Brent Fultz, Tim Kelley, Mike McKerns, Jiao Lin, JaeDong Lee, Olivier Delaire, Max Kresch, Michael Aivazis

Experimental Inelastic Neutron Scattering

Introduction to DANSE

This book is distributed as an Acrobat pdf document, and we recommend keeping it in this form. Many details of the software documentation are far too extensive to include inside this document directly, and are provided on web sites. Especially in Chapter 7, the software module documentation is accessed through web links in the pdf document. You are welcome to print hardcopies, but please keep the Acrobat pdf form too.

This book is dedicated to the new user – may you benefit from our mistakes.

Preface

A bright future lies ahead for neutron scattering studies of materials, molecules, and condensed matter. The international science community and national science agencies have invested heavily in new instruments and new sources of neutrons that will overcome many of the historical limitations of neutron scattering research. Higher intensities and instrument sensitivities are major and obvious improvements. Experimental *inelastic* neutron scattering has been particularly constrained by low countrates, forcing experimental compromises in energy resolution and intensity, for example. The ARCS inelastic neutron spectrometer with its location at the high-power target station of the Spallation Neutron Source and with its high detection efficiency, for example, will provide unprecedented experimental productivity, overcoming many of the restrictions caused by the low countrates that have accompanied inelastic neutron scattering experiments to date.

As features in the data from inelastic scattering experiments become more clear, and averaging processes are less necessary, more information can be extracted from experimental measurements. Advanced software is necessary to use this new information for sophisticated experiments. The goal of this book is to describe the underlying scattering physics and dynamic processes in materials, and show how modern software can be used to elevate the level of science done with inelastic neutron spectrometers. A large body of specialized knowledge is required to design modern experiments for inelastic instruments such as time-of-flight chopper spectrometers. This body of knowledge will only grow as new capabilities become available. Unfortunately, the underlying concepts are scattered over many disciplines. For example, the books on the theory of thermal neutron scattering by S. W. Lovesey and G. L. Squires are superb. Similarly, excellent solid-state physics texts by J. M. Ziman, C. Kittel, U. Rössler, and N. W. Ashcroft and N. D. Mermin are available for understanding the principles of excitations in condensed matter. Unfortunately, the concepts from neutron scattering and condensed matter physics are not connected well by existing written texts. A more serious frustration for an experimentalist is that there is no coherent source of explanations for the computing methods used for the analysis of inelastic neutron scattering data. A vast chasm separates *The C Programming Language* by B. W. Kernigan and D. M. Ritchie from *The Theory of Neutron Scattering from Condensed Matter* by S. W. Lovesey. The intellectual challenge for us au-

thors was organizing a coherent presentation of this wide body of knowledge, connected by the needs of experimental inelastic neutron scattering.

This document did not begin as a textbook. Our original concept was a manual of specifications for the ARCS data analysis software. Defining specifications is a major step in planning a software project and setting its scope. Writing a manual of specifications forces a high degree of detailed planning of classes and modules. As we struggled with these details on software structure, it became obvious that they should parallel as closely as possible the science and practice of experimental inelastic neutron scattering research. Besides the challenge of organizing the higher-level intellectual concepts, practical problems with notation became apparent almost immediately. (Should the scattering vector be $\boldsymbol{Q}$, $\boldsymbol{\kappa}$ or $\boldsymbol{\Delta k}$? What about its sign?)

This book is intended for a spectrum of readers spanning from graduate students beginning their doctoral research in inelastic neutron scattering, researchers who need to learn how to use *DANSE* (Distributed Data Analysis for Neutron Scattering Experiments) for data analysis, and ourselves, the authors, who need a reference manual explaining the purpose of the software modules of *DANSE*. The focus of this text, and our our heartfelt concern, however, was for the graduate student who enters the field of inelastic neutron scattering with no experience with instruments, probably only a sketchy understanding of the scientific principles, and perhaps limited knowledge of modern concepts in software engineering. This text was designed to help the reader in all three areas, and do so as efficiently.

Our philosophy is to present the minimal level of detail required to understand physical concepts. There is some sacrifice of the care in development found in *The Theory of Neutron Scattering from Condensed Matter* by S. W. Lovesey and *Introduction to the Theory of Inelastic Neutron Scattering* by G. L. Squires, but our goal was to provide explanations that are "best buys," producing the most physical insight for the amount of intellectual effort required to understand them. Another goal was to present the field of inelastic neutron scattering as a codified intellectual discipline, showing inter-relationships between different topics. To do so, the notation from other books has been altered in places, for example $\boldsymbol{Q}$, defined as $\boldsymbol{k}_\mathrm{i} - \boldsymbol{k}_\mathrm{f}$ was selected for the scattering vector so that κ could be consistent with its usage in *The Theory of Lattice Dynamics in the Harmonic Approximation* by A. A. Maradudin, et al. (It is an unfortunate, but well-established convention that the scattering vector in the diffraction literature is of opposite sign, $\boldsymbol{\Delta k} \equiv \boldsymbol{k}_\mathrm{f} - \boldsymbol{k}_\mathrm{i}$.)

Graduate students learning the concepts presented in this book are assumed to have some understanding of scattering experiments – a good understanding of x-ray diffraction would be suitable preparation. The student should have some competence with the manipulation of Patterson functions in Fourier space to the level developed, for example, in *Transmission Electron Microscopy and Diffractometry of Materials* by B. Fultz and J. W. Howe, and should have some understanding of solid-state physics at the level of *Prin-*

ciples of the Theory of Solids by J. M. Ziman or *Solid-State Physics* by H. Ibach and H. Lüth. Finally, the authors recommend a book such as *Learning Python* by M. Lutz and D. Ascher.

Concepts from neutron scattering, solid-state physics, and computer programming lay the foundation for the reader's path through this book on experimental inelastic neutron scattering. We hope your adventure through energy and momentum space will be as fun for you as it has been for us.

Brent Fultz, Tim Kelley, Mike McKerns, Jiao Lin,
JaeDong Lee, Olivier Delaire, Max Kresch, Michael Aivazis
Pasadena
May, 2007

Contents

Table 0.1. Symbols and Notation — ‡ denotes difference with Squires

General Notation	
$\boldsymbol{A}$	magnetic vector potential
a, $a^\dagger$	phonon annihilation and creation operators
$\{\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3\}$	primitive lattice translation vectors
‡ $\{\boldsymbol{a}_1^*, \boldsymbol{a}_2^*, \boldsymbol{a}_3^*\}$	primitive translation vectors of the reciprocal lattice
a_0	Bohr radius $a_0 = \hbar^2/(m_\mathrm{e}e^2)$
α	coefficient of linear thermal expansion
α, β	Cartesian indicies $\{x, y, z\}$
b	scattering length, sometimes coherent scattering length
$\widehat{\boldsymbol{b}}$, b, $b^\dagger$	scattering length operator, and eigenvalues for spin-$\frac{1}{2}$ system
B	bulk modulus
$\boldsymbol{B}_s$	magnetic field from electron spins in sample
$\boldsymbol{B}_L$	magnetic field from electron orbital momentum
c	velocity of sound or velocity of light
c, $c^\dagger$	annihilation and creation operators for electronic excitations
C_p	heat capacity at constant pressure
C_V	heat capacity at constant volume
$\delta(\boldsymbol{x})$	Dirac delta function
δ_{ij}	Kroneker delta function
$\boldsymbol{\Delta k}$	scattering vector $\equiv \boldsymbol{k}_\mathrm{f} - \boldsymbol{k}_\mathrm{i} = -\boldsymbol{Q}$
$\mathrm{d}^2\sigma/\mathrm{d}\Omega\,\mathrm{d}E$	double-differential cross-section
E	energy transfer $E = E_\mathrm{i} - E_\mathrm{f}$
$\boldsymbol{e}(\boldsymbol{k}, \sigma, \lambda)$	phonon polarization vector (with components for all κ in unit cell)
f	form factor of the scatterer (F.T. of potential in space and time)
f_σ	number of eigenvectors for each degenerate energy ($1 \le a \le f_\sigma$)
$G(\boldsymbol{r}, t)$	"Van Hove" function: space-time correlation function
‡ $G_\mathrm{s}(t)$	"self-correlation" function: time-time correlation function
γ	Grüneisen parameter
$\Gamma(R, \omega)$	space-energy corrrelation function
‡ $g(\omega)$	phonon density of states
‡ $\boldsymbol{g}$	arbitrary reciprocal lattice vector
‡ $\{h, h, l\}$	indicies of reciprocal lattice vector $\boldsymbol{g} = h\boldsymbol{a}_1^* + k\boldsymbol{a}_2^* + l\boldsymbol{a}_3^*$
H	Hamiltonian
‡ $I(\boldsymbol{Q})$	intensity (for diffraction)
‡ $I(\boldsymbol{Q}, \omega)$	intensity for scattering (same as $S(\boldsymbol{Q}, \omega)$)
$I(\boldsymbol{Q}, t)$	intermediate function
$J(\boldsymbol{r} - \boldsymbol{r}')$	exchange integral in Heisenberg Hamiltonian
‡ κ, κ'	indices of atom in basis
‡ $\boldsymbol{k}_\mathrm{i}$	incident neutron wavevector $k_\mathrm{i} = 2\pi/\lambda$
‡ $\boldsymbol{k}_\mathrm{f}$	scattered (final) wavevector $k_\mathrm{f} = 2\pi/\lambda$

Table 0.1. Symbols and Notation

General Notation (cont.)	
l, l'	indices of unit cells
λ	neutron wavelength
$\lambda_{\rm i}$, $\lambda_{\rm f}$	initial and final states of scattering system
λ (or a)	denotes (or index for) independent eigenvector for each σ or ω_σ^2
$\boldsymbol{M}_L(\boldsymbol{r})$	magnetization of sample (electron orbital part)
$\boldsymbol{M}_s(\boldsymbol{r})$	magnetization of sample (electron spin part)
$\boldsymbol{M}(\boldsymbol{r})$	magnetization of sample $\boldsymbol{M}(\boldsymbol{r}) = \boldsymbol{M}_L(\boldsymbol{r}) + \boldsymbol{M}_s(\boldsymbol{r})$
‡ $\widetilde{\boldsymbol{M}}(\boldsymbol{k})$	Fourier transform of sample magnetization
‡ $\widetilde{\boldsymbol{M}}_\perp(\boldsymbol{k})$	projected part of $\widetilde{\boldsymbol{M}}(\boldsymbol{k})$, perpendicular to both $\boldsymbol{Q}$ and $\widetilde{\boldsymbol{M}}(\boldsymbol{k}) \times \boldsymbol{Q}$
M_κ	mass of atom in basis
$\mathcal{M}(\tau)$	memory function of time-time correlations
$\{m, n, o\}$	indicies of real-space lattice vector $\boldsymbol{r}_l = m\boldsymbol{a}_1 + n\boldsymbol{a}_2 + o\boldsymbol{a}_3$
$m_{\rm e}$	electron mass
$m_{\rm n}$	neutron mass
$\mu_{\rm B}$	Bohr magneton $\mu_{\rm B} = e\hbar/(2m_{\rm e}c)$
$\mu_{\rm N}$	nuclear magneton $\mu_{\rm B} = e\hbar/(2m_{\rm n}c)$
$\boldsymbol{\mu}_{\rm N}$	magnetic dipole moment of neutron
ω	angular frequency in $\hbar\omega = E_{\rm i} - E_{\rm f}$
$P(\boldsymbol{r})$	"Patterson function": space-space correlation function
$\phi(v)$	velocity distribution of incident neutron flux
Φ	incident neutron flux
$\Phi_{\alpha\beta}(l\kappa; l'\kappa')$	force constant between two atoms
$\psi(\boldsymbol{r}, t)$	wavefunction of incident or scattered neutron
‡ $\boldsymbol{Q}$	scattering vector $\boldsymbol{Q} = \boldsymbol{k}_{\rm i} - \boldsymbol{k}_{\rm f}$
$\boldsymbol{r}_i$	position of a scattering atom or spin
‡ $\boldsymbol{r}_{l,\kappa}$	atom site in a crystal
‡ $\boldsymbol{r}_l$	lattice site in a crystal
‡ $\boldsymbol{r}_\kappa$	basis vector
r	number of atoms in basis
‡ $r_{\rm e}$	classical electron radius $r_{\rm e} = e^2/(m_{\rm e}c^2)$
$r_{\rm p}$	"classical proton radius" $r_{\rm p} = e^2/(m_{\rm p}c^2)$
$\boldsymbol{R}$	position vector

Table 0.1. Symbols and Notation

General Notation (cont.)	
$\boldsymbol{s}$	spin of electrons in sample
$S(\boldsymbol{Q},\omega)$	"scattering law" (scattered intensity: same as $I(\boldsymbol{Q},\omega)$)
σ	cross-section
σ_{coh}	coherent cross-section
σ_{inc}	incoherent cross-section
σ_{tot}	total cross-section
σ (or s)	denotes (or index for) a distinct value of ω_σ^2
‡ $2\theta = \phi$	scattering angle
‡ θ_{B}	Bragg angle
θ_{D}	Debye temperature
$\boldsymbol{u}(t)$	(small) displacement of nucleus
$\boldsymbol{u}_{l,\kappa}(t)$	displacement of nucleus in unit cell l and basis point κ
U	$U = -\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{u}(0)$
$\boldsymbol{u}(\kappa)$ and $u_\alpha(\kappa)$, $u_\beta(\kappa)$	atom displacement in a phonon, and Cartesian components
$\boldsymbol{v}$	velocity of neutron
V	$V = \mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{u}(t)$
W	exponent for Debye–Waller factor $= \mathrm{e}^{-2W}$ $W = \frac{1}{2}\langle(\boldsymbol{Q}\cdot\boldsymbol{u})^2\rangle$
ξ	extinction distance
$Y(Q,\tau)$	momentum-time corrrelation function
$\mathcal{Z}$	partition function

Table 0.1. Symbols and Notation

Group Theory Specialized Notation	
$\underline{\mathcal{S}} \equiv \{\mathbf{S}\,\vert\,\boldsymbol{v}(S)+\boldsymbol{x}(m)\}$	Seitz space group operator
h	order of the group (number of elements)
$\mathbf{S}$	matrix of rotation or improper rotation
$\boldsymbol{v}(S)$	little displacement vector for a screw axis operation
$\boldsymbol{x}(m)$	lattice translation vector
$\boldsymbol{\Gamma}_{\underline{\mathcal{S}}}(\boldsymbol{k};\{\mathbf{S}\,\vert\,\boldsymbol{v}(S)+\boldsymbol{x}(m)\})$	$3r\times 3r$ unitary matrix of symmetry operator (in recip. space)
$\mathbf{R}_i \equiv \{\mathbf{R}\,\vert\,\boldsymbol{v}(S)+\boldsymbol{x}(m)\}$	symmetry operator for a single wavevector $\boldsymbol{k}$

1. Introduction

1.1 Overview of this Book

Welcome to inelastic neutron scattering – theory, experiment, and data analysis. Welcome also to distributed data analysis for neutron scattering experiments, *DANSE*, and to the software engineering that makes it possible. This Chapter 1 begins with a short overview of the rest of the book, which spans from the quantum mechanics of scattering to the structure of the *DANSE* software.[1]

Chapters 2 and 3 are the most pedagogically formal chapters, written in the style of a graduate-level physics textbook. These textbook explanations were included for two reasons. Collecting the basic physics provides a handy reference for new users who are less familiar with inelastic neutron scattering. Second, textbook-style derivations provide both context and a common notational standard for *DANSE*. The descriptions of data processing steps in Chapter 6, and the descriptions of the software components in Chapter 7 refer directly to the variables, equations, and physical processes described in Chaps. 2 and 3.

Throughout Chap. 2, neutron scattering is treated as wave scattering. The phase of the neutron wavefunction scattered from a point, the complex exponent of $e^{i(\boldsymbol{Q}\cdot\boldsymbol{r}-\omega t)}$, is studied for its interference with waves emitted from other points in space, and from other instants in time. The intensity is the double Fourier transform of the Van Hove space-time correlation function: $I(\boldsymbol{Q}, E) = F_{\boldsymbol{r}} F_t G(\boldsymbol{r}, t)$. This relationship is developed in proper detail for both nuclear and magnetic scattering.

Chapters 2-4 are a subset of a course on condensed matter physics. They covers the known dynamical processes that cause the scattering potential to vary with time. For neutron scattering, these are the motions of nuclei, which are analyzed as phonon dynamics, and the motions of electron spins, which have several types of excitations. Thermodynamics aspects of these excitations are also described, as are some fundamental issues in modern research on correlated electron systems.

[1] Additional explanation about why this book was written, and how its scope was selected, is provided in the Preface.

Chapter 5 explains the components of inelastic neutron spectrometers. The focus is on the direct-geometry Fermi chopper spectrometer, which allows measurements over wide ranges of energy and momentum transfers between the neutron and the sample. Chopper spectrometers such as ARCS send a burst of neutrons down an incident flight path to the sample, and then time the arrival of neutrons at detectors. A number of arguments are presented to explain the operational issues and the technological challenges that were faced in the design of the ARCS instrument. When planning an experiment with ARCS, critical decisions must be made concering the size and shape of the sample, and the operation of the Fermi chopper. Chapter 5 helps explain these choices.

Chapter 6 describes the reduction of data from neutron scattering experiments. Large datasets, typically 0.5 GB/run, are acquired from a chopper spectrometer with a pixelated detector spanning a wide range in angle. The descriptions of data reduction focus on the types of data structures that must be manipulated at each step – their dimensions, variables, and physical meaning. Other types of data corrections are explained, including multiple scattering (where a neutron is scattered more than once as it traverses a thick sample), and multiphonon scattering (where a neutron generates more than one inelastic excitation when it is scattered).

Chapter 7 is a description of the software packages in *DANSE*. To some extent, this chapter follows a conventional style for software documentation familiar to programmers and scientists. It does, however, contain embedded web links so that updated, thorough developers' information on specific packages can be obtained promptly by clicking on the links. These web pages include developer's explanations, and documentation (including UML diagrams) generated by doxygen and epydoc. Maintaining these thousands of pages is practical with an online format, but is not practical in LaTeX documents.

Chapter 8 explains the broad principles of computer science. There are certainly many other books on modern object-oriented programming, but this tends to be a topic neglected by scientists working on neutron scattering. It is hoped that Chapter 8 will reveal the beautiful correspondence that is possible between the structure of well-designed software and the hierarchy of physical concepts. To some extent, summarizing physical laws in software is similar to summarizing them in a textbook. Such background on the structure of computer programs is helpful for understanding Chapter 9.

Chapter 9 explains the software architecture of *DANSE*. It covers the data analysis capabilities presently available to a scientist, but the emphasis is on explaining the architecture. *DANSE* is composed of software encapsulations called "components" that communicate with each other through standard types of data streams. This modularized design, together with its control structure, makes it possible to distribute the data analysis over multiple computers, as in Grid computing.

Another important part of Chapter 9 is its explanation of how the software components are built. In essence, modularization was performed by extending the open-source language Python. Python is an interpreted language, so it is possible to restructure Python programs interactively without recompilation. Interpreted languages have always had an an advantage for quick rearrangement and experimentation. This programming flexibility is inherited by *DANSE* because the components of *DANSE* are actual Python functions. Within reason, they can be rearranged flexibly into new Python programs that are run through the Python interpreter. Some of the actual components are written as Python code, but this is not practical for many components that require the high performance of a compiled language such as C++.

Incorporating compiled code into *DANSE* requires some explanation. Everything that runs on a computer is a program, including the language Python itself. Since Python is written in the C language, and its source code is openly available, it is practical to use the C language to write new functions for Python. A standard applications program interface (API) is provided in a Python installation to help programmers extend Python by adding extensions written in C or C++. Chapter 9 explains how to use this API to build new Python functions from code written in C, using examples from neutron scattering data analysis.

1.2 Python and C++

1.2.1 What is Python?

An interesting, short explanation of Python is provided by the Python open source community. The following quotation from their website,

`http://www.python.org`,

also conveys some of the flavor of Python documentation and perhaps gives some insight into the open source community itself.

> *python*, (Gr. Myth. An enormous serpent that lurked in the cave of Mount Parnassus and was slain by Apollo) 1. any of a genus of large, non-poisonous snakes of Asia, Africa and Australia that suffocate their prey to death. 2. popularly, any large snake that crushes its prey. 3. totally awesome, bitchin' language that will someday crush the $'s out of certain other so-called VHLL's ;-). Python was developed by Guido van Rossum, who named it after the classic British television comedy series *Monty Python's Flying Circus*.
>
> Python is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, Scheme or Java.
>
> Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types,

> and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC). New built-in modules are easily written in C or C++. Python is also usable as an extension language for applications that need a programmable interface.
>
> The Python implementation is portable: it runs on many brands of UNIX, on Windows, DOS, OS/2, Mac, Amiga... If your favorite system isn't listed here, it may still be supported, if there's a C compiler for it. Ask around on comp.lang.python – or just try compiling Python yourself.
>
> Python is copyrighted but freely usable and distributable, even for commercial use.

An excellent tutorial on Python is distributed with the software itself, and the `python.org` website has links to others.

1.2.2 What is C++?

Compared to Python, C++ and C are lower-level languages. They are important when pushing computing efficiency to its limits, as is often needed in numerical computation. In essence, C++ is the computer language C with new extensions for object-oriented programming.

Although the language C is approximately a subset of C++, there are some differences between these languages even for the same capabilities. Compared to C, the C++ language provides for better error checking by the compiler, and increased flexibility in working with variables and functions. Some improvements C++ are in input/output, flexibility in naming functions, typing and initialization of variables, and keywords.

The powerful new capability of C++ is its extension to object-oriented programming. The old programming paradigm of functions and data is replaced by objects that include both. Important concepts include abstraction, where the object interface is specified in detail, so the code for the object may be replaced readily or written at a later stage of a software project. The C++ language alows control over encapsulation, where the code for an object can be hidden from other parts of a program, reducing the liklihood of unwanted interactions. C++ provides a hierarchial structure that encourages the building of larger objects from smaller ones, and for deriving new objects from a standard base object, which inherit its characteristics. C++ also supports virtual functions, which allow a set of different objects to be treated in a similar way.

The C++ language is powerful, but it is dangerous in that it offers many ways to write cryptic code with very obscure bugs. This is a serious concern for a project where programming is done in part by physical scientists, who are more interested in results than in the elegance of software structures. In the year 2001 when the project was started, there was no clear alternative

programming language that would have been supported for more than a decade into the future. On the other hand, there is elegance in the structure of computer programs, and this elegance compensates in part for the discipline required for robust C++ programming.

1.3 *DANSE*

1.3.1 What is *DANSE*?

DANSE or "Distributed Data Analysis for Neutron Scattering Experiments," is an integrated software system for doing neutron scattering science on a computer. "Doing neutron scattering" means 1) performing all essential transformations of raw data acquired with neutron instruments, plus 2) modeling and simulating the structure and dynamics of the samples, and predicting the neutron scattering. "Integrated" means providing a uniform environment for comparing 1 and 2. The modular structure of the proposed system parallels the steps of data analysis performed by scientists, making it natural to use. It is extensible so that working scientists can easily transform their computer programs into interoperating *DANSE* components. The distributed nature of the architecture makes it possible to use high-performance computing resources that enable new types of science with neutron scattering data.

A full understanding of *DANSE* requires the understanding of two subjects. The first, summarized in Section 1.3.2, is a software framework that permits the interoperability of modular software components. The second, summarized in Section 1.3.3, is the set of software modules needed for data analysis for the different subfields of neutron scattering research. A more detailed description of the software architecture is presented in Section 1.4 of this chapter, and in Chapters 7 and 9.

1.3.2 *DANSE* Programming Environment

Technically, *DANSE* is a "component-based runtime environment." This means that the components are pre-compiled, and interconnected by the user at runtime. Examples of components are: readers for data files, C++ codes for data reduction, user-interactive graphics packages, commercial data analysis environments such as Matlab, `FORTRAN` codes for electronic structure calculations on Linux clusters, and Monte Carlo simulations of instruments for neutron scattering. The user directs the interconnections of these components, using either a graphical programming interface or a command-line interface, depending on need or preference. In a graphical programming interface, components are depicted naturally as boxes to be retrieved from library shelves, and their interconnections are depicted as wires.

Interconnections between components provide data flow and control. An important design decision for the *DANSE* architecture was to build the components and interconnections with a high-level programming language. Program execution is directed by a Python interpreter [6][7], using XML [8][9] for describing the data and control functions. Using a high-level language, Python, for these interconnections frees developers from working with compilers, low-level linkers and header files. The developers of software for neutron scattering research are scientists. Many gravitate towards commercial data analysis environments such as Matlab [10], IDL [11] or Igor [12] to escape from the lower levels of computing. The challenge is to offer them both a more powerful environment and ease of use.

Python is readily extensible to incorporate FORTRAN programs, C++ programs, Matlab, and IDL. All components in the *DANSE* system are Python objects, and in a single process space these objects interoperate through the Python interpreter [13][14]. Interoperation in a distributed environment requires XML-based data exchange protocols, but neutron scattering research requires the exchange of only three generic types of data as explained below. A possible concern about an interpreted high-level language such as Python is that it involves the tradeoff of computing performance for convenience. This presents a design opportunity, however. The most computationally-intensive tasks should be performed within the components compiled from FORTRAN and C++ codes, and given Python bindings. Minimal time is lost at the level of the Python interpreter when transferring execution threads between components. The advantage of having the interconnections of components at a high level is that more scientists are enabled to contribute new components to the system.

1.3.3 Enabling Neutron Science

The *DANSE* system offers opportunities for the design of more efficient experiments by use of use of prior simulations, or simulations that could be performed as measurements are underway and experimental trends begin to emerge. Some of the most exciting new experiments made possible by the most recent high-performance neutron instruments can be executed only with new software. Examples include single crystal analysis with chopper spectrometers, or beam time optimization for stress tensor measurements with an engineering diffractometer. Detailed simulations of the structure and dynamics of materials and molecules are readily possible with the *ab initio* calculations of today [15][16][17][18][19], but these simulations and neutron scattering experiments cannot yet be compared in the same environment. Scientists should be exploiting fully the data from neutron instruments, performing sophisticated experiments with detailed input from theory, comparing the experimental data to simulations, and having more creative freedom to analyze the data in a broader scientific context than is possible with a

stand alone software package running on a personal computer. Offering powerful software and hardware to the user will elevate the standards for data analysis, and elevate the level of science.

The *DANSE* data analysis software is being developed by scientists who work with real data. Neutron scattering research is organized into subfields specializing in different types of science, approximately clustered around instruments that perform particular types of measurements. To best accommodate the needs of these subfields, the development of scientific software follows a similar decomposition into subfields:

1. *Diffraction.* This method has the largest user community. Experiments includes studies of crystal structure and microstructure, both on liquids and amorphous materials, polycrystalline materials, and single crystals.
2. *Engineering Diffraction.* Research in this field includes measurements and interpretations of internal strains in materials, and studies of crystalline textures in polycrystallie materials.
3. *Small-Angle Neutron Scattering (SANS).* Users in this field have interests that span from biochemistry to solid-state magnetism. SANS research includes a large activity in polymer structure, and the structural evolution of polymers under temperature and flow.
4. *Reflectometry*, which measures the depth profile of neutron scattering near a surface. The science includes structures of large molecules at surfaces and interfaces, and surface magnetism probed with polarized neutrons.
5. *Inelastic scattering*, which studies dynamical processes such as the elementary excitations of phonons and magnons in solids, and vibrations and motions of molecules.

1.3.4 Inelastic Neutron Scattering with *DANSE*

The focus of this book is on experimental inelastic neutron scattering theory and data analysis.[2] For extracting science from inelastic neutron scattering data, *DANSE* provides a set of Python language scripts, modules, and shared objects. To some extent, *DANSE* can be viewed as a part of the Python language, although of course it is not part of the standard Python distribution.

A number of working applications built from components have been tested and documented. The flexibility of *DANSE* should allow for minor changes of these data analysis procedures to accommodate experiments that differ slightly from previous ones, or major changes to accommodate entirely new experiments or types of data analyses. Some components of *DANSE* are illustrated schematically in Fig. 1.1.

DANSE provides a rich set of tools for many standard operations on neutron inelastic scattering data, such as conversion of time-of-flight data

[2] It is expected that the other subfields of neutron scattering science will develop documentation in a parallel way.

