of its cumulative average lattice spacing would very quickly dissipate to much less than the original amplitude, since we are calculating a cumulative average; since we only started around .8 μm, this could be a region dominated by noise. Second, note that the strained lattice spacing is *always larger*; this implies either that the depth profile's periodic wavelength is at least twice our range of penetration depths, or that the profile is not periodic, or that an effect of thermal expansion has been neglected. Unfortunately, without more insight into the form of the depth profile at the time of our snapshot, it is difficult to separate noise from feature. For example, one might interpret the peak-trough-peak feature from .83 to .86 μ m to be the cumulative average over a sinusoidal profile, but it is more constructive to work backwards from the theoretical depth profile, which is beyond the scope of this paper, to ensure that one is not analyzing noise.

One major trend that is clear across all measured energies above the K-edge is a downward slope in expansion as penetration depth increases. This means that the more we measure, the closer we get to the

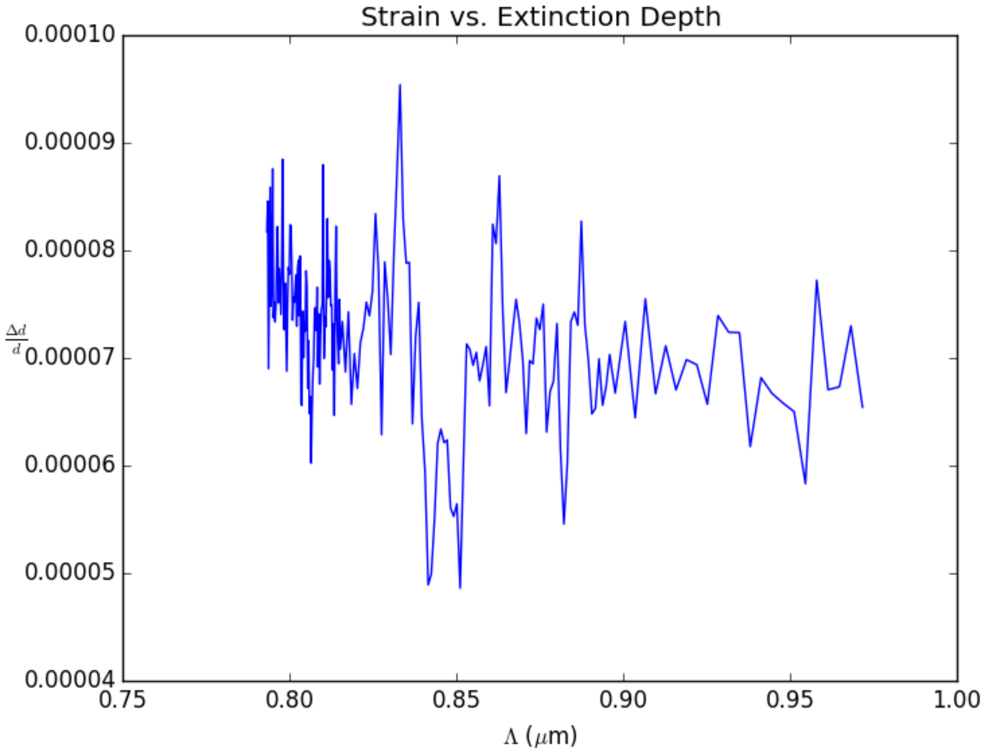


Figure 4.9: Cumulative average interplanar spacing difference ratio, this time as a function of penetraton depth.

unstrained depth, which is what we would expect from any strain profile.

Chapter 5

Conclusions

By analyzing data of cumulative average interplanar distance obtained from Darwin curves of a laser-strained and unstrained crystal lattice at the same snapshot in time, we have observed a crystal whose layers are expanded in our depth regime, whose average interplanar distance nears the unstrained interplanar spacing with increasing depth. We have concluded that while a compression pulse is known to be located at 1.3 μm within the material at our snapshots all taken 250 ps after the laser pulse strain, the K-edge (the highest-reaching edge) only reaches just above 1 μm where the depth profile is not known, making discernment between features and noise difficult. Nevertheless, an overall slope of the strained average interplanar spacing tending towards the unstrained spacing is a promising feature.

5.1 Future Work

Many areas of this paper call for further analysis in multiwavelength x-ray diffraction. The most fundamental such expansion is multiwavelength x-ray diffraction over a known depth profile with maximal data resolution (ie, data at more closely spread energies near the K-edge, and at more angles). The resolution would call for the adaptation of the code to perform chi-square fits with Darwin curves instead of Lorentzians to account for the asymmetry that comes with higher resolution data, whereas the known depth profile can help discern data from noise. With this two-front approach, a method can be validated for understanding the depth profile from the average relative interplanar spacing. As mentioned in §4.2, the chi-square method is not difficult to scale in this regard.

Further analysis is also warranted on the volume of the Darwin curves; below the K-edge, the strained peak is wider and taller than the unstrained, whereas the opposite is true of the curves above the K-edge. The interplanar spacing data exhibits irregularities about the K-edge implying discontinuity, which could point to distortion of the electrons in the material above the K-edge, and this has not been examined. The volumes should also play a part in normalization and accuracy of the chi-square fit over the Darwin curve mentioned above. Most importantly, the relative volume gives key insight into the density of planes in going from the cumulative average spacing from one penetration depth to the next, which can contribute to results concerning the depth profile.

With a theoretical depth profile to discern noise and an analysis of the volume as well as the interplanar spacing, methods for the reformulation of the depth profile can be derived and validated.

Bibliography

1. Jens Als-Nielsen and Des McMorrow. *Elements of Modern X-Ray Physics*. John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom, 2 edition, 2011.

2. Lattice constants. `http://7id.xray.aps.anl.gov/calculators/crystal_lattice_parameters.html`. [online, accessed 08-09-14].

3. Overview — advanced photon source. `http://www.aps.anl.gov/About/APS_Overview/`. [online, accessed 08-08-14].

4. Mda file viewer. `http://sector7.xray.aps.anl.gov/cgi-bin/mdaview.cgi?file=old/idd/2006/Feb/7idd_0046.mda`, Feb 2006. [online, accessed 08-09-14].

5. U. W. Arndt, D. C. Creagh, R. D. Deslattes, J. H. Hubbell, P. Indelicato, E. G. Kessler Jr, and E. Lindroth. X-rays. In *International Tables for Crystallography*, volume C, chapter 4.2, pages 191–258. Wiley, 2006. [doi:10.1107/97809553602060000600].

6. P. J. Brown, A. G. Fox, E. N. Maslen, M. A. O'Keefe, and B. T. M. Willis. Intensity of diffracted intensities. In *International Tables for Crystallography*, volume C, chapter 6.1, pages 554–595. Wiley, 2006. [doi:10.1107/97809553602060000600].

7. C. T. Chantler. Theoretical form factor, attenuation, and scattering tabulation for z=1-92 from e=1–10 ev to e=.4–1.0 mev. *Journal of Physical and Chemical Reference Data*, 24(1):71–643, 1995.

8. Mechanical properties, elastic constants, lattice vibrations of germanium (ge). `http://www.ioffe.rssi.ru/SVA/NSM/Semicond/Ge/mechanic.html`. [online, accessed Aug 2014].

9. Nist x-ray form factor, atten. scatt. tables form page. `http://physics.nist.gov/PhysRefData/FFast/html/form.html`. [online, accessed Aug 2014].

10. Codata value: speed of light in vacuum. `http://physics.nist.gov/cgi-bin/cuu/Value?c`, 2010. [online, accessed Aug 2014].

11. Codata value: Planck constant in ev s. `http://physics.nist.gov/cgi-bin/cuu/Value?hev`, 2010. [online, accessed Aug 2014.

12. Codata value: classical electron radius. `http://physics.nist.gov/cgi-bin/cuu/Value?re`, 2010. [online, accessed Aug 2014].

13. Sergey Stepanov. X-ray dynamical diffraction on the web. `http://x-server.gmca.aps.anl.gov/x0h.html`, December 2013. [online, accessed Aug 2014].

14. G. Jackson Williams. Probing impulsive strain and lattice dynamics by time-resolved ultrafast x-ray diffraction. Master's thesis, DePaul University, June 2010.

Appendix A

Matlab Code

A.1 populateGlobals.m—Imports scattering coefficients and correction factors from custom file.

```
%% Load scattering coefficients and correction factors
%  From converted Crystallographic Tables data

function []=populateGlobals(dataPath)
% Inputs:
%   dataPath*:   Absolute path of scatcoef.txt and corrfax.txt
%                converted files.
% Outputs: None.  Instead updates global variables:
%   scatcoef:    Scattering coefficients in the form
%                {{'elementName' [a1 b1 a2 b2 a3 b3 a4 b4 c]},..}
%   corrfax:     Correction factors, of the form
%                {{'elementName' [9 f' values] [9 f'' values]},...}
%                at different energies as per the code.
%   paperDir     Directory of Paper folder, to which everything is relative.

    global scatcoef corrfax paperDir

    % Set Paper directory to two directories higher (remove two ending /)
    paperDir=regexprep(cd,'[/\\][^/\\]*[/\\][^/\\]*[/\\]?$','/');

    if(nargin==0)
        dataPath=[paperDir 'Code/Data/coefs/CTs'];
    end

    [fid,msg]=fopen(strcat(dataPath,'scatcoef.txt'),'r');
    if(fid==-1)
        disp(msg)
        return
    end
    tline=fgetl(fid);
    scatcoef={};
    while ischar(tline)
        s=regexp(tline,'\t','split');
        s{1}=regexprep(regexprep(s{1},'+','PLUS'),'-','MINUS');
        %s{2}=[str2num(s{2})];
        %scatcoef{end+1}=s;
        %disp(scatcoef{end}{2}(1))
        scatcoef.(s{1})=[str2num(s{2:end})];
        %scatcoef(end+1)={{s(1) [str2num(char(s(2:end)))]}};
        tline=fgetl(fid);
    end
    fclose(fid);
```

```
[fid,msg]=fopen(strcat(dataPath,'corrfax.txt'),'r');
if(fid==-1)
    disp(msg);
    return
end
tline=fgetl(fid);
corrfax={};
while ischar(tline)
    s=regexp(tline,'\t','split');
    corrfax.(s{1})=[str2num(s{2});str2num(s{3})];
    %disp(corrfax{end}{2}(1,:))
    tline=fgetl(fid);
end
fclose(fid);
```

A.2 f0.m—Atomic form factor

```
%% Returns the atomic form factor
%  Run populateGlobals.m first
%
% Inputs:
%   ele:            --- Element name
%   a:              \AA Element Lattice Parameter
%   Q:              1/m Difference wavevector
% Outputs:
%   lef0:               Form factor without corrections or absorption

function [lef0]=f0(ele,Q)

global scatcoef

if(ele=='Si')                           % Match CT
    ele='Siv';
end
ele=regexprep(regexprep(ele,'+','PLUS'),'-','MINUS');

i=1;
while(i<length(scatcoef) && scatcoef{i}{1}~=ele)
    i=i+1;
end

lef0=scatcoef.(ele)(9);
for(i=1:4)
    lef0=lef0+scatcoef.(ele)(2*i-1)*exp(-scatcoef.(ele)(2*i)*(Q/(4*pi))^2);
end
```

A.3 fp.m—Correction & Absorption factors

```
%% Return constant reduction factors
%  Run populateGlobals.m first
%
% Inputs:
%   ele:        --- Element name
%   E:          eV  Energy of incident x-ray
%   wavelength: m   wavelength of incident x-ray, overrides E
% Outputs:
%   f'-i f'':        Correction factor, from Crystallographic tables.


function [lefp]=fp(ele,wavelength)
global corrfax

% The following accords with the data gathered from the CTs
wavelengths=[2.74851 2.28962 1.93597 1.788965 1.54052 0.70926 0.55936 0.215947 ...
             0.20901 0.180195]*1E-10;

fpp=interp1(wavelengths',corrfax.(ele)',wavelength);
lefp=fpp(1)-1j*fpp(2);
```

A.4 F.m—Atomic form factor

```
%% F.m: Return atomic form factor
%  Be sure to run populateGlobals.m first.
%  Run as F(ele,a,wavelength,hkl).  Set flags to enter energy instead of
%  wavelength or angle instead of Miller index.  See examples for details.
%
% Inputs:
%   ele:            ——— Element name
%   a:              \AA Element Lattice Parameter
%   wavelength:     m   Wavelength of incident beam (default)
%   hkl:            ——— Miller index, in the form hkl for single integers
%                       or [h k l] for multi—integer or negative indices.
%   E:              eV  Energy of incident beam (overrides wavelength)
%   angle:          rad Angle (overrides hkl)
%   flag*:          ——— To enter energy instead of wavelength, flag 'E'
%                       to enter angle instead of miller index, flag 'a'
%                       for both, flag 'Ea'.
% Outputs:
%   leF:                Atomic form factor
%
% Examples:
%   F(ele,a,E         ,hkl)
%   F(ele,a,wavelength,hkl,  'w')
%   F(ele,a,E         ,angle,'a')
%   F(ele,a,wavelength,angle,'wa')

function [leF]=F(ele,a,varargin)
% F(ele,a,wavelength[E],hkl[angle],'[Ea]'*)

PlanckH=4.135667516E—15;
c=299792458;

if(nargin~=4 && nargin ~=5)
    disp('Incorrect usage F.  Requires 4 or 5 inputs.  See "help F".');
    return
end

if(nargin==5 && ~isempty(strfind(varargin{3},'E')))        % energy passed
    E=varargin{1};
    wavelength=PlanckH*c/E;
else                                                       % wavelength passed
    wavelength=varargin{1};
    E=PlanckH*c/wavelength;
end

if(nargin==5 && ~isempty(strfind(varargin{3},'a')))        % angle specified
    Q=4*pi*sin(varargin{2})/wavelength;
else                                                       % hkl specified
    if(length(varargin{2})==1)
        hkl=num2cell(str2num(regexprep(sprintf('%03d',varargin{2}),'(\d)','$1 ')));
        [h,k,l]=hkl{:};
    else
        hkl=num2cell(varargin{2});
        [h,k,l]=hkl{:};
    end
    if(h==k && k==l && l==0)
        Q=0;
    else
        d=a/sqrt(h*h+k*k+l*l);
        Q=2*pi/d;
    end
    angle=asin(Q*wavelength/(4*pi));
end

leF=f0(ele,Q)+fp(ele,wavelength);
```

A.5 Fuc.m—Unit cell form factor

```
%% Fuc.m: Return unit cell form factor of 2—element zinc—blende structure
%  Be sure to run populateGlobals.m first
%  Run as F(ele1,ele2,a,wavelength,hkl).  Set flags to enter energy instead of
%  wavelength or angle instead of Miller index.  See F.m help for details.
%
% Inputs:
%   ele1:           ——— First element name
%   ele1:           ——— Second element name
%   a:              \AA Element Lattice Parameter
%   wavelength:     m   Wavelength of incident beam (default)
%   hkl:            ——— Miller index, in the form hkl for single integers
%                       or [h k l] for multi—integer or negative indices.
%   E:              eV  Energy of incident beam (overrides wavelength)
%   angle:          rad Angle (overrides hkl)
%   flag*:          ——— To enter energy instead of wavelength, flag 'E'
%                       to enter angle instead of miller index, flag 'a'
%                       for both, flag 'Ea'.
% Outputs:
%   leFuc:              Atomic form factor

function [leFuc]=Fuc(ele1,ele2,a,varargin)
% Fuc(ele1,ele2,a,wavelength[E],hkl[angle],'[Ea]')

PlanckH=4.135667516E—15;
c=299792458;

if(nargin==6)
    flags=varargin{3};
elseif(nargin==5)
    flags='';
else
    disp('Incorrect usage Fuc.  Requires 5 or 6 inputs.  See "help Fuc".');
    return
end

% Parse flags
if(nargin==6 && ~isempty(strfind(varargin{3},'E')))       % energy passed
    E=varargin{1};
    wavelength=PlanckH*c/E;
else                                                      % wavelength passed
    wavelength=varargin{1};
    E=PlanckH*c/wavelength;
end

if(nargin==6 && ~isempty(strfind(varargin{3},'a')))       % angle specified
    Q=4*pi*sin(varargin{2})/wavelength;
else                                                      % hkl specified
    if(length(varargin{2})==1)
        hkl=num2cell(str2num(regexprep(sprintf('%03d',varargin{2}),'(\d)','$1 ')));
        [h,k,l]=hkl{:};
    else
        hkl=num2cell(varargin{2});
        [h,k,l]=hkl{:};
    end
    if(h==k && k==l && l==0)
        Q=0;
    else
        d=a/sqrt(h*h+k*k+l*l);
        Q=2*pi/d;
    end
    angle=asin(Q*wavelength/(4*pi));
end

leFuc=F(ele1,a,varargin{1},varargin{2},flags)*...
```

```
       abs(1+exp(2*pi*1j*(h/4+k/4+l/4)))*...
       (1+exp(1j*pi*(h+k))+exp(1j*pi*(k+l))+exp(1j*pi*(h+l)));

if(false)
    disp(ele1);
    disp(ele2);
    disp(a);
    disp(varargin{1});
    disp(varargin{2});
    disp(varargin{3});
    disp(exp(2*pi*1j*(h/4+k/4+l/4)));
    disp((1+exp(1j*pi*(h+k))+exp(1j*pi*(k+l))+exp(1j*pi*(h+l))));
end
```

A.6 Darwin.m—Darwin plot

```
%% Darwin.m: Plot Darwin rocking curve
%  Be sure to run populateGlobals.m first
%  Run as Darwin(ele1,ele2,a,wavelength,hkl).  Set flags to enter energy instead of
%  wavelength or angle instead of Miller index.  See F.m help for details.
%
% Inputs:
%   ele1:           ——  First element name
%   ele1:           ——  Second element name
%   a:              \AA Element Lattice Parameter
%   wavelength:     m   Wavelength of incident beam (default)
%   hkl:            ——  Miller index, in the form hkl for single integers
%                        or [h k l] for multi-integer or negative indices.
%   E:              eV  Energy of incident beam (overrides wavelength)
%   angle:          rad Angle (overrides hkl)
%   flag*:          ——  single characters that combine, 6th fn parameter.
%   Flags:
%     E:    Treat parameter 4 as energy instead of wavelength
%     a:    Treat parameter 5 as angle instead of Miller index
%     n:    No absorption
%     s:    Save image (predetermined path/name), can be overriden by
%           extra 'S' argument.
%     h:    Hide plot (do not show figure)
%   Extra Args:         Extra arguments may be specified beginning with the
%                       7th parameter (you can leave flags as an empty
%                       string) in the form Darwin(...,'X0h','Si10kev').
%     X0h:  Specify X0h file for comparison on the same figure
%     S:    Save with this filename instead of the default.
% Outputs:
%   leFuc:              Atomic form factor

function []=Darwin(ele1,ele2,a,varargin)
% Darwin(ele1,ele2,a,wavelength[E],hkl[angle],'[Ea]')

global paperDir

PlanckH=4.135667516E-15;                % eVs Planck's in eV s (NIST/hev)
c=299792458;                            % m/s Speed of light (NIST/c)
r0=2.8179403267E-5;                     % \AA Electron radius (NIST/re)

if(nargin>=6)
    flags=regexprep(varargin{3},'[n]',''); % These flags make it no further
    if(nargin>6)                        % Extra arguments
        if(~mod(nargin,2)==0)
            disp('Incorrect usage of Darwin; see help file on extra arguments.');
            return
        end
        for i=4:2:nargin-3
            switch varargin{i}
              case 'X0h'
                X0hName=varargin{i+1};
              case 'S'
                saveName=varargin{i+1};
              otherwise
                disp(['Unrecognized option ' varargin{i} '; see "help ' ...
                      'Darwin".']);
            end
        end
    end
elseif(nargin<=5)
    flags='';
    if(nargin<5)
        disp('Incorrect usage Darwin.  Requires 5 or 6 inputs.  See "help Darwin".');
        return
    end
end

% Parse flags
if(nargin>=6 && ~isempty(strfind(varargin{3},'E')))       % energy passed
    E=varargin{1};
    wavelength=PlanckH*c/E;
else                                                      % wavelength passed
    wavelength=varargin{1};
    E=PlanckH*c/wavelength;
end

if(nargin>=6 && ~isempty(strfind(varargin{3},'a')))       % angle specified
    Q=4*pi*sin(varargin{2})/wavelength;
    d=lambda/(2*sin(theta));                              % (m) Assume first order
else                                                      % hkl specified
    if(length(varargin{2})==1)
        hkl=num2cell(str2num(regexprep(sprintf('%03d',varargin{2}),'(\d)','$1 ')));
        [h,k,l]=hkl{:};
    else
        hkl=num2cell(varargin{2});
        [h,k,l]=hkl{:};
    end
    if(h==k && k==l && l==0)            % This should never be called.
        Q=0;
    else
        d=a/sqrt(h*h+k*k+l*l);
        Q=2*pi/d;
    end
    angle=asin(Q*wavelength*1E10/(4*pi));
end

% Start Rocking Curve calculations

vc=a^3;                                 % \AA^3 Unit cell volume
m=1;

F0=Fuc(ele1,ele2,a,varargin{1},0,flags);
F=Fuc(ele1,ele2,a,varargin{1},varargin{2},flags);
if(nargin>=6 && ~isempty(strfind(varargin{3},'n'))) % no absorption
    F=abs(F);
    F0=abs(F0);
    nowith='no';
    absno=0;
else
    nowith='with';
    absno=1;
end
g0=2*d^2*r0/vc*F0;
g=2*d^2*r0/vc*F;

x=-5:.01:5;
zetas=real(g0/pi+x.*g/pi);
xc=pi.*zetas/g-g0/g;
thetas=zetas.*tan(angle)*180000/pi;
```

```
Rs=abs(r(xc)).^2;

hold off;plot(thetas,Rs,'b-');hold on;

if(ele1==ele2)
    ele12=ele1;
else
    ele12=[ele1 ele2];
end
plotName=sprintf('%s(%d%d%d)%d',ele12,h,k,l,absno);
if(~exist('saveName'))
    saveName=plotName;
end

basepath=regexprep(cd,'/[^/\\]*/[^/\\]*[/\\]?$','/'); % Up to Paper directory
tmptitle=[plotName ' wavelength=' num2str(round(wavelength*1E12)/100) ...
          ' Angstroms, ' nowith ' absorption'];
title(tmptitle,'Interpret','tex');
xlabel('\omega (millidegrees)');
ylabel('Intensity reflectivity');
% Plot X0h
if(exist('X0hName'))
    xoh=load([paperDir 'Code/Data/X0h/' X0hName  '.dat']);
    plot(xoh(:,1)/3.6,xoh(:,2),'k--');
    legend('me','X0h');
end

print('-dpng',[paperDir 'Images/Background/X0h_Comps/matlab/' ...
               saveName]);


function [R]=r(xc)
    R=[xc(real(xc)<=-1)+sqrt(xc(real(xc)<=-1).^2-1) ...
       xc(abs(real(xc))<1)-1j*sqrt(1-xc(abs(real(xc))<1).^2) ...
       xc(real(xc)>=1)-sqrt(xc(real(xc)>=1).^2-1)];
```