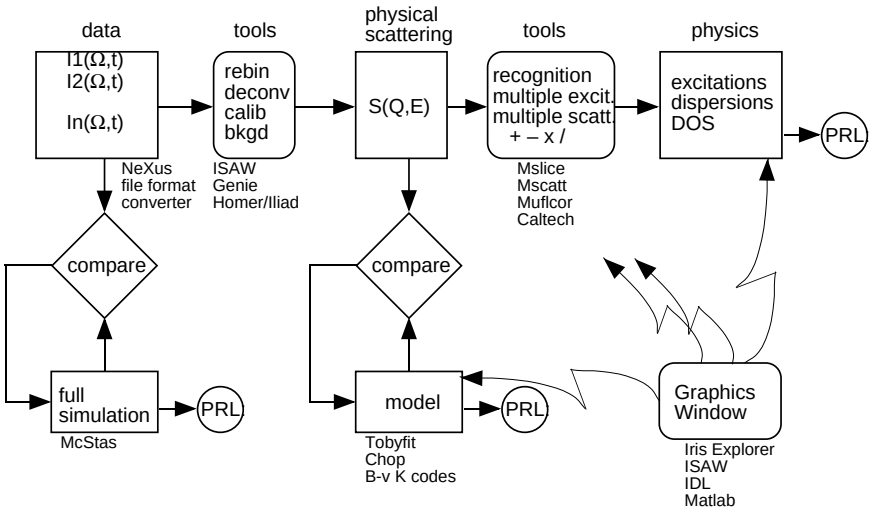


Fig. 1.1. Some components of *DANSE*, which need not necessarily be connected simultaneously. The traditional forward analysis of data is shown in the path across the top, with simulations and modeling accessing this path vertically. PRL denotes a publication. This figure was developed in the year 2000 before the standardization of UML in the *DANSE* project, and is shown for historical interest.

into physical distributions in momentum and energy, background corrections, deconvolutions, corrections for multiple scattering and multiple excitations, and data visualization. In addition to this "traditional" analysis, *DANSE* offers new capabilities for working with models of structure and dynamics. For example, a module `spinwave` allows the user to select a model for the dynamics of thermal excitations on an ordered array of spins, and tune the exchange interactions in the model for a best fit to the $S(Q, \omega)$ obtained from a traditional forward analysis of experimental data.

DANSE offers non-traditional types of data analysis by full simulation of time-of-flight data. Monte Carlo simulations of the instrument characteristics have been integrated with simulations of the spin and nuclear dynamics of the sample, allowing a direct calculation of the counts measured at the detector bank. A set of graphics tools can be inserted at various locations of *DANSE* scripts to examine the results of data operations, modeling, or simulations. The outputs from these three types of operations can be piped simultaneously into the visualization packages to facilitate direct comparisons between theory and experiment, or between different types of data analysis.

1.3.5 User Interactions with *DANSE*

Distributed Computing. Figure 1.2 illustrates data analysis as a service accessed through remote procedure calls to a central compute server. Com-

ponents can be maintained and run centrally on well-tested platforms. A compute server, not the user, arranges for computation on the appropriate hardware. A user could elect for little code and no raw data to reside on his or her local computer, while still directing the reuse and reconfiguring of the needed Python components, including those that run on specialized hardware such as Beowulf clusters. Our intent is to ensure that any user can utilize the highest-performance hardware without buying and maintaining it.

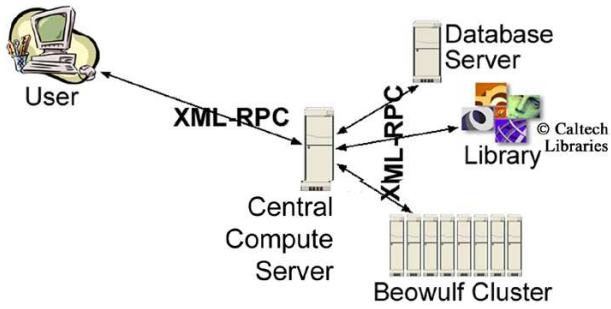


Fig. 1.2. Architecture for distributed computing. The XML-RPC protocol was an early choice that was typically run through ssh tunnels.

Nevertheless, many of the data analysis components could run locally on the user's computer. Bindings for proprietary scientific data analysis packages such as IDL or Matlab are available. If a user owns one of these proprietary packages, it can be seamlessly integrated into a *DANSE* analysis procedure. Nevertheless, software running on a user's local computer will include automatic and transparent Internet access to the *DANSE* compute server. Such a remote access to the *DANSE* system will overcome a traditional problem with software releases for a user's local computer, where software developers must design for the least-capable hardware. With such access to the central service, capabilities for large simulations on the *DANSE* system, for example, would be available to the user through a familiar software client.

User interfaces. User interfaces, especially graphical user interfaces (GUIs), can simplify access to the software system while offering access to much of its flexibility and power. These two requirements of simplicity and flexibility are usually in conflict, and different users will prefer different balances between them. User interfaces are usually controversial, and some designs are fraught with peril, as illustrated in Fig. 1.3

There are fundamental reasons why different users will always need different interfaces to the data analysis framework. A simple set of controls is appropriate not only for new users, but also for advanced users who have repetitive tasks. Such "dashboard"-style controls are also appropriate when real time analyses are performed during data acquisition at 3:00 AM. A second

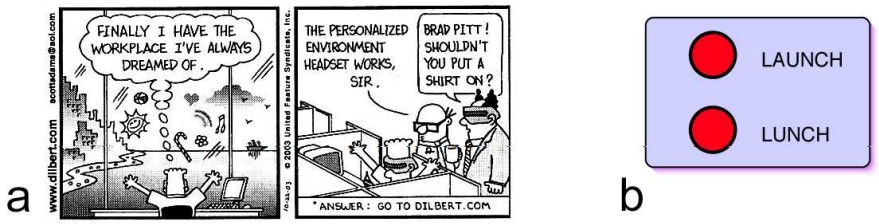


Fig. 1.3. User interfaces. (**a**) Good. (**b**) Bad.

user interface is appropriate for rearrangements of data analysis procedures, such as performing a different type of background correction or a fit to a different type of analytical model. A "wiring diagram" metaphor (see Fig. 1.4) is an intuitive way for the user to specify these operations, where the user can select data analysis components from both local and remote libraries, and drag-and-drop the components onto the GUI canvas. A command line interface allows a user to access the full capabilities of the Python language and might be the preference of some users, especially for development work.

Graphical user interfaces are expected to change with time. Changing the *DANSE* GUI should not be difficult because of a clean separation between the user's depiction of the data analysis procedure, and the server's execution of the data analysis procedure. Parts of the first *DANSE* GUI were adapted with major modifications from the "Visual Python Programming Environment" (ViPEr) [26] developed by Michel Sanner's group at the Scripps Oceanographic Institute. ViPEr is based on open source software[3]. This user interface is shown in Fig. 1.4. The execution engine of ViPEr has been modified substantially to be compatible with the *DANSE* system, but the interface is shown in Fig. 1.4.

1.4 High-Level Architecture of *DANSE*

1.4.1 UML

This section uses "UML Diagrams," which are part of the Unified Modeling Language 2.0 (a standard maintained by the Object Management Group, an open consortium). UML offers a set of graphical diagrams that describe software systems, and these diagrams are especially suited for describing object-oriented software systems such as *DANSE*. There are several different types of UML diagrams. Each type of diagram has its own scope, in which information is presented accurately, and each type of diagram has known limitations that bound its scope. For example, a deployment diagram shows the computers on

[3] Python, Tkinter, PyXML, Python Mega Widgets, 4Suite, PyOpenGL (and subpackage: gle), Gnuplot.py, Numeric.py

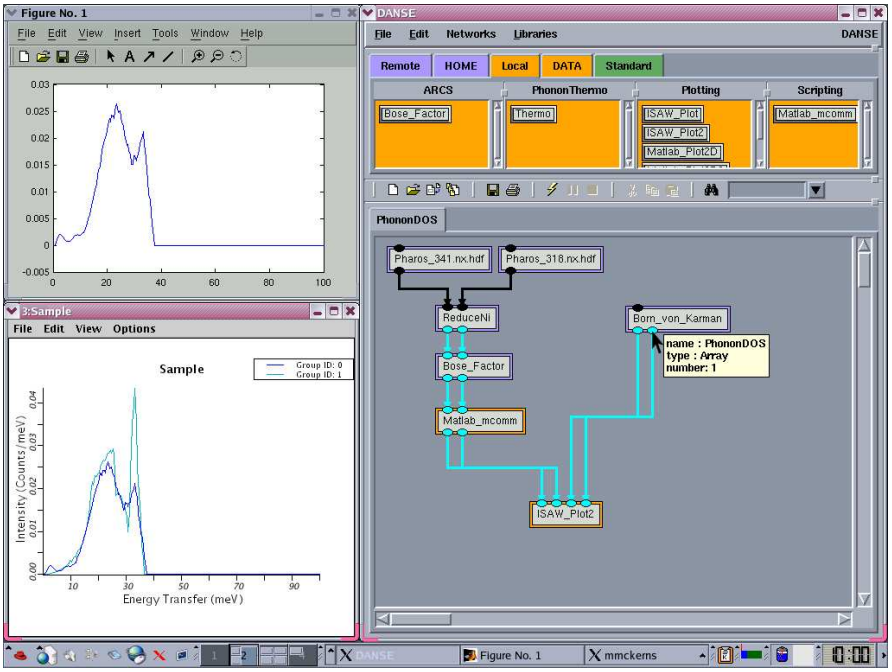


Fig. 1.4. Screen shot of the present ViPEr interface to the *DANSE* client software. Boxes on upper right represent libraries of data analysis components, local or remote, on five file systems. The workspace for arranging data analysis procedures is on the lower right, with components in place for reducing a data set from the Pharos instrument. As the cursor is placed over input or output ports, a description of data types is presented in a dialog box. Upper left shows the Matlab window launched by the analysis network, in which the data files were corrected and normalized. The output from Matlab was sent into the graphical display in the lower left window for comparison with a lattice dynamics simulation on the right of the window. (This display was generated by ISAW, an integrated spectral analysis workbench.) Components that ran remotely are bordered in purple, local components are bordered in yellow.

which software components run, but has no information about the sequence of component execution. The strength of UML diagrams is that within their scope of information they are unambiguous, and are well documented. Two recommended references are:

1. M. Kratochvil and B. McGibbon, "UML Xtra-Light" (Cambridge Univ. Press, Cambridge, 2003). Very short and easy to read.
2. M. Fowler, "UML Distilled" (Addison-Wesley, Boston, 2004). Excellent descriptions of important UML diagrams such as class diagrams, but sketchy descriptions of others that are less important to *DANSE*. The *DANSE* project has adopted the notation used in this book.

1.4.2 Software Packges

A high-level UML diagram is presented in Figure 1.5. This figure is a UML package diagram, and the boxes are many of the actual software packages in the alpha release of the ARCS software of July, 2005 (Monte Carlo, ARCStest, and simulation packages are not shown). The software packages are at the scale of application or utility programs, and some of them can be used as standalone applications. The arrows in the figure depict dependencies of the software packages. For example, the package "stdVector" must be installed for "reduction" to operate, but "visualization" does not need "stdVector." Documentation on these packages is the subject of Chapter 7, which contains web links to more thorough and recent information. The full *DANSE* system will contain many more packages, perhaps by a factor of ten. These packages are not yet formulated, but their specific functionalities are listed in the Work Breakdown Structure for *DANSE*, which to date is an NSF Construction proposal.

http://wiki.cacr.caltech.edu/danse/index.php/Top-level_doc_for_ARCS_alpha

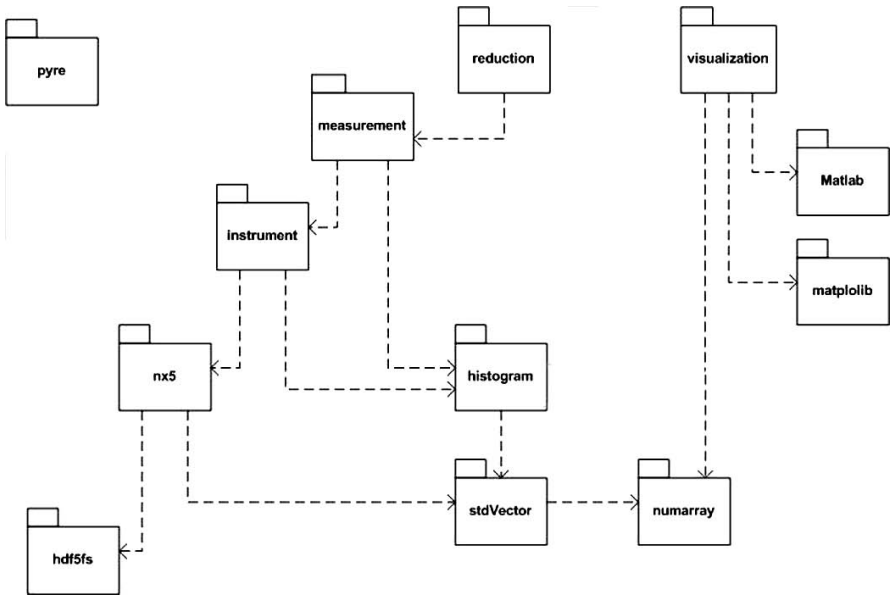


Fig. 1.5. Packages in the alpha release of the ARCS software for data reduction and visualization, showing dependencies. The package "reduction" has the internal structure shown in Fig. 1.9.

1.4.3 Use Cases

Use cases illustrate how a software system is intended to be used. An actor (in the sense of a person starting actions of the software) follows a usage scenario. It is unclear if a natural language description is of more value than a graphical UML use case diagram, but we use both in the three examples presented in Figure 1.6.

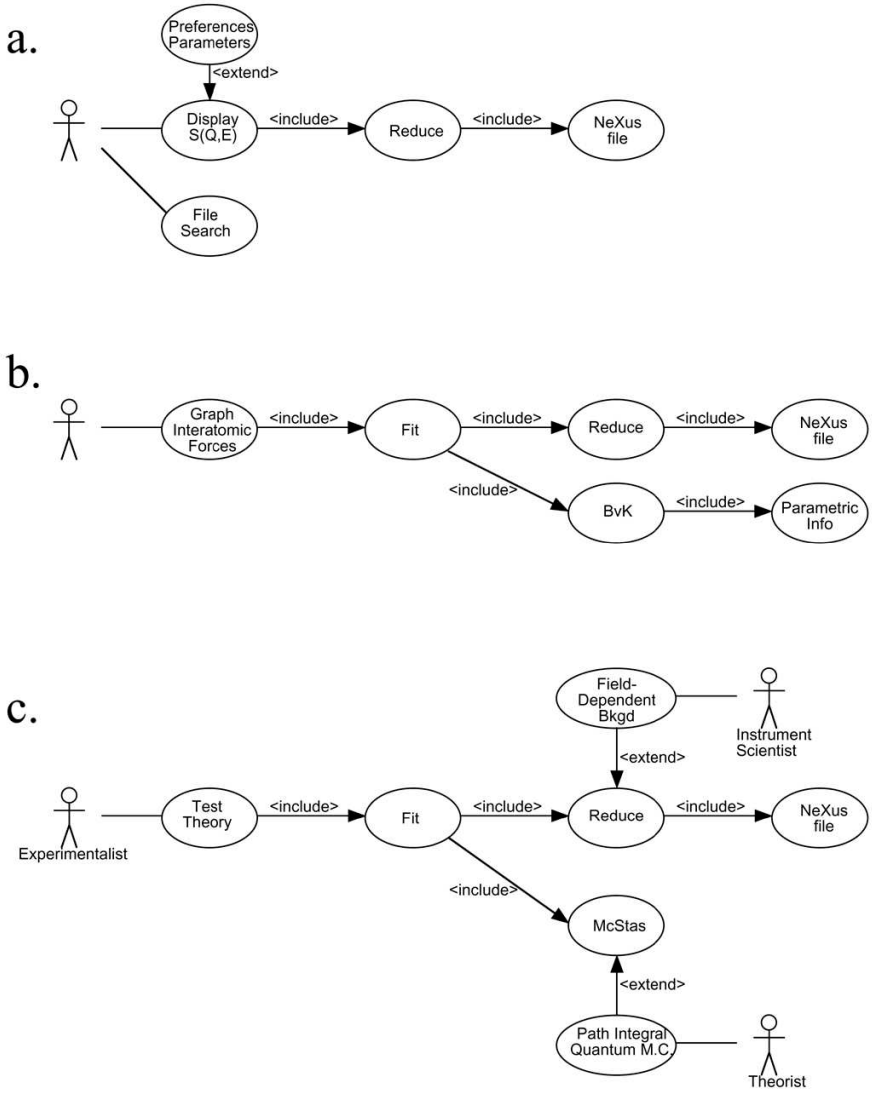


Fig. 1.6. Three use cases for *DANSE* (**a**) Reduction. (**b**) Modeling. (**b**) Simulation.

Figure 1.6a depicts a common scenario when a neutron scattering experiment is running. The actor (user) at left locates disk files containing recent data. The user seeks the display of $S(Q, E)$, but this user has a preferred format for the display, perhaps preferences that were set for him by another person some time in the past. The display requires the use of a software package called "Reduce," which accesses the NeXus files containing the raw data.

Figure 1.6b depicts a typical scenario after data acquisition, perhaps performed at the user's home or office. This user has discovered a large shift in phonon frequencies with temperature, and seeks to make a plot of the interatomic forces in a crystal versus temperature. The software package to graph the interatomic forces requires access to reduced data as in the first use case, but it uses an iterative algorithm where the parameters in a Born-von Kármán lattice dynamics model of Section 4.1 are optimized to fit the reduced experimental data.[4]

Use case diagrams are interesting when multiple actors are involved. Figure 1.6c depicts a collaborative effort to measure an energy gap in a one-dimensional quantum magnet. The goal is to test a theory of spin dynamics. Unfortunately, the experimenter noticed an increased instrument background associated with the applied magnetic field, and requested help from the instrument scientist in correcting the background used by the package Reduce. Being a persuasive fellow, the experimentalist was also able to interest a theoretically-inclined colleague to adapt a high-powered model for the imaginary part of the spin susceptibility for use in a McStas simulation of the experimental scattering. The theorist wants to run the Path Integral Quantum Monte Carlo code on her own computing cluster, for which the code was optimized.

These use cases for *DANSE* are of course not a complete set. Nevertheless, they already show how and why users demand a wide variety of software capabilities. These vary from the self-contained and quick analysis of Figure 1.6a to large collaborative efforts involving multiple persons at different locations. Flexibility of software functionality and adaptable usage of computing resources is important for the analysis of neutron scattering data.

1.4.4 Deployment Diagrams

Computer hardware continues to follow the trend of Moore's Law, where processing performance per unit cost, P, continues to double every 1.5 years as $P \propto 2^{y/1.5}$, where y is in years. In some scientific domains it has been found that software algorithms follow a similar trend, although we know of no such

[4] An astute reader may wonder if the fit is performed at the level of the $S(Q, E)$ from Figure 1.6a, or is there a further reduction to a phonon density of states before the fit. Although the diagram could be refined further to provide this detail, there will invariably be further questions about data formats that are used by Fit, Reduce and BvK that are beyond the scope of a use case diagram.

assessment for neutron scattering software. Unfortunately, the complexity of physical systems that can be simulated is not increasing with time as $2^{y/0.75}$. Scaling of electronic structure calculations with the number of electrons, N, goes as $N \propto P^{1/3}$, and the algorithms are improving slowly. Perhaps the number of atoms for which these computations are possible is increasing with time as $N \propto 2^{y/3}$, i.e., doubling every three years. The improvement in computing capability is an exciting trend, but scientific computing will be constrained by computing resources for at least the next decades. Access to computing resources will remain a concern for scientists, and flexibility in using these resources is a central priority for the *DANSE* system.

DANSE offers flexiblility in its use of computing resources, as shown by the UML deployment diagrams of Figure 1.7. UML deployment diagrams show how software uses various parts of hardware and other software systems. Figure 1.7a is particularly simple. It is often practical to carry data files in a laptop computer, and some types of visualization and analysis can be performed efficiently with this deployment of *DANSE*. Such a deployment is not sufficient for computations that include molecular dynamics, finite element methods, or LDA electronic structure calculations, of course. The valuable computing resources for such computations are negotiated by science teams at universities, companies, and national laboratories, following many different procedures and rules of access. Because *DANSE* supports computing in user-configurable networks, deployments such as in Figure 1.7b are possible. A design philosophy of *DANSE* is that it requires no more, but no less, access to computing resources than a user would receive on these various systems. In most cases this means that distributed computing with *DANSE* is practical when the user has a standard user account on a computing system.

1.4.5 Classes and Inheritance

Inheritance is a powerful concept in object-oriented programming. When a programmer writes a new class, he or she need not include code for all the required functionality if the new class can "inherit" this functionality from a "superclass."[5] For example, it takes work to write a software component that sends data across a network to a file on a different computer. Assume that such a code is already in hand, however. A user who has a small program to generate an important data string might seek to use this network transmission code by including all the lines of this code in his program. His short code is overwhelmed by the addition of this new functionality.

Inheritance is an easier way to give small components powerful capabilities. For example, by inheriting attributes of the `pyre` framework, a simple component will be endowed with capabilities of stream communications across a network. The author of the component can expect that the streams

[5] If possible, it is easier to get rich by inheriting than by doing actual work.

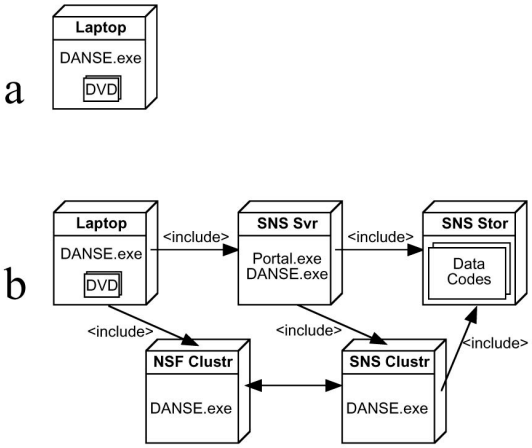


Fig. 1.7. Deployment diagrams for *DANSE*. (**a**) A minimal deployment on a laptop computer without network access, useful on airplanes for example. (**b**) A distributed analysis network, assuming authorized user access through firewalls.

from a simple component are fully separated from streams from other components. This separation is assured because when a simple component, here a Python class, is instantiated into an object[6], it occupies a part of the computer memory independent of the other components.

Consider an example of inherited functionality. First consider a standard case where a text string is written to a file. This can be done with built-in Python functions, but redirecting the standard i/o is another way to do this. We make use of the module `sys`, and direct its output to a disk file:

```
import sys
sys.stdout = open('log.txt', 'a')
print "This text message will go into the file <log.txt>."
```

Running this three-line program generates a new file, `log.txt`, that contains the text string as above. Our simple program inherited the attributes contained in the module `sys`, including its attribute `stdout`. This approach could be used to write a simple code to write a file across a network. We need to also import `pyre`:

```
import sys pyre
sys.stdout = pyre.gsl.scp.open('btf@caltech.edu:~/log.txt', 'a')
print "This text message will go into the file <log.txt>."
```

The result is a new file, `log.txt`, located in the home directory of btf on the computer caltech.edu. Here `pyre.gsl` provides the network services in

[6] A class is part of the code. To use it, an instance is created and named. This is an "object" in memory. Although it has the same functionality as other objects instantiated from the same class, these multiple objects have their own states and operate independently. Consider instantiation as a "cloning" process, for example.

a convenient and consistent way, even if the remote system is not running *DANSE*. Importing this functionality is quite easy.

Classes in Python and `pyre` are based on namespaces. When a Python class statement is executed, assignments create names local to the class. These become class attributes. By instantiating a class, making it into a named object in memory, the new object has its own local attributes. The object inherits from superclasses, however, by searching upwards for attributes not available locally. Each usage of features from a superclass is kept in a distinct namespace. Each can be reached by a detailed address following the convention `highobject.midobject.lowobject.attribute`.

1.4.6 Class Diagrams

A class diagram shows the structure of real code. Class diagrams can be constructed by automatic tools that scan software modules for imports and class declarations, then graph how components are inter-related. These tools also identify the function declarations and data structures internal to the components.

Notation and details of class diagrams are covered by excellent books on UML, especially the text by M. Fowler. For the purpose of the present discussion of the *DANSE* system, however, we show the high-level class diagram of the `pyre` framework in Figure 1.8. A component, perhaps written by a user in the Python language, is shown right of center in this diagram.

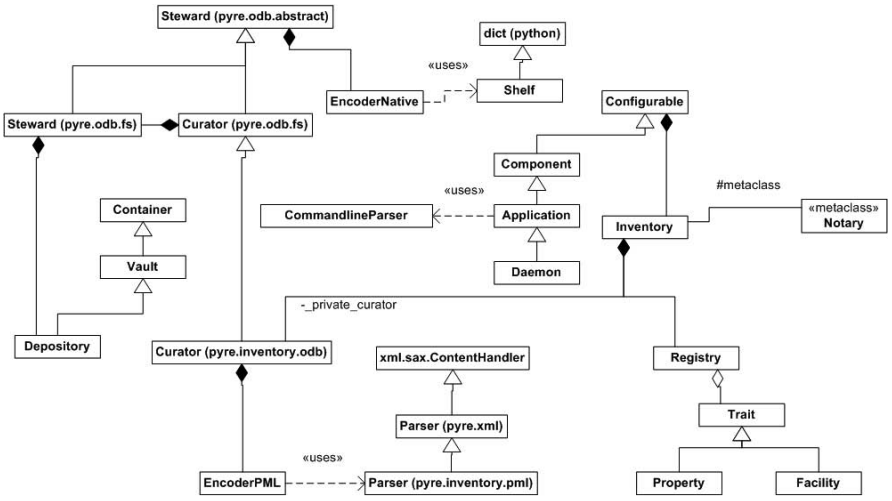


Fig. 1.8. Class diagram of `pyre` 0.6.

Figure 1.8 shows how `pyre` interacts with one component. A real application, such as for data reduction, may include a hundred components. It is

most useful to think of `pyre` not as all the classes of Figure 1.8 and their subclasses, but rather as a supporting infrastructure for the *DANSE* system. Pyre manages the life cycles of components and endows them with standard functionality. Two versions of a component diagram are shown in Figure 1.9. For many pruposes, it is sufficient to know that components are supported by `pyre` as in Fig. 1.9b so that the arrows between them operate as expected, for example. After all, in such diagrams we draw neither the components of the operating system, nor the components of the Python language, even though we know that these are essential. Figure 1.9b is a standard UML 2.0 component diagram showing how components interact on the `pyre` framework.

1.4.7 A View of a Software Component Framework

The value of a software component framework can be understood by analogy to an operating system, but at a higher level of abstraction. Without an operating system, it is possible to boot a computer into the entry point in memory for an application program. The application program would need to include input/output services so that output could be saved to a disk file. It would then be possible to reboot the computer to run a second application that uses the saved results. Today nobody uses a computer this way. All users demand a robust operating system that allows application programs to run concurrently, facilitates interchanges of data through memory or pointers, manages the memory, and hides many machine-specific details of the input/output routines beneath the application layer. An analogous set of basic services is provided by a component framework, conceptually located in a layer above the operating system. The framework allows a scientific programmer to focus on scientific cores of components, with less attention to specifics of operating systems, data structures, error handling, or even the physical location of the computation.

Technically, pyre is a "component-based runtime environment." This means that the components are pre-compiled, and interconnected by the user at runtime. The user directs the interconnections of these components, using either a menu, a graphical programming interface, or a command-line interface, depending on need or preference. The user interface is a layer independent of the components, and can be replaced or modified without affecting the core functionalities of *DANSE*. For either distributed or local computing, the user could select a favorite interface for all neutron instruments, shortening the learning curve for new users, and encouraging expert users to experiment with new types of data analysis and computational science.

DANSE will be organized with the *data flow paradigm* as the basic abstraction, with a layer for execution control as shown in Fig. 1.9a. Typical scientific analysis codes, written in languages such as `Python`, `Fortran` and C, will be the *cores* of software *components*. A component mediates in several ways between the core and its environment. The components inherit methods from the framework, including methods for passing data and handling

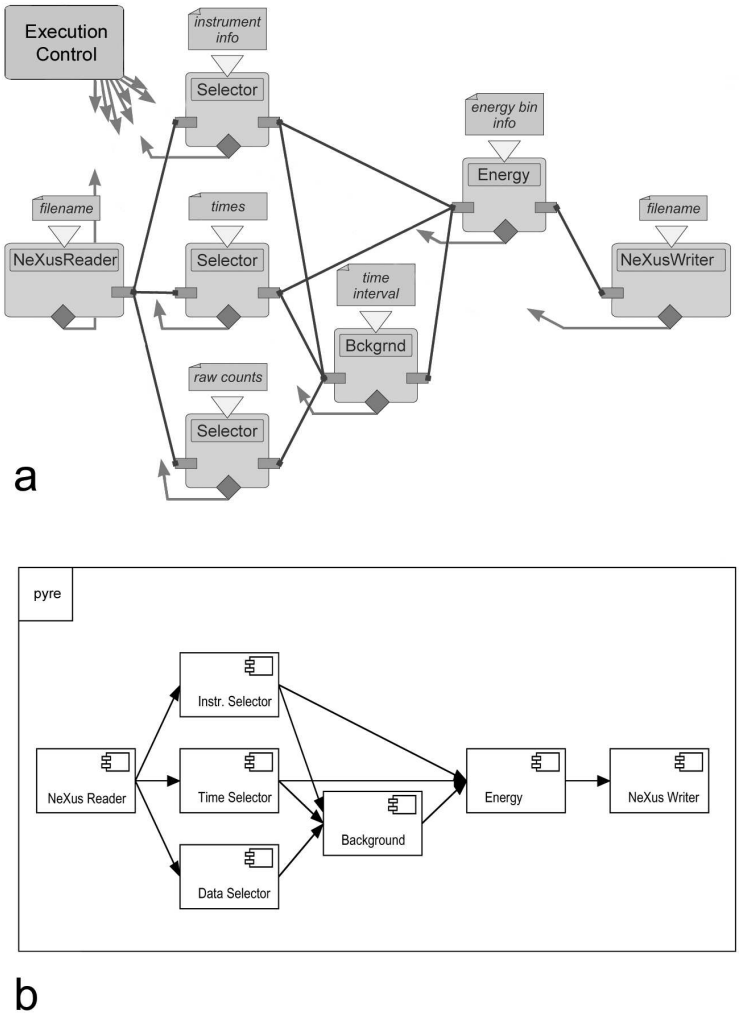


Fig. 1.9. (**a**) Schematic of the conversion of raw data into an energy spectrum. The component `NeXusReader` is responsible for reading a file, and converting it into a data object. After further analysis and user-supplied information, the component `Energy` produces a histogram of intensity as a function of energy. Components such as `Bckgnd` and `Energy` are themselves composed of lower-level components. (**b**) Conventional UML component diagram, showing components interacting on a framework labeled "pyre."

errors. The component is responsible for the initialization of its core, which may require user-supplied information (depicted in Fig. 1.9a as information above the component boxes). After a component is instantiated, it provides information to the framework that could be passed to the user interface. Components will negotiate their data exchanges with the help of `XML`-based data exchange protocols when they are first connected.