A.7 extractspec.m—Code for parsing spec file, written by G. Jackson Williams 2013

```
% extractspec—Written by G Jackson Williams, Summer 2013
% Extracts from a given .spec file 'filename' the specified scan(s) 'scan'.
function Data = extractspec(filename, scan)

%% Convert to a format matlab will like
% Test to see if the .spec needs to be edited
[fid, message] = fopen(filename,'r','b');
if(fid < 0)
    fprintf('Warning: No such file exists in directory! \n');
    fprintf('          Please choose new filename         \n');
    disp(message)
    Data = [];
    return
end
tline = fgetl(fid);
if(strcmp(tline(1),'#'))
    fprintf('\nConverting original .spec into something Matlab will like!\n');
    tic
    i = 1;
    while ischar(tline)
        tline = strrep(tline,'#','%');
        NewFile{i} = tline;
        i = i+1;
        tline = fgetl(fid);
    end
    fclose(fid);

    fid_new = fopen(filename, 'w');
    for n = 1:i-1
        fprintf(fid_new,'%s\n',NewFile{n});
    end
    fclose(fid);
    pause(0.1);
    t = toc;
    fprintf(['Conversion took ' num2str(t,3) ' seconds\n']);
end

%% Handle all lines
% List the indexes of where the scans start
[fid, message] = fopen(filename,'r','b');
i = 1;
k = 1;
ScanData = cell(scan,1);     % cell(a,b) creates an a x b cell array of empty matrices
                             % The 'scan' size is due to not reading more
                             % than we need.
% The following while loop reads every line, so the code in it must call
% every case.
while ischar(tline)          % Originally set in ln. 17
    if(isempty(tline))
        tline = fgetl(fid);
        continue
    end

    % The line with %S gives you the scan number
    if(strncmp(tline,'%S ',3))
        for j = 4:length(tline)
            if(double(tline(j)) == 32)           % 32 is space ASCII
                break
            end
        end
        ScanNum(k,1) = str2double(tline(4:j-1));% Just scan number: 4 skips the "%S #", ends at
            space
        ScanHeader{k} = tline;                  % Save entire line to SH
        ScanIndex(k,1) = i;

        % Remember that scan can be a vector.
        if(ScanNum(k,1) == max(scan(:))+1)      % Don't read more than needed
            break
        end
    end
    if(strncmp(tline,'%L ',3))                  % These are column headers
        Header{k} = tline;
        k = k+1;
    end

    if(strcmp(tline(1),'%'))                    % Ignore further comments
    else
        % Put the data as a new col in ScanData{ScanNum(see ln. 70)}
        % sscanf returns a column vector of all %gs read.
        ScanData{ScanNum(k-1)} =  [ScanData{ScanNum(k-1)}; sscanf(tline,'%g')'];
        i = i+1;
    end
    tline = fgetl(fid);
end

if(scan > k - 1)    % Didn't find your scan #, reached eof
    fprintf(['Warning: This spec file does not have a Scan #' num2str(scan) '\n']);
    fprintf(['         The maximum scan number for this file is ' num2str(k-1) '\n']);
    Data = [];
    return
end

%% Formulate Data struct to return
%data = load(filename);
data = ScanData{scan};                          % Only return those scans we asked for

% strread: read a string based on a delimiter
headerparts = strread(ScanHeader{k-1},'%s','delimiter',' ');% Get scan header (%S SCANNUMBER ...)
headerp = strread(Header{k-1},'%s','delimiter',' ');        % Get data header (%L COLNAMES ...)

Data.scantype = headerparts{3};                 % Send back in struct with metadata
Data.motorname = headerparts{4};
Data.startpos = str2double(headerparts{5});
Data.endpos = str2double(headerparts{6});
Data.numsteps = str2double(headerparts{7});
Data.counttime = str2double(headerparts{8});
Data.dataheader = headerp;
Data.data = data;
imax = length(headerp) - 1;                     % Return each column in its own member
for i = 1:imax;
    eval(['Data.' headerp{i+1} ' = data(:,' num2str(i) ');']);
end


%% Close file
status = fclose('all');
if(status < 0)
    fprintf('Warning: Opened files did not close properly!');
end



end
```

Appendix B

Python Code

B.1 MakePlots.py—Code generating figures in this paper.

```
#!/usr/bin/python2
import re,os,sys
from cmath import *
import matplotlib.pyplot as plt
from numpy import arange
from scipy import interpolate

import FormFactor.formfactor as ff
import Spec.Spec as spec
import Darwin.Darwin as dwn
import MDA.MDA as mda

saveDir='/'.join(os.path.dirname(os.path.realpath(__file__)).split('/')[:-2])+'/Images/'

# Define physical constants
PlanckH=4.135667516E-15;                # eVs Planck's in eV s (NIST/hev)
c=299792458;                            # m/s Speed of light (NIST/c)
r0=2.8179403267E-5;                     # \AA Electron radius (NIST/re)

def main(show=False,save=True):
    """Create and save every figure used in the paper."""
    ff.init(['Ge','Si','Ga','As','In','Sb','C'])

    Fig_2_10(show,save)
    Fig_2_13(show,save)
    Fig_2_14(show,save)
    Fig_2_16(show,save)
    Fig_3_02(show,save)
    Fig_3_03(show,save)
    Fig_3_04(show,save)
    Fig_4_01(show,save)
    Fig_4_02(show,save)
    Fig_4_03(show,save)
    Fig_4_04(show,save)
    Fig_4_05(show,save)
    Fig_4_06(show,save)
    Fig_4_07(show,save)
    Fig_4_08(show,save)
    Fig_4_09(show,save)

def Fig_2_10(show=False,save=True):
    """Kinematical result for many layers as given by Equation 2.35, with diverging parts
        replaced by |r_N|^2=1."""
    x=arange(0.001,10,.01)
    a=lambda x: 1/(x-4)**2      # zeta_0/m=1
    plt.plot(x,[min(a(s),1) for s in x])
    plt.xticks([4],[r'$\frac{\zeta_0}{m}$'])
    plt.ylim([0,1.03])
    plt.ylabel('Intensity reflectivity',rotation=90)
    plt.xlabel(r'$\zeta$')
    plt.tick_params(axis='x',labelsize=20)
    if save: plt.savefig(saveDir+'2_10_Kinematical.png',dpi=200)
    if show: plt.show()
    plt.clf()

def Fig_2_13(show=False,save=True):
    """Darwin curve without absorption effects--the reflectivity is 100% in the middle, but
        tapers off to the sides.  The intensity reflectivity is the square of the amplitude
        reflectivity given in Equation 2.42 when |x|>=1 and equals 1 (the product of the complex
        number and its conjugate in Equation 2.42) when |x|<=1"""

    dwn.plotRockingCurve('Si','Si',5.4309,111,wavelength=1.5495e-10,xaxis='xc',absorption=False,
        normalShift=True,save=save,show=show,saveAs='2_13_DarwinNoAbsorption.png',xwidth=3,
        plotTitle='Darwin Curve Without Absorption')

def Fig_2_14(show=False,save=True):
    """Darwin reflectivity curve for Ge(111) probed with an 8 keV at sigma orientation."""
    dwn.plotRockingCurve('Ge','Ge',5.6461,111,E=8000.,save=save,saveAs='2_14_DarwinNoAbsorption.
        png',show=show,absorption=False,plotTitleEnergy=True,normalShift=True)

def Fig_2_16(show=False,save=True):
    """Darwin reflectivity curve for Ge(111) probed with an 8 keV at sigma orientation accounting
         for absorption effects."""
    dwn.plotRockingCurve('Ge','Ge',5.6461,111,E=8000.,save=save,saveAs='2_16_DarwinAbsorption.png
        ',show=show,absorption=True,plotTitleEnergy=True,normalShift=True)

def Fig_3_02(show=False,save=True):
    """Graph of dispersion corrections from the Crystallographic Tables 5 (top) and NIST 9 (
        bottom). The NIST data is more abundant near the band edges, where the form factors
        display erratic behavior. The lines are the linear interpolation (used in this paper)
        between the points, which represent energies for which the source provides explicit data.
        """
    global PlanckH,c

    a=5.6461;hkl=111;
    energies=arange(5,30,.01)

    Nfs=[[q.real,q.imag] for q in [ff.fp('Ge',E=E*1000,source='NIST') for E in energies]]
    Cfs=[[q.real,q.imag] for q in [ff.fp('Ge',E=E*1000,source='CTs') for E in energies]]
    [Nfp,Nfpp]=zip(*Nfs)
    [Cfp,Cfpp]=zip(*Cfs)

    [NIP,CIP]=[ff.getInterpPoints(source=sources,ele='Ge') for sources in ['NIST','CTs']]
    [Nfpo,Nfppo]=zip(*[[q.real,q.imag] for q in [ff.fp('Ge',E=E*1000,source='NIST') for E in NIP
        ]])
    [Cfpo,Cfppo]=zip(*[[q.real,q.imag] for q in [ff.fp('Ge',wavelength=wl,source='CTs') for wl in
         CIP]])

    f1,(ax1,ax2)=plt.subplots(2,sharex=True,sharey=True)

    CIPE=[PlanckH*c/q/1000. for q in CIP]
    ax1.plot(energies,Cfp,'b-')
    ax1.plot(energies,Cfpp,'r-')
    ax1.plot(CIPE,Cfpo,'bo')
    ax1.plot(CIPE,Cfppo,'ro')
    axfp,=ax1.plot([1e4],[1e4],'ro-')
    axfpp,=ax1.plot([1e4],[1e4],'bo-')
    ax1.legend([axfp,axfpp],["f'","f''"],loc=4)
    ax1.set_ylabel('Crystallographic Tables')

    ax2.plot(energies,Nfp,'b-')
    ax2.plot(energies,Nfpp,'r-')
    ax2.plot(NIP,Nfpo,'bo')
    ax2.plot(NIP,Nfppo,'ro')
    axfp,=ax2.plot([1e4],[1e4],'ro-')
    axfpp,=ax2.plot([1e4],[1e4],'bo-')
    ax2.legend([axfp,axfpp],["f'","f''"],loc=4)
    ax2.set_ylabel('NIST')

    ax1.set_title('Dispersive Form Factor Corrections for Ge(111)')
    plt.xlabel('Energy (keV)')
    plt.ylim([-7,1])
    plt.xlim([5,30])

    if save: plt.savefig(saveDir+'3_02_ff.png',dpi=200)
    if show: plt.show()
    plt.clf()

def Fig_3_03(show=False,save=True):
    """Raw data for energies at the extremum and median of data, unnormalized and shifted from
Bragg angle."""
    AnalyzeFrame(fitL=False,save=save,show=show,scanNums=[1,124,232])

def Fig_3_04(show=False,save=True):
    """Superposition of raw data with Lorentz fits at different energies. This demonstrates the
        variance of fluctuations in height, width, and absolute angle between energies, none of
        which play a major role in data gathering thanks to reference data of the unstrained
        crystal for each measurement."""

    Ge=mda.MDA('7idd_0046.mda','all',['7cart:m3.VAL','S16','S15'])
    ax=[]
    colors=['c','r','b','k']
    for scan,c in zip([Ge.scans[q] for q in [231,194,80,1]],colors):
        x=[xx*1000. for xx in scan.data['7cart:m3.VAL']]
        p=scan.fitLorentzian(x,scan.data['S15'])
        xxs=arange(min(x),max(x),(max(x)-min(x))/1000.)

        plt.errorbar(x,scan.data['S15'],yerr=[sqrt(q) for q in scan.data['counts2']],fmt=(c+'o'))
        axi=plt.errorbar(1e4,1e4,yerr=.01,fmt=c+'o-')
        axL1,=plt.plot([xx for xx in xxs],[scan.Lorentzian(p[0],xx) for xx in xxs],c+'-')
        ax.append([axi,r'$E$=%.3f keV'%scan.Y])

    plt.xlabel(r'$\theta$ (millidegrees)')
    plt.ylabel('Counts (per 10 s)')
    plt.title('Effect of energy on strained rocking curves')

    plt.legend(*zip(*ax),loc=2)
    plt.ylim([0,1150])
    plt.xlim([-16,6])
    if save: plt.savefig(saveDir+'3_04_merGe.png',dpi=200)
    if show: plt.show()
    plt.clf()

def Fig_4_01(show=False,save=True):
    """Darwin curves for Gesigma at different miller indices, compared with X0h."""
    dwn.plotRockingCurve('Ge','Ge',5.6461,400,1.54056e-10,X0hComp='Ge400_154',save=save,show=show
        ,saveAs='4_01_Ge_400',absorption=True,noabsorbbg=True)
    dwn.plotRockingCurve('Ge','Ge',5.6461,111,1.54056e-10,X0hComp='Ge111_154',save=save,show=show
        ,saveAs='4_01_Ge_111',absorption=True,noabsorbbg=True)

def Fig_4_02(show=False,save=True):
    """Gallium Arsenide at miller index 400."""
    dwn.plotRockingCurve('Ga','As',5.6461,400,wavelength=1.54056e-10,X0hComp='GaAs400',absorption
        =True,save=save,saveAs='4_02_GaAs_400',show=show,noabsorbbg=True)

def Fig_4_03(show=False,save=True):
    """General computational agreement of varying Darwin curves with X0h at different x-ray
        energies near the K-edge."""
    for i in range(11000,11151,50):
        dwn.plotRockingCurve('Ge','Ge',5.6461,400,E=i,X0hComp='Ge'+str(i),saveAs='4_03_Ge(400)'+
            str(i),absorption=True,save=save,show=show,noabsorbbg=True)
```

```
def Fig_4_04(show=False,save=True):
    """The effect of varying energies on Si(111)"""
    # Different energies of Si(111)
    a=5.4309;hkl=111;wavelength=None

    axnames=[]                                 # plot handles
    for E in [5000,10000,50000]:
        [wavelength,E,Q,angle,[h,k,l]]=ff.__parseParameters(a,hkl,wavelength,E)
        [thetas,rs]=dwn.RockingCurve('Si','Si',a,hkl,wavelength,E,'thetas',absorption=True,
            normalShift=False)
        axi,=plt.plot(thetas,rs)
        axnames.append([axi,r'$E$=%d keV'%E])
    plt.title('Darwin curves of Si(111) for varying energies')
    plt.xlabel(r'$\omega$ (millidegrees)')
    plt.ylabel(r'Intensity reflectivity')
    plt.legend(*zip(*axnames))
    plt.ylim([0,1.05])

    if save==True: plt.savefig(saveDir+'4_04_Si(111)E.png',dpi=200)
    if show: plt.show()
    plt.clf()

def Fig_4_05(show=False,save=True):
    """Effects of varying Miller planes on the Darwin curve for Ge at 8.05 keV"""
    axnames=[]
    # Ge at different Miller indices
    minds=[111,200,220,113,222,400]
    a=5.6461;wavelength=1.5405e-10;
    for mind in minds:
        [wavelength,E,Q,angle,[h,k,l]]=ff.__parseParameters(a,mind,wavelength)
        [thetas,rs]=dwn.RockingCurve('Si','Si',a,mind,wavelength,E,'thetas',absorption=True,
            normalShift=False)
        axi,=plt.plot(thetas,rs)
        axnames.append([axi,str(mind)])
    plt.title('Darwin curves of Ge at different Miller indices (E=%.2f keV).'%(E/1000.))
    plt.xlabel(r'$\omega$ (millidegrees)')
    plt.ylabel(r'Intensity reflectivity')
    plt.legend(*zip(*axnames))
    plt.ylim([0,1.05])

    if save==True: plt.savefig(saveDir+'4_05_GeIndex.png',dpi=200)
    if show: plt.show()
    plt.clf()

def Fig_4_06(show=False,save=True):
    """This is the effect of different zinc blende materials at Miller index 111. These have not
        been centered to show the relative offsets (different Bragg angles) due to the variant d-
        spacings."""
    axnames=[]
    # Si, Ge, GaAs, InSb, diamond on same graph
    ases={'Si':5.4309,'Ge':5.6461,'Ga':5.6535,'In':6.48,'C':3.5668}
    eles=[['Si','Si'],['Ge','Ge'],['Ga','As'],['In','Sb'],['C','C']]
    wavelength=1.5405e-10;hkl=111
    for ele1,ele2 in eles:
        [wavelength,E,Q,angle,[h,k,l]]=ff.__parseParameters(ases[ele1],hkl,wavelength)
        [thetas,rs]=dwn.RockingCurve('Si','Si',ases[ele1],hkl,wavelength,E,'thetas',absorption=
            True,normalShift=False)
        axi,=plt.plot(thetas,rs)
        ele12=ele1 if ele1==ele2 else ele1+ele2
        axnames.append([axi,ele12])
    plt.title('Darwin curves of different silicon blende materials, (111).')
    plt.xlabel(r'$\omega$ (millidegrees)')
    plt.ylabel(r'Intensity reflectivity')
    plt.legend(*zip(*axnames))
    plt.ylim([0,1.05])

    if save==True: plt.savefig(saveDir+'4_06_Eles(111).png',dpi=200)
    if show: plt.show()
    plt.clf()

def Fig_4_07(show=False,save=True):
    """Darwin plots of strained (Bon ) and unstrained (Boff ) rocking curves for Ge(400) at
        decreasing energies (increasing penetration depths), used to model the strain profile.
        Fits are Lorentzian."""

    AnalyzeFrame(fitL=True,save=save,show=show,scanNums=[1,124,232])

def Fig_4_08(show=False,save=True):
    """Graph of relative difference between strained and unstrained lattice spacing against
        energy (top) and extinction depth of energies (bottom). Note the abberant behavior near
        the K-edge of 11.1 keV for Ge (top), and the nonlinearity about it in the bottom graph.
        """

    f,(ax1,ax2)=plt.subplots(2,sharex=True)

    Es=arange(11020,11500,.1)
    Ds=[dwn.extDepth('Ge','Ge',5.6461,400,theta=0,E=E,source='NIST') for E in Es]
    ax1.plot([E/1000. for E in Es],[D*1.e6 for D in Ds],'b-')

    nistPts=ff.getInterpPoints(source='NIST',ele='Ge')
    Ds2=[dwn.extDepth('Ge','Ge',5.6461,400,theta=0,E=1000.*E,source='NIST') for E in nistPts]
    ax1.plot([E for E in nistPts],[Ds2s*1.e6 for Ds2s in Ds2],'bo')
    ax1.set_ylabel(r'$\Lambda_{ext}$ ($\mu$m)')
    ax1.set_ylim([.75,1.02])

    Ge=mda.MDA('7idd_0046.mda','all',['7cart:m3.VAL','S16','S15'])
    ds=[];Es=[]
    for scan in Ge.scans:
        scan.finddd2('7cart:m3.VAL','S16','S15',scan.Y,5.6461,400)
        ds.append(scan.dd)
        Es.append(scan.Y)
    ax2.plot(Es,ds,'-')

    ax1.set_title('Extinction Depth and Strain, via Energy')
    plt.xlabel(r'Energy (keV)')
    plt.ylabel(r'$\frac{\Delta d}{d}$ (unitless)')
    plt.xlim([11.02,11.5])

    if save: plt.savefig(saveDir+'4_08_extDepthStrain.png',dpi=200)
    if show: plt.show()
    plt.clf()

def Fig_4_09(show=False,save=True):
    """Cumulative average lattice spacing difference ratio, this time as a function of penetraton
         depth."""

    Ge=mda.MDA('7idd_0046.mda','all',['7cart:m3.VAL','S16','S15'])
    ds=[];Es=[]
    for scan in Ge.scans:
        scan.finddd2('7cart:m3.VAL','S16','S15',scan.Y,5.6461,400)
        ds.append(scan.dd)
        Es.append(scan.Y)
    dEs=[[d,E] for (d,E) in zip(ds,Es) if E>11.11]
    [ds,Es]=zip(*dEs)
    Ds=[]                                    # ds is the strain, Ds is the depth
    for E in Es:
        Ds.append(dwn.extDepth('Ge','Ge',5.6461,400,theta=0,E=1000.*E,source='NIST'))
    plt.plot([D*1.e6 for D in Ds],ds)
    plt.title('Strain vs. Extinction Depth')
    plt.xlabel(r'$\Lambda$ ($\mu$m)')
    plt.ylabel(r'$\frac{\Delta d}{d}$',rotation=0)

    if save: plt.savefig(saveDir+'08_Ed.png')
    if show: plt.show()
    plt.clf()
```

```
def createGifFromMDA():
    """Creates gif from MDA."""
    AnalyzeFrame(save=True,show=False,errbars=True,dtxt=True,scanNums='all')
    os.system('convert -delay 20 -loop 0 {0}Ge/Ge_*.png {0}Ge.gif'.format(saveDir))

def AnalyzeFrame(fitL=True,x0hFile=None,dtxt=False,save=True,show=False,errbars=True,scanNums
    =[1,124,232]):
    Ge=mda.MDA('7idd_0046.mda','all',['7cart:m3.VAL','S16','S15'])
    scanIter=Ge.scans if scanNums=='all' else [Ge.scans[q-1] for q in scanNums]
    for scan in scanIter:
        scanRaw='7cart:m3.VAL'+('' if fitL else 'raw') # unshifted angles for raw data plot
        [x,y1,y2]=[scan.data[scanRaw],scan.data['S16'],scan.data['S15']]
        [ERR1,ERR2]=[[sqrt(10*qq) for qq in scan.data[q]] for q in ['counts1','counts2']]

        if fitL:
            [p1,p2]=[scan.fitLorentzian(x,q) for q in [y1,y2]]
            xxs=arange(min(x),max(x),(max(x)-min(x))/1000.)
            axL1,=plt.plot([1000*xx for xx in xxs],[scan.Lorentzian(p1[0],xx) for xx in xxs],'b-'
                )
            axL2,=plt.plot([1000*xx for xx in xxs],[scan.Lorentzian(p2[0],xx) for xx in xxs],'r-'
                )
        if x0hFile!=None:
            ax0h,=plt.plot(*dwn.getX0hData(x0hFile),linestyle='-',color='k')
        x=[1000*xx for xx in x]
        ax1=plt.errorbar(x,y1,yerr=ERR1,fmt='bo')
        ax2=plt.errorbar(x,y2,yerr=ERR2,fmt='ro')
        cheatfmt=['ro-','bo-'] if fitL==True else ['ro','bo']

        plt.xlabel(r'$\theta$ (millidegrees)')
        plt.ylabel('Counts (per 10 s)')
        plt.title('Ge E=%.4f keV (scan #%d)'%(scan.Y,scan.scanNum))

        if dtxt:
            scan.finddd2('7cart:m3.VAL','S16','S15',scan.Y,5.6461,400)
            plt.text(-13,.9,'d=%7.3f'%scan.d,ha='left',va='center')

        lgdnames=['Laser Off']
        lgdlist=[ax1]
        if fitL:
            lgdlist.append(axL1)
            lgdnames.append('Lorentzian Fit (Off)')
        lgdnames.append('Laser Off')
        lgdlist.append(ax2)
        if fitL:
            lgdlist.append(axL2)
            lgdnames.append('Lorentzian Fit (On)')
        if x0hFile!=None:
            lgdlist.append(ax0h)
            lgdnames.append('X0h data')
        plt.legend([ax2,ax1],['Laser On','Laser Off'])
        saveText=('Ge/Ge_%03d.png' if scanNums=='all' else ('4_07_Ge_%03d.png' if fitL else '3
            _03_Ge_Raw_%03d'))
        if save: plt.savefig(saveDir+saveText%scan.scanNum,dpi=200)
        if show: plt.show()
        plt.clf()


if __name__=='__main__':
    main()
```