The dark lines between components in Fig. 1.9a depict *data streams* between the input and output ports of different components. The conceptual decoupling of components from each other through data streams facilitates their *physical decoupling*. With the serialization of data streams,[7] it becomes possible to distribute the computation among multiple computers. Alternatively, the user may prefer to organize a computation with multiple threads, or as separate processes on the same computer. The control of component execution is encapsulated in the *executive layer* of the framework. The executive layer manages the life cycles of components, and handles error conditions. With the assistance of the executive layer, components will have access to a centralized mechanism for the logging of status, errors, and for preserving a record of the computation.

The role of the `pyre` framework in *DANSE* is analogous to the role of Python itself. Pyre provides a set of debugged superclasses for the following services:

- creation and destruction of components in memory on different computers
- standardized inter-component communication
- distributed computing
- user interfaces
- handling of errors
- journaling of debugging streams for networks

The component framework also offers these important advantages:

- *encapsulation* Users of a functionality are insulated from changes to the underlying code in future releases of pyre
- *structure* Local scope minimizes name conflicts in a large system
- *maintenance* Even in a large system, there can be only one copy of code to change
- *consistency* Components have common interfaces

Further Reading

The contents of the following are described in the Bibliography.

[7] For the domain of computational neutron scattering research, the main types of data to be exchanged between components are histograms and tables. Other lightweight information that describes the data will be passed as *metadata* described by `XML`.

H. A. Abelson and G. J. Sussman: *Structure and Interpretation of Computer Programs* (MIT Press, Cambridge Mass, 2001).

Mark Lutz and David Ascher: *Learning Python* (O'Reilly & Associates, Inc. 1999).

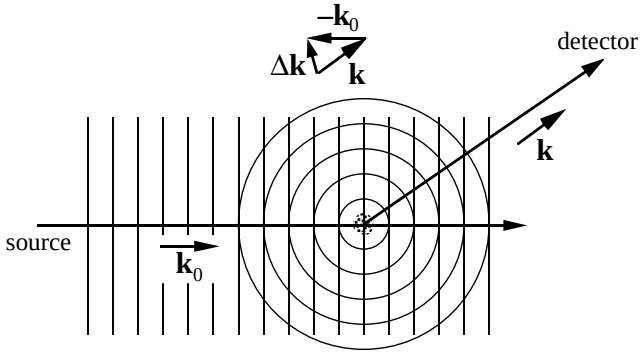
Brent Fultz and Doug Abernathy: *ARCS Spectrometer web site*,
`http://www.cacr.caltech.edu/projects/ARCS/`

Tim Kelley, Mike McKerns, Jiao Lin, Michael Aivazis and Brent Fultz: *DANSE wiki web site*,
`http://wiki.cacr.caltech.edu/danse/index.php/Main_Page`

M. Kratochvil and B. McGibbon: *UML Xtra-Light* (Cambridge Univ. Press, Cambridge, 2003).

M. Fowler: *UML Distilled* (Addison-Wesley, Boston, 2004).

2. Scattering



This chapter on scattering assumes that the reader is reasonably knowledgeable about elastic scattering as it is used in diffraction studies of atomic structure. (After all, x-ray diffraction is common currency for chemists, materials scientists and condensed matter physicists.) The chapter builds on this basic understanding of diffraction, although selected background concepts are developed here. We begin with a discussion of coherence and energy, in part because these concepts are sometimes taken for granted by persons having extensive experience with diffraction experiments. In Sect. 2.2, elastic scattering from static potentials is developed in the Born approximation, giving the basic Fourier transform relationship between the wave and the scattering factor distribution: $\psi(\boldsymbol{Q}) = Ff(\boldsymbol{r})$. This is followed by a brief explanation that shows how time-varying potentials cause inelastic scattering by modulating the frequency of the scattered wave.

More progress in understanding scattering experiments is possible by analyzing the correlation functions that are derived from the measured intensity, rather than the neutron wavefunction (which is not measured directly). Section 2.4 returns to elastic scattering to develop the concept of the Patterson function, $P(\boldsymbol{r})$. The Patterson function is the spatial correlation function that includes all information about the diffracted intensity, as opposed to the diffracted wave. The corresponding correlation function for inelastic scatter-

ing is the Van Hove space-time correlation function. It is developed in Sect. 3.1 in Chapter 3 in a path that parallels the Patterson function, but is more general. The double Fourier transform of the Van Hove function, $G(\boldsymbol{r}, t)$, provides all information about the scattered intensity (elastic and inelastic). The Patterson function is a special case of the Van Hove function for static scattering potentials. Graphical examples are used to demonstrate fundamental features of the scattered intensity for the elastic case with the Patterson function: $I(\boldsymbol{Q}) = F_{\boldsymbol{r}} P(\boldsymbol{r})$, and for the inelastic case with the Van Hove function: $I(\boldsymbol{Q}, E) = F_{\boldsymbol{r}} F_t G(\boldsymbol{r}, t)$. The two Fourier relationships between the scattered intensity and the correlation functions are discussed in detail in this chapter and the next:

- $I(\boldsymbol{Q}) = F_{\boldsymbol{r}} P(\boldsymbol{r})$, which relates the diffracted intensity, $I(\boldsymbol{Q})$, to the Fourier transform (from space to momentum) of the Patterson function, $P(\boldsymbol{r})$.
- $I(\boldsymbol{Q}, E) = F_{\boldsymbol{r}} F_t G(\boldsymbol{r}, t)$, which relates the scattered intensity, $I(\boldsymbol{Q}, E)$, to two Fourier transforms (from space to momentum, and time to energy) of the Van Hove function, $G(\boldsymbol{r}, t)$.

2.1 Coherence and Incoherence

Diffraction requires "coherent scattering," characterized by a precise relationship between the phases of the incident and scattered waves. The scattered wave is the sum of component waves, "wavelets" as we call them, emanating from the different atoms in the sample. In diffraction, phase differences between these outgoing wavelets cause constructive or destructive interferences at different angles around the sample, e.g., the appearance of Bragg diffraction peaks.

2.1.1 Wavefunctions

Phase. A wavefunction $\psi(x, t)$ describes the structure of a wave (its crests and troughs) along position x, at any time t. The mathematical form $\psi(kx - \omega t)$ accounts for how the wave amplitude shifts in position with increasing time, t. The argument of the wavefunction, $kx - \omega t$, is called the "phase" of the wave. It includes two constants: k (the wavevector), and ω (the angular frequency). The phase $kx - \omega t$ is dimensionless, so it can be used as the argument of a sine function or a complex exponential, for example. Our mathematical form causes the entire structure of the wave $\psi(kx - \omega t)$ to move to more positive x with increasing t. This is clear if we recognize that a particular wavecrest in ψ exists at a particular value of phase, so for larger t, the wave amplitude moves to larger x for the same value of $kx - \omega t$.[1]

[1] We say $\psi(kx - \omega t)$ travels to the right with a "phase velocity" of ω/k. The wave $\psi(kx + \omega t)$ travels to the left.

One-Dimensional Wave. One-dimensional waves are simple because they have no vector character. Suppose the wave is confined a region of length L. The wavefunction and its intensity are:

$$\psi_{1\mathrm{D}}(x,t) = \frac{1}{\sqrt{L}}\ \mathrm{e}^{+\mathrm{i}(kx-\omega t)}\ , \tag{2.1}$$

$$I_{1\mathrm{D}} = \psi_{1D}(x,t)\ \psi^*_{1D}(x,t) = \frac{1}{\sqrt{L}}\ \mathrm{e}^{+\mathrm{i}(kx-\omega t)}\frac{1}{\sqrt{L}}\ \mathrm{e}^{-\mathrm{i}(kx-\omega t)}\ , \tag{2.2}$$

$$I_{1\mathrm{D}} = \frac{1}{L}\ . \tag{2.3}$$

If $\psi_{1\mathrm{D}}(x,t)$ were an electron wavefunction, the intensity, $I_{1\mathrm{D}}$, would be a probability density. The prefactor in (2.1) ensures proper normalization in the interval L, with a probability of 1 for finding the electron in the interval:

$$P = \int_0^L I_{1D}\ \mathrm{d}x = \int_0^L \frac{1}{L}\ \mathrm{d}x = 1\ . \tag{2.4}$$

Plane Wave. In three dimensions, a plane wave is:

$$\psi_{3\mathrm{Dpl}}(\boldsymbol{r},t) = \frac{1}{\sqrt{V}}\ \mathrm{e}^{+\mathrm{i}(\boldsymbol{k}\cdot\boldsymbol{r}-\omega t)}\ , \tag{2.5}$$

which has an intensity and a normalization analogous to those for the one-dimensional wavefunction (at a snapshot in time). The spatial part of the phase, $\boldsymbol{k}\cdot\boldsymbol{r}$, is illustrated in Fig. 2.1a with $\boldsymbol{k}\cdot\boldsymbol{r} = 0$, and 2.1b with $\boldsymbol{k} \parallel \boldsymbol{r}$ for two orientations of $\boldsymbol{r}$. Along the direction of $\boldsymbol{r}$ in Fig. 2.1a there is no change in the phase of the wave (here $\psi_{3\mathrm{Dpl}}(\boldsymbol{r},t) = 1/\sqrt{V}\ \mathrm{e}^{+\mathrm{i}(0-\omega t)}$), whereas in Fig. 2.1b the phase changes most rapidly along $\boldsymbol{r}$ (here $\psi_{3\mathrm{DPlan}}(\boldsymbol{r},t) = 1/\sqrt{V}\ \mathrm{e}^{+\mathrm{i}(kr-\omega t)}$). The dot product $\boldsymbol{k}\cdot\boldsymbol{r}$ for the phase in (2.5) gives the plane wave its anisotropy.

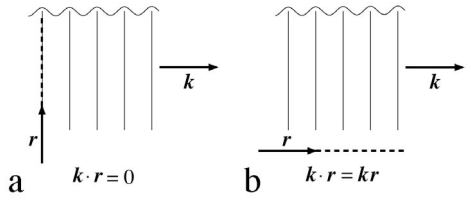


Fig. 2.1. Plane wave with $\boldsymbol{k}$ oriented to the right, with orientations of $\boldsymbol{r}$ being (**a**) along the wave crests, perpendicular to $\boldsymbol{k}$, (**b**) parallel to $\boldsymbol{k}$.

Spherical Wave. By placing the origin of a spherical coordinate system at the center of the spherical wave, the spherical wave has its simplest form:

$$\psi_{3\mathrm{Dsph}}(\boldsymbol{r},t) = \frac{1}{\sqrt{V}}\ \frac{\mathrm{e}^{+\mathrm{i}(kr-\omega t)}}{r}\ . \tag{2.6}$$

If the center of the spherical wave is the distance $\boldsymbol{r}_0$ away from the origin of the coordinate system:

$$\psi_{\rm 3Dsph}(\boldsymbol{r},t) = \frac{1}{\sqrt{V}}\ \frac{{\rm e}^{+{\rm i}(k|\boldsymbol{r}-\boldsymbol{r}_0|-\omega t)}}{|\boldsymbol{r}-\boldsymbol{r}_0|}\ . \tag{2.7}$$

Figure 2.2 shows a vector construction for $\boldsymbol{r}-\boldsymbol{r}_0$, which can be obtained by connecting the tail of $-\boldsymbol{r}_0$ to the arrow of $\boldsymbol{r}$. At distances far from the scattering center, where the curvature of the spherical wave is not important, it is often useful to approximate the spherical wave as a plane wave with $\boldsymbol{r}-\boldsymbol{r}_0$ pointing along the direction of $\boldsymbol{k}$.[2]

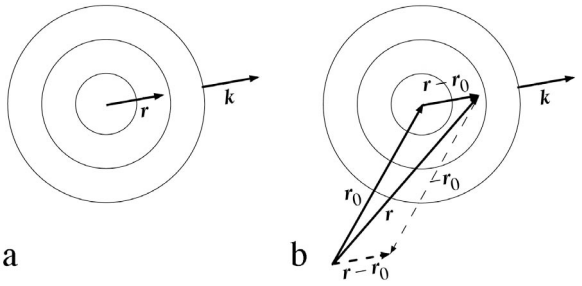


Fig. 2.2. Spherical wave with $\boldsymbol{k}$ oriented away from the center of wave emission. (**a**) with coordinate system for $\boldsymbol{r}$ having its origin at center of wave emission. (**b**) with coordinate system for $\boldsymbol{r}$ having an arbitrary origin.

Phase Factor. A phase factor, ${\rm e}^{-{\rm i}\boldsymbol{\Delta k}\cdot\boldsymbol{R}}$ or ${\rm e}^{-{\rm i}(\boldsymbol{\Delta k}\cdot\boldsymbol{R}+\omega t)}$, has the mathematical form of a plane wave (2.5), and is associated with a particular wavelet, but beware. *A phase factor is not a wave.* A phase factor proves handy when two or more wavelets are scattered from different points in space at $\{\boldsymbol{R}_j\}$, typically separated by some atomic distances. What is important after the long path to the detector is how the wavelets interfere with each other – constructively or destructively – and this is accounted for by sums of phase factors like this:

$$\psi_{\rm phf}(\boldsymbol{\Delta k}) = \sum_{\{\boldsymbol{R}_j\}} {\rm e}^{-{\rm i}\boldsymbol{\Delta k}\cdot\boldsymbol{R}_j}\ . \tag{2.8}$$

The definition $\boldsymbol{\Delta k}\equiv\boldsymbol{k}-\boldsymbol{k}_0$ (illustrated in the chapter title image) is repeated a number of times in this book. This $\boldsymbol{\Delta k}$ is a difference in the wavevectors of two actual waves. Dot products like $\boldsymbol{\Delta k}\cdot\boldsymbol{R}_j$ give phase differences between wavelets, but $\boldsymbol{\Delta k}$ is not an actual wavevector. Chapter 5 develops these concepts, but the reader is now forewarned that exponentials containing $\boldsymbol{\Delta k}$ are not waves, but phase factors.

2.1.2 Coherent and Incoherent Scattering

Coherent scattering preserves the relative phases of the wavelets, $\{\psi_{\boldsymbol{r}_j}\}$, scattered from different locations, $\{\boldsymbol{r}_j\}$, in a material. For coherent scattering,

[2] This is often useful because real scatterers typically emit spherical waves, but Fourier transforms require plane waves.

the total scattered wave, $\Psi_{\rm coh}$, is constructed by adding the *amplitudes* of the scattered wavelets:

$$\Psi_{\rm coh} = \sum_{\boldsymbol{r}_j} \psi_{\boldsymbol{r}_j} \ . \tag{2.9}$$

The total coherent wave therefore depends on the constructive and destructive interferences of the wavelet amplitudes. Diffraction experiments measure the total coherent *intensity*, $I_{\rm coh}$:

$$I_{\rm coh} = \Psi^*_{\rm coh}\Psi_{\rm coh} = \left|\sum_{\boldsymbol{r}_j} \psi_{\boldsymbol{r}_j}\right|^2 \ . \tag{2.10}$$

On the other hand, "incoherent scattering" does not preserve a phase relationship between the incident wave and the scattered wavelets. For incoherent scattering it is incorrect to add the amplitudes of the scattered wavelets, $\{\psi_{\boldsymbol{r}_j}\}$. Incoherently-scattered wavelets do not maintain phase relationships, so they cannot interfere constructively or destructively. The total intensity of incoherent scattering, $I_{\rm inc}$, is the sum of individual scattered intensities:

$$I_{\rm inc} = \sum_{\boldsymbol{r}_j} I_{\boldsymbol{r}_j} = \sum_{\boldsymbol{r}_j} \left|\psi_{\boldsymbol{r}_j}\right|^2 \ . \tag{2.11}$$

Because measurable intensities are added in incoherent scattering, the angular distribution of incoherent scattering from a group of N identical atoms is the same as for a single atom, irrespective of how these N atoms are positioned in space. The total intensity is simply N times larger. Some types of incoherent scattering occur with a transfer of energy from the wave to the material, and these processes can be useful for spectroscopic analysis of the atom species in a material.

It is important to emphasize the difference between the right-hand sides of (2.10) and (2.11). Because the intensity of coherent scattering in (2.10) first involves the addition of wave amplitudes, coherent scattering depends on the relative phases of the scattered wavelets and the relative positions of the N atoms in the group. Coherent scattering is useful for diffraction experiments. Incoherent scattering is not. This chapter describes in sequence the four types of scattering having coherent components that allow for diffraction experiments on materials:

- x-rays, which are scattered when they cause the atomic electrons to oscillate and re-radiate,
- electrons, which are scattered by Coulomb interactions when they penetrate the positively-charged atomic core,
- neutrons, which are scattered by nuclei (or unpaired electron spins), and
- γ-rays, which are scattered when they resonantly excite a nucleus, which later re-radiates.

2.1.3 Elastic and Inelastic Scattering

Besides being "coherent" or "incoherent," scattering processes are "elastic" or "inelastic" when there is, or is not, a change in energy of the wave after scattering. We can therefore construct four word pairs:

(coherent elastic) (coherent inelastic)
(incoherent elastic) (incoherent inelastic)

Diffraction experiments need coherent elastic scattering, whereas spectroscopies that measure intensity versus energy often use incoherent inelastic scattering. The case of incoherent elastic scattering is also common, and occurs, for example, when phase relationships between scattered wavelets are disrupted by disorder in the material. Incoherent elastic intensity does not show the sharp diffractions associated with crystalline periodicities, but has a broad angular dependence. Finally, coherent inelastic scattering is used in neutron scattering studies of excitations in materials, such as such as phonons (vibrational waves) or magnons (spin waves), that have precise energy-wavevector relationships. In some phonon studies, a neutron loses energy when creating a phonon (so it is inelastic), but the scattering amplitude depends on the phases of the atom movements in the phonon with respect to the neutron wavevectors (so it is coherent).

A deeper and more rigorous distinction between coherent and incoherent scattering involves our knowledge about the internal coordinates of the scatterer:

- Consider a simple oscillator (a bound electron, for example) that is driven by an incident wave and then re-radiates. There is a transfer of energy from the incident wave to the oscillator, and then to the outgoing wave. Suppose we know in full detail how the coordinates of the oscillator respond to the incident wave. Since the scattering process is determined fully, the phases of all outgoing wavelets have a precise and known relationship to the phase of the incident wave. The scattering is coherent.
- On the other hand, suppose the coordinates of this oscillator were coupled to another system within the material (a different electron, for example), and furthermore suppose there is freedom in how the oscillator can interact with this other system. (Often differing amounts of energy can be transferred from the oscillator to the other system because the transfer occurs by a quantum mechanical process that is not deterministic.) If this energy transfer is different for different scatterings, we cannot predict reliably the phase of the scattered wavelet. The scattering is incoherent.

It is therefore not surprising that incoherence is often associated with inelastic scattering, since inelastic scattering involves the transfer of energy from the scatterer to another component of the material. Incoherence does not imply inelastic scattering, however, and inelastic scattering is not necessarily incoherent.

2.1.4 Wave Amplitudes and Cross-Sections

Cross-Sections. X-rays, electrons, neutrons, and γ-rays are detected one-at-a-time in scattering experiments. For example, the energy of an x-ray is not sensed over many positions, as are ripples that spread to all edges of a pond of water. Either the entire x-ray is detected or not within the small volume of a detector. For x-ray scattering by an individual atomic electron as described in the next section, the scattering may or may not occur, depending on a probability for the x-ray–electron interaction.

An important quantity for scattering problems is the "cross-section," σ, which is the effective "target area" presented by each scatterer. With cross-sections it is handy to think of a number, N, of scatterers in a sample of area A as in Fig. 2.3. The probability of scattering is equal to the fraction of sample area "blocked" by all N scatterers. For thin samples when the scatterers do not overlap, the N scatterers block an area equal to $N\sigma$. The fraction of rays removed from the incident beam is the blocked area divided by the total area:

$$N\,\frac{\sigma}{A} = N\,\frac{\sigma x}{Ax} = \rho\,\sigma x\ . \tag{2.12}$$

Here the density of scatterers, $\rho \equiv N/(Ax)$ has units [scatterers cm^{-3}].

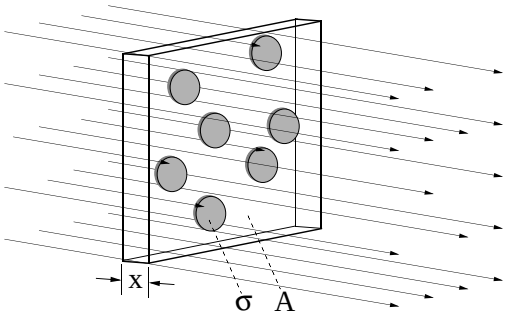


Fig. 2.3. These 7 scatterers occupy the fraction 0.2 of the sample area, A, and therefore remove the fraction 0.2 of the rays from the incident beam. From (2.12): $\sigma = (0.2/7)A$. In the thin sample limit, the number of scatterers and the amount of scattering increase in proportion to thickness, x, but σ remains constant.

To illustrate a salient feature of coherent scattering, consider the elastic scattering of neutrons through the interaction of their spin with the spin polarization of electrons in an antiferromagnet. The total cross-section depends on the total number of unpaired electrons in the material. As mentioned after (2.11), for incoherent scattering the picture would then be complete – the spatial distribution of the scattered intensity is obtained by adding the intensities from independent scattering events from different atoms.

Coherent scattering requires further consideration of the wave amplitudes before calculating the cross-section. A hierarchy of wave interference processes can occur between spin structures on different length scales:

- the unpaired electrons in the same atom (atomic form factor),

- the atoms in the unit cell of the crystal (structure factor),
- the unit cells in the crystal (shape factor), and
- density variations across a material (small angle scattering).

The spatial redistribution of scattered intensity can be spectacularly large in the case of Bragg diffractions, but the total coherent cross-section remains constant. By rearranging the atom positions in a material, the constructive and destructive interferences of coherent scattering are altered and the angles of scattering are redistributed, but for the same incident flux the scattered energy is conserved (for x-rays or γ-rays), or the total number of scattered particles remains the same (electrons and neutrons).

The flux of scattered x-rays, electrons, neutrons, or γ-rays at the distance $\boldsymbol{r}$ from the scatterer decreases as $1/r^2$ along $\hat{\boldsymbol{r}}$. A scattered photon carries energy, so the radiated energy flux also decreases as $1/r^2$ from the scatterer. The energy of a photon is proportional to E^*E, so the electric field, E, has an amplitude that must decrease as $1/r$ from the center of scattering. For scattered x-rays, we relate the electric field along $\widehat{\boldsymbol{r}}$ to the incident electric field at the scatterer, E_0:

$$E(\boldsymbol{r}) \propto \frac{E_0}{r} \ , \tag{2.13}$$

where the constant of proportionality would include any angular dependence. The electric fields $E(\boldsymbol{r})$ and E_0 in (2.13) have the same units, of course, so the constant of proportionality has units of length. The square of this "scattering length" is the cross section per steradian, as we next show for electron scattering (but the argument pertains to all waves).

Cross-Section for Wave Scattering. Here we find the cross-section for wave scattering. Consider the total flux, $J_{\rm sc}(R)$, scattered through a unit area of surface of a sphere at radius R around the scatterer. The incident beam has a flux $J_{\rm in}$ over an area A. The ratio of all scattered electrons to incident electrons, $N_{\rm sc}/N_{\rm in}$, is:

$$\frac{N_{\rm sc}}{N_{\rm in}} = \frac{J_{\rm sc}(R)\ 4\pi R^2}{J_{\rm in}\ A} = \frac{v\ |\psi_{\rm sc}(R)|^2\ 4\pi R^2}{v\ |\psi_{\rm in}|^2\ A} \ . \tag{2.14}$$

We consider elastic scattering for which the incident and scattered electrons have the same velocity, v, but for inelastic scattering these factors do not cancel. We use the spherical wave (2.6) for $\psi_{\rm sc}(R)$ and the plane wave (2.5) for $\psi_{\rm in}$. For both waves, the exponential phase factors, multiplied by their complex conjugates, give the factor 1. The normalization factors also cancel, so (2.14) becomes:

$$\frac{N_{\rm sc}}{N_{\rm in}} = \frac{|f_{\rm el}|^2\ 4\pi R^2}{R^2\ A} \ , \tag{2.15}$$

where $f_{\rm el}/R$ is the fraction of the incident electron amplitude that is scattered into a unit area of the sphere at radius R. Figure 2.3 helps demonstrate the

fact that the ratio of the cross-section σ to the area A of the incident beam equals the ratio of scattered to incident electrons, $N_{\rm sc}/N_{\rm in}$:

$$\frac{\sigma}{A} = \frac{N_{\rm sc}}{N_{\rm in}} = \frac{4\pi|f_{\rm el}|^2}{A} \,, \tag{2.16}$$

$$\sigma = 4\pi|f_{\rm el}|^2 \,. \tag{2.17}$$

The scattering of an x-ray by a single atomic electron can be treated in the same way, but we need to account for the electric dipolar pattern of x-ray radiation with a factor of 2/3 in the cross-section,

$$\sigma_{\rm x1e} = \frac{8\pi}{3}\left|f_{\rm x1e}\right|^2 \,, \tag{2.18}$$

where $f_{\rm x1e}$ is the scattering length. This $f_{\rm x1e}$ is the actual constant of proportionality to convert (2.13) into an equality.

Anisotropic scattering is the rule rather than the exception, however, so simple cross-sections like those of (2.17) are usually inadequate, even if altered by factors like the 2/3 used in (2.18). The "differential scattering cross-section," written as $\mathrm{d}\sigma/\mathrm{d}\Omega$, contains the angular detail missing from the total cross-section, σ.

The differential scattering cross-section, $\mathrm{d}\sigma/\mathrm{d}\Omega$*, is the piece of area offered by the scatterer,* $\mathrm{d}\sigma$*, for scattering an incident x-ray (or electron or neutron) into a particular increment in solid angle,* $\mathrm{d}\Omega$*.*

The concept of $\mathrm{d}\sigma/\mathrm{d}\Omega$ is depicted Fig. 2.4. Note that $\mathrm{d}\sigma/\mathrm{d}\Omega$ relates an increment in area (on the left) to an increment in solid angle (on the right).

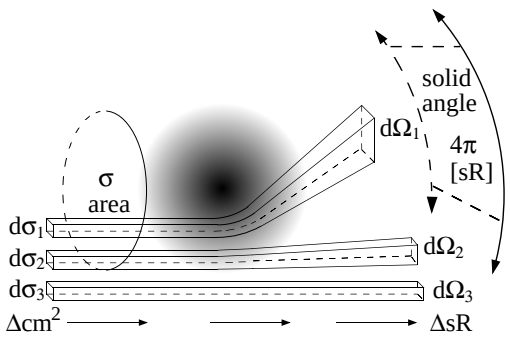


Fig. 2.4. The differential scattering cross-section, $\mathrm{d}\sigma/\mathrm{d}\Omega$, for three paths past a scatterer. The third path, $\mathrm{d}\sigma_3/\mathrm{d}\Omega_3$, misses the scatterer and contributes only to the forward beam. The paths with areas $\mathrm{d}\sigma_1$ and $\mathrm{d}\sigma_2$ make contributions to the total cross-section for scattering, σ, and these contributions are included when the intensity is integrated over the differential solid angles $\mathrm{d}\Omega_1$ and $\mathrm{d}\Omega_2$.

For the simple case of isotropic scattering,

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega} = \left|f\right|^2 \,, \tag{2.19}$$

which is a constant. For anisotropic scattering, (2.19) is generalized with a scattering length, $f(\boldsymbol{k}_0, \boldsymbol{k})$, that depends on the directions of the incident and outgoing wavevectors, $\boldsymbol{k}_0$ and $\boldsymbol{k}$, respectively:

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega} = \left|f(\boldsymbol{k}_0, \boldsymbol{k})\right|^2 . \tag{2.20}$$

We expect to recover the total cross-section, σ, by integrating $\mathrm{d}\sigma/\mathrm{d}\Omega$ over all solid angle,

$$\sigma = \int\limits_{\text{sphere}}^{4\pi} \frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\mathrm{d}\Omega . \tag{2.21}$$

Substituting (2.19) into (2.21) and integrating gives (2.17), as expected.

Special Characteristics of Coherent Scattering. Compare the differential scattering cross-sections for coherent x-ray scattering by a single electron at $\boldsymbol{r}_j$, $\mathrm{d}\sigma_{\mathrm{x1e},\boldsymbol{r}_j}/\mathrm{d}\Omega$, and an atom having Z electrons, $\mathrm{d}\sigma_{\mathrm{atom}}/\mathrm{d}\Omega$:

$$\frac{\mathrm{d}\sigma_{\mathrm{x1e},\boldsymbol{r}_j}}{\mathrm{d}\Omega}(\boldsymbol{k}_0, \boldsymbol{k}) = \left|f_{\mathrm{x1e},\boldsymbol{r}_j}(\boldsymbol{k}_0, \boldsymbol{k})\right|^2 , \tag{2.22}$$

$$\frac{\mathrm{d}\sigma_{\mathrm{atom}}}{\mathrm{d}\Omega}(\boldsymbol{k}_0, \boldsymbol{k}) = \left|f_{\mathrm{atom}}(\boldsymbol{k}_0, \boldsymbol{k})\right|^2 . \tag{2.23}$$

In coherent scattering we sum wave amplitudes (cf., (2.9)), so for coherent scattering we sum the scattering lengths of all Z electrons to obtain the scattering length of an atom:

$$f_{\mathrm{atom}}(\boldsymbol{k}_0, \boldsymbol{k}) = \sum_{\boldsymbol{r}_j}^{Z} f_{\mathrm{x1e},\boldsymbol{r}_j}(\boldsymbol{k}_0, \boldsymbol{k}) . \tag{2.24}$$

Note that (2.24) is a sum of the $f_{\mathrm{x1e},\boldsymbol{r}_j}$, but (2.23) is the square of this sum. Equation (2.23) can predict that the coherent x-ray scattering from an atom with Z electrons is Z^2 times stronger than for a single electron, and this proves to be true in the forward direction. The total cross-section for coherent scattering must increase linearly with the number of scatterers (here the number of electrons, Z). Consequently the coherent scattering is suppressed in other directions if a scaling with Z^2 is allowed in special directions. The angular distribution of coherent scattering must be different for the atom and for the single electron. That is, $f_{\mathrm{x1e}}(\boldsymbol{k}_0, \boldsymbol{k})$ and $f_{\mathrm{atom}}(\boldsymbol{k}_0, \boldsymbol{k})$ must have different shapes (they must depend differently on $\boldsymbol{k}_0$ and $\boldsymbol{k}$). The following is an inequality for coherent scattering (although its analog for incoherent scattering is an equality):

$$\frac{\mathrm{d}\sigma_{\mathrm{atom,coh}}}{\mathrm{d}\Omega}(\boldsymbol{k}_0, \boldsymbol{k}) \neq \sum_{\boldsymbol{r}_j}^{Z} \frac{\mathrm{d}\sigma_{\mathrm{x1e},\boldsymbol{r}_j,\mathrm{coh}}}{\mathrm{d}\Omega}(\boldsymbol{k}_0, \boldsymbol{k}) . \tag{2.25}$$

Integrating (2.25) gives an equality for coherent (and incoherent) scattering:

$$\int\limits_{\text{sphere}}^{4\pi} \frac{\mathrm{d}\sigma_{\mathrm{atom,coh}}}{\mathrm{d}\Omega}(\boldsymbol{k}_0, \boldsymbol{k})\mathrm{d}\Omega = \int\limits_{\text{sphere}}^{4\pi} \sum_{\boldsymbol{r}_j}^{Z} \frac{\mathrm{d}\sigma_{\mathrm{x1e},\boldsymbol{r}_j,\mathrm{coh}}}{\mathrm{d}\Omega}(\boldsymbol{k}_0, \boldsymbol{k})\mathrm{d}\Omega , \tag{2.26}$$

because with (2.21) we see that (2.26) equates the individual electron cross-sections to the total cross-section of the atom:

$$\sigma_{\text{atom,coh}} = Z\sigma_{\text{x1e,coh}} \; . \tag{2.27}$$

The process of actually performing the sum in (2.24) evidently requires delicacy in accounting for the phase relationships between the x-ray wavelets scattered into different angles, and knowledge about the electron density of the atom. This is the subject of atomic form factor calculations (see Chapter 3 in Fultz and Howe, for example, or (2.55)).

2.2 Born Approximation

Almost without a second thought, we treat neutron scattering as a wave phenomenon with the neutron wavefunction satisfying the Schrödinger wave equation. A neutron diffraction pattern, with its sharp peaks, is certainly evidence of wave behavior. The interpretation of the neutron wavefunction is different from that of a simple wave, however. Suppose we were to turn on an neutron beam and watch the formation of a diffraction pattern, using an area detector capable of displaying the impacts of individual neutrons. When the neutron beam is turned on, bright flashes are seen at points on the detector screen. Each individual event occurs at a particular point on the detector, and does not appear as a continuous ring. With time, an obvious bias appears, where the points of detection are most frequently at the positions of the rings and spots of the diffraction pattern. This behavior motivates the interpretation of the neutron wavefunction in terms of probabilities – specifically, the neutron probability is the neutron wavefunction times its complex conjugate (which gives a real number). Usually this probabilistic interpretation can be ignored when we consider a diffraction pattern from many neutrons, and we can consider neutron diffraction as the diffraction of any other type of wave. When individual neutron events are considered, however, we may have to recall the probabilistic interpretation of the neutron wavefunction because individual neutron detections look like particles rather than waves.

Another point to remember is that the wave behavior is a characteristic of an *individual* neutron. When considering a diffraction pattern involving multiple neutrons, we do not add the amplitudes of multiple wavefunctions. Neutrons are fermions, and do not form coherent states as in Bose condensation, for example. At the viewing screen, we add the intensities of individual neutrons. The interactions between different neutrons are not coherent.

Our picture of scattering begins with one neutron as a wave incident on an atom. This wave looks like a plane wave because it comes from a distant source. The wave interacts with the nucleus or magnetic electron cloud of the atom, and an outgoing wave is generated. This outgoing wave is something like a spherical wave originating at the atom, although its intensity is not

isotropic. Figure 2.5 shows the geometry, wavevectors and position vectors for our neutron scattering problem. Here both $\boldsymbol{r}$ and $\boldsymbol{r}'$ are large compared to the size of the scatterer. Our plane wave incident from the left, $\Psi_{\rm inc}$, is of the standard form:

$$\Psi_{\rm inc} = \mathrm{e}^{\mathrm{i}(\boldsymbol{k}_{\rm i}\cdot\boldsymbol{r}'-\omega t)} \; . \tag{2.28}$$

In what follows we neglect the time dependence to emphasize the manipulations of the spatial coordinates. We later recover the time-dependence by multiplying our results by $\mathrm{e}^{-\mathrm{i}\omega t}$. A spherical wave, $\Psi_{\rm sc}$, travels outwards from the center of scattering. The scattered wave has the form:

$$\Psi_{\rm sc} = f(\boldsymbol{k}_{\rm i},\boldsymbol{k}_{\rm f})\frac{\mathrm{e}^{\mathrm{i}k_{\rm f}|\boldsymbol{r}-\boldsymbol{r}'|}}{|\boldsymbol{r}-\boldsymbol{r}'|} \; , \tag{2.29}$$

where the scattering length $f(\boldsymbol{k}_{\rm i},\boldsymbol{k}_{\rm f})$ of Sect. 2.1.4 varies with the orientation of $\boldsymbol{k}_{\rm i}$ and $\boldsymbol{k}_{\rm f}$, $\boldsymbol{r}'$ is now used to locate the center of the scatterer, and the difference, $\boldsymbol{r}-\boldsymbol{r}'$, is the distance from the scatterer to the detector. The intensity of $\Psi_{\rm sc}$ falls off with distance as $1/r^2$, as we expect:

$$I_{\rm sc} = \Psi^*_{\rm sc}\Psi_{\rm sc} = \left|f(\boldsymbol{k}_{\rm i},\boldsymbol{k}_{\rm f})\right|^2 \frac{\mathrm{e}^{-\mathrm{i}k_{\rm f}|\boldsymbol{r}-\boldsymbol{r}'|}}{|\boldsymbol{r}-\boldsymbol{r}'|}\frac{\mathrm{e}^{\mathrm{i}k_{\rm f}|\boldsymbol{r}-\boldsymbol{r}'|}}{|\boldsymbol{r}-\boldsymbol{r}'|} \; , \tag{2.30}$$

$$I_{\rm sc} = \left|f(\boldsymbol{k}_{\rm i},\boldsymbol{k}_{\rm f})\right|^2 \frac{1}{|\boldsymbol{r}-\boldsymbol{r}'|^2} \; . \tag{2.31}$$

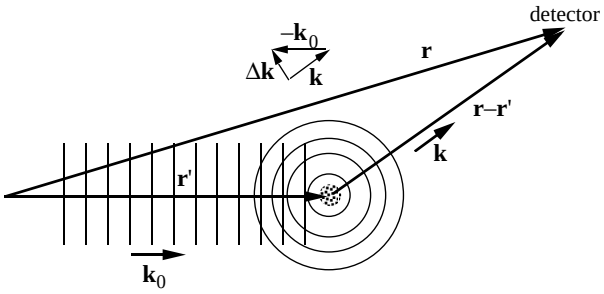


Fig. 2.5. Wavevectors and position vectors for neutron scattering.

2.2.1 Green's Function

To obtain the scattering length $f(\boldsymbol{k}_{\rm i},\boldsymbol{k}_{\rm f})$, we must solve the Schrödinger equation for the incident neutron inside the scattering atom (the mass of the neutron is m, and its coordinates in the atom are $\boldsymbol{r}'$):

$$-\frac{\hbar^2}{2m}\nabla^2\Psi(\boldsymbol{r}') + V(\boldsymbol{r}')\,\Psi(\boldsymbol{r}') = E\,\Psi(\boldsymbol{r}') \; , \tag{2.32}$$

$$\frac{\hbar^2}{2m}\nabla^2\Psi(\boldsymbol{r}') + E\,\Psi(\boldsymbol{r}') = V(\boldsymbol{r}')\,\Psi(\boldsymbol{r}') \; , \tag{2.33}$$

which we write as:

$$\left(\nabla^2 + k_{\rm i}^2\right) \Psi(\boldsymbol{r}') = U(\boldsymbol{r}')\, \Psi(\boldsymbol{r}') \ , \tag{2.34}$$

after having made the two definitions:

$$k_{\rm i}^2 \equiv \frac{2mE}{\hbar^2} \ , \tag{2.35}$$

$$U(\boldsymbol{r}') \equiv \frac{2mV(\boldsymbol{r}')}{\hbar^2} \ . \tag{2.36}$$

The formal approach to finding the solution of the Schrödinger equation in this problem makes use of Green's functions. A Green's function, $G(\boldsymbol{r}, \boldsymbol{r}')$, provides the response at $\boldsymbol{r}$ for a point scatterer at $\boldsymbol{r}'$:

$$\left(\nabla^2 + k_{\rm i}^2\right) G(\boldsymbol{r}, \boldsymbol{r}') = \delta(\boldsymbol{r}') \ . \tag{2.37}$$

We find the Green's function in a quick way by starting with an identity:

$$\nabla^2 \frac{{\rm e}^{{\rm i}kr}}{r} = {\rm e}^{{\rm i}kr} \nabla^2 \frac{1}{r} - k^2 \frac{{\rm e}^{{\rm i}kr}}{r} \ , \tag{2.38}$$

$$\left(\nabla^2 + k^2\right) \frac{{\rm e}^{{\rm i}kr}}{r} = {\rm e}^{{\rm i}kr} \nabla^2 \frac{1}{r} \ , \tag{2.39}$$

Recall that:

$$\nabla^2 \frac{1}{r} = -4\pi\delta(r) \ , \text{ so} \tag{2.40}$$

$$\left(\nabla^2 + k^2\right) \frac{{\rm e}^{{\rm i}kr}}{r} = -{\rm e}^{{\rm i}kr} 4\pi\delta(r) \ . \tag{2.41}$$

The right hand side simplifies because it equals zero everywhere except at $r = 0$, due to the nature of the δ-function. At $r = 0$, however, ${\rm e}^{{\rm i}kr} = 1$. From our identity (2.38) we therefore obtain:

$$\left(\nabla^2 + k^2\right) \frac{{\rm e}^{{\rm i}kr}}{r} = -4\pi\delta(r) \ . \tag{2.42}$$

We make a shift of the origin: $\boldsymbol{r} \to \boldsymbol{r} - \boldsymbol{r}'$ (so we can see more easily how the outgoing wave originates at the scatterer – see Fig. 2.5). After doing so, we can identify our Green's function by comparing (2.37) and (2.42):

$$G(\boldsymbol{r}, \boldsymbol{r}') = -\frac{1}{4\pi} \frac{{\rm e}^{{\rm i}k_{\rm f}|\boldsymbol{r}-\boldsymbol{r}'|}}{|\boldsymbol{r} - \boldsymbol{r}'|} \ . \tag{2.43}$$

With our Green's function in hand, we construct $\Psi_{\rm scatt}(\boldsymbol{r})$ by integrating. The idea is that to obtain the total wave amplitude at $\boldsymbol{r}$, we need to add up the spherical wavelet amplitudes emanating from all $\boldsymbol{r}'$ (each of form (2.43)), weighted by their strengths. This weight is the right-hand side of (2.34). Formally, the limits of integration cover all of space, but in fact it is only important to extend them over the $\boldsymbol{r}'$ where $U(\boldsymbol{r}')$ is non-zero (approximately the volume of the atom).

$$\Psi_{\rm sc}(\boldsymbol{r}) = \int U(\boldsymbol{r}')\,\Psi(\boldsymbol{r}')\,G(\boldsymbol{r},\boldsymbol{r}')\,{\rm d}^3\boldsymbol{r}' \ . \tag{2.44}$$

The total wave at $\boldsymbol{r}$, $\Psi(\boldsymbol{r})$, has both incident and scattered components:

$$\Psi = \Psi_{\rm inc} + \Psi_{\rm sc} \ , \tag{2.45}$$

$$\Psi(\boldsymbol{r}) = {\rm e}^{{\rm i}\boldsymbol{k}_{\rm i}\cdot\boldsymbol{r}} + \frac{2m}{\hbar^2}\int V(\boldsymbol{r}')\,\Psi(\boldsymbol{r}')\,G(\boldsymbol{r},\boldsymbol{r}')\,{\rm d}^3\boldsymbol{r}' \ . \tag{2.46}$$

2.2.2 First Born Approximation

Up to here our solution is exact. It is, in fact the Schrödinger equation itself, merely transformed from a differential equation to a integral equation appropriate for scattering problems. The problem with this integral equation (2.44) is that Ψ appears both inside and outside the integration, so an approximation is generally required to proceed further. The approximation that we use is the "first Born approximation." It amounts to using a plane wave, the incident plane wave, for Ψ in the integral:

$$\Psi(\boldsymbol{r}') \simeq {\rm e}^{{\rm i}\boldsymbol{k}_{\rm i}\cdot\boldsymbol{r}'} \ . \tag{2.47}$$

The first Born approximation assumes that the wave is undiminished and scattered only once by the material. This assumption is valid when the scattering is weak.

We simplify (2.43) by making the approximation that the detector is far from the scatterer. This allows us to work with plane waves at the detector, rather than outgoing spherical waves. To do so we align the outgoing wavevector $\boldsymbol{k}_{\rm f}$ along $(\boldsymbol{r}-\boldsymbol{r}')$ as shown in Fig. 2.5. The product of scalars, $k_{\rm f}\,|\boldsymbol{r}-\boldsymbol{r}'|$, in the exponential of a spherical wave emitted from $\boldsymbol{r}'$, is then equal to $\boldsymbol{k}_{\rm f}\cdot(\boldsymbol{r}-\boldsymbol{r}')$ of a plane wave,

$$G(\boldsymbol{r},\boldsymbol{r}') \simeq -\frac{1}{4\pi}\frac{{\rm e}^{i\boldsymbol{k}_{\rm f}\cdot(\boldsymbol{r}-\boldsymbol{r}')}}{|\boldsymbol{r}|} \ . \tag{2.48}$$

In (2.48) we also assumed that the origin is near the scatterer, so $|\boldsymbol{r}| \gg |\boldsymbol{r}'|$, simplifying the denominator of our Green's function.[3]

Returning to our exact integral equation (2.46), we obtain the approximate scattered wave (the first Born approximation for the scattered wave) by using (2.47) and (2.48) in (2.46):

$$\Psi(\boldsymbol{r}) \simeq {\rm e}^{{\rm i}\boldsymbol{k}_{\rm i}\cdot\boldsymbol{r}} - \frac{m}{2\pi\hbar^2}\int V(\boldsymbol{r}')\,{\rm e}^{{\rm i}\boldsymbol{k}_{\rm i}\cdot\boldsymbol{r}'}\frac{{\rm e}^{{\rm i}\boldsymbol{k}_{\rm f}\cdot(\boldsymbol{r}-\boldsymbol{r}')}}{|\boldsymbol{r}|}{\rm d}^3\boldsymbol{r}' \ , \tag{2.49}$$

$$\Psi(\boldsymbol{r}) = {\rm e}^{{\rm i}\boldsymbol{k}_{\rm i}\cdot\boldsymbol{r}} - \frac{m}{2\pi\hbar^2}\frac{{\rm e}^{{\rm i}\boldsymbol{k}_{\rm f}\cdot\boldsymbol{r}}}{|\boldsymbol{r}|}\int V(\boldsymbol{r}')\,{\rm e}^{{\rm i}(\boldsymbol{k}_{\rm i}-\boldsymbol{k}_{\rm f})\cdot\boldsymbol{r}'}{\rm d}^3\boldsymbol{r}' \ . \tag{2.50}$$

[3] If we neglect a constant prefactor, this assumption of $|\boldsymbol{r}-\boldsymbol{r}'| = |\boldsymbol{r}|$ is equivalent to assuming that the scatterer is small compared to the distance to the detector.

If we define:[4]

$$\boldsymbol{Q} \equiv \boldsymbol{k}_\mathrm{i} - \boldsymbol{k}_\mathrm{f} \ , \tag{2.51}$$

$$\Psi(\boldsymbol{r}) = \mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\boldsymbol{r}} - \frac{m}{2\pi\hbar^2}\frac{\mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{f}\cdot\boldsymbol{r}}}{|\boldsymbol{r}|}\int V(\boldsymbol{r}')\,\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}'}\mathrm{d}^3\boldsymbol{r}' \ . \tag{2.52}$$

The scattered part of the wave is:

$$\Psi_\mathrm{sc}(\boldsymbol{Q},\boldsymbol{r}) = \frac{\mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{f}\cdot\boldsymbol{r}}}{|\boldsymbol{r}|}f(\boldsymbol{Q}) \ , \text{ where :} \tag{2.53}$$

$$f(\boldsymbol{Q}) \equiv -\frac{m}{2\pi\hbar^2}\int V(\boldsymbol{r}')\,\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}'}\mathrm{d}^3\boldsymbol{r}' \ . \tag{2.54}$$

The factor $f(\boldsymbol{Q})$ is the scattering factor of (2.29), which we have found to depend on the incident and outgoing wavevectors only through their difference, $\boldsymbol{Q} \equiv \boldsymbol{k}_\mathrm{i} - \boldsymbol{k}_\mathrm{f}$. We recognize the integral of (2.54) as the Fourier transform of the potential seen by the incident neutron as it goes through the scatterer. In the first Born approximation:

The scattered wave is proportional to the
Fourier transform of the scattering potential.

The factor $f(\boldsymbol{Q})$ of (2.54) is given various names, depending on the potential $V(\boldsymbol{r})$ (we changed notation: $\boldsymbol{r}' \to \boldsymbol{r}$). When $V(\boldsymbol{r})$ is the potential of a single atom, $V_\mathrm{at}(\boldsymbol{r})$, we define $f_\mathrm{at}(\boldsymbol{Q})$ as the "atomic form factor":

$$f_\mathrm{at}(\boldsymbol{Q}) \equiv -\frac{m}{2\pi\hbar^2}\int V_\mathrm{at}(\boldsymbol{r})\,\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}}\mathrm{d}^3\boldsymbol{r} \ . \tag{2.55}$$

Alternatively, we can use the potential for the entire crystal for $V(\boldsymbol{r})$ in (2.54) (and develop the kinematical theory of diffraction). When $V(\boldsymbol{r})$ refers to the entire crystal, however, the first Born approximation of 2.52 is generally not reliable because multiple scattering will invalidate the assumption of (2.47). This assumption is, nevertheless, the basis for the "kinematical theory of diffraction," which we develop for its clarity and its qualitative successes. It is possible to transcend formally the single scattering approximation, and develop a "dynamical theory" of neutron diffraction by considering higher-order Born approximations, but this has not proved a particularly fruitful direction. Modern dynamical theories take a completely different path.

2.2.3 Higher-Order Born Approximations

Nevertheless, it is not difficult in principle to extend the Born approximation to higher orders. Instead of using an undiminished plane wave for $\Psi(\boldsymbol{r}')$, we could use a $\Psi(\boldsymbol{r}')$ that has been scattered once already. Equation (2.46) gives the second Born approximation if we use do not use the plane wave of (2.47) for $\Psi(\boldsymbol{r}')$, but rather:

[4] Sadly, the diffraction vector for elastic scattering is defined as $-\boldsymbol{Q} = \boldsymbol{\Delta k} \equiv \boldsymbol{k}_\mathrm{f} - \boldsymbol{k}_\mathrm{i}$

$$\Psi(\boldsymbol{r}') = \mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\boldsymbol{r}'} + \frac{2m}{\hbar^2}\int V(\boldsymbol{r}'')\,\Psi(\boldsymbol{r}'')\,G(\boldsymbol{r}',\boldsymbol{r}'')\,\mathrm{d}^3\boldsymbol{r}'' \;, \tag{2.56}$$

where we now use a plane wave for $\Psi(\boldsymbol{r}'')$:

$$\Psi(\boldsymbol{r}'') \simeq \mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\boldsymbol{r}''} \;. \tag{2.57}$$

The second Born approximation involves two centers of scattering. The first is at $\boldsymbol{r}''$ and the second is at $\boldsymbol{r}'$ (as shown in Fig. 2.6). The second and higher Born approximations are not used very frequently. If the scatterer is strong enough to violate the condition of weak scattering used in the first Born approximation, the scattering will also violate the assumptions of the second Born approximation.

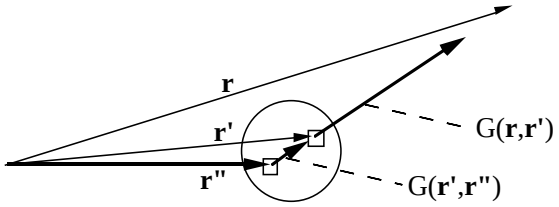


Fig. 2.6. Coordinates for the second Born approximation. The neutron path is shown as the dark arrows, which are labeled by the relevant Green's functions.

2.3 Essence of Coherent Inelastic Scattering

2.3.1 Spherical Waves from Point Scatterers

An intuitive shortcut from (2.37) to (2.44) is to regard $(\nabla^2 + \boldsymbol{k}_\mathrm{i}^2)$ as a scattering operator that generates a scattered wavelet proportional to $U(\boldsymbol{r}')\Psi(\boldsymbol{r}')$. The scattered wavelet must have the properties of (2.29) for its amplitude and phase versus distance. The scattered wavelet amplitude emitted from a small volume, $\mathrm{d}^3\boldsymbol{r}'$, centered about $\boldsymbol{r}'$ is:

$$\mathrm{d}\Psi_\mathrm{sc}(\boldsymbol{r},\boldsymbol{r}') = U(\boldsymbol{r}')\Psi(\boldsymbol{r}')\frac{\mathrm{e}^{\mathrm{i}(k_\mathrm{f}|\boldsymbol{r}-\boldsymbol{r}'|-\omega_0 t)}}{|\boldsymbol{r}-\boldsymbol{r}'|}\mathrm{d}^3\boldsymbol{r}' \;. \tag{2.58}$$

This is an expression for a spherical wave at $\boldsymbol{r}$ originating from the small volume $\mathrm{d}^3\boldsymbol{r}$ surrounding $\boldsymbol{r}'$. It is isotropic – note that the exponential factor $k_\mathrm{f}\,|\boldsymbol{r}-\boldsymbol{r}'|$ is a product of scalars.

In using the approach from the previous section with static potentials, $|\boldsymbol{k}_\mathrm{i}| = |\boldsymbol{k}_\mathrm{f}|$. This remains true in an average sense in the present section where we assign a time-dependence to $U(\boldsymbol{r}')$. Notice from (2.58) that the amplitude of the scattered wave is modulated by $U(\boldsymbol{r}')$. In the present section, we see

that the Fourier component of $U(\boldsymbol{r}')$ with frequency ω gives a frequency modulation to the scattered wave, whose frequency of ω_0 is changed to $\omega_0 \pm \omega$.

We obtain the total scattered wave by integrating around all volume of the scatterer. The incident plane wave, $\propto \mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\boldsymbol{r}'}$ (2.47), helps sets the phase of the scattering at each volume interval. The phase of the outgoing wave also depends on the orientation of the outgoing $\boldsymbol{k}_\mathrm{f}$ with respect to the position of the scattering point, $\boldsymbol{r}'$. The *relative* phase from each scattering point depends on the change in wavevector, $\boldsymbol{Q} \equiv \boldsymbol{k}_\mathrm{i} - \boldsymbol{k}_\mathrm{f}$, as $\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}'}$.

$$\Psi_\mathrm{sc}(\boldsymbol{Q},\boldsymbol{r}) = -\frac{\mathrm{e}^{\mathrm{i}(\boldsymbol{k}_\mathrm{f}\cdot\boldsymbol{r}-\omega_0 t)}}{|\boldsymbol{r}|}\frac{m}{2\pi\hbar^2}\int V(\boldsymbol{r}')\,\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}'}\,\mathrm{d}^3\boldsymbol{r}' \ . \tag{2.59}$$

In arriving at (2.58) we have repeated, in an intuitive way, the steps to (2.53) and (2.54).

The trick now is to replace the potential, $V(\boldsymbol{r})$, with a suitable potential for neutron scattering. For nuclear scattering, relevant to phonon measurements, we use the "Fermi pseudopotential," which places all the potential at a point nucleus:

$$V_\mathrm{nuc}(\boldsymbol{r}) = 4\pi\frac{\hbar^2}{2m}\,b\,\delta(\boldsymbol{r}) \ , \tag{2.60}$$

where b is a simple constant (perhaps a complex number). For thermal neutrons, the δ-function is an appropriate description of the shape of a nucleus.[5] We place independent Fermi pseudopotentials at the set of positions, $\{\boldsymbol{R}_j\}$, of all atomic nuclei in the crystal:

$$V(\boldsymbol{r}) = 4\pi\frac{\hbar^2}{2m}\sum_j b_j\,\delta(\boldsymbol{r}-\boldsymbol{R}_j) \ . \tag{2.61}$$

2.3.2 Time-Varying Potentials

In the key step that leads to inelastic scattering, we allow the nuclei to move around as the atoms vibrate. Our potential is:

$$V(\boldsymbol{r}) = 4\pi\frac{\hbar^2}{2m}\sum_j b_j\,\delta(\boldsymbol{r}-\boldsymbol{R}_j(t)) \ , \tag{2.62}$$

Substituting (2.62) into (2.59), we note the elegant cancellation of prefactors:

$$\Psi_\mathrm{sc}(\boldsymbol{Q},\boldsymbol{r},t) = -\frac{\mathrm{e}^{\mathrm{i}(\boldsymbol{k}_\mathrm{f}\cdot\boldsymbol{r}-\omega_0 t)}}{|\boldsymbol{r}|}\int\sum_j b_j\,\delta(\boldsymbol{r}'-\boldsymbol{R}_j(t))\,\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}'}\,\mathrm{d}^3\boldsymbol{r}' \ . \tag{2.63}$$

The integration over the δ-functions of (2.63) fixes the exponentials at the nuclear positions $\{\boldsymbol{R}_j(t)\}$:

[5] For magnetic scattering, however, the δ-function should be convoluted with a real-space form factor for the magnetic electrons. This could be done at the end of the calculation by multiplying the k-space result with a form factor.

$$\Psi_{\rm sc}(\boldsymbol{Q},\boldsymbol{r},t) = -\frac{{\rm e}^{{\rm i}(\boldsymbol{k}_{\rm f}\cdot\boldsymbol{r}-\omega_0 t)}}{|\boldsymbol{r}|}\sum_j b_j\,{\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{R}_j(t)}\ . \tag{2.64}$$

For isolating the phonon scattering, we separate the static and dynamic parts of the nuclear positions. For completeness we do so for each atom in each unit cell (unit cell indices are $\{l,\kappa\}$ for {lattice, basis}, each requiring its own sum):

$$\boldsymbol{R}_j(t) = \boldsymbol{x}_{l,\kappa} + \boldsymbol{u}_{l,\kappa}(t)\ , \tag{2.65}$$

transforming (2.64):

$$\Psi_{\rm sc}(\boldsymbol{Q},\boldsymbol{r},t) = -\frac{{\rm e}^{{\rm i}(\boldsymbol{k}_{\rm f}\cdot\boldsymbol{r}-\omega_0 t)}}{|\boldsymbol{r}|}\sum_{l,\kappa} b_{l,\kappa}\,{\rm e}^{{\rm i}\boldsymbol{Q}\cdot(\boldsymbol{x}_{l,\kappa}+\boldsymbol{u}_{l,\kappa}(t))}\ . \tag{2.66}$$

When $\boldsymbol{Q}\cdot\boldsymbol{u}$ is small, we can expand the exponential in (2.66) to obtain:

$$\Psi_{\rm sc}(\boldsymbol{Q},\boldsymbol{r},t) = -\frac{{\rm e}^{{\rm i}(\boldsymbol{k}_{\rm f}\cdot\boldsymbol{r}-\omega_0 t)}}{|\boldsymbol{r}|} \times \sum_{l,\kappa} b_\kappa\,{\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{x}_{l,\kappa}}\left(1+{\rm i}\boldsymbol{Q}\cdot\boldsymbol{u}_{l,\kappa}(t) - \frac{1}{2}(\boldsymbol{Q}\cdot\boldsymbol{u}_{l,\kappa}(t))^2+\ldots\right) \tag{2.67}$$

The terms in the large parentheses in (2.67) account for the elastic scattering, one-phonon scattering, two-phonon scattering, ..., as we next show.