B.2 Darwin.py—Darwin functions for diamond and zinc-blende structures

```
#!/usr/bin/python2
import FormFactor.formfactor as ff
import re,os,sys
from cmath import *
import matplotlib.pyplot as plt
from numpy import arange

# Physical Constants
PlanckH=4.135667516E-15;                # eVs Planck's in eV s (NIST/hev)
c=299792458;                            # m/s Speed of light (NIST/c)
r0=2.8179403267E-5;                     # \AA Electron radius (NIST/re)

def r(xc):
    if xc.real>=1:
        ret=xc-sqrt(xc**2-1)
    elif xc.real<=-1:
        ret=xc+sqrt(xc**2-1)
    else:
        ret=xc-1j*sqrt(1-xc**2)
    return ret

def RockingCurve(ele1,ele2,a,hkl,wavelength=None,E=None,xaxis='thetas',source='NIST',absorption=
    True,normalShift=False,xwidth=5):
    """Return x and y vectors of a rocking curve:
    Inputs:
        ele1,ele2:      ---   Element names
        a:              Ang   Lattice spacing
        hkl:            ---   Miller index as three-digit integer hkl
                              or three-element list [h,k,l].  REQUIRED
        wavelength:     m     Wavelength of incident beam
        E:              eV    Energy of incident beam (overrides wavelength)
        xaxis:          ---   What to return as x-axis.
                              Options 'x','thetas','zetas','xc'.
      Optional inputs follow..
        absorption:     ?     Include absorption effects?
        source:         ---   Where do we get form factors?
                              Options 'CTs' 'NIST'
        normalShift:    ?     Should the curve be centered?
        xwidth:         ---   How many g/pi zetas should we plot?

    Output:
        [xaxis,yaxis]"""

    # Calculate unpassed parameters
    [wavelength,E,Q,angle,[h,k,l]]=ff.__parseParameters(a,hkl,wavelength,E)
    d=wavelength*1.e10/(2*sin(angle)) if Q!=0 else 0
    vc=a**3

    [F0,F]=[ff.Fuc(ele1,ele2,a,hkli,wavelength,source=source) for hkli in [0,hkl]]
    if not absorption: [F,F0]=[abs(F),abs(F0)]
    [g0,g]=[2*d**2*r0/vc*FF for FF in [F0,F]]

    x=arange(-xwidth,xwidth,.01)

    zetas=[(g0/pi+xs*g/pi).real for xs in x]
    thetas=[((z-g0/pi*normalShift)*tan(angle)*180000./pi).real for z in zetas]
    xc=[pi*z/g-g0/g for z in zetas]
    rs=[abs(r(x))**2 for x in xc]

    if xaxis=='thetas':
        xax=thetas
    elif xaxis=='xc':
        xax=xc
    elif xaxis=='zetas':
        xax=zetas
    elif xaxis=='x':
        xax=x
    else:
        print 'Unrecognized x axis '+str(xaxis)+'.  See help(RockingCurve) for options.'
        xax=None
    return [xax,rs]


def plotRockingCurve(ele1,ele2,a,hkl,wavelength=None,E=None,xaxis='thetas',source='NIST',
    absorption=True,normalShift=False,xwidth=5,X0hComp=None,save=False,saveAs=None,latexOut=False
    ,latexOutput=None,show=False,plotTitleEnergy=False,plotBragg=False,plotTitle=None,noabsorbbg=
    False):
    """Plot a rocking curve:
    Inputs:
        ele1,ele2:      ---   Element names
        a:              \AA   Lattice spacing
        hkl:            ---   Miller index as three-digit integer hkl
                              or three-element list [h,k,l].  REQUIRED
        wavelength:     m     Wavelength of incident beam
        E:              eV    Energy of incident beam (overrides wavelength)
      Optional inputs follow..
        xaxis:          ---   What to plot as x-axis.
                              Options 'x','thetas','zetas','xc'.
        source:         ---   Where do we get form factors?
                              Options 'CTs' 'NIST'
        absorption:     ?     Include absorption effects?
        normalShift:    ?     Should the curve be centered?
        xwidth:         ---   How many g/pi zetas should we plot?
        X0hComp:        path  Superimpose X0h data by including .dat path here
                              (requires xaxis=='thetas')
        save:           ?     Save figure?
        saveAs:         path  Optional figure name for saving
        latexOut:       ?     Output data in latex form?
        latexOutput:    path  Set to a path to output all data in latex form.
        show:           ?     Show the graph?
        plotTitleEnergy:?     In plot title, include energy instead of lambda?
        plotBragg:       ?     Plot the bragg angle?
                              (requires xaxis=='thetas')
        plotTitle:      str   (optional) plot title
        noabsorbbg:     ?     Plot no absorption in the background?
                              (meaningless unless absorption==True)

    Output:
        None"""

    if xaxis!='thetas':
        X0hComp=None
        plotBrag=False
    if absorption==False:
        noabsorbbg=False

    [wavelength,E,Q,angle,[h,k,l]]=ff.__parseParameters(a,hkl,wavelength,E)
    [thetas,rs]=RockingCurve(ele1,ele2,a,hkl,wavelength,E,xaxis,source,absorption,normalShift,
        xwidth)
    if noabsorbbg:
        [nothetas,nors]=RockingCurve(ele1,ele2,a,hkl,wavelength,E,xaxis,source,False,normalShift,
            xwidth)

    if noabsorbbg:
        ax4,=plt.plot(nothetas,nors,'k-')
    ax1,=plt.plot(thetas,rs,'b-')
    if X0hComp!=None:
        ax2,=plt.plot(*getX0hData(X0hComp),linestyle='--',color='r')
    if plotBragg:
        d=a/sqrt(h*h+k*k+l*l)
        braggAngle=asin(wavelength/d)
        ax3,=plt.plot([braggAngle]*2,[0,1],'r-')

    if latexOut:
        print writeLatexData(ele1,ele2,a,[h,k,l],wavelength,filename='LatexOut')

    if plotTitle==None:
        nowith='with' if absorption else 'no'
        ele12=ele1 if ele1==ele2 else ele1+ele2
        titleEnergy=r'$E$=%.2f keV'%(E/1000.) if plotTitleEnergy else r'$\lambda$=%.2f $\AA$'%(
            wavelength*1e10)
        plotTitle=ele12+'('+''.join([str(s) for s in [h,k,l]])+r')$\hat\sigma$, '+titleEnergy+(''
             if noabsorbbg else ', '+nowith+' absorption')
    plt.title(plotTitle)
    plt.ylabel(r'Intensity reflectivity')
    xlabel=r'$\omega$ (millidegrees)'
    if xaxis=='xc':
        xlabel=r'$x$'
    elif xaxis=='x':            # should not be plotting x
        xlabel=r'Arb.'
    elif xaxis=='zetas':
        xlabel=r'$\zeta$'
    plt.xlabel(xlabel)


    if noabsorbbg or X0hComp!=None or plotBragg: # no axes if only one plot
        axLegend=[[ax1],[r'Computed']]
        if X0hComp!=None:
            axLegend[0].append(ax2)
            axLegend[1].append(r'$\chi_{0,H}$')
        if plotBragg:
            axLegend[0].append(ax3)
            axLegend[1].append(r'Bragg angle')
        if noabsorbbg:
            axLegend[0].append(ax4)
            axLegend[1].append(r'No Absorption')

        plt.legend(*axLegend)

    plt.ylim([0,1.05])

    saveDir='/'.join(os.path.dirname(os.path.realpath(__file__)).split('/')[:-3])+'/Images/'

    if save:
        if saveAs!=None:
            plt.savefig(saveDir+saveAs,dpi=200)
        else:
            plt.savefig(saveDir+ele12+'('+''.join([str(s) for s in [h,k,l]])+')'+('2' if
                noabsorbbg else ('1' if absorption else '0'))+'.png',dpi=200)
    if show: plt.show()
    plt.clf()

def extDepth(ele1,ele2,a,hkl,theta,wavelength=None,E=None,source='NIST'):
    """Returns penetration depth; theta is the relative Braggish angle in millidegrees"""
    global r0

    [wavelength,E,Q,angle,[h,k,l]]=ff.__parseParameters(a,hkl,wavelength,E)
    d=wavelength*1.e10/(2*sin(angle)) if Q!=0 else 0
    vc=a**3

    [F0,F]=[ff.Fuc(ele1,ele2,a,hkli,wavelength,source=source) for hkli in [0,hkl]]
    [g0,g]=[2*d**2*r0/vc*FF for FF in [F0,F]]

    x=(pi/(180000.*tan(angle))*theta-g0/pi)*pi/g
    x=0.

    eta=g*sqrt(1-x**2)
    extDepth=d.real*1e-10/(2*eta.real)    # in m

    return extDepth

def __writeLatexData(ele1,ele2,a,hkl,wavelength=None,E=None,filename=None):
    """Describe several parameters in the form of a specific LaTeX table.
    Inputs:
        ele1,ele2:      ---   Element names
        a:              \AA   Lattice spacing
        wavelength:     m     Wavelength of incident beam
        type:           ---   Always 'zinc blende' for now
        E:              eV    Energy of incident beam (overrides wavelength)
        hkl:            ---   Miller index as three-digit integer hkl
                              or three-element list [h,k,l]
        filename:       path  If specified, write to file [filename] (in working dir)

    Output:
        Data in specific LaTeX format, if filename=None."""

    r0=2.8179403267E-5;                     # \AA Electron radius (NIST/re)

    [wavelength,E,Q,angle,[h,k,l]]=ff.__parseParameters(a,hkl,wavelength,E)
    d=a/sqrt(h*h+k*k+l*l)
    vc=a**3
    theta=asin(wavelength*1e10/(2*d))

    f00=ff.f0(ele1,0)
    fp0=ff.fp(ele1,wavelength=wavelength).real
    fpp0=ff.fp(ele1,wavelength=wavelength).imag
    f0Q=ff.f0(ele1,Q)
    fpQ=ff.fp(ele1,wavelength=wavelength).real
    fppQ=ff.fp(ele1,wavelength=wavelength).imag
    Fat0=ff.F(ele1,a,0,wavelength)
    FatQ=ff.F(ele1,a,[h,k,l],wavelength)

    ele12=ele1 if ele1==ele2 else ele1+ele2

    F0=ff.Fuc(ele1,ele2,a,0,wavelength)
    F=ff.Fuc(ele1,ele2,a,[h,k,l],wavelength)
    g0=2*d**2*r0/vc*F0
    g=2*d**2*r0/vc*F
    zeta0=g0/pi
    x0=pi*zeta0/g
    theta0=(zeta0*tan(theta)*180000./pi).real
    rr0=r(x0)

    text=r"""\begin{table}[htpb]\centering
    \begin{tabular}{|*{6}{>{\centering\arraybackslash}m{2.1cm}|}}
        \hline
        %(hkl)s & $\lambda$=%(wavelength).2f \AA & $E=$%(E).2f keV & $d=$%(d).2f \AA & $v_c$=%(vc
            ).1f \AA$^3$ & $\theta$=%(theta).2f$\deg$\\ \hline
        $f^0( 0)$=%(f00).2f & $f'( 0)=$%(fp0).2f & $f''( 0)=$%(fpp0).2f & $f^0( Q)=$%(f0Q).2f &
            $f'( Q)=$%(fpQ).2f & $f''( Q)=$%(fppQ).2f \\ \hline
        \multicolumn{2}{|c|}{$F_{\mbox{\small at}}( 0)$=%(Fat0)s} & $\abs{F_{\mbox{\tiny at}}( 0)
            }$=%(Fat0a).1f & $\abs{F_{\mbox{\tiny at}}( Q)}$=%(FatQa).1f & \multicolumn{2}{c|}{
            $F_{\mbox{\small at}}( Q)$=%(FatQ)s}  \\ \hline
        \multicolumn{2}{|c|}{$F_0$=%(F0)s} & $\abs{F_0}$=%(F0a).1f & $\abs{F( Q)}$=%(FQa).1f & \
            multicolumn{2}{c|}{$F( Q)$=%(FQ)s} \\ \hline
        \multicolumn{2}{|c|}{$g_0$=%(g0)s} & $\abs{g_0}$=%(g0a).6f & $\abs g$=%(ga).6f & \
            multicolumn{2}{c|}{$g$=%(g)s} \\ \hline
        \multicolumn{2}{|c|}{$\zeta_0$=%(z0)s \AA} &  \multicolumn{2}{c|}{$x_0$=%(x0).2f} & \
            multicolumn{2}{c|}{$\theta_0$=%(theta0).2f m$\deg$} \\ \hline
        \multicolumn{3}{|c|}{$r_0=$%(rr0)s} & \multicolumn{3}{c|}{$\abs{r_0}^2$=%(rr0a2).3f} \\ \
            hline
    \end{tabular}
    \caption{ %(caption)s }
    \label{tbl:%(label)s}
\end{table}"""%{'hkl':''.join([str(s) for s in [h,k,l]]),'wavelength':wavelength*1e10,
                'E':E/1000.,'d':d.real,'vc':vc,'theta':theta.real*180./pi,
         'f00':f00.real,'fp0':fp0,'fpp0':fpp0,'f0Q':f0Q.real,'fpQ':fpQ,'fppQ':fppQ,
         'Fat0':writeRI(Fat0),'Fat0a':abs(Fat0),'FatQa':abs(FatQ),'FatQ':writeRI(FatQ),
         'F0':writeRI(F0),'F0a':abs(F0),'FQa':abs(F),'FQ':writeRI(F),
         'g0':writeRI(g0,'6'),'g0a':abs(g0),'ga':abs(g),'g':writeRI(g,'6'),
         'z0':writeRI(zeta0,'6'),'x0':x0.real,'theta0':theta0.real,
         'rr0':writeRI(rr0),'rr0a2':abs(rr0)**2,
         'caption':ele12+'('+''.join([str(s) for s in [h,k,l]])+r')$\ha\sigma$ data.',
         'label':ele12+'data'}

    if filename!=None:
        with open(filename+'.txt','wa') as lf:
            lf.write(text)
    else:
        return text

def writeRI(comp,prec='2'):
    """Return formatted string of complex float"""
    return ("%."+prec+"f%+."+prec+"fi")%(comp.real,comp.imag)

def getX0hData(filename,plot=False):
    ledata=[]
    X0hPath=os.path.normpath(os.path.join(os.getcwd(),'../Data/X0h'))+'/'
    with open(X0hPath+filename+'.dat','r') as data:
        for line in data:
            ledata.append([float(s) for s in re.split('\s+',line)[1:3]])
    ledata=list(zip(*ledata))
    ledata[0]=[dat/3.6 for dat in ledata[0]]
    if plot:
        ax,=plt.plot(*zip(*ledata))
        plt.grid()
        plt.yscale('log')
        plt.show()
    return ledata
```