2.3.3 Elastic Neutron Scattering

First neglect the time-dependence of the scattering potential, which restricts us to the case of elastic scattering. Without dynamics, we note that b_κ depends only on κ, not l, since all unit cells have identical atom arrangements. The first term in parentheses in (2.67), the 1, gives the familiar coherent elastic scattering, $\Psi^{\rm el}_{\rm sc}(\boldsymbol{Q},\boldsymbol{r})$, from the static part of the structure:

$$\Psi^{\rm el}_{\rm sc}(\boldsymbol{Q},\boldsymbol{r}) = -\frac{{\rm e}^{{\rm i}(\boldsymbol{k}_{\rm f}\cdot\boldsymbol{r}-\omega_0 t)}}{|\boldsymbol{r}|}\sum_{l,\kappa} b_\kappa\,{\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{x}_{l,\kappa}}\ . \tag{2.68}$$

By writing $\boldsymbol{x}_{l,\kappa}$ as a sum of a lattice vector $\boldsymbol{x}_l$ plus a basis vector $\boldsymbol{x}_\kappa$:

$$\boldsymbol{x}_{l,\kappa} = \boldsymbol{x}_l + \boldsymbol{x}_\kappa\ , \tag{2.69}$$

and neglecting the factor for the outgoing spherical wave:

$$\Psi^{\rm el}_{\rm sc}(\boldsymbol{Q},\boldsymbol{r}) = \sum_{\boldsymbol{x}_\kappa} b_\kappa\,{\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{x}_\kappa}\sum_{\boldsymbol{x}_l}{\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{x}_l} \equiv \mathcal{F}(\boldsymbol{Q})\,\mathcal{S}(\boldsymbol{Q})\ . \tag{2.70}$$

Here we separated the sums over basis and lattice vectors into a structure factor, $\mathcal{F}(\boldsymbol{Q})$, and a shape factor, $\mathcal{S}(\boldsymbol{Q})$. We could use (2.70) (with a Debye-Waller factor) to calculate the neutron diffraction pattern from a crystal. Equation (2.70) is central to neutron diffractometry of materials.

2.3.4 Phonon Scattering

The next term in parentheses in (2.67), the $\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{u}_{l,\kappa}(t)$, is new. It gives the inelastic scattered wave, $\Psi_{\mathrm{sc}}^{\mathrm{inel}}(\boldsymbol{Q},\boldsymbol{r},t)$. To calculate it, we use the phonon solution, $\boldsymbol{u}_{l,\kappa}(\omega(\boldsymbol{q}),t)$, for a particular $\omega(\boldsymbol{q})$:

$$\boldsymbol{u}_{l,\kappa}(\omega(\boldsymbol{q}),t)=\frac{\boldsymbol{U}_{\kappa}(\boldsymbol{q})}{\sqrt{2M_{\kappa}\omega(\boldsymbol{q})}}\,\mathrm{e}^{\mathrm{i}(\boldsymbol{q}\cdot\boldsymbol{x}_l-\omega(\boldsymbol{q})t)}\ . \tag{2.71}$$

The phase factor, $\mathrm{e}^{\mathrm{i}\boldsymbol{q}\cdot\boldsymbol{x}_l}$, provides all the long-range spatial modulation of $\boldsymbol{u}_{l,\kappa}(\boldsymbol{q},t)$. The dependence on $\boldsymbol{\kappa}$, a short-range basis vector index, is taken out of the phase factor and placed in the complex constant $\boldsymbol{U}_{\kappa}(\boldsymbol{q})$.

This is the source of the denominator in (2.71). It is convenient for the $\boldsymbol{U}_{\kappa}(\boldsymbol{q})$ of (2.71) to have modulus unity, as does the exponential. Nevertheless, the $\boldsymbol{u}_{l,\kappa}(\omega(\boldsymbol{q}),t)$ must have a maximum displacement, $\mathcal{X}$, which gives the total energy of the harmonic oscillator. Since this total energy must equal $\hbar\omega$ for each additional phonon excitation:

$$\hbar\omega=\frac{1}{2}M\omega^2\mathcal{X}^2\ , \tag{2.72}$$

$$\mathcal{X}=\sqrt{\frac{2\hbar}{M\omega}}\ . \tag{2.73}$$

Normalizing the amplitude in (2.71) by $\sqrt{M\omega}$ ensures this consistency between amplitude and energy, while allowing $\boldsymbol{U}_{\kappa}(\boldsymbol{q})$ to have modulus unity. Details such as the factor of 2 require a more complete quantum mechanical development.

After substitution of (2.71), the second term in (2.67) gives an inelastically-scattered wave:

$$\begin{aligned}\Psi_{\mathrm{sc}}^{\mathrm{inel}}(\boldsymbol{Q},\boldsymbol{r},t)&=-\frac{\mathrm{i}\,\mathrm{e}^{\mathrm{i}(\boldsymbol{k}_{\mathrm{f}}\cdot\boldsymbol{r}-\omega_0 t)}}{|\boldsymbol{r}|}\\&\quad\times\sum_{l,\kappa}\frac{b_{\kappa}}{\sqrt{2M_{\kappa}\omega(\boldsymbol{q})}}\Big(\boldsymbol{Q}\cdot\boldsymbol{U}_{\kappa}(\boldsymbol{q})\Big)\,\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l}\,\mathrm{e}^{\mathrm{i}(\boldsymbol{q}\cdot\boldsymbol{x}_l-\omega(\boldsymbol{q})t)}\ ,\end{aligned} \tag{2.74}$$

and making a parallel decomposition into lattice plus basis vectors as in (2.69):

$$\begin{aligned}\Psi_{\mathrm{sc}}^{\mathrm{inel}}(\boldsymbol{Q},\boldsymbol{r},t)&=-\frac{\mathrm{i}\,\mathrm{e}^{\mathrm{i}\boldsymbol{k}_{\mathrm{f}}\cdot\boldsymbol{r}}}{|\boldsymbol{r}|}\,\mathrm{e}^{-\mathrm{i}(\omega_0+\omega)t}\\&\quad\times\sum_{\boldsymbol{x}_{\kappa}}\frac{b_{\kappa}}{\sqrt{2M_{\kappa}\omega(\boldsymbol{q})}}\Big(\boldsymbol{Q}\cdot\boldsymbol{U}_{l,\kappa}(\boldsymbol{q})\Big)\,\mathrm{e}^{\mathrm{i}(\boldsymbol{q}+\boldsymbol{Q})\cdot\boldsymbol{x}_{\kappa}}\sum_{\boldsymbol{x}_l}\mathrm{e}^{\mathrm{i}(\boldsymbol{q}+\boldsymbol{Q})\cdot\boldsymbol{x}_l}\end{aligned} \tag{2.75}$$

We recognize the last sum (over $\boldsymbol{x}_l$) as a sum over phase factors that generate a diffracted wave from a simple lattice. It will produce a series of Bragg peaks at reciprocal lattice vectors:

$$\sum_{\boldsymbol{x}_l}\mathrm{e}^{\mathrm{i}(\boldsymbol{q}+\boldsymbol{Q})\cdot\boldsymbol{x}_l}=\frac{(2\pi)^3}{V_0}\sum_{\boldsymbol{g}}\delta\big((\boldsymbol{q}+\boldsymbol{Q})-\boldsymbol{g}\big)\ . \tag{2.76}$$

Note that this is the momentum conservation condition: $(\boldsymbol{q}+\boldsymbol{Q})\cdot\boldsymbol{r}_l = 2\pi\textit{integer}$. Momentum conservation is seen directly for coherent inelastic scattering – the scattering wavevector, $\boldsymbol{Q}$, must equal the (negative of the) phonon vector $\boldsymbol{q}$ plus a reciprocal lattice vector $\boldsymbol{g}$. The prefactor in (2.76) is the volume of a unit cell in the reciprocal lattice, where V_0 is the volume of the unit cell of the crystal. Substituting into (2.75):

$$\Psi_{\rm sc}^{\rm inel}(\boldsymbol{Q},\boldsymbol{r},t) = -\frac{\mathrm{i}\,\mathrm{e}^{\mathrm{i}\boldsymbol{k}_{\rm f}\cdot\boldsymbol{r}}}{|\boldsymbol{r}|}\,\mathrm{e}^{-\mathrm{i}(\omega_0+\omega)t}\,\frac{(2\pi)^3}{V_0} \times\sum_{\boldsymbol{x}_\kappa}\frac{b_\kappa}{\sqrt{2M_\kappa\omega(\boldsymbol{q})}}\Big(\boldsymbol{Q}\cdot\boldsymbol{U}_{l,\kappa}(\boldsymbol{q})\Big)\,\mathrm{e}^{\mathrm{i}(\boldsymbol{q}+\boldsymbol{Q})\cdot\boldsymbol{x}_\kappa}\,\delta\big((\boldsymbol{q}+\boldsymbol{Q})-\boldsymbol{g}\big)\ . \quad (2.77)$$

The scattered wave, $\Psi_{\rm sc}^{\rm inel}(\boldsymbol{Q},\boldsymbol{r},t)$, has the temporal frequency $\omega_0+\omega$, differing from the frequency ω_0 of the incident wave. The inelastically-scattered neutron therefore differs in energy by the amount $\hbar\omega$ from the incident neutron. This energy conservation condition can also be written as a δ-function, $\delta(\omega-\omega(\boldsymbol{q}))$.

In what follows, the intensity, $I(\boldsymbol{Q},E)$, will be obtained from $\Psi_{\rm sc}^{*}(\boldsymbol{Q},\boldsymbol{r},t)\times\Psi_{\rm sc}(\boldsymbol{Q},\boldsymbol{r}',t')$. The intensity depends on the four positions and times, $\{\boldsymbol{r}',t',\boldsymbol{r},t\}$, but not on all four independently. Instead, the intensity depends on correlations between them. Section 3.1 shows that the Van Hove correlation function in space and time expresses what we need to know to calculate the inelastic intensity, $I(\boldsymbol{Q},E)$.

2.3.5 Intensity from Wave Amplitude

The next step is to get the double-differential scattering cross-section for coherent inelastic scattering, $\mathrm{d}^2\sigma/\mathrm{d}\Omega\,\mathrm{d}E$. With reference to Fig. 2.4, this quantity assigns an energy spectrum to each cone of beam filling the range in solid angle $\mathrm{d}\Omega$ (to the right of the scatterer). The intensity at each piece of solid angles and in each energy range is the maximum information that we could expect from experimental data. We obtain $\mathrm{d}^2\sigma/\mathrm{d}\Omega\,\mathrm{d}E$ by calculating $|\Psi_{\rm sc}^{\rm inel}|^2$ of (2.77) by taking $|\Psi_{\rm sc}|^2$, and ignoring the intensity factor of r^{-2} (cf. (2.20), (2.31)). A question that might arise is if there could be cross-terms between $\Psi_{\rm sc}^{\rm inel}$ and $\Psi_{\rm sc}^{\rm el}$. If so, it might be necessary to calculate the total scattered wave from (2.66), rather than (2.77), a greater effort since there will be twice as many sums that will have to be reduced by taking differences between different atom positions at different times. These cross-terms are zero, however, and this can be understood by a physical argument. As described after (2.77), the frequencies of the incident and scattered waves are mismatched. Integrated over many periods of the neutron wavefunction, these mismatched waves will not add constructively or destructively. There is no coherent interaction between $\Psi_{\rm sc}^{\rm inel}$ and $\Psi_{\rm sc}^{\rm el}$ that requires calculation. Incidentally, such interactions if they were to exist, would be quantum beats

of the wavetrains, but we ignore them in the present analysis because we do not have sufficient time resolution at the detector to sense them.

The only subtlety in calculating $|\Psi_{\rm sc}^{\rm inel}(\boldsymbol{Q},\boldsymbol{r})|^2$ is handling the δ-functions. The square of a δ-function is a δ-function, $\delta^2(x) = \delta(x)$. Alternatively, one can think of $\delta(\boldsymbol{Q}+\boldsymbol{q}\pm\boldsymbol{g})$ and $\delta(\omega-\omega(\boldsymbol{q}))$ as enforcing the conservation of momentum and conservation of energy on the neutron scattering process, and these conditions must hold for both the neutron wavefunction amplitude and its probability distribution. To keep an appropriate density of δ-functions in k-space requires that we keep the same factor $(2\pi)^3/2V_0$. For coherent inelastic phonon scattering:

$$\begin{aligned}\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_{\rm f}}{k_{\rm i}}\frac{(2\pi)^3}{2V_0}\sum_{\boldsymbol{g},\boldsymbol{q}}\frac{n(\boldsymbol{q})+\frac{1}{2}\pm\frac{1}{2}}{\omega(\boldsymbol{q})} \\ \times\Big|\sum_{\kappa}\frac{b_\kappa}{\sqrt{M_\kappa}}\,\mathrm{e}^{-W_\kappa(\boldsymbol{Q})}\,\big[\boldsymbol{Q}\cdot\boldsymbol{U}_{l,\kappa}(\boldsymbol{q})\big]\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_\kappa}\Big|^2 \\ \times\,\delta(\boldsymbol{Q}+\boldsymbol{q}\pm\boldsymbol{g})\,\delta(\omega-\omega(\boldsymbol{q}))\;, \end{aligned} \tag{2.78}$$

where the phonon occupancy factor, $n(\boldsymbol{q})$, is the Bose-Einstein distribution:

$$n(\boldsymbol{q}) = \frac{1}{\mathrm{e}^{\hbar\omega(\boldsymbol{q})/k_{\rm B}T}-1}\;. \tag{2.79}$$

The signs in the factor $n(\boldsymbol{q})+\frac{1}{2}\pm\frac{1}{2}$ in (2.78) are such that it is $n(\boldsymbol{q})+1$ for phonon creation, and $n(\boldsymbol{q})$ for phonon annihilation – it is always possible to create a phonon, even at $T=0$ when no phonon excitations are present. A new factor in (2.78) is the ratio $k_{\rm f}/k_{\rm i}$, which expresses the effect on flux caused by the rate at which neutrons leave the sample. Compared to an elastically-scattered neutron, fewer neutrons per second will pass into a solid angle if they are slowed to smaller values of $k_{\rm f}$. Another new factor is the Debye–Waller factor, $\mathrm{e}^{-W_\kappa(\boldsymbol{Q})}$. Finally, we note that the phonons on different branches must be considered separately in (2.78), and it is traditional to add a "branch index," sometimes denoted γ, to $\omega(\boldsymbol{q})$, since more than one ω_γ may correspond to a specific $\boldsymbol{q}$.

2.4 Correlation Function for Elastic Scattering – The Patterson Function

2.4.1 Overview

In much of Chapter 2, scattering theory has been developed by calculating the amplitude of the wave scattered from crystals with excitations or disorder. The amplitude of the diffracted wave, ψ, is the sum of phase factors of wavelets emitted from individual atoms. For elastic scattering, which we consider presently, the phase information in $\psi(\boldsymbol{Q})$ includes details of atom

positions, which can be obtained by inverse Fourier transformation, $F^{-1}\psi$. We then calculate the intensity $I(\boldsymbol{Q}) = \psi^*\psi$.

This Sect. 2.4 takes a different approach of calculating directly the diffracted intensity $I(\boldsymbol{Q})$, rather than calculating it as $\psi^*\psi$. In this new approach, the real space information is obtained with the Fourier inversion $F^{-1}I$, rather than $F^{-1}\psi$, but this sacrifices some information about atom positions. Nevertheless, the intensity is the actual quantity measured in a diffraction experiment, so this new approach offers a more rigorous understanding of what structural information is available from diffraction experiments. Furthermore, in cases of severely disordered materials, there may be no obvious way to obtain the atom positions needed for a calculation of $\psi(\boldsymbol{Q})$. For problems involving severe structural disorder, another advantage of direct manipulations of $I(\boldsymbol{Q})$ is that a convenient reference state proves to be a homogeneous distribution of scatterers, or uncorrelated scatterers as in an ideal gas. A powerful tool for calculating diffraction intensities from such materials (and regular crystals too) is the "Patterson function," defined in Sect. 2.4.3 as an autocorrelation function of the scattering factor distribution.

Whereas the diffracted wave, $\psi(\boldsymbol{Q})$, is the Fourier transform of the scattering factor distribution, the diffracted intensity, $I(\boldsymbol{Q})$, is the Fourier transform of the Patterson function of the scattering factor distribution.

The Patterson function is a function in real space, with argument $\boldsymbol{r}$. The Patterson function is a convolution, so the reader should be familiar with convolutions and the convolution theorem (Sect. A.1) before reading the present chapter. The presentation here of real-space correlation functions is good preparation for the discussion that follows on space-time correlation functions. We begin by proving the emphasized statement above. The subsequent section uses the Patterson function to explain diffraction phenomena involving displacements of atoms off of periodic positions owing to temperature.

2.4.2 Atom Centers at Points in Space

The most important results in this chapter are obtained by assuming the scatterers are points. At each point, $\boldsymbol{r}_j$, resides the scattering strength of one entire atom, $f_{\boldsymbol{r}_j}$ (or one unit cell). The actual shape of the atom is included later by convolution, and does not change the main results obtained with point atoms.

It proves convenient to consider a distribution of scatterers, $f(\boldsymbol{r})$, with a continuous variable, $\boldsymbol{r}$, rather than a sum over discrete points, $\{\boldsymbol{r}_j\}$. We change variables as:

$$\psi(\boldsymbol{Q}) = \sum_{\boldsymbol{r}_j}^{N} f_{\boldsymbol{r}_j} \mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}_j} = \int_{-\infty}^{\infty} f(\boldsymbol{r})\, \mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}} \mathrm{d}^3\boldsymbol{r}\ . \tag{2.80}$$

To equate a continuous integral to a discrete sum requires that $f(\boldsymbol{r})$ is not a smooth function of position. Over most of space $f(\boldsymbol{r})$ is zero, but at atom

centers such as $\boldsymbol{r} = \boldsymbol{r}_i$, $f(\boldsymbol{r}_i)$ is a Dirac delta function times a constant, $f_{\boldsymbol{r}_i}$:

$$f(\boldsymbol{r}_i) = f_{\boldsymbol{r}_i}\delta(\boldsymbol{r} - \boldsymbol{r}_i) \; . \tag{2.81}$$

Recall the important property of the Dirac delta function:

$$y(x') = \int\limits_{-\infty}^{\infty} \delta(x - x')\, y(x)\, \mathrm{d}x \; . \tag{2.82}$$

Equation (2.82) requires that $\delta(x-x')$ is zero everywhere, except at the point $x = x'$. At this point the delta function is infinitely high, but of unit area, so the integral of (2.82) picks out only the value of $y(x)$ at x'. To extend (2.81) to include many atom centers, we take the sum over $\boldsymbol{r}_j$:

$$f(\boldsymbol{r}) = \sum_{\boldsymbol{r}_j}^{N} f_{\boldsymbol{r}_j}\delta(\boldsymbol{r} - \boldsymbol{r}_j) \; , \tag{2.83}$$

so we satisfy the equality in (2.80) between points in space, $\{\boldsymbol{r}_j\}$, and a continuous function of $\boldsymbol{r}$. We include the shape of the atomic form factor, $f_{\mathrm{at}}(\boldsymbol{r})$, in Sect. 2.4.5.

2.4.3 Definition of the Patterson Function

We define the "Patterson function," $P(\boldsymbol{r})$:

$$P(\boldsymbol{r}) \equiv \int\limits_{-\infty}^{\infty} f^*(\boldsymbol{r}')f(\boldsymbol{r} + \boldsymbol{r}')\, \mathrm{d}^3\boldsymbol{r}' \; . \tag{2.84}$$

Equation (2.84) is a convolution. Since the function $f(\boldsymbol{r})$ is not inverted in the usual way for a convolution, we write:

$$P(\boldsymbol{r}) = f^*(\boldsymbol{r}) * f(-\boldsymbol{r}) \; , \tag{2.85}$$

This is a specific type of convolution known as an "autocorrelation function," sometimes denoted with a special symbol:

$$P(\boldsymbol{r}) = f(\boldsymbol{r}) \circledast f(\boldsymbol{r}) \; . \tag{2.86}$$

The most important feature of the Patterson function is that its Fourier transform is the diffracted intensity in kinematical theory. To show this, we use (2.80) to write $I(\boldsymbol{Q}) = \psi^*\psi$ as:

$$I(\boldsymbol{Q}) = \int\limits_{-\infty}^{\infty} f^*(\boldsymbol{r}')\, \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}'}\mathrm{d}^3\boldsymbol{r}' \int\limits_{-\infty}^{\infty} f(\boldsymbol{r}'')\, \mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}''}\mathrm{d}^3\boldsymbol{r}'' \; . \tag{2.87}$$

Since $\boldsymbol{r}'$ and $\boldsymbol{r}''$ are independent variables:

$$I(\boldsymbol{Q}) = \int\limits_{-\infty}^{\infty} \Big(\int\limits_{-\infty}^{\infty} f^*(\boldsymbol{r}')f(\boldsymbol{r}'')\, \mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot(\boldsymbol{r}''-\boldsymbol{r}')}\mathrm{d}^3\boldsymbol{r}'' \Big)\, \mathrm{d}^3\boldsymbol{r}' \; . \tag{2.88}$$

Define $\boldsymbol{r} \equiv \boldsymbol{r}'' - \boldsymbol{r}'$, and change variables $\boldsymbol{r}'' \to \boldsymbol{r} + \boldsymbol{r}'$. In so doing, the limits of integration for $\boldsymbol{r}$ are shifted by $-\boldsymbol{r}'$, but this is not of concern for integrations performed over all of space:

$$I(\boldsymbol{Q}) = \int_{-\infty}^{\infty} \Big(\int_{-\infty}^{\infty} f^*(\boldsymbol{r}') f(\boldsymbol{r} + \boldsymbol{r}') \, \mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}} \mathrm{d}^3\boldsymbol{r} \Big) \, \mathrm{d}^3\boldsymbol{r}' \; , \tag{2.89}$$

$$I(\boldsymbol{Q}) = \int_{-\infty}^{\infty} \Big(\int_{-\infty}^{\infty} f^*(\boldsymbol{r}') f(\boldsymbol{r} + \boldsymbol{r}') \mathrm{d}^3\boldsymbol{r}' \Big) \mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}} \mathrm{d}^3\boldsymbol{r} \; . \tag{2.90}$$

Using the definition of (2.84), we rewrite (2.90):

$$I(\boldsymbol{Q}) = \int_{-\infty}^{\infty} P(\boldsymbol{r}) \, \mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}} \mathrm{d}^3\boldsymbol{r} \; . \tag{2.91}$$

Equation (2.91) shows that the diffracted intensity is the Fourier transform of the Patterson function:

$$I(\boldsymbol{Q}) = FP(\boldsymbol{r}) \; , \tag{2.92}$$

and by the inverse transformation we must have:

$$P(\boldsymbol{r}) = F^{-1} I(\boldsymbol{Q}) \; . \tag{2.93}$$

For comparison, the diffracted wave, $\psi(\boldsymbol{Q})$ of (2.80), is the Fourier transform of the scattering factor distribution, $f(\boldsymbol{r})$. We therefore have another relationship between $I(\boldsymbol{Q})$ and $f(\boldsymbol{r})$:

$$I(\boldsymbol{Q}) = \psi^*(\boldsymbol{Q}) \, \psi(\boldsymbol{Q}) \; , \tag{2.94}$$

$$I(\boldsymbol{Q}) = \big(Ff(\boldsymbol{r})\big)^* \; Ff(\boldsymbol{r}) = \big|Ff(\boldsymbol{r})\big|^2 \; . \tag{2.95}$$

Comparing (2.92) and (2.95):

$$FP(\boldsymbol{r}) = \big|Ff(\boldsymbol{r})\big|^2 \; . \tag{2.96}$$

Equation (2.96) is consistent with the convolution theorem of Sect. A.1 – a convolution in real space (the Patterson function of (2.84)) corresponds to a multiplication in Fourier space (right-hand side of (2.96)). Note how (2.95) shows the effects of the flip and the complex conjugation of $f(\boldsymbol{r})$ in the convolution of (2.84):

$$F\big[f^*(\boldsymbol{r}) * f(-\boldsymbol{r})\big] = \big(Ff(\boldsymbol{r})\big)^* \; Ff(\boldsymbol{r}) = \big|f(\boldsymbol{Q})\big|^2 \; , \tag{2.97}$$

as compared to:

$$F\big[f(\boldsymbol{r}) * f(\boldsymbol{r})\big] = Ff(\boldsymbol{r}) \; Ff(\boldsymbol{r}) = \big(f(\boldsymbol{Q})\big)^2 \; . \tag{2.98}$$

2.4.4 Properties of Patterson Functions

It is instructive to illustrate the steps in constructing a Patterson function (2.84). The steps in any convolution are shift, multiply, and integrate, and are shown in Fig. 2.7. Figure 2.7a shows the overlap of a function shifted by the distance $\boldsymbol{r}$ against the original position shown as a dashed curve. To obtain the Patterson function in Fig. 2.7b, at each shift the function was multiplied by its shifted counterpart, then integrated.

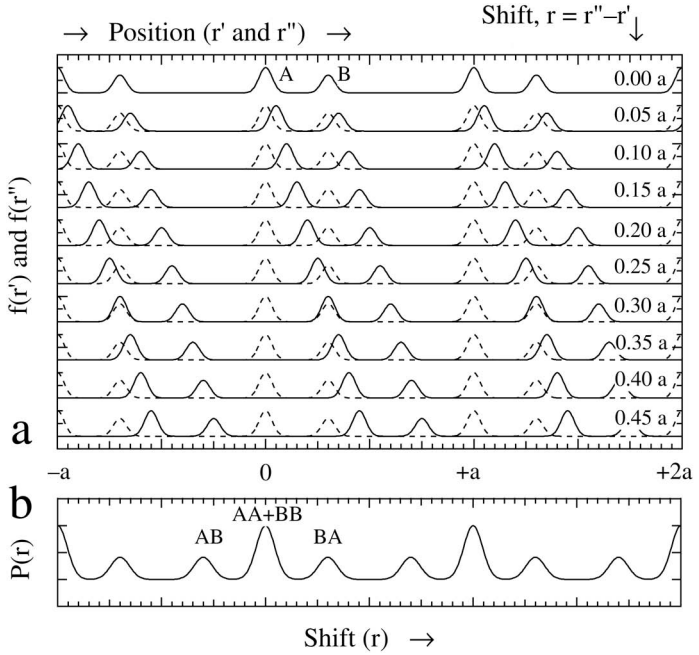


Fig. 2.7. (**a**) Shifts of a function of period, a, with respect to itself. The shift, $r = r'' - r'$, is labeled at right in units of a. (**b**) The Patterson function, obtained by integrating the product of the solid and dashed curves for all shifts, r.

Note that the peaks of the Patterson function in Fig. 2.7b are broader than the peaks in the scattering factor distribution of Fig. 2.7a. Since the peaks in Fig. 2.7a are Gaussian functions of equal width, the peaks in the Patterson function are broadened by a factor of $\sqrt{2}$. Second, the periodicity of the Patterson function is one lattice constant, a. This is expected, since the overlap of the peaks in the function of Fig. 2.7a is maximized each time the shift equals an integral number of lattice constants. The intensities of these primary maxima are proportional to $A^2 + B^2$. There are secondary maxima that occur at shifts of ± 0.3 when the large peak overlaps the small peak. The intensities of these secondary maxima are proportional to AB. The Patterson

function has a peak at each distance corresponding to a separation between the peaks in Fig. 2.7a.

The Patterson function, $P(\boldsymbol{r})$ of Fig. 2.7b, has a higher symmetry than the $f(\boldsymbol{r})$ of Fig. 2.7a. Identical secondary peaks occur in $P(\boldsymbol{r})$ when the large peak is shifted to the right by $+0.3a$ and overlaps the small peak, or when the small peak is shifted to the left by $-0.3a$ and overlaps the large peak. For this reason, even when $f(\boldsymbol{r})$ has no center of inversion, $P(\boldsymbol{r})$ has inversion symmetry. The Patterson function is unchanged if the original function is inverted.[6] Equation (2.93) shows that the measured x-ray diffraction intensity provides the Patterson function, not the scattering factor distribution. We therefore have "Friedel's law":

Diffraction experiments cannot distinguish between an atom arrangement and the atom arrangement when it is inverted.

This is sometimes called the "phase problem" in structure determination, since the phase of the diffracted wave $\psi(\boldsymbol{Q})$ is not measured, only its intensity, $\psi^*\psi$.

2.4.5 Perfect Crystals

In working problems with Patterson functions, it is often convenient to write the scattering factor distribution for an entire crystal, $f(\boldsymbol{r})$, in the following way:

$$f(\boldsymbol{r}) = f_{\mathrm{at}}(\boldsymbol{r}) * \sum_{\boldsymbol{R}_n} \delta(\boldsymbol{r} - \boldsymbol{R}_n) \; . \tag{2.99}$$

Here $f_{\mathrm{at}}(\boldsymbol{r})$ is the form factor of one atom. In (2.99) the form factor of the atom is convoluted with a sum of delta functions, each centered at a different atom site, $\boldsymbol{R}_n$. We evaluate (2.99) by first writing explicitly the convolution:

$$f(\boldsymbol{r}) = \int_{-\infty}^{\infty} f_{\mathrm{at}}(\boldsymbol{r}') \sum_{\boldsymbol{R}_n} \delta\big(\boldsymbol{r} - (\boldsymbol{r}' - \boldsymbol{R}_n)\big) \mathrm{d}^3\boldsymbol{r}' \; . \tag{2.100}$$

Rearranging the operations on independent variables:

$$f(\boldsymbol{r}) = \sum_{\boldsymbol{R}_n} \int_{-\infty}^{\infty} f_{\mathrm{at}}(\boldsymbol{r}') \, \delta\big(\boldsymbol{r} - (\boldsymbol{r}' - \boldsymbol{R}_n)\big) \mathrm{d}^3\boldsymbol{r}' \; . \tag{2.101}$$

The integral of (2.101) serves to pick out the value of $f_{\mathrm{at}}(\boldsymbol{r}')$ at the location of the delta function, cf., (2.82). By shifting the delta function continuously by $\boldsymbol{r}'$, the shape of $f_{\mathrm{at}}(\boldsymbol{r})$ is generated around the center of each delta function. These centers are each atom site, $\boldsymbol{R}_n$, so after the integration of (2.101):

[6] You can obtain the same $P(\boldsymbol{r})$ by taking the mirror image of the $f(\boldsymbol{r})$ in Fig. 2.7a (with the small peak to the immediate left of the large peak), and repeating the construction.

$$f(\boldsymbol{r}) = \sum_{\boldsymbol{R}_n} f_{\mathrm{at}}(\boldsymbol{r} - \boldsymbol{R}_n) \; . \tag{2.102}$$

Please compare (2.99) and (2.102).

The Patterson function of an infinite one-dimensional perfect crystal, $P_0(x)$, is:

$$P_0(x) = f^*(x) * f(-x) \; , \tag{2.103}$$

which we write using (2.99) for N atoms:

$$P_0(x) = \Big(f^*_{\mathrm{at}}(x) * \sum_{n'=-\infty}^{+\infty} \delta(x - n'a) \Big) * \Big(f_{\mathrm{at}}(-x) * \sum_{n''=+\infty}^{-\infty} \delta(n''a - x) \Big) \; . \tag{2.104}$$

Convolutions are commutative and associative, so we rearrange (2.104):

$$P_0(x) = \Big(f^*_{\mathrm{at}}(x) * f_{\mathrm{at}}(-x) \Big) * \Big(\sum_{n'=-\infty}^{+\infty} \delta(x - n'a) \Big) * \Big(\sum_{n''=+\infty}^{-\infty} \delta(n''a - x) \Big) \; . \tag{2.105}$$

Recall that a convolution of two functions requires a shift, overlap, multiplication, and integration. Because the δ-functions are infinitesimally narrow, there is zero overlap of the two series of δ-functions unless the shift, x, satisfies the condition $x = na$, where n is an integer. Therefore:

$$\Big(\sum_{n'=-\infty}^{\infty} \delta(x - n'a) \Big) * \Big(\sum_{n'=-\infty}^{\infty} \delta(x - n'a) \Big) = N' \Big(\sum_{n=-\infty}^{\infty} \delta(x - na) \Big) \; . \tag{2.106}$$

Here $N' = \infty$, which is as expected for an infinite number of overlaps of an infinite chain of atoms. For a chain of N atoms, the Patterson function is:

$$P_0(x) = N \Big(f^*_{\mathrm{at}}(x) * f_{\mathrm{at}}(-x) \Big) * \Big(\sum_{n=-\infty}^{\infty} \delta(x - na) \Big) \; , \tag{2.107}$$

The Fourier transformation of $P_0(x)$ provides the diffracted intensity, $I(Q)$. By the convolution theorem of Sect. A.1, the two convolutions and one multiplication of (2.107) become, after Fourier transformation, two multiplications and one convolution. Using (2.97):

$$I(Q) = N \big| f_{\mathrm{at}}(Q) \big|^2 * F\Big[\sum_{n'=-\infty}^{\infty} \delta(x - n'a) \Big] \; . \tag{2.108}$$

The Fourier transform of the δ-function series is:

$$F\Big[\sum_{n'=-\infty}^{\infty} \delta(x - n'a) \Big] = \int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}Qx} \sum_{n'=-\infty}^{\infty} \delta(x - n'a) \; \mathrm{d}x \; . \tag{2.109}$$

The condition $Qa = 2\pi h$ (where h is an integer) must be satisfied, or the integration over an infinite range of x is zero. The k-space Fourier transform is therefore zero except when $Q = 2\pi h/a$ precisely, so:

$$F\Big[\sum_{n'=-\infty}^{\infty}\delta(x-n'a)\Big] = N\sum_{h=-\infty}^{\infty}\delta(Q-2\pi h/a) = N\sum_{g}\delta(Q-g)\ . \quad (2.110)$$

Here again N is the number of terms in the sum in (2.110). In a formal problem, N becomes a mathematical infinity, but it is useful to keep the N because it shows the proportionality to the size of the crystal. The diffraction intensity of (2.108) is:

$$I(Q) = N^2\big|f_{\rm at}(Q)\big|^2\Big[\sum_{h=-\infty}^{\infty}\delta(Q-2\pi h/a)\Big]\ . \quad (2.111)$$

Equation (2.111) is a familiar result in a new form. The series of δ-functions gives the centers of the Bragg peaks from the crystal. These peaks are still sharp, but are attenuated at large Q by the atomic form factor intensity, $|f_{\rm at}(Q)|^2$.

2.4.6 Deviations from Periodicity

In many cases of interest, a scattering factor distribution, $f(\boldsymbol{r})$, can be expressed as the sum of a perfectly periodic function, $\langle f(\boldsymbol{r})\rangle$, plus a deviation function, $\Delta f(\boldsymbol{r})$, which provides the random or semi-random deviations from perfect periodicity. We know that the perfectly periodic function, $\langle f(\boldsymbol{r})\rangle$, provides sharp Bragg diffractions, but how does the deviation function, $\Delta f(\boldsymbol{r})$, affect the diffracted intensity? To find out, we calculate the Patterson function of $f(\boldsymbol{r})$:

$$f(\boldsymbol{r}) = \langle f(\boldsymbol{r})\rangle + \Delta f(\boldsymbol{r})\ , \quad (2.112)$$

$$P(\boldsymbol{r}) \equiv f^*(\boldsymbol{r}) * f(-\boldsymbol{r})\ , \quad (2.113)$$

$$\begin{aligned} P(\boldsymbol{r}) = {} & \langle f^*(\boldsymbol{r})\rangle * \langle f(-\boldsymbol{r})\rangle + \langle f^*(\boldsymbol{r})\rangle * \Delta f(-\boldsymbol{r}) \\ & + \Delta f^*(\boldsymbol{r}) * \langle f(-\boldsymbol{r})\rangle + \Delta f^*(\boldsymbol{r}) * \Delta f(-\boldsymbol{r})\ . \end{aligned} \quad (2.114)$$

Look at the second term in (2.114). We rewrite it with the aid of (2.96):

$$\langle f^*(\boldsymbol{r})\rangle * \Delta f(-\boldsymbol{r}) = \Big[\langle f^*_{\rm at}(\boldsymbol{r})\rangle * \sum_{\boldsymbol{R}_n}\delta(\boldsymbol{r}-\boldsymbol{R}_n)\Big] * \Delta f(-\boldsymbol{r})\ . \quad (2.115)$$

Convolutions are associative, so we can group the second and third factors in (2.115), and consider the new convolution:

$$\sum_{\boldsymbol{R}_n}\delta(\boldsymbol{r}-\boldsymbol{R}_n) * \Delta f(-\boldsymbol{r}) = \sum_{\boldsymbol{R}_n}\Delta f(-\boldsymbol{R}_n)\ , \quad (2.116)$$

where we used (2.82) in the same way as for (2.101)–(2.102). We assume that the deviation function, $\Delta f(-\boldsymbol{R}_n)$, has zero mean value.[7] Therefore:

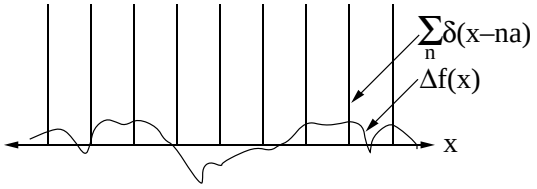


Fig. 2.8. Overlap of periodic delta functions, $\sum_n \delta(x-na)$, with a random function of zero mean, $\Delta f(x)$. Since the deviation function $\Delta f(\boldsymbol{r})$ has zero mean and is non-periodic, the periodic delta functions overlap $\Delta f(-\boldsymbol{r})$ at as many positive values as negative values, demonstrating (2.117).

$$\sum_{\boldsymbol{R}_n} \delta(\boldsymbol{r}-\boldsymbol{R}_n) * \Delta f(-\boldsymbol{r}) = 0 \; . \tag{2.117}$$

The second term for $P(\boldsymbol{r})$ in (2.114) is therefore zero (see also Fig. 2.8). Because $\boldsymbol{R}_n$ has precise periodicity over an infinite distance, (2.117) also holds true when $\Delta f(\boldsymbol{r})$ has short-range structure. By the same argument, the third term in (2.114) is also zero. Equation (2.114) becomes:

$$P(\boldsymbol{r}) = \langle f^*(\boldsymbol{r})\rangle * \langle f(-\boldsymbol{r})\rangle + \Delta f^*(\boldsymbol{r}) * \Delta f(-\boldsymbol{r}) \; . \tag{2.118}$$

The Patterson function for an alloy with disorder is reduced to two parts defined as the two terms in (2.118): 1) a Patterson function from the average crystal, $P_{\text{avge}}(\boldsymbol{r})$, and 2) a Patterson function from the deviation crystal, $P_{\text{devs}}(\boldsymbol{r})$:

$$P(\boldsymbol{r}) = P_{\text{avge}}(\boldsymbol{r}) + P_{\text{devs}}(\boldsymbol{r}) \; . \tag{2.119}$$

The diffracted intensity is the Fourier transform of the Patterson function of the alloy:

$$I(\boldsymbol{Q}) = F\big[P_{\text{avge}}(\boldsymbol{r}) + P_{\text{devs}}(\boldsymbol{r})\big] \; , \tag{2.120}$$

and since Fourier transforms are distributive:

$$I(\boldsymbol{Q}) = F\big[P_{\text{avge}}(\boldsymbol{r})\big] + F\big[P_{\text{devs}}(\boldsymbol{r})\big] \; . \tag{2.121}$$

Equation (2.121) shows that the diffraction patterns from the average crystal, $\langle f(\boldsymbol{r})\rangle$, and the deviation crystal, $\Delta f(\boldsymbol{r})$, are additive. In terms of the diffracted waves from these average and deviation crystals (cf., (2.96)):

$$I(\boldsymbol{Q}) = \big|F\left\langle f(\boldsymbol{r})\right\rangle\big|^2 + \big|F\big[\Delta f(\boldsymbol{r})\big]\big|^2 \; . \tag{2.122}$$

[7] This does not restrict generality because any non-zero mean could have been transferred into $\langle f(\boldsymbol{r})\rangle$ in (2.112).

We are familiar with the first term in (2.122), $|F\langle f(\boldsymbol{r})\rangle|^2$, which gives the sharp Bragg diffractions from the average crystal.

The second term in (2.122), $\left|F\left[\Delta f(\boldsymbol{r})\right]\right|^2$, is new. It is often a broad, diffuse intensity, as we show next. We will also show that with increasing disorder and larger $\Delta f(\boldsymbol{r})$, the sharp Bragg diffractions become weaker, and the diffuse intensity becomes stronger. Two important sources of $\Delta f(\boldsymbol{r})$ in a crystalline alloy are atomic displacement disorder and chemical disorder. Atomic displacement disorder comprises small deviations of atoms from the sites of a perfect crystal. These displacements may be static, or dynamic as in the case of thermal motion. Chemical disorder exists when there is randomness in the species of atoms that occupy the sites of a crystal. We consider these two types of disorder in sequence.

2.4.7 Uncorrelated Displacements

Atomic displacement disorder exists when atoms do not sit precisely on the periodic sites of a crystal. Atomic size differences in an alloy cause static displacements from lattice sites, and thermal vibrations cause dynamic displacement disorder. Both cause diffuse scattering. Here we consider a simple type of displacement disorder where each atom has a small, random shift, δ, off its site of a periodic lattice as shown in Fig. 2.9.

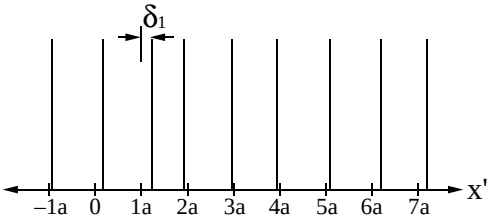


Fig. 2.9. Atomic displacement disorder in a one-dimensional crystal.

For now we assume there are no correlations between the displacements, δ_j, of neighboring atoms.[8] The Patterson function, $f(x) * f(-x)$, for this displacement distribution is shown in Fig. 2.10a. To understand this Patterson function, consider the overlap of the atom center distribution with itself after a shift of $x = na + \xi$, where a is the lattice parameter, n is an integer, and ξ is a small distance (typically $\xi < a$). With no correlation between the displacements of neighboring atoms, the probability of overlap of two atom centers is the same for a shift of the crystal by many lattice constants, $na+\xi$, as it is for a shift of one lattice constant, $1a + \xi$. The important exception occurs around $x = 0$, i.e., when $n = 0$. All the atom centers overlap perfectly with themselves when ξ is exactly zero, but even for the smallest shift, $\xi \neq 0$, there are zero overlaps of atom centers.

[8] For example, we assume that if one atom is displaced to the left, its neighbor to the right is equally likely to be displaced to the left or to the right.

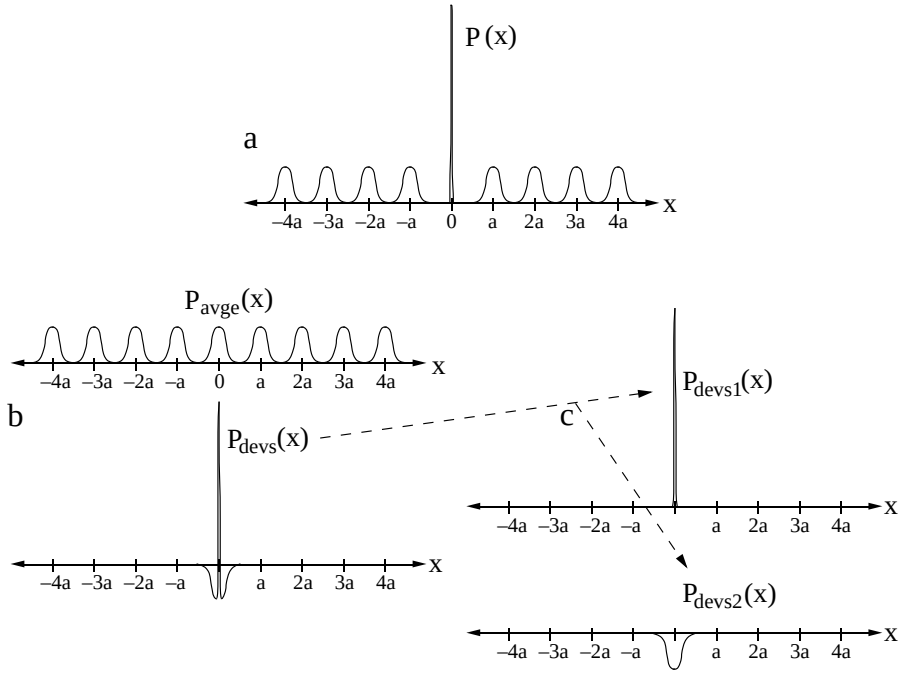


Fig. 2.10. (**a**) Patterson function for the random displacements of Fig. 2.9 and (2.119). (**b**) The Patterson function at top is the sum of $P_{\rm avge}(x)$ and $P_{\rm devs}(x)$. (**c**) $P_{\rm devs}(x)$ is the sum of $P_{\rm devs1}(x)$ and $P_{\rm devs2}(x)$.

The best way to work with the Patterson function in Fig. 2.10a is to break it into periodic and non-periodic parts (2.119), as shown in the two plots in Fig. 2.10b. The diffracted intensity from our crystal with displacement disorder is obtained from (2.121) as the sum of the Fourier transforms of these two functions, $P_{\rm avge}(x)$ and $P_{\rm devs}(x)$. The Fourier transform of $P_{\rm avge}(x)$ is the well-known series of Bragg peaks. These peaks are suppressed at large values of Q owing to the breadths of the peaks in $P_{\rm avge}(x)$ caused by displacement disorder (see top of Fig. 2.11). This suppression of the Bragg peaks at large Q is similar to the suppression caused by the atomic form factor, which also broadens the scattering centers of the atoms.

The Fourier transform of $P_{\rm devs}(x)$ is new for us. To understand its contribution to the diffraction intensity, we split $P_{\rm devs}(x)$ into two parts, $P_{\rm devs1}(x)$ and $P_{\rm devs2}(x)$ (Fig. 2.10c). The first, $P_{\rm devs1}(x)$, is a Dirac delta function, whose Fourier transform is a constant in k-space ($F\big[P_{\rm devs1}(x)\big]$ in Fig. 2.11). The second part, $P_{\rm devs2}(x)$, is a short, broadened function with negative sign. (In Sect. 2.4.8 we will consider it to be a Gaussian function.) Its Fourier transform, $F\big[P_{\rm devs2}(x)\big]$, is also shown in Fig. 2.11. The areas of these two parts, $P_{\rm devs1}(x)$ and $P_{\rm devs2}(x)$, are equal, since both arise from the same total number of atom-atom overlaps (equal to the number of atoms, N). This has an

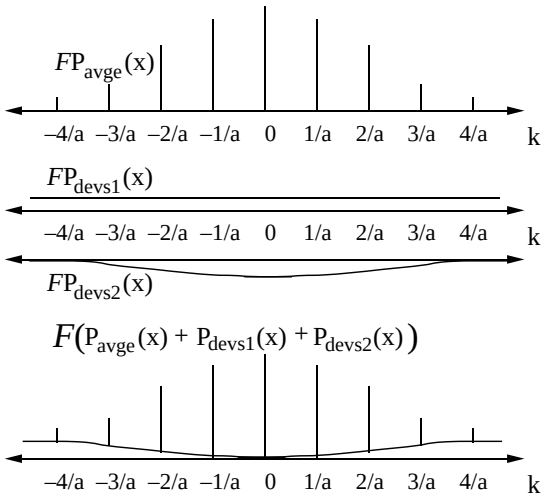


Fig. 2.11. The Fourier transform of the Patterson functions of Fig. 2.10. Fourier transform of $P_{\rm avge}(x)$ (top), Fourier transforms of the two components of $P_{\rm devs}(x)$ (middle). The sum of all three components (bottom) is the diffraction intensity from our linear crystal with Gaussian displacement disorder.

important consequence for the diffracted intensity at $Q = 0$:

$$I(Q{=}0) = \int_{-\infty}^{\infty} P_{\rm devs}(x)\, {\rm e}^{-{\rm i}0x}\, {\rm d}x = \int_{-\infty}^{\infty} P_{\rm devs}(x)\, {\rm d}x \ , \tag{2.123}$$

which is simply the area of the Patterson function, $P_{\rm devs}(x)$. Since $P_{\rm devs1}(x)$ and $P_{\rm devs2}(x)$ have equal and opposite areas, at $Q = 0$ there is zero diffuse scattering from atomic displacement disorder.

The $F\big[P_{\rm devs2}(x)\big]$ has a negative sign that decreases in magnitude with Q. The diffuse scattering therefore increases with Q, as the flat contribution originating from $F\big[P_{\rm devs1}(x)\big]$ increasingly dominates over $F\big[P_{\rm devs2}(x)\big]$. The function $P_{\rm devs2}(x)$, incidentally, has the same shape as the individual peaks in $P_{\rm avge}(x)$. In this case the Q-dependence of the rolloff of the Bragg peaks is the same as the Q-dependence of the diffuse scattering. The effects of displacement disorder increase with the characteristic size of the displacements, δ_j. The larger the characteristic δ, the faster the rolloff of the Bragg peaks with Q, and the greater the intensity of the diffuse scattering.

2.4.8 Temperature

During thermal vibrations, the distances between atoms undergo rapid changes with time. It is useful, however, to think of each x-ray scattering event as taking an instantaneous snapshot of the atom positions. The diffraction data are averages of many different instantaneous atom configurations. Over a large crystal, however, each instantaneous snapshot looks approximately the same. The same argument of the previous section on atomic displacement disorder is then appropriate for understanding the diffraction

effects caused by thermal disorder in atom positions. This section uses a simple model of atom vibrations to calculate two effects of temperature:

- the Debye–Waller factor that causes the Bragg peaks to lose intensity,
- the thermal diffuse scattering, which is where the "lost" intensity reappears.[9]

A detailed analysis of thermal vibrations is not simple, because it should be performed with knowledge of the polarizations and numbers of all phonon modes. A complete analysis considers the contribution of each phonon to the relative separation of each atom-atom pair in the solid. In phonons with long wavelengths, for example, neighboring pairs of atoms tend to move together.[10] In contrast, high frequency phonons affect strongly the mutual displacements of neighboring atoms. In addition, it is important to know how the atom motions within each phonon are oriented with respect to $\boldsymbol{Q}$ – atom motions nearly perpendicular to $\boldsymbol{Q}$ have weak effects on the scattering. Calculating the Patterson function from the densities of phonons with all polarizations is a problem for computers, and is beyond the scope of this book.

Thermal vibrations broaden the Patterson function of the scattering factor distribution. To develop a simple analytical model, we assume each atom center has a thermal spread around its crystal site that is a Gaussian function of characteristic width, σ. (A plausibility argument for a Gaussian function is provided in Appendix A.11.) For any n^{th} neighbor pair of atoms, we expect the vibrations of both atoms to affect the Patterson function. Suppose we place a stationary atom 1 at a point in space. When an atom 2 vibrates with respect to atom 1, there is a probability distribution for their interatomic separation, x:

$$p_{\text{atom2}}(x) = \frac{1}{\sqrt{\pi}\,\sigma}\,\mathrm{e}^{-x^2/\sigma^2}\ . \tag{2.124}$$

This function is shown schematically in Fig. 2.12a. Now let atom 1 vibrate. For every interatomic separation provided by the thermal motion of the atom 2, make a thermal distribution for the position of atom 1. To obtain the distribution of separations between atoms 1 and 2, the displacement distribution of atom 2 serves to weight the displacement distribution of atom 1. The various weights are shown in Fig. 2.12b, and the weighted sum of the net thermal distribution of atom 1 with respect to atom 2 is shown schematically in Fig. 2.12c as $p * p(x)$. The procedure we followed was in fact a convolution: the distribution of atom 1 was shifted, $p_{\text{atom1}}(x - x')$, multiplied by $p_{\text{atom2}}(x')$, and summed (integrated) over all values of x':

[9] The total coherent cross-section remains constant.

[10] Another aspect of the problem is that a crystal has fewer long-wavelength than short-wavelength vibrational modes. However, the lower energy of the long-wavelength modes means that their occupancy is higher at all temperatures, especially low temperatures.

$$P_{\text{therm}}(x) = \int_{-\infty}^{\infty} p_{\text{atom2}}(x')\, p_{\text{atom1}}(x - x')\, \mathrm{d}x' \ . \tag{2.125}$$

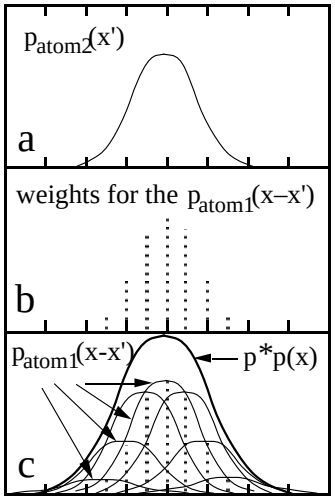


Fig. 2.12. (**a**) The thermal spread of centers for atom 2. (**b**) Weights for the centers of the thermal distribution of the atom 1. (**c**) The distribution of all thermal separations of atom 1 with respect to atom 2.

For n^{th} neighbor pairs of atoms, the Patterson function of the thermal spread, $P_{\text{therm}}(x)$, is the convolution of the Gaussian thermal spread functions of both atoms (cf., (8.24)):

$$P_{\text{therm}}(x) = \left(\frac{1}{\sqrt{\pi}\sigma}\mathrm{e}^{-x^2/\sigma^2}\right) * \left(\frac{1}{\sqrt{\pi}\sigma}\mathrm{e}^{-x^2/\sigma^2}\right) \text{ when } n \neq 0 \ . \tag{2.126}$$

$$P_{\text{therm}}(x) = \frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-x^2/(2\sigma^2)} \text{ when } n \neq 0 \ . \tag{2.127}$$

A detailed analysis treats closer pairs of atoms differently from more distant pairs, but here we ignore this difference except for the case of $n = 0$. In the special case of $n = 0$, we are considering the autocorrelation function between the positions *of the individual atoms with themselves*. Each atom sees itself as being at rest, so the Patterson function for the thermal spread is:

$$P_{\text{therm}}(x) = N\delta(x) \text{ when } n = 0 \ . \tag{2.128}$$

We now obtain the Patterson function for the entire crystal by convoluting the thermal spread function, $P_{\text{therm}}(x)$, with the Patterson function of the perfect crystal (2.107). The Patterson function, $P(x)$, for the crystal with thermal displacement disorder is the following modification of (2.107). Note the special treatment of the $n = 0$ term, which provides the δ-function:

$$P(x) = N\big[f_{\rm at}^*(x) * f_{\rm at}(-x)\big] * \Bigg[\,\delta(x) + \Big(\sum_{n\neq 0;n=-\infty}^{n=\infty} \delta(x-na)\Big) * \Big(\frac{1}{\sqrt{2\pi}\sigma} {\rm e}^{-x^2/(2\sigma^2)}\Big)\Bigg] \,. \quad (2.129)$$

We rewrite the sum in (2.129) by adding and subtracting the $n = 0$ term (the same trick used in Fig. 2.10 to give an uninterrupted infinite series for $P_{\rm avge}(x)$):

$$P(x) = N\big[f_{\rm at}^*(x) * f_{\rm at}(-x)\big] * \Bigg[\,\delta(x) - \frac{1}{\sqrt{2\pi}\sigma} {\rm e}^{-x^2/(2\sigma^2)} + \Big(\sum_{n=-\infty}^{n=\infty} \delta(x-na)\Big) * \Big(\frac{1}{\sqrt{2\pi}\sigma} {\rm e}^{-x^2/(2\sigma^2)}\Big)\Bigg] \,. \quad (2.130)$$

The diffracted intensity is the Fourier transform of the Patterson function of (2.130). The transformation from (2.130) to (2.131) follows that from (2.107) to (2.111), plus the fact that the Fourier transform of a Gaussian is a Gaussian:

$$I(Q) = N\big|f_{\rm at}(Q)\,\big|^2 \Big[\,(1 - {\rm e}^{-\sigma^2(Q)^2/2}) + {\rm e}^{-\sigma^2(Q)^2/2} \sum_h \delta(Q - 2\pi h/a)\Big] \,. \quad (2.131)$$

The last term in the square brackets is the expected set of sharp Bragg peaks, but attenuated at larger values of Q by the "Debye–Waller factor," $D(Q)$:

$$D(\sigma, Q) = {\rm e}^{-\sigma^2(Q)^2/2} \,. \quad (2.132)$$

The Debye–Waller factor suppresses the intensity of Bragg peaks at high Q, as does the size of the atom through the factor $|f_{\rm at}(Q)|^2$ of Sect. 3.3.2, so the Debye–Waller factor can be considered a "thermal fattening of the atoms." The intensity lost from the Bragg peaks reappears[11] as the first term in brackets in (2.131), $1 - {\rm e}^{-\sigma^2(Q)^2/2}$, which is the "thermal diffuse scattering." The thermal diffuse scattering has no distinct peaks, but usually has gradual modulations that increase with Q as shown in Fig. 2.11.