B.3 formfactor.py—Form factor module

```
#!/usr/bin/python2

import re,os,sys,json
import matplotlib as mpl
import matplotlib.pyplot as plt
from time import time
from numpy import arange,matrix,linalg
from cmath import *
from scipy import interpolate

# Define physical constants
PlanckH=4.135667516E—15;                # eVs Planck's in eV s (NIST/hev)
c=299792458;                            # m/s Speed of light (NIST/c)
r0=2.8179403267E—5;                     # \AA Electron radius (NIST/re)

# import Crystallographic Table data (form factors and correction coefficients):
dataPath=os.path.normpath(os.path.join(os.getcwd(),'../Data/coefs/'))+'/'
with open(dataPath+'CTs/scatcoef.json','r') as f:
    scatcoef=json.loads(f.read())
with open(dataPath+'CTs/corrfax.json','r') as f:
    corrfax=json.loads(f.read())
with open(dataPath+'NIST/factors.json','r') as f:
    NISTfax=json.loads(f.read())
NISTintrp={}

def init(eles):
    """Initialize interpolation(s) for element for faster processing."""
    global NISTintrp
    for ele in eles:
        NISTintrp[ele]= \
            [interpolate.interp1d(NISTfax[ele]['E'],NISTfax[ele]['f1'],'linear'),
             interpolate.interp1d(NISTfax[ele]['E'],NISTfax[ele]['f2'],'linear')]

def getInterpPoints(source='NIST',ele=None):
    if source=='NIST':
        return NISTfax[ele]['E']
    elif source=='CTs':
        return [q*1.e—10 for q in [2.74851,2.28962,1.93597,1.788965,1.54052,
                0.70926,0.55936,0.215947,0.20901,0.180195]]

def f0(ele,Q):
    """Atomic form factor without absorption or reductions.  Uses CTs.
    Inputs:
        ele: ———  Element name
        Q:   1/m  Magnitude of difference vector

    Output:
        The atomic form factor of the given element at the given index or angle
            * Returns 0 on error."""

    ele = ele=='Si' and 'Siv' or ele     # Based on Crystallographic Tables
    return sum([scatcoef[ele][2*i]*exp(—scatcoef[ele][2*i+1]*(Q/(4*pi))**2) for i in range(4)])+
        scatcoef[ele][8]

def fp(ele,E=None,wavelength=None,source='NIST'):
    """Return constant reduction factors:
    Inputs:
        ele:        ——— Element name
        wavelength: m   Wavelength of incident ray —or—
        E:          eV  Energy of incident ray
        source:     ——— Source, 'CTs' or 'NIST'*
                        *Note that the NIST f' is actually f'+Z!
    Outputs:
        f'—if'': Correction factor to atomic form factor,
                 taken from Crystallographic Tables."""
    PlanckH=4.135667516E-15;                # eVs Planck's in eV s (NIST/hev)
    c=299792458;                            # m/s Speed of light (NIST/c)

    if source=='CTs':
        if E!=None:
            wavelength=PlanckH*c/E
        #print wavelength
        wavelengths=[a*1e-10 for a in [2.74851,2.28962,1.93597,1.788965,1.54052,
                                       0.70926,0.55936,0.215947,0.20901,0.180195]]
        intrp=[interpolate.interp1d(wavelengths[::-1],corrfax[ele][0][::-1]),
               interpolate.interp1d(wavelengths[::-1],corrfax[ele][1][::-1])]
        return intrp[0](wavelength)-1j*intrp[1](wavelength)
    elif source=='NIST':
        if E==None:
            E=PlanckH*c/wavelength
        energies=NISTfax[ele]['E']
        return NISTintrp[ele][0](E/1000.)-f0(ele,0)- \
               1j*NISTintrp[ele][1](E/1000.)
    else:
        print 'Unrecognized source '+source+' in F.'
        return

def F(ele,a,hkl,wavelength=None,E=None,source='NIST'):
    """Total atomic form factor.
    Inputs:
        ele:        --- Element name
        a:          \AA Lattice parameter
        wavelength: m   Wavelength of incident xray
        hkl:            Miller index as three-digit integer hkl
                        or three-element list [h,k,l]
        E:          eV  Energy (overrides wavelength)
        source:     --- Do we get f,f',f'' from NIST or CTs?"""

    [wavelength,E,Q,angle,[h,k,l]]=__parseParameters(a,hkl,wavelength,E)

    return f0(ele,Q)+fp(ele,E,wavelength,source)

def Fuc(ele1,ele2,a,hkl,wavelength=None,E=None,source='NIST'):
    """Unit cell form factor.
    Inputs:
        ele1:       --- First element name
        ele2:       --- Second element name
        a:          \AA Lattice parameter
        wavelength: m   Wavelength of incident xray
        hkl:            Miller index as three-digit integer hkl
                        or three-element list [h,k,l]
        E:          eV  Energy (overrides wavelength)
    Output:
        The unit cell form factor of the given element at the given index or angle"""

    [wavelength,E,Q,angle,[h,k,l]]=__parseParameters(a,hkl,wavelength,E)

    # The conjugate of everything EXCEPT fs are taken:
    [fa,fb]=[F(ele,a,hkl,wavelength,source=source) for ele in [ele1,ele2]]
    latticeSum=(1+exp(1j*pi*(h+k))+exp(1j*pi*(k+l))+exp(1j*pi*(h+l)))
    fafb=fa+fb*exp(1j*pi/2.*(h+k+l))
    fafbhalfconj=fa+fb*exp(-1j*pi/2.*(h+k+l))
    return sqrt(fafb*fafbhalfconj)*latticeSum

def rFuc(ele1,ele2,a,hkl,wavelength=None,E=None,source='NIST'):
    """Same as Fuc, without the funky step---used to calculate extinction depth as Als-Nielsen
        does."""

    [wavelength,E,Q,angle,[h,k,l]]=__parseParameters(a,hkl,wavelength,E)

    [fa,fb]=[F(ele,a,hkl,wavelength,source=source) for ele in [ele1,ele2]]
    latticeSum=(1+exp(1j*pi*(h+k))+exp(1j*pi*(k+l))+exp(1j*pi*(h+l)))
    return (fa+fb*exp(1j*pi/2.*(h+k+l)))*latticeSum

def printOutfpfpp(eles):
    """To be exported into LaTeX."""

    global scatcoef,corrfax
    print '\t'+('\t%.3f'*10)%tuple( \
                    [2.74851,2.28962,1.93597,1.788965,1.54052,
                     0.70926,0.55936,0.215947,0.20901,0.180195])
    for ele in eles:
        print ele+"\tf'"+('\t%.3f'*10)%tuple(corrfax[ele][0])+'\n'+ \
              "\tf''"+('\t%.3f'*10)%tuple(corrfax[ele][1])

    print '\n'*3
    for ele in eles:
        ele = ele=='Si' and 'Siv' or ele     # Based on Crystallographic Tables
        print '\t'.join([ele]+[str(j) for j in scatcoef[ele]])

def printOutNISTfpfpp(eles,indepVar='Energy'):
    """Same as above, for NIST data."""

    Es=[.01069,.8739896,6.915365,22.98,39.19543,51.18542,139.2553,181.8539,271.3869,405.0001]
    wls=[PlanckH*c/E for E in Es]
    table={'Es':Es,'wls':wls}
    for ele in eles:
        fpEs=[[float(q.real),float(q.imag)] for q in [fp(ele,E=E*1000,source='NIST') for E in Es
            ]]
        table[ele]=zip(*fpEs)
    tableText=r"""
\begin{table}[htbp]
\begin{center}
\begin{tabular}{|l|l|r|r|r|r|r|r|r|r|r|r|}
\hline
\multicolumn{ 12}{|c|}{Selected NIST Forward-Scattering Dispersion Corrections to Form Factors}
    \\ \hline\multicolumn{2}{|c|}{"""+(r"$E$ (keV)" if indepVar=='Energy' else r"$\lambda$ (\AA)"
    )+'} '+('& {:.2f} '*len(Es)).format(*(table['Es'] if indepVar=='Energy' else table['wls']))+r
    """ \\ \hline"""
    for ele in eles:
        tableText+=r"""
\multirow{2}{*}{"""+ele+"""} & f' """+('& {:.3f} '*10).format(*table[ele][0])+r""" \\ \cline
    {2-12}
 & f''  """+('& {:.3f} '*10).format(*table[ele][1])+r""" \\ \hline"""
    tableText+=r"""
\end{tabular}
\end{center}
\caption{Dispersion corrections for forward scattering, provided by NIST.\cite{NISTcff}  Data is
    more abundant than the Crystallographic Tables near the band edges, making this the optimal
    data source for our calculations.  Note that NIST provides $f'+f^0$ and $f''$, and the atomic
     form factor $f^0$ was subtracted from the latter in compiling this table.}
\label{tbl:NISTcff}
\end{table}"""
    print tableText

def __parseParameters(a,hkl,wavelength=None,E=None):
    """Calculates wavelength from energy, angle from Miller index, returns them all.
    Inputs:
        a:          \AA Lattice parameter
        wavelength: m   Wavelength of incident xray
        hkl:            Miller index as three-digit integer hkl
                        or three-element list [h,k,l]
        E:          eV  Energy (overrides wavelength)
    Output:
        As a single list, [wavelength,E,Q,angle,[h,k,l]]."""

    PlanckH=4.135667516E-15;              # eVs Planck's in eV s (NIST/hev)
    c=299792458;                          # m/s Speed of light (NIST/c)
```

```
if E!=None:                           # Override wavelength
    wavelength=PlanckH*c/E
elif wavelength!=None:
    E=PlanckH*c/wavelength
else:
    print 'parseParameters received neither wavelength nor energy.'
    return

if isinstance(hkl,list):          # [h,k,l]
    [h,k,l]=hkl
else:
    [h,k,l]=[int(q) for q in '%03d'%hkl]
if h==k and k==l and l==0:
    Q=0
else:
    d=a/sqrt(h*h+k*k+l*l)
    Q=2*pi/d
    #print 'pp',d.real,a

angle=asin(Q*wavelength*1.e10/(4*pi))

return [wavelength.real,E.real,Q.real,angle.real,[h,k,l]]
```