The Debye–Waller factor can provide quantitative information about the mean-squared displacement, $\langle x^2 \rangle$, during thermal motion of the atoms. The larger is $\langle x^2 \rangle$, the smaller the Debye–Waller factor (and the larger the suppression of the Bragg diffractions). We first relate $\langle x^2 \rangle$ to the σ^2 in the thermal spread function of the individual atoms. This is the second moment of the Gaussian function:

$$\langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 \frac{1}{\sqrt{\pi}\,\sigma} \, {\rm e}^{-x^2/\sigma^2} \, {\rm d}x = \frac{1}{2}\sigma^2 \,, \quad (2.133)$$

so from (2.132) and (5.20):

[11] Never forget that the total cross-section for coherent scattering is constant.

$$D(\sigma, Q) = \mathrm{e}^{-\langle x^2 \rangle (Q)^2} = \mathrm{e}^{-\langle x^2 \rangle (4\pi \sin\theta/\lambda)^2} \ . \tag{2.134}$$

At modest temperatures and small Q we can often linearize the exponential to predict a suppression of the Bragg peaks that is quadratic in Q:

$$D(\sigma, Q) \simeq 1 - \langle x^2 \rangle \left(\frac{4\pi \sin\theta}{\lambda} \right)^2 \ . \tag{2.135}$$

Physically, the Debye–Waller factor accounts for the loss of constructive interference in diffraction when the mean-squared atomic displacements become comparable to the x-ray wavelength. The Debye–Waller factor always suppresses the intensity of Bragg peaks.

Equations (2.134) or (2.135) can be used to determine $\langle x^2 \rangle$ from experimental data on diffraction intensities.[12] Conversely, it is often important to predict the Debye–Waller factor for a material at a known temperature. In essence, $\langle x^2 \rangle$ is proportional to the potential energy of a harmonic oscillator, and scales linearly with temperature, T. Although $\langle x^2 \rangle$ can be calculated easily for the single oscillator in the Einstein model, it is more handy to express the Debye–Waller factor in terms of a Debye temperature, θ_{D}, since tabulations of θ_{D} are conveniently available. For the Debye model the Debye–Waller factor has been worked out, and at temperatures comparable to the Debye temperature or higher, the Debye–Waller factor is:

$$D(T, Q) \simeq \exp\left[\frac{-12 h^2 T}{m\, k_B\, \theta_{\mathrm{D}}^2} \left(\frac{\sin\theta}{\lambda} \right)^2 \right] , \tag{2.136}$$

$$D(T, \theta) \simeq 1 - \frac{22,800\, T}{m\, \theta_{\mathrm{D}}^2} \left(\frac{\sin\theta}{\lambda} \right)^2 \tag{2.137}$$

Here the units of mass are the atomic weight (e.g., 55.847 for Fe), T and θ_{D} are in Kelvin, and λ is in Å. For use in (2.134) and (2.135), in the Debye model:

$$\langle x^2 \rangle = 144.38\, \frac{T}{m\, \theta_{\mathrm{D}}^2} \ . \tag{2.138}$$

Although the Debye–Waller factor pertains to the thermal spread of distances between pairs of atoms, a Debye–Waller factor is often assigned to the scattering from a single atom. With this approximation, the atomic form factor, f, of each atom is replaced with $f \exp(-M)$. The Debye–Waller factor for the intensity is therefore $\exp(-2M)$. Also defined is the parameter B, related to $\langle x^2 \rangle$. Standard relationships are:

$$2M = \langle x^2 \rangle \left(\frac{4\pi \sin\theta}{\lambda} \right)^2 , \tag{2.139}$$

$$M = B \left(\frac{\sin\theta}{\lambda} \right)^2 . \tag{2.140}$$

[12] Note that $\langle x^2 \rangle$ is along the direction of Q. In an isotropic material $\langle x^2 \rangle$ would equal 1/3 of the mean-squared atomic displacement.

In the case of an alloy, it is typical to assign different Debye–Waller factors for each type of atom, A or B, written as $\mathrm{e}^{-M_\mathrm{A}}$ and $\mathrm{e}^{-M_\mathrm{B}}$:

$$\psi(\boldsymbol{Q}) = \sum_{\boldsymbol{r}} \Big[\mathrm{e}^{-M_\mathrm{A}} f_\mathrm{A}\, \delta_\mathrm{A}(\boldsymbol{r}) + \mathrm{e}^{-M_\mathrm{B}} f_\mathrm{B}\, \delta_\mathrm{B}(\boldsymbol{r}) \Big] \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}} \ . \tag{2.141}$$

Here the δ-functions are Kroneker delta functions indicating the presence of an A or B atom at $\boldsymbol{r}$.

At temperatures below approximately half the Debye temperature, and especially below a quarter of the Debye temperature, (2.136) is no longer reliable for calculating the Debye–Waller factor. Two quantum effects are important at low temperatures. First, owing to Bose–Einstein phonon population statistics, the higher frequency phonons are not excited in simple proportion to the ratio kT/ε, where ε is the phonon energy. Second, at temperatures below about half the Debye temperature, the "zero-point" vibrations of the solid account for an increasingly large fraction of the atom displacements. Owing to zero-point vibrations, the thermal diffuse scattering can never be eliminated, even by cooling to arbitrarily low temperature.

The derivation of (2.131) was clean because we assumed the same Gaussian thermal spread for all interatomic correlations. For long wavelength phonons, however, adjacent atoms tend to move together in a group. In general, the nearest-neighbor pair correlations are less broadened than the correlations for more distant neighbor pairs. If atoms tend to move in groups, as in acoustic modes, the displacement has long-range modulations, and the thermal diffuse scattering intensity is concentrated near the reciprocal lattice points. The detailed shape of the thermal diffuse scattering depends on the lattice dynamics of the crystal vibrations [9.2]. With a Born–von Kármán model of lattice dynamics, for example, it is possible to calculate the projected components of the atom movements normal to the diffracting planes, and obtain a more accurate $P_\mathrm{therm}(x)$ of (2.127). Alternatively, the phonon spectrum of the crystal can be deduced from measurements of the thermal diffuse scattering, at least in principle. In practice, such measurements require careful correction for other sources of diffuse intensity (such as atomic size and displacement effects).

3. Inelastic Scattering

Elastic scattering involves momentum transfers and positions $\{\boldsymbol{Q}, \boldsymbol{r}\}$, which are complementary variables in quantum mechanics: $\boldsymbol{Q} \leftrightarrow \boldsymbol{r}$. Inelastic scattering is an extension into energy and time $\{\boldsymbol{Q}, E, \boldsymbol{r}, t\}$, which provides pairs of complementary variables: $(\boldsymbol{Q}, E) \leftrightarrow (\boldsymbol{r}, t)$. The amplitude of the scattered wave, ψ, is the sum of phase factors of wavelets emitted from individual atoms, but now we allow for a time variation. The phase information in $\psi(\boldsymbol{Q}, E)$ includes details of atom positions and their motions. Recall the case for elastic scattering, where we obtained by inverse Fourier transformation: $f(\boldsymbol{r}) = F_{\boldsymbol{Q}}^{-1}\psi(\boldsymbol{Q})$. For inelastic scattering the analogous result is: $f(\boldsymbol{r}, t) = F_{\boldsymbol{Q}}^{-1}F_{E}^{-1}\psi(\boldsymbol{Q}, E)$.

An inelastic experiment measures an intensity and not a wave amplitude. The experimental information on $\{\boldsymbol{r}, t\}$ can be obtained directly from the scattered intensity $I(\boldsymbol{Q}, E)$ by Fourier inversion $F_{\boldsymbol{Q}}^{-1}F_{E}^{-1}I(\boldsymbol{Q}, E)$, rather than $F_{\boldsymbol{Q}}^{-1}F_{E}^{-1}\psi(\boldsymbol{Q}, E)$. As for the case with the Patterson function for diffraction experiments, there is a similar loss of information about atom positions and the phases of their motions. Nevertheless, the intensity is the actual quantity measured in a scattering experiment and we must make do with it. Inelastic scattering has an important analog to the Patterson function of elastic scattering, the "Van Hove function," defined in Sect. 3.1.2 as an autocorrelation function of the scattering factor distribution in space and time.

Whereas the inelastically scattered wave, $\psi(\boldsymbol{Q}, E)$, is the double Fourier transform of the moving scattering factor distribution, the scattered intensity, $I(\boldsymbol{Q}, E)$, is the double Fourier transform of the Van Hove correlation function of the scattering factor distribution.

We begin by proving this emphasized statement. The subsequent section uses the Van Hove function to explain scattering phenomena involving various dynamical excitations. Then starting anew with Fermi's Golden Rule, inelastic scattering is calculated in a more precise and general way. The Van Hove space-time correlation function is developed by taking careful account of the non-commutivity of the position and momentum operators. A proper treatment of magnetic scattering is then presented. These latter sections are parallel to similar sections in the books by Squires and Lovesey, which are recommended to all practitioners of inelastic neutron scattering, especially persons inclined towards theory.

3.1 Correlation Function for Inelastic Scattering – The Van Hove Function

3.1.1 Atom Centers at Points in Space and Time

As was the case for elastic scattering, most of the important results can be obtained by assuming the scatterers are points. The point scatterer emits a wavelet from position $\boldsymbol{r}_j$ at time t_k. This $\boldsymbol{r}_j$ and t_k provide the phase of the wavelet from the point emitter relative to wavelets from other point emitters: $\boldsymbol{Q}\cdot\boldsymbol{r}_j - \omega t_k$. The amplitude of the scattering is given by the scattering strength of the point emitter, $f(\boldsymbol{r}_j, t_k)$, which is an amplitude at a point in space and an instant in time. For a nucleus we may consider f as being at a point, although for magnetic spin distributions the shape of electron orbitals may be included later by convolution. For the distribution in time, we will usually consider a Fourier series with different frequencies, or energies $E = \hbar\omega$.

It proves convenient to consider a distribution of scatterers, $f(\boldsymbol{r}, t)$, with continuous variables, $\boldsymbol{r}$ and t, rather than a sum over discrete points, $\{\boldsymbol{r}_j\}$, and snapshots in time t_k. We change variables as:

$$\psi(\boldsymbol{Q}, E) = \sum_{\boldsymbol{r}_j}^{N} \sum_{t_k} f_{\boldsymbol{r}_j, t_k} \mathrm{e}^{-\mathrm{i}(\boldsymbol{Q}\cdot\boldsymbol{r}_j - \omega t)} = \iint_{-\infty}^{+\infty} f(\boldsymbol{r}, t)\, \mathrm{e}^{-\mathrm{i}(\boldsymbol{Q}\cdot\boldsymbol{r} - \omega t)} \mathrm{d}^3\boldsymbol{r}\, \mathrm{d}t \ . \quad (3.1)$$

To equate a continuous integral to a discrete sum requires that $f(\boldsymbol{r}, t)$ is not a smooth function of position or time. Over most of space and time, $f(\boldsymbol{r}, t)$ is zero, but when the scattering amplitude exists at $\boldsymbol{r} = \boldsymbol{r}_i$ and $t = t_k$, $f(\boldsymbol{r}_i, t_k)$ is a Dirac delta function times a constant, $f_{\boldsymbol{r}_j, t_k}$:

$$f(\boldsymbol{r}_j, t_k) = f_{\boldsymbol{r}_j, t_k} \delta(\boldsymbol{r} - \boldsymbol{r}_i)\delta(t - t_k) \ . \quad (3.2)$$

To extend (3.2) to include many atom centers, we take the sum over $\boldsymbol{r}_j$ and t_k:

$$f(\boldsymbol{r}, t) = \sum_{\boldsymbol{r}_j}^{N} \sum_{t_k} f_{\boldsymbol{r}_j, t_k} \delta(\boldsymbol{r} - \boldsymbol{r}_j)\delta(t - t_k) \ , \quad (3.3)$$

so we satisfy the equality in (3.1) between points in space and time, $\{\boldsymbol{r}_j, t_k\}$, and continuous functions of $\boldsymbol{r}, t$.

3.1.2 Definition of the Van Hove Function

We define the "Van Hove function," $G(\boldsymbol{r}, t)$:

$$G(\boldsymbol{r}, t) \equiv \iint_{-\infty}^{+\infty} f^*(\boldsymbol{r}', t') f(\boldsymbol{r} + \boldsymbol{r}', t + t')\, \mathrm{d}^3\boldsymbol{r}' \mathrm{d}t' \ . \quad (3.4)$$

Equation (3.4) is a double autocorrelation function – a space-time correlation function with limits of integration over all space and all time.

The most important feature of the Van Hove function is that its Fourier transform is the scattered intensity in kinematical theory. To show this, we use (3.1) to write $I(\boldsymbol{Q}) = \psi^* \times \psi$ as:

$$I(\boldsymbol{Q},E) = \iint_{-\infty}^{+\infty} f^*(\boldsymbol{r}')\,\mathrm{e}^{\mathrm{i}(\boldsymbol{Q}\cdot\boldsymbol{r}'-\omega t')}\mathrm{d}^3\boldsymbol{r}'\mathrm{d}t' \times \iint_{-\infty}^{+\infty} f(\boldsymbol{r}'',t'')\,\mathrm{e}^{-\mathrm{i}(\boldsymbol{Q}\cdot\boldsymbol{r}''-\omega t'')}\mathrm{d}^3\boldsymbol{r}''\mathrm{d}t'' \;. \tag{3.5}$$

Since $\boldsymbol{r}'$ and $\boldsymbol{r}''$ are independent variables, as are t' and t'':

$$I(\boldsymbol{Q},E) = \iint_{-\infty}^{+\infty} \Big(\iint_{-\infty}^{+\infty} f^*(\boldsymbol{r}',t') f(\boldsymbol{r}'',t'') \times \mathrm{e}^{-\mathrm{i}[\boldsymbol{Q}\cdot(\boldsymbol{r}''-\boldsymbol{r}')-\omega(t''-t')]}\;\mathrm{d}^3\boldsymbol{r}''\mathrm{d}t''\Big)\,\mathrm{d}^3\boldsymbol{r}'\mathrm{d}t' \;. \tag{3.6}$$

Define $\boldsymbol{r} \equiv \boldsymbol{r}'' - \boldsymbol{r}'$ and $t \equiv t'' - t'$, and change variables $\boldsymbol{r}'' \rightarrow \boldsymbol{r} + \boldsymbol{r}'$ and $t'' \rightarrow t + t'$. In so doing, the limits of integration for $\boldsymbol{r}$ are shifted by $-\boldsymbol{r}'$ and $-t'$, but this is not of concern for integrations performed over all of space and all of time:

$$I(\boldsymbol{Q},E) = \iint_{-\infty}^{+\infty} \times \Big(\iint_{-\infty}^{+\infty} f^*(\boldsymbol{r}',t') f(\boldsymbol{r}+\boldsymbol{r}',t+t')\,\mathrm{e}^{-\mathrm{i}(\boldsymbol{Q}\cdot\boldsymbol{r}-\omega t)}\;\mathrm{d}^3\boldsymbol{r}\,\mathrm{d}t\Big)\,\mathrm{d}^3\boldsymbol{r}'\,\mathrm{d}t' \;, \tag{3.7}$$

$$I(\boldsymbol{Q},E) = \iint_{-\infty}^{+\infty} \Big(\iint_{-\infty}^{+\infty} f^*(\boldsymbol{r}',t') f(\boldsymbol{r}+\boldsymbol{r}',t+t')\mathrm{d}^3\boldsymbol{r}'\,\mathrm{d}t\Big) \times \mathrm{e}^{-\mathrm{i}(\boldsymbol{Q}\cdot\boldsymbol{r}-\omega t)}\;\mathrm{d}^3\boldsymbol{r}\,\mathrm{d}t \;. \tag{3.8}$$

Using the definition of (3.4), we rewrite (3.8):

$$I(\boldsymbol{Q},E) = \iint_{-\infty}^{+\infty} G(\boldsymbol{r},t)\,\mathrm{e}^{-\mathrm{i}(\boldsymbol{Q}\cdot\boldsymbol{r}-\omega t)}\;\mathrm{d}^3\boldsymbol{r}\,\mathrm{d}t \;. \tag{3.9}$$

Equation (3.9) shows that the scattered intensity is the Fourier transform of the Van Hove function:

$$I(\boldsymbol{Q},E) = F_{\boldsymbol{r}}F_t G(\boldsymbol{r},t) \;, \tag{3.10}$$

and by the inverse transformation we must have:

$$G(\boldsymbol{r},t) = F_{\boldsymbol{Q}} F_E I(\boldsymbol{Q},E) \; . \tag{3.11}$$

For comparison, the scattered wave, $\psi(\boldsymbol{Q},E)$ of (3.1), is the Fourier transform of the scattering factor distribution, $f(\boldsymbol{r},t)$. We therefore have another relationship between $I(\boldsymbol{Q},E)$ and $f(\boldsymbol{r},t)$:

$$I(\boldsymbol{Q},E) = \psi^*(\boldsymbol{Q},E)\,\psi(\boldsymbol{Q},E) \; , \tag{3.12}$$

$$I(\boldsymbol{Q},E) = \left(F_{\boldsymbol{r}} F_t f(\boldsymbol{r},t)\right)^* \; F_{\boldsymbol{r}} F_t f(\boldsymbol{r},t) = \left|F_{\boldsymbol{r}} F_t f(\boldsymbol{r},t)\right|^2 \; . \tag{3.13}$$

Comparing (3.10) and (3.13):

$$F_{\boldsymbol{r}} F_t G(\boldsymbol{r},t) = \left|F_{\boldsymbol{r}} F_t f(\boldsymbol{r},t)\right|^2 \; . \tag{3.14}$$

Equation (3.14) is consistent with the convolution theorem of Sect. A.1 – a (double) convolution in real space (the Van Hove function of (3.4)) corresponds to a multiplication in Fourier space (right-hand side of (3.14)).

3.1.3 Examples of Van Hove Functions

In this section we examine the scattering from a simple form of the Van Hove function from 3.9 with one spatial dimension:

$$I(Q,E) = \int\limits_{-\infty}^{\infty} \int\limits_{-\infty}^{\infty} G(x,t)\, \mathrm{e}^{-\mathrm{i}(Qx-\omega t)}\, \mathrm{d}x\, \mathrm{d}t \; . \tag{3.15}$$

In all our examples we assume coherent scattering so there are predictable phase relationships between different scattering centers. Section 3.1.4 develops further the cases where incoherent averaging of the space and time correlations produce a "self-correlation function" and the "Patterson function."

We later treat the incoherent case by following the "self-correlation function" of the individual scatterers.

First consider an elementary excitation in a solid that provides a periodic modulation of the scattering factor in space and time:

$$f(x,t) = \mathrm{e}^{\mathrm{i}(qx-\omega_0 t)} \; . \tag{3.16}$$

We use (3.4) to obtain its Van Hove function:

$$G(x,t) = \int\limits_{-\infty}^{\infty} \int\limits_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}(qx'-\omega_0 t')}\, \mathrm{e}^{\mathrm{i}[q(x+x')-\omega_0(t+t')]}\, \mathrm{d}x' \mathrm{d}t' \; , \tag{3.17}$$

$$G(x,t) = \int\limits_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}qx} \mathrm{d}x' \int\limits_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}\omega_0 t}\, \mathrm{d}t' = \mathrm{e}^{\mathrm{i}qx} \int\limits_{-\infty}^{\infty} \mathrm{d}x'\, \mathrm{e}^{-\mathrm{i}\omega_0 t} \int\limits_{-\infty}^{\infty} \mathrm{d}t' \; , \tag{3.18}$$

$$G(x,t) = \mathrm{e}^{\mathrm{i}(qx-\omega_0 t)} \; , \tag{3.19}$$

where we have ignored the normalization, and we find that $G(x,t) = f(x,t)$. In this case where $G(x,t)$ has the form of a wave (3.19), (3.15) becomes:

$$I(Q,E) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}(qx-\omega_0 t)}\,\mathrm{e}^{-\mathrm{i}(Qx-\omega t)}\,\mathrm{d}x\,\mathrm{d}t\ , \tag{3.20}$$

$$I(Q,E) = \int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}(q-Q)x}\,\mathrm{d}x \int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}(\omega-\omega_0)t}\,\mathrm{d}t\ , \tag{3.21}$$

$$I(Q,E) = \delta(q-Q-g)\ \delta(\omega-\omega_0)\ . \tag{3.22}$$

Equation (3.22) shows that the momentum transfer must match that of the wave, modulo a reciprocal lattice vector, g (which provides a factor of $\mathrm{e}^{\mathrm{i}gx} = 1$ in the integrand).

An elementary excitation in a solid with unique $\{q, \omega_0\}$ provides intensity at a single point in energy, and at a distance of Q away from each reciprocal lattice point. This of course assumes coherent scattering – incoherent scattering places a restriction on ω only.

Local Excitations. It is instructive to develop the Van Hove function and the scattered intensity with a pictorial approach, as in Fig. 3.1. The scattering factor distribution is shown in Fig. 3.1a as it undergoes a full cycle of oscillation.[1] The construction of the Van Hove function $G(x,t)$ parallels that of the Patterson function in Fig. 2.7. The detailed steps of shifting and integrating in Fig. 2.7 are not shown here. Suffice to say that situation along the x-dimension is quite analogous to that of Fig. 2.7. For the case where $t = 0$, for example, it is necessary to average the shifting and integrating of Fig. 2.7 for all nine times shown in Fig. 3.1a. Notice that although some of the scattering factor distributions have zero intensity at odd multiples of a, the result for $G(x,t)$ always has intensity at these locations (although weaker). The time dimension of $G(x,t)$ is obtained in a similar way. Consider the first time interval for time differences of t_1. For this case it is necessary to match all pairs of the scattering factor distributions that differ by one in their vertical stacking in Fig. 3.1a. These eight cases are then overlapped and shifted along the x-axis as before, and their results are averaged. The other time intervals require taking pairs separated further in time, but always it was eight cases whose shift and overlap are evaluated for constructing Fig. 3.1b.

The final step is to obtain the $I(Q,\omega)$ of Fig. 3.1c. In this case the situation is fairly simple. There is only one time frequency in Fig. 3.1a, equal to $2\pi/t_8$. If we allow creation and annihilation of such an excitation, we expect intensity at frequencies $\omega = \pm 2\pi/t_8$. The spatial periodicity of the problem includes a superlattice periodicity, so peaks appear at intervals of $\pm n\pi/a$, where n is either even or odd.

[1] The convention is that times t_k with larger subscripts are later times.

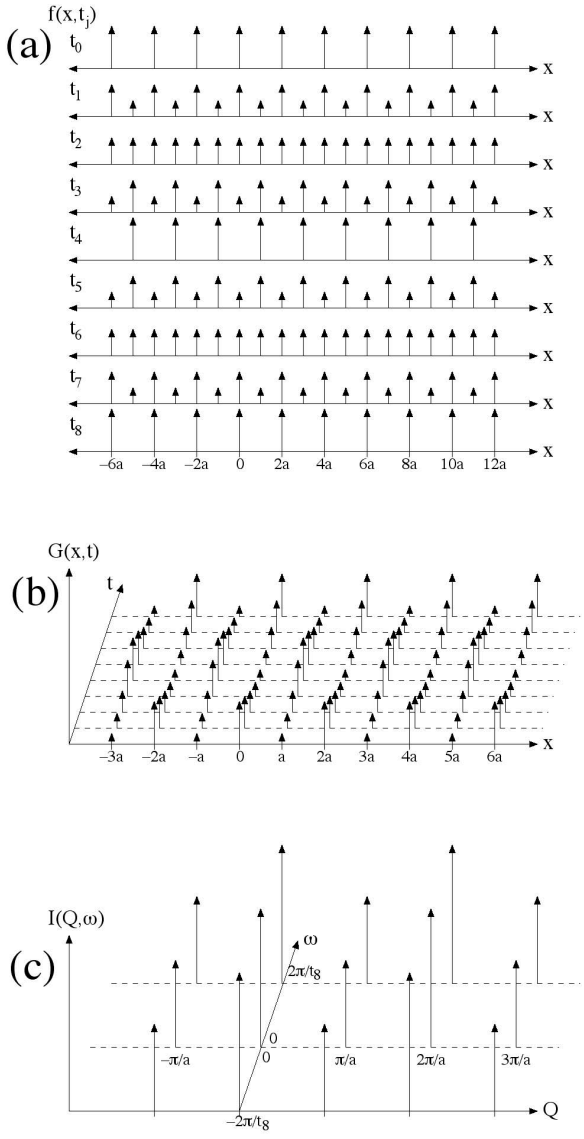


Fig. 3.1. (**a**) Variations of a scattering function $f(x, t_i)$ in space and time. Nine snapshots are shown vertically for nine t_i. (**b**) The Van Hove function, $G(x, t)$, obtained by overlapping all pairs of scattering factor distributions in (a), and summing the resulting product of overlaps for all pairs separated by the same number of time intervals. (**c**) The scattered intensity, $I(Q, \omega)$. It is assumed possible to both create and annihilate an excitation as shown in (a), hence points at $\pm\omega_0$.

Dispersive Excitations. The next example in Fig. 3.2a is essentially the scattering factor distribution shown in Fig. 2.7, but set in motion to the right. It therefore has the same inversion symmetry along x as shown in Fig. 2.7 (which was used to demonstrate Friedel's law). The scattering factor distribution is assumed to move continuously to the right with increasing time. In assessing the overlap of the scattering factor distributions, for all cases of time zero (i.e., the overlap of the scattering factor with itself before

it has a chance to move), the largest intensity occurs at lattice translation vectors. For larger times, t, evaluation of $G(x,t)$, requires the overlap of pairs of $f(x,t_k)$ in Fig. 3.2a that are separated in time. The best overlaps of these time-shifted $f(x,t_k)$ generally do not occur at lattice translations. The structures of $f(x,t_k)$ remain rigidly unchanged with time, however, so the same structure exists around the best overlap. We see in Fig. 3.2b that the shape of $G(x,t)$ is the same at each t, but it moves to the right with increasing time.

The scattered intensity $I(Q,\omega)$ in Fig. 3.2c shows peaks much as in Fig. 3.1c. It is evident that along the time dimension of Figs. 3.2a,b the pattern repeats itself with a periodicity of eight time snapshots in Figs. 3.2a. The inelastic part of the scattering therefore occurs at $\pm 2\pi/t_8$, assuming that excitations of the type shown in Fig. 3.2a can be both created and annihilated. There is another point worth noting about the time dependence in Fig. 3.2c. Along the time dimension we encounter sets of three δ-functions for each fundamental periodicity of the wave. The δ-functions, and their grouping into threes, requires that we have higher-order Fourier components in the frequency domain – formally, we expect a whole series of excitations over an infinite range of energies: $\pm n\hbar\omega_0 = \pm n\hbar 2\pi/t_8$. These high energies are improbable in nature, so the use of sharp scattering centers (drawn as arrows in Fig. 3.2a) may be misleading for problems of time dynamics. Delta-functions are not a problem for the spatial coordinates, however, since diffraction experiments occur with zero energy transfer. Along the time dimension, however, we expect our excitations to have the shape of a sine wave, since additional Fourier components are quite costly in energy.

Also shown in Fig. 3.2c, however, are other related types of excitations (dashed vertical lines). Suppose that the velocity of the arrows in Fig. 3.2a is a constant, but the horizontal separation of the arrows can be varied. For example, assume it is possible to double their spatial periodicity from a in Fig. 3.2a, to $2a$. In this case it takes twice as long to repeat the pattern of Fig. 3.2a, so the frequency is halved. As long as the velocity of the excitation in Fig. 3.2a is the same, there will be a linear relationship between Q and ω. This is shown for a whole dispersion of such excitations as dashed vertical lines in Fig. 3.2c. Their arrangement in straight lines is characteristic of the excitations all moving with the same velocity. Normally there would also be excitations expected where the scattering factor distribution moves to the left in Fig. 3.2a, so each reciprocal lattice point at $\omega = 0$ would also be crossed by a dispersion of excitations with a negative slope in the space of $\{Q,\omega\}$.

Disordered Excitations. Another example is presented in Fig. 3.3. This example is a space-time analog of displacement disorder presented in Sect. 2.4.7. Here we assume that the scattering factor distribution is initially periodic, but is set in motion. Each atom is first displaced to the right, but each atom oscillates with a slightly different frequency. We assume that the frequencies have a narrow spread $\Delta\omega$ about a central frequency ω_0. Likewise,

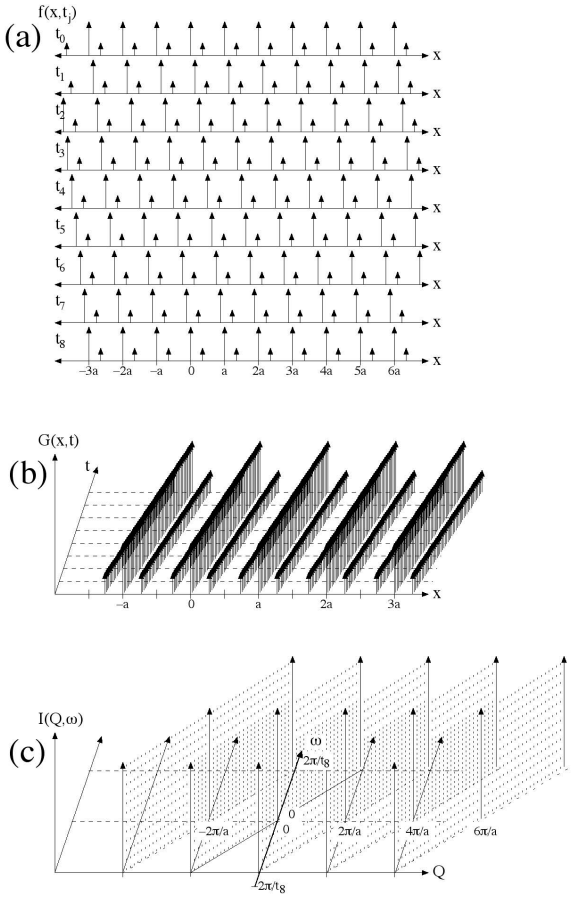


Fig. 3.2. (**a**) Variations of a scattering function in space and time. Nine snapshots in time are shown vertically. (**b**) The Van Hove function, obtained by overlapping all pairs of scattering factor distributions in (a), and summing the resulting product of overlaps for all pairs separated by the same number of time intervals. It was assumed that the scattering factor distribution moves to the right continuously between the time snapshots shown in (a). (**c**) The scattered intensity. It is assumed possible to both create and annihilate an excitation as shown in (a), hence points at $\pm\omega_0$.

we assume that the amplitudes of the displacements are not large, so each arrow does not travel far from its lattice site. Eventually the oscillation damps away, and the scatterers are back to their initial periodic configuration. It is easiest to first analyze the situation for long times. Here the scatterers are in periodic positions, and have stopped moving. As the time interval becomes long with respect to the damping time, most of the spatial correlations will involve correlations between precisely periodic structure. For long times, $G(x,t)$ will exhibit a set of peaks at equal intervals of $x = a$.