B.4 Spec.py—Module for parsing data from spec files, with heavy contributions from Appendix A.7

```
#!/usr/bin/python2

import re,os,sys
import matplotlib as mpl
import matplotlib.pyplot as plt

class Scan():
    """Scan class: represents a single scan.
        Argument 'data' is a dictionary of both scalars (like T) and lists (like 'Cbon')"""

    def __init__(self,scanNum,scanTime,data):
        self.scanNum=scanNum
        self.scanTime=scanTime
        self.data=data

    def printValues(what):
        """Prints to stdout the values of Scan[argument]"""
        if not what in self.data:
            print 'No such argument.  Arguments include: %s.'%', '.join([s for s in self.data])
            return
        for s in self.data:
            print '%s: '%s,self.data[s];
        for i in range(10):
            print ''


class Spec:
    """Spec class: Represents all of the data in a single spec file as a list of Scans."""
    specPath=os.path.normpath(os.path.join(os.getcwd(),'../../Data/Spec/'))+'/'

    def __init__(self,filename,scannums,filedir=specPath):
        self.filename=filename
        self.scannums=scannums
        self.filedir=filedir
        self.scans=[]
        self.availScans=[]     # Array of available scans in spec file
        self.readSpecFile()

    def readSpecFile(self):
        # Open data and scan in only what we need to conserve memory

        with open(''.join([self.filedir,self.filename]),'rb') as f:
            x=1
            unwantedScan=True
            scanNumbers=[]                   # The scans we pick up, in order
            scanTimes=[]
            data=[]
            currentScan=-1
            for line in f:
                #print 'We are on line',x
                if line[0]=='#' or line[0]=='%': # Something we can process
                    if line[1]=='S':            # We've hit a new scan
                        tmpScanNum=int(re.search(r'\s(\d+)\s',line).group(1))
                        self.availScans.append(tmpScanNum)
                        #print 'Found scan #%d.'%tmpScanNum
                        if tmpScanNum in self.scannums:
                            #print 'Found wanted scan',tmpScanNum
                            unwantedScan=False
                            currentScan+=1
                            scanNumbers.append(tmpScanNum)
                        else:
                            unwantedScan=True
                    elif unwantedScan:
                        pass              # Do nothing
                    elif line[1]=='D':
                        #print 'Found date!'
                        pass
                    elif line[1]=='T':
                        tmpTime=float(re.search(r'\s([\d.-]+)\s',line).group(1))
                        #print 'This scan has time',tmpTime
                        scanTimes.append(tmpTime)
                    elif line[1]=='L':
                        #print 'Found values: ',re.findall('\s(\w+)\s',line[2:])
                        data.append([])
                        data[currentScan].append(re.findall('\s(\w+)\s',line[2:]))
                elif not line in ('\n','\r\n') and not unwantedScan:
                    #print re.split('\s+',line.strip())
                    data[currentScan].append([float(s) for s in re.split('\s+',line.strip())])
                else:
                    # print 'line',x,'is empty!'
                    pass
                x+=1

        # Error checking
        if len(scanNumbers) != len(self.scannums):
            for j in self.scannums:
                if not j in scanNumbers:
                    print 'Alert: did not find scan #%d!'%j

        realLen=len(data)
        if len(scanTimes)!=realLen:
            print 'Alert: Scan time missed!'
        if len(scanNumbers)!=realLen:
            print 'Alert: Scan numbers do not match!'


        # Makes Scan objects out of data
        for j in range(len(data)):
                data[j]=zip(*data[j]) # Simply transpose data matrix of jth scan
                q=dict()
                for i in range(len(data[j])):     # Assign to each column its list of values
                    q[str(data[j][i][0])]=data[j][i][1:]
                self.scans.append(Scan(scanNumbers[j],scanTimes[j],q))
```

B.5 MDA.py—Module for parsing MDA data

```
#!/usr/bin/python2

import re,os,sys,time
import matplotlib.pyplot as plt
from cmath import *
from string import strip
from numpy import arange
from scipy import interpolate,optimize
from FormFactor.formfactor import __parseParameters as pp
import Darwin.Darwin as dwn

class Scan():
    """Scan class: Represents single imported MDA scan"""

    def __init__(self,scanNum,scanTime,Y,data):
        self.scanNum=scanNum
        self.scanTime=scanTime
        self.Y=Y
        self.data=data
        self.data['counts1']=self.data['S16']
        self.data['counts2']=self.data['S15']
        self.dtheta=self.normalizeCurvesLorentz('7cart:m3.VAL','S16','S15')

    def normalizeCurvesLorentz(self,x,y1,y2):
        """Normalize the position (not width) of both curves
        according to a Lorentzian fit to y1.  Scales self.data[x].
        Returns distance between centroids in units of x.
        Inputs:
          x:   independent variable, common to y1 and y2
          y1:  guiding (more constant) dependent variable
          y2:  other dependent variable
        Outputs:
          d:   Distance between centroids in units of x"""
        [sx,sy1,sy2]=[self.data[x],self.data[y1],self.data[y2]]
        [hwhm,x0,h]=self.fitLorentzian(sx,sy1)[0]
        self.data[x+'raw']=self.data[x] # preserve raw data
        self.data[x]=[xx-x0 for xx in sx]
        #self.data[y1]=[yy/h for yy in sy1]
        #self.data[y2]=[yy/h for yy in sy2]
        [hwhm,x0,h]=self.fitLorentzian(self.data[x],self.data[y2])[0]
        return -x0*1000

    def fitLorentzian(self,x,y):
        """Fits curve on, returns Lorentzian fit factors.
        Inputs:
            x,y:   Independent and dependent variable.
        Outputs:
            p:     List of fit params HWHM, x-displacement, peak height."""
        return optimize.leastsq(self.__errorL,self.__estimates(x,y),args=(x,y))

    def Lorentzian(self,p,x):
        """Returns a Lorentzian based on parameters p
        and independent variable x.
        Inputs:
            p:    HWHM, x-displacement, peak height
            x:    independent variable
        Outputs:
            Calculated lorentzian (used for leastsq fit)"""
        [hwhm,x0,h]=p
        return h*hwhm**2/((x-x0)**2+hwhm**2)


    def finddd2(self,x,y1,y2,E,a,hkl):
        """Same as finddd, but we assume y1 at the Bragg angle and offset y2.  This accounts for
```

```
            small fluctuations in crystal position"""
        [hwhm2,x02,h2]=self.fitLorentzian(self.data[x],self.data[y2])[0]
        [wavelength,E,Q,angle,[h,k,l]]=pp(a,hkl,E=1000.*E)
        d=a/sqrt(h*h+k*k+l*l)
        x02=x02*pi/180.+angle
        d2=wavelength*1.e10/(2*sin(x02))   # Ang
        self.d,self.dd=[d2.real,((d2-d)/d).real]

    def finddd(self,x,y1,y2,E):
        PlanckH=4.135667516E-15;                # eVs Planck's in eV s (NIST/hev)
        c=299792458;                            # m/s Speed of light (NIST/c)
        wavelength=PlanckH*c/(1000*E)           # don't forget E was in keV
        """Run this before normalizing--takes absolute thetas, then subtracts d."""
        [hwhm1,x01,h1]=self.fitLorentzian(self.data[x],self.data[y1])[0]   # find th1
        [hwhm2,x02,h2]=self.fitLorentzian(self.data[x],self.data[y2])[0]
        [x01,x02]=[q*pi/180. for q in [x01,x02]]
        d1=(wavelength*1.e10/(2*sin(x01))).real
        d2=(wavelength*1.e10/(2*sin(x02))).real

        [wavelength,E,Q,angle,[h,k,l]]=pp(5.6578,400,E=self.Y)
        d=5.6578/sqrt(h*h+k*k+l*l)
        return (d2-d1)/d1

    def findd(self,a,hkl,E):
        [wavelength,E,Q,angle,[h,k,l]]=pp(a,hkl,E=self.Y)
        d=a/sqrt(h*h+k*k+l*l)
        theta0=asin(wavelength*1.e10/(2*d))*180000./pi   # mdeg
        thetanew=(theta0-self.dtheta)*pi/180000.         # rad
        return d-wavelength*1.e10/(2*sin(thetanew))      # Ang

    def __errorL(self,p,x,y):
        return y-self.Lorentzian(p,x)

    def __estimates(self,x,y):
        """Estimates the Lorentzian parameters of dataset (x,y).
        Inputs:
            x:   xaxis list
            y:   yaxis list
        Outputs:
            p:   HWHM, x-displacement, peak height"""

        # Rough but effective
        hwhm=.0005

        # For x0, just find the middle x
        x0=x[len(x)/2]
        return [hwhm,x0,max(y)]

    def normalizeCurvesDarwin(self,x,y1,y2):
        [sx,sy1,sy2]=[self.data[x],self.data[y1],self.data[y2]]
        [scaley,ashift]=self.fitLorentzian(sx,sy1)[0]
        self.data[x]=[xx-ashift for xx in sx]
        self.data[y1]=[yy/scaley for yy in sy1]
        self.data[y2]=[yy/scaley for yy in sy2]
        [scaley,ashift]=self.fitDarwinCurve(self.data[x],self.data[y2])[0]
        return -ashift*1000

class MDA():
    """MDA class: Represents all imported data in a single MDA file as a list of scans."""

    def __init__(self,filename,scannums,scanvars,filedir='../../Data/MDA/',darwinParams=None,
        darwinDyn=['Y','E']):
        self.filename=filedir+filename
        self.scannums=scannums
        self.scanvars=scanvars
        self.scans=[]                     # list of Scan objects, to be populated
        self.availScans=[]                # available scan numbers
        self.darwinParams=darwinParams
        self.darwinDyn=darwinDyn
        self.readMDAFile(scannums,scanvars)

    def readMDAFile(self,scannums,scanvars):
        with open(self.filename,'r') as f:
            currentScan=-1
            x=0
            nowon='header'
            columnTime=False                  # '# Column Descriptions:' tag not unique
            for line in f:
                x=x+1
                if nowon!='scan' and line[0]!='#':
                    continue
                if currentScan==-1 and not 'Scan Divider' in line:
                    continue                  # skip file header information
                if 'Scan Divider' in line:
                    currentScan+=1
                    nowon='header'
                    columns=[]
                    data=[]
                    wantedColumns=[]
                    wantedScan=False
                    continue
                elif 'Column Descriptions:' in line:
                    if not columnTime:
                        columnTime=True
                    else:
                        columnTime=False
                        nowon='column'
                        continue
                elif '1-D Scan Values' in line:
                    nowon='scan'
                    continue
                if nowon=='header':
                    if '# Current point =' in line:
                        scanNum=int(re.search('=\s+(\d+)\s*of\s*\d*\s*$',line).group(1))
                        self.availScans.append(scanNum)
                        if scannums=='all' or scanNum in scannums:
                            wantedScan=True
                    if '# Scan time =' in line:
                        scanTime=re.search('= (.*)$',line).group(1)
                    elif '# 2-D Scan Values:' in line:
                        y=float(re.search(':\s*(\S+).*$',line).group(1))
                elif nowon=='column' and wantedScan==True:
                    m=re.search('#\s+(\d+)\s*\[\s*(.+)\s*\]\s*(.*)$',line)
                    for ss in scanvars:
                        if scanvars=='all' or ss in m.group(3):   # We want this column
                            columns.append(m.groups())
                            wantedColumns.append([int(m.group(1)),ss])
                elif nowon=='scan' and wantedScan==True:
                    if line[0]=='#':
                        continue
                    elif line[0]=='\n':   # Scan finished
                        data=zip(*data)   # swap rows/columns
                        ledata={}
                        for column in wantedColumns:
                            ledata[column[1]]=data[column[0]-1]
                        self.scans.append(Scan(scanNum,scanTime,y,ledata))
                    else:
                        data.append([float(q) for q in re.split('\s*',strip(line))])
```

B.6 Gatherer.py—Scrapes Crystallographic Tables and NIST factor data into JSON [2014]

```python
#!/usr/bin/python
import urllib2,re,string,sys,os,json
from time import sleep
from bs4 import BeautifulSoup as bs

# This file gathers coefficient data and outputs them to the proper folders with the proper
    formats.
# Data parsed by this file includes:
#     Crystallographic Tables data for f', f'': from CTs/CT Tables/ to CTs/
#     NIST form factor data for f', f'': from the NIST websites to NIST/

def load_scatcoef(eles='all'):
    """This function goes through Table 6.1.1.1 of the Crystallographic Tables to find selected
    mean atomic scattering factors.
    Input: eles       List of elements for which to load data
                      All elements loaded if left empty
    Output: {elementName: [a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4 c]}"""
    url='../6_1_1_4_data.html'            # HTML stripped of all but this table
    returndata={}
    with open(url,'r') as data:
        s=data.read().replace("&#8722;",'-')
        s=bs(s).table.find_all('table')[2].tbody.find_all('tr')
        for elem in s:
            ele=re.sub(r'[^A-Za-z0-9\+-]','',elem.td.getText())
            # getText() concatenates all child strings
            #print ele
            if eles=='all' or ele in eles:
                returndata[ele]=[float(r.span.string) for r in elem.find_all('td')[2:11]]
    return returndata

def load_corrfax(eles='all'):
    """This function goes through Table 4.2.6.8 of the Crystallographic Tables to find selected
    correction factors for forward scattering f' and f''.
    Input: eles       List of elements for which to load data
                      All elements loaded if left empty
    Output: {elementName: [[f'],[f'']],...}"""
    url='../4_2_6_8_data.html'            # HTML stripped of all but this table
    returndata={}
    with open(url,'r') as data:
        s=data.read().replace("&#8722;","-")
        s=bs(s).table.find_all('table')[2].tbody.find_all('tr')
        for i in range(len(s)/2):
            ele=s[2*i].td.span.string.strip()           # Element name
            if eles=='all' or ele in eles:
                q1=[float(r.span.string) for r in s[2*i].find_all('td')[2:]]    # f'
                q2=[float(r.span.string) for r in s[2*i+1].find_all('td')[1:]] # f''
                returndata[ele]=[q1,q2]
    return returndata

def createCTJSON(eles='all'):
    """This function creates a file with JSON dump of selected elements.
    Input: eles       List of elements for which to load data
                      All elements loaded if left empty
    Output: None"""
    # Modified for matlab use (no native JSON)
    corrfax=load_corrfax(eles)
    scatcoef=load_scatcoef(eles)

    with open('CTs/corrfax.txt','w') as f:    # For matlab
        for s in corrfax:
            f.write('\t'.join([s,' '.join([str(q) for q in corrfax[s][0]]),
                               ' '.join([str(q) for q in corrfax[s][1]])])+'\n')
            # Each line has 21 values: [elementName]\t10*[f's ] 10*[f''s ]
    with open('CTs/scatcoef.txt','w') as f:   # For matlab
        for s in scatcoef:
            f.write('\t'.join([s,' '.join([str(q) for q in scatcoef[s]])])+'\n')
            # Each line has 10 elements [ele]\t[a1 b1 a2 b2 a3 b3 a4 b4 c]

    with open('CTs/corrfax.json','w') as f:
        json.dump(corrfax,f)
    with open('CTs/scatcoef.json','w') as f:
        json.dump(scatcoef,f)

def convertNISTDB(Z):
    """Convert NIST formfactor data from
    http://physics.nist.gov/cgi-bin/ffast/ffast.pl?Z=[ZVALUE]&gtype=4
    Using http://physics.nist.gov/PhysRefData/FFast/Text2000/sec02.html#eq04 and
    http://physics.nist.gov/PhysRefData/FFast/Text2000/sec08.html into JSON format"""

    url="http://physics.nist.gov/cgi-bin/ffast/ffast.pl?Z=%s&gtype=4"%str(Z)
    site=bs(urllib2.urlopen(url).read())
    eleName=re.match('\s*(\w+)',site.dl.b.contents[0]).group(1)
    print 'Parsing '+eleName+'..'
    text=''.join([q for q in site.dl.get_text() if ord(q)<128]) # strip non-unicode

    m=re.search((r'\s*(?P<ele>\S+)\s*\(Z.*Relativistic correction estimate[^\n]+,'
                 r'\s*(?P<frel>\S+)\s*e.*Nuclear Thomson correction[^\n]+='
                 r'\s*(?P<fNT>\S+)\s*e'),text,re.DOTALL)
    ele=m.group('ele')
    frel=float(m.group('frel'))
    fNT=float(m.group('fNT'))

    rawData=[]
    for line in site.find_all('pre')[-1].contents[50].split('\n')[1:-1]:
        rawData.append([float(q) for q in re.split('\s+',line)])
    rawData=zip(*rawData)[:3]

    return [eleName,fNT,frel,rawData[0],rawData[1],rawData[2]];

def createLocalArchiveNIST():
    """Convert data to JSON for easier parsing."""
    allEles={}
    for Z in range(1,93):
        sleep(2)
        [ele,fNT,frel,E,f1,f2]=convertNISTDB(Z)
        allEles[ele]={'fNT':fNT,'frel':frel,'Z':Z,
                      'E':E,'f1':f1,'f2':f2}
    with open('NIST/factors.json','w') as f:
        json.dump(allEles,f)

createLocalArchiveNIST()
```

Appendix C

Data

C.1 Sample MDA Data

```
# ******************************  Scan Divider  ******************************


# 2—D Scan Point
# Current point = 1 of 251
# Scanner = 7idd:scan2
# Scan time = Feb 05, 2006 16:32:14.596798416

# Column Descriptions:
#    1  [2—D Positioner 1]  7id:HHLM:energy_set, , LINEAR, keV, 7id:HHLM:energy_get.VAL, , keV

# 2—D Scan Values: 11.5000007


# 1—D Scan
# Points completed = 21 of 21
# Scanner = 7idd:scan1
# Scan time = Feb 05, 2006 16:32:24.730131344

#  Positioner: name, descr, step mode, unit, rdbk name, rdbk descr, rdbk unit
#  Detector: name, descr, unit

# Column Descriptions:
#    1  [     Index      ]
#    2  [1—D Positioner 1]  7cart:m3.VAL, , LINEAR, deg, 7cart:m3.RBV, Huber_Theta, deg
#    3  [1—D Detector  16]  7idd:scaler1.S1, ,
#    4  [1—D Detector  17]  7idd:scaler1.S2, ,

[...]

#   47  [1—D Detector  60]  7idd:scaler1.S16, ,
#   48  [1—D Detector  61]  7idd:scaler1.S11, ,
#   49  [1—D Detector  62]  7idd:scaler1.S13, ,
#   50  [1—D Detector  63]  7idd:scaler1.S14, ,
#   51  [1—D Detector  64]  7idd:scaler1.S15, ,

# 1—D Scan Values
#  1          2        3     4 5   6 7 8    9     10   11    12         13          14
#  15        16            17          18         19         20         21       22
#  23          24          25         26   27         28         29         30          31
#  32           33         34         35        36    37        38 39 40         41
#  42      43    44 45  46  47 48 49      50  51
   1 22.1402988 10000000 23416 0 956 0 0   37   29913 2400 70750 29.4791660 0.636758983
0.636758983 2.5333333 3.8006001e—06  1.428571463 0.219593495 102.219688 24.8647499 11.55019
—75.4308472 0.462000579 —6.71698856 0.156793401 11.5 7.52733660 1.81200886 447.064728
```

```
—950.166992 2.53820633e+11 —0.974999964 89.6338959 22.1402988 46.3866997 314.5 3.3499999
0  0 13.4931192 13.1827192 —381.25 —1000  0 —10   7  0  0   30649  10
   2 22.1415997 10000000 23385 0 942 0 0   68   37174 2314 75274 32.5298195 0.637369335
0.637369335 2.5333333 3.8006001e—06  2.857142925 0.219837397 102.179123 24.8647499
11.55019 —75.4306488 0.462127775 —6.72246408 0.156813204 11.5 7.52849245 1.81628132
447.803070 —949.402100 2.53484597e+11 —0.974999964 89.6338959 22.1415997 46.3866997
314.5 3.3499999  0  0 13.4931192 13.1827192 —381.25 —1000  0 —10   7  0  0   38257  20
```