At short time separations, the individual arrows in Fig. 3.3b have not displaced much. At the shortest times after the neutron impact, the scatterers and have nearly the same displacements since they move nearly in phase. It is incorrect, however, to assume that the peaks in $G(x,t)$ are displaced to the right. Recall that it is an average over many different time-separated snapshots of $f(x,t)$ in Fig. 3.3a, many of which are moving to the left. For

small t, $G(x,t)$ is therefore similar to that of a perfect crystal in its spatial periodicities, with sharp peaks at each lattice translation.

At intermediate times, however, this this regularity is lost, even between neighboring arrows in this example. Figure 3.3c shows the intensity from this $G(x,t)$. For $\omega = 0$ the situation is as in Fig. 2.11, but for $\omega = 2\pi/t_8$ the intensity is broadened along the ω axis, owing the spread in frequencies. It is also expected that the disorder along the time domain will produce extra incoherent scattering between the sharp peaks in $G(x,t)$.

The initial coherence of the arrows is lost at intermediate times (assuming that the dephasing time $2\pi/\Delta\omega$ is less than the damping time). If we assume that there are no short-range spatial correlations, we have precisely the situation considered in Sect. 2.4.7. Figures 2.10 and 2.11 show the situation for long times, or equivalently for $\omega = 0$ in the $I(Q,\omega)$ of Fig. 3.3.

3.1.4 Autocorrelation Functions

The Van Hove function is worthy of deep respect, because it includes all the information available from all types of scattering: (coherent, inelastic), (coherent, elastic), (incoherent, inelastic), (incoherent, elastic). Subsets of this total information are often important, perhaps because the scattering is primarily incoherent, or is primarily elastic, for example. In particular, two important, although less general, correlation functions are:

- The Patterson function, $P(\boldsymbol{r})$, contains the spatial information obtained from diffraction measurements with *elastic coherent* scattering.
- The "self-correlation" function, $G_{\mathrm{s}}(t)$, contains the time information obtained from measurements of *incoherent inelastic* scattering. It involves an averaging over Q.

General Concept of Patterson and Self-Corrrelation Functions. Figure 3.4 shows the essential ideas. A "snapshot in time" is obtained by horizontal sampling across the figure. In a real experiment the samplings will be at many different times, but here we see that all such snapshots have the same periodic structure. As we know from Sect. 2.4.3, the intensity for this time snapshot is obtained by the Patterson function, whose periodicity in space gives the Bragg diffractions. In this particular case of a sine wave, however, we have Bragg peaks only at $Q = 0$ and $Q = \pm g$. If we had a series of δ-functions with the same periodicity, however, we would have a series of Bragg peaks to arbitrarily large orders of $Q = ng$. To obtain the Patterson function, the horizontal lines in Fig. 3.4a, each at a particular t_k, are convoluted with themselves, and the results are averaged for all t_k. The $f(x,t_k)$ for dfferent t_k in Fig. 3.4a do not interact with each other until after the convolution is performed. Specifically:

$$P(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f^*(x',t')\; f^*(x'+x,t')\,\mathrm{d}x'\,\mathrm{d}t' \;, \tag{3.23}$$

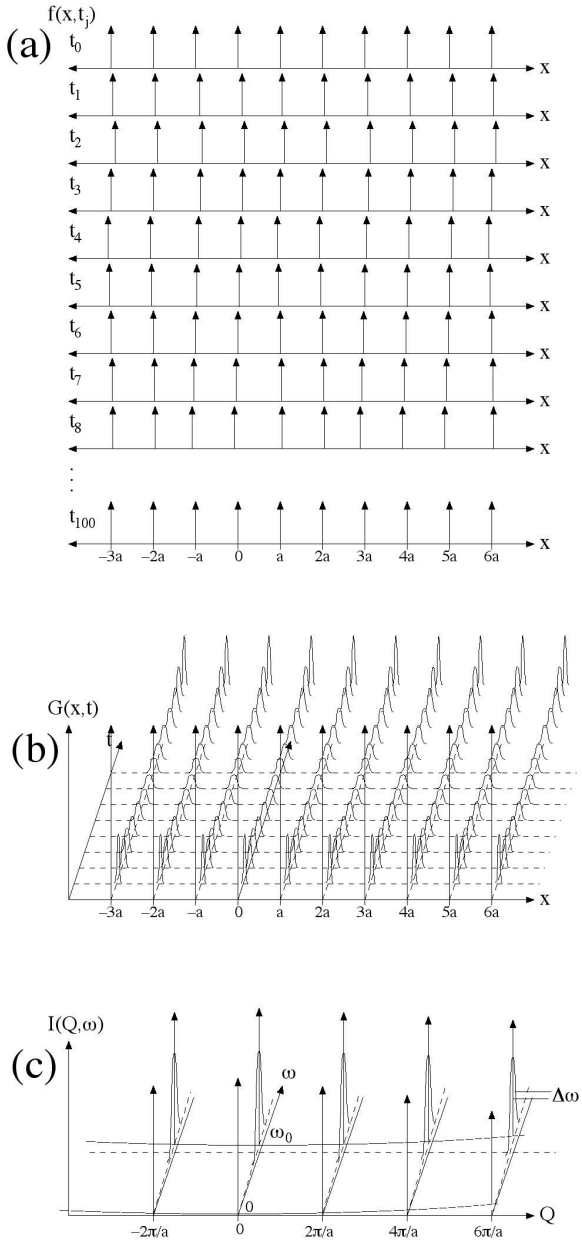


Fig. 3.3. (**a**) Variations of a scattering function in space and time. Nine snapshots in time are shown vertically. (**b**) The Van Hove function, obtained by overlapping all pairs of scattering factor distributions in (a), and summing the resulting product of overlaps for all pairs separated by the same number of time intervals. (**c**) The scattered intensity. It is assumed possible to both create and annihilate an excitation as shown in (a), hence points at $\pm\omega_0$.

where we use the same value of t' in both arguments of the scattering factor. The $P(x)$ includes an averaging over time, but does not consider correlations in time. This information is probed by elastic scattering experiments, which do not allow for measuring the time variations and hence energy transfers.

Likewise we can take a "snapshot in space" by taking a vertical sampling of the moving wave, at $\boldsymbol{r}_1$ for example (although the results for all other values of $\boldsymbol{r}_j$ are the same in this example). Notice how the wave in Fig. 3.4 shows a time periodicity along a vertical line. This time periodicity of our moving wave is, of course, ω, in the wave function: $\mathrm{e}^{\mathrm{i}Qx-\omega t}$. The restriction to a single value of Q eliminates the possibility of obtaining any spatial information on the excitation, but the ω-dependence is as expected. We obtain the time correlations for a specific position:

$$G_{\mathrm{s}}(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f^*(x',t')\; f^*(x',t'+t)\,\mathrm{d}x'\,\mathrm{d}t'\;, \tag{3.24}$$

where we use the same value of x' in both arguments of the scattering factor. The $G_{\mathrm{s}}(t)$ includes an averaging over position, but does not consider correlations in position. This information is probed by incoherent inelastic scattering experiments, which do not allow for measuring the Q-dependences and hence spatial information.

More information comes by identifying correlations in both time and space, however, as is indicated by the diagonal line in Fig. 3.4. This requires the full Van Hove function, $G(\boldsymbol{r},t)$, without summing the correlation functions of different space or time shapshots to obtain $G_{\mathrm{s}}(t)$ or $P(\boldsymbol{r})$. Nevertheless, it is instructive to evaluate the Patterson function and self-correlation function for each of the three examples shown in Figs. 3.1, 3.2, 3.3.

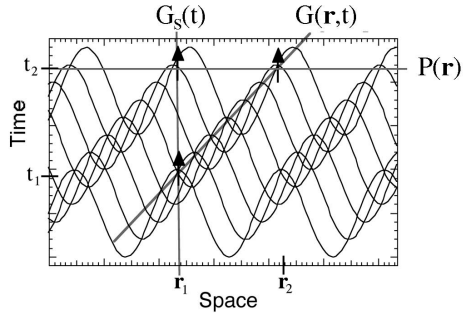


Fig. 3.4. A simple sine wave, moving to the right with increasing time. Three methods of sampling the intensity of the wave in space and time are shown as straight lines, as explained in the text. The relevant correlation functions for each sampling are shown for each line.

Local Excitations. Along the horizontal axis of $\omega = 0$ in Fig. 3.1c, we obtain a diffraction pattern with strong fundamentals at $Q = \pm n\pi/a$, where n is an even integer, and somewhat weaker superlattice diffractions for odd n. The same result is true for two other values of $\omega = \pm 2\pi/t_8$. This periodic

structure in Q is evident from the Van Hove function $G(x,t)$ in Fig. 3.1b. The spatial correlation function of (3.23), $P(x) = \int\int f(x',t_k)\, f(x'+x,t_k)\mathrm{d}x'\mathrm{d}t_k$ has spatial periodicities that are similar at any time t_k, although different weights for the superlattice diffractions. The average over all t_k is of course what makes the final result for $I(Q,\omega=0)$ in Fig. 3.1c.

Now look at the correlation functions along lines parallel to the time axis for fixed x' in $G(x',t)$ of Fig. 3.1b. The time periodicity of $G(x',t)$ in Fig. 3.1b differs in phase for even and odd n in $x' = \pm na$. All time periodicities are the same, however, and include only the frequencies $-2\pi/t_8, 0, -2\pi/t_8$.[2] The frequency spectrum of $G(x',t)$, measured along $x' = 0$, is generally the same as the inelastic scattering along any other $x' = na$. If we average the intensities along these individual x', we lose the information on phases of scattering between the scatterers at different x'. The mixed constructive and destructive interference between these different scatterers results in intensities between individual atoms only. The result is a "self-correlation" function of (3.24), or a case of incoherent inelastic scattering, $G_\mathrm{s}(t) = \int\int f(x_j,t')\, f(x_j+x,t'+t)\mathrm{d}x_j\mathrm{d}t'$. In this particular case, the incoherent average of these (obtained by summing the intensities at all Q) looks generally the same as the spectrum along a slice along any particular Q.

Dispersive Excitations. The Patterson and self-correlation functions of Fig. 3.2 can be understood in much the same way. The largest horizontal periodicity in Fig. 3.2b is a, so this Patterson function gives diffraction peaks at multiples of $2\pi/a$. The self-correlation function likewise has the simple periodicities of $-2\pi/t_8, 0, -2\pi/t_8$, neglecting the detailed structure in cuts along the time axis that would give higher-order Fourier components. More interesting are the dispersions of different excitations that have the same wave speed (presented as dashed vertical lines in Fig. 3.2c). These longer-wavelength excitations could be obtained by periodically deleting rows of arrows in Fig. 3.2b, for example. All these excitations would be measured in an incoherent inelastic scattering experiment, since the averaging over all x' of $G(x',t)$ in Fig. 3.2b would intersect all time periodicities. An incoherent inelastic scattering experiment would collapse all the inelastic intensity onto a single ω-axis, and lose the relationship between ω and Q that is so evident from the coherent inelastic scattering shown in Fig. 3.2c. For dispersive excitations, the self-correlation function has considerably less information compared to the full Van Hove function. On the other hand, if one is interested in obtaining an accurate density of states (i.e., a total energy spectrum), irrespective of the values of Q, the self-correlation function may be advantageous. It does not require accurate measurements of intensities over all values of Q and ω.

Disordered Excitations. Finally we consider the Patterson function and self-correlation function of Fig. 3.3. The scatterers oscillate about their lattice

[2] Incidentally, if the arrows were to oscillate through zero in Fig. 3.1a, we would have no value of $\omega = 0$, and no elastic scattering. This corresponds to half the scatterings occurring out-of-phase with the other half.

sites, and the Patterson function is therefore as expected of a perfect crystal. For the time average, there is displacement disorder in the arrangement, however, so we expect a diffuse background, increasing with Q as shown in Fig. 2.11. The self-correlation function for incoherent inelastic scattering is the same for all x, since the scatterers are assumed independent in their motions, without positional correlations that may case neighbors to move together in phase. In this case we expect the same time structure for $G_{\rm s}(t)$ and $G(x,t)$, and hence inelastic incoherent scattering will have the same energy spectrum as the coherent inelastic scattering for all Q. Both will show a broadening in frequency of the excitation around ω_0. The broadening arises from two effects. First is the frequency spread of the oscillators, postulated to be $\Delta\omega$. The second effect is the damping of the oscillations. The damping time of τ provides a broadening in energy of $\hbar/\tau$, known as "lifetime broadening." In the present problem we have assumed that the broadening of the excitation at ω_0 is dominated by the frequency spread $\Delta\omega$, since the dephasing time was assumed shorter than the damping time.

3.2 Relationships Between Intensities, Correlation Functions, Waves, and Scattering Lengths

Much of neutron scattering science involves relationships between the neutron wavefunction $\psi(Q,\omega)$ and physical scattering lengths $f(r,t)$ in the sample. Many important functional relationships are through Fourier transformation F, autocorrelation $\circledast$, multiplication $||^2$, or averaging $\langle\rangle$. A map of the important physical functions and how they are transformed into one another is presented in Fig. 3.5.

When using Fig. 3.5, reverse transformations are possible in all cases, but information is lost by averaging over an argument of a function. For example, $I(\omega)$ can be transformed to $S(Q,\omega)$ in the incoherent approximation, but dispersive information about $\omega(Q)$ cannot be obtained. (Of course, if the scattering is incoherent, there is no dispersive information and this is not a problem.) Averaging over ω is not shown in Fig. 3.5 because a more common manipulation in neutron scattering science is to identify the elastic scattering from inelastic data. Finally, the functions $f(r,\omega)$ and $\psi(Q,t)$ are not shown. They may prove useful as intermediate steps in some calculations, but they mix the phase information about the neutron and the scatterer. Some definitions are:

- $S(Q,\omega)$ is the "scattering law."
- $Y(Q,\tau)$ is the (unnamed) momentum-time corrrelation function.
- $\Gamma(R,\omega)$ is the (unnamed) space-energy corrrelation function.
- $G(R,\tau)$ is the Van Hove space-time correlation function.
- $P(R)$ is the Patterson function.

- $\mathcal{M}(\tau)$ is the "memory function," a time correlation function for dynamics at a site (sometimes called $G_{\rm s}(\tau)$, a self-correlation function).
- $I(Q)$ is a diffraction pattern.
- $I(\omega)$ is an inelastic spectrum.
- $f(r,t)$ is a scattering length density.
- $\psi(Q,\omega)$ is a neutron wavefunction.

The two unnamed correlation functions in Fig. 3.5 have special uses. The function $\mathit{\Gamma}(R,\omega)$ is used for projections of excitations onto specific sites, such as projected densities of states that describe the vibrational spectrum of a particular atom. The function $Y(Q,\tau)$ is useful for studies of transient phenomena that may occur after an impulsive perturbation.

3.3 General Formulation of Nuclear Scattering

A more general treatment of nuclear scattering is possible than was presented in the previous Sect. 2.3.5. It starts with Fermi's golden rule, and avoids the explicit use of wavefunctions for the scatterer (i.e., it does not require phonon solutions like $\boldsymbol{u}_{l,\kappa}(\boldsymbol{q},t)$ of (2.71)).

3.3.1 Fermi's Golden Rule

Fermi's golden rule gives the transition rate from an initial state to a final state, W, at time t':

$$W(t') = \frac{2\pi}{\hbar}|\langle\Psi_{\rm f}(\boldsymbol{r},t')|V(\boldsymbol{r},t')|\Psi_{\rm i}(\boldsymbol{r},t')\rangle|^2 \ . \tag{3.25}$$

For nuclear scattering of a neutron, the states $|\Psi\rangle$ include coordinates for the neutron and coordinates for the crystal. The nuclear forces of interaction are very short range, compared to the atom motions in the crystal, so the potential in (3.25) is $V_{\rm nuc}(\boldsymbol{r},t')$, which involves the coordinates of the neutron and nuclei only. It includes the crystal only insofar as the positions of the nuclei alter the location of the δ-functions of the Fermi pseudopotential of (2.60). Moving the atoms will connect the neutron and crystal coordinates through the conservation of momentum and energy only. The electronic interactions between the atoms are not affected by the neutron, however, and the forces between the nucleus and the neutron and the nucleus are not affected by the atom positions. We can therefore separate the state $|\Psi_{\rm i}(\boldsymbol{r},t)\rangle$ into a lattice part $|\lambda_{\rm i}\rangle$ and a neutron part $|\boldsymbol{k}_{\rm i}\rangle$:

$$|\Psi_{\rm i}(\boldsymbol{r},t)\rangle = |\lambda_{\rm i}(\boldsymbol{r}_{\rm nu},t)\rangle\,|\boldsymbol{k}_{\rm i}(\boldsymbol{r}_{\rm ne},t)\rangle \ , \tag{3.26}$$

where the independent coordinates $\boldsymbol{r}_{\rm nu}$ and $\boldsymbol{r}_{\rm ne}$ refer to the postions of the nucleus and neutron. We have assumed the neutron states are plane-wave states characterized by the wavevector $\boldsymbol{k}_{\rm f}$ (as in (2.28) and (2.47), but here

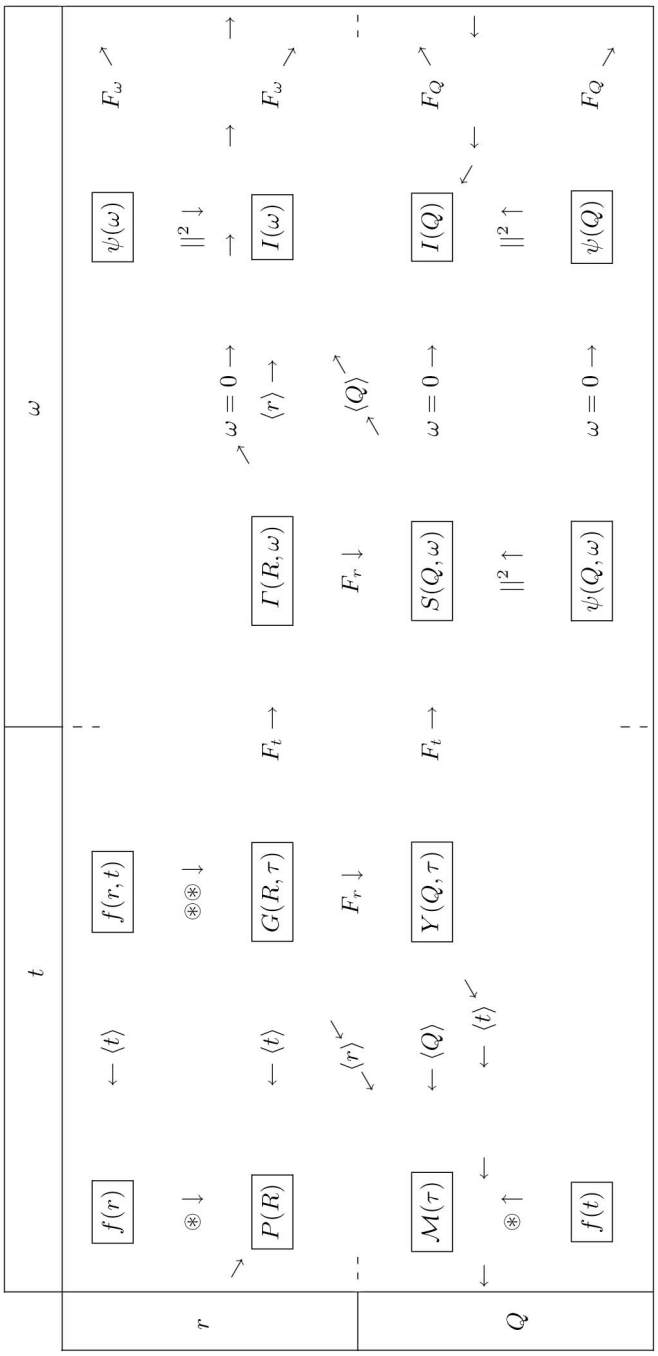


Fig. 3.5. Interrelationships between Correlation Functions, Scattering Length Densities, Intensities, Wavefunctions

$k_\mathrm{i} \neq k_\mathrm{f}$). In practice, the $\langle\,|\,|\,\rangle$ in (3.25) denotes an integration over all positional coordinates at the instant t' when W is evaluated. To get the total probability of the transition, $P_{\mathrm{i}\to\mathrm{f}}$, we integrate this over all times when the two states interact in the presence of the perturbing potential $V(\boldsymbol{r},t)$:

$$P_{\mathrm{i}\to\mathrm{f}} = \frac{2\pi}{\hbar}\int \langle\lambda_\mathrm{i}(\boldsymbol{r},t)|\,\langle \boldsymbol{k}_\mathrm{i}(\boldsymbol{r},t)|V^*(\boldsymbol{r},t)|\lambda_\mathrm{f}(\boldsymbol{r},t)\rangle\,|\boldsymbol{k}_\mathrm{f}(\boldsymbol{r},t)\rangle \\ \langle\lambda_\mathrm{f}(\boldsymbol{r},t)|\,\langle \boldsymbol{k}_\mathrm{f}(\boldsymbol{r},t)|V(\boldsymbol{r},t)|\lambda_\mathrm{i}(\boldsymbol{r},t)\rangle\,|\boldsymbol{k}_\mathrm{i}(\boldsymbol{r},t)\rangle\;\mathrm{d}t\;, \tag{3.27}$$

where we have substituted (3.26) into (3.25), written out the $|\,|^2$, and integrated over all times.

Recall that the time evolution of the state of the scatterer is:

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}|\psi(t{=}0)\rangle\;, \tag{3.28}$$
$$\langle\psi(t)| = \langle\psi(t{=}0)|\mathrm{e}^{+\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}\;, \tag{3.29}$$

We assume that the states of the crystal $\{|\lambda_\mathrm{j}\rangle\}$ are eigenstates with specific energies $\{\varepsilon_\mathrm{j}\}$, so we can simplify (3.28) and (3.29) using $\varepsilon_\mathrm{i} = \hbar\omega_\mathrm{i}$ as:

$$|\lambda_\mathrm{i}(t)\rangle = \mathrm{e}^{-\mathrm{i}\omega_\mathrm{i}t}|\lambda_\mathrm{i}(t{=}0)\rangle\;, \tag{3.30}$$
$$\langle\lambda_\mathrm{i}(t)| = \langle\lambda_\mathrm{i}(t{=}0)|\mathrm{e}^{+\mathrm{i}\omega_\mathrm{i}t}\;, \tag{3.31}$$

The general formulation makes use of these expressions because they allow us to work with states of the crystal at $t = 0$ such as $|\lambda_1(0)\rangle$ and $|\lambda_2(0)\rangle$, which are constant and can be pulled out of any integration over time. Similarly for the position evolution of plane-wave states of the neutron $\{|\boldsymbol{k}\rangle\}$, which have constant momentum:[3]

$$|\boldsymbol{k}_\mathrm{i}(\boldsymbol{r})\rangle = \mathrm{e}^{-\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\underline{\boldsymbol{r}}}|\boldsymbol{k}_\mathrm{i}(\boldsymbol{r}{=}0)\rangle\;, \tag{3.32}$$
$$\langle\boldsymbol{k}_\mathrm{i}(\boldsymbol{r})| = \langle\boldsymbol{k}_\mathrm{i}(\boldsymbol{r}{=}0)|\mathrm{e}^{+\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\underline{\boldsymbol{r}}}\;. \tag{3.33}$$

Returning to the evaluation of (3.27), into it we substitute (3.30), (3.31) and (3.32), (3.33):

$$P_{\mathrm{i}\to\mathrm{f}} = \frac{2\pi}{\hbar}\int \langle\lambda_\mathrm{i}(0)|\mathrm{e}^{\mathrm{i}\omega_\mathrm{i}t}\,\langle\boldsymbol{k}_\mathrm{i}(0)|\mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\underline{\boldsymbol{r}}}\,V^*(\boldsymbol{r},t)\,\mathrm{e}^{-\mathrm{i}\boldsymbol{k}_\mathrm{f}\cdot\underline{\boldsymbol{r}}}|\boldsymbol{k}_\mathrm{f}(0)\rangle\,\mathrm{e}^{-\mathrm{i}\omega_\mathrm{f}t}|\lambda_\mathrm{f}(0)\rangle \\ \langle\lambda_\mathrm{f}(0)|\mathrm{e}^{\mathrm{i}\omega_\mathrm{f}t}\,\langle\boldsymbol{k}_\mathrm{f}(0)|\mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{f}\cdot\underline{\boldsymbol{r}}}\,V(\boldsymbol{r},t)\,\mathrm{e}^{-\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\underline{\boldsymbol{r}}}|\boldsymbol{k}_\mathrm{i}(0)\,\rangle\mathrm{e}^{-\mathrm{i}\omega_\mathrm{i}t}|\lambda_\mathrm{i}(0)\rangle\;\mathrm{d}t\;, \tag{3.34}$$

where the operators $\underline{\boldsymbol{r}}$ refer to the neutron coordinates, and the (0) refer to $t = 0$ and $\boldsymbol{r} = 0$ (although we are not concerned about the time-dependence of the neutron wavefunction or the position-dependence of the crystal excitation).

The next step in simplifying (3.34) is to substitute the scattering potential for V. As before, we use the sum of Fermi pseudopotentials at all crystal sites (i.e., (2.62)). These δ-functions are most convenient in selecting the $\boldsymbol{r}$ where the neutron sees the nuclei, because the integrations over spatial coordinates become trivial — we delete the integral over position, and in each matrix

[3] Plane waves prove their convenience here. For other states we would have to integrate over k-space.

element we substitute the nuclear positions with $\boldsymbol{R}_j$ and $\boldsymbol{R}_k$, and the times t_j and t_k. We sum over all nuclei in the crystal:

$$P_{\mathrm{i}\to\mathrm{f}} = \frac{2\pi}{\hbar}\int \sum_j \langle\lambda_\mathrm{i}(0)|\mathrm{e}^{\mathrm{i}\omega_\mathrm{i}t_j}\,\langle\boldsymbol{k}_\mathrm{i}(0)|\mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\underline{\boldsymbol{R}}_j}\,b_j^*\,\mathrm{e}^{-\mathrm{i}\boldsymbol{k}_\mathrm{f}\cdot\underline{\boldsymbol{R}}_j}|\boldsymbol{k}_\mathrm{f}(0)\rangle\,\mathrm{e}^{-\mathrm{i}\omega_\mathrm{f}t_j}|\lambda_\mathrm{f}(0)\rangle$$
$$\sum_k \langle\lambda_\mathrm{f}(0)|\mathrm{e}^{\mathrm{i}\omega_\mathrm{f}t_k}\,\langle\boldsymbol{k}_\mathrm{f}(0)|\mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{f}\cdot\underline{\boldsymbol{R}}_k}\,b_k\,\mathrm{e}^{-\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\underline{\boldsymbol{R}}_k}|\boldsymbol{k}_\mathrm{i}(0)\rangle\mathrm{e}^{-\mathrm{i}\omega_\mathrm{i}t_k}|\lambda_\mathrm{i}(0)\rangle\,\mathrm{d}t\ . \quad (3.35)$$

The notation (0) refers to both $t = 0$ and $\boldsymbol{r} = 0$, since we made use of the relations (3.30)–(3.33). At $t = 0$), $\boldsymbol{r} = 0$, the phase factors, $\mathrm{e}^{\mathrm{i}((Q)\cdot\boldsymbol{r}-\omega t)}$, of the plane-wave states of (2.28) and (2.47) are equal to 1. Therefore:

$$\langle\boldsymbol{k}_\mathrm{i}(0)|\boldsymbol{k}_\mathrm{f}(0)\rangle\langle\boldsymbol{k}_\mathrm{f}(0)|\boldsymbol{k}_\mathrm{i}(0)\rangle = 1\ . \quad (3.36)$$

We define the difference in frequency as ω:

$$\omega_i - \omega_f \equiv \omega\ . \quad (3.37)$$

Substituting (3.36) and (3.37) into (3.35):

$$P_{\mathrm{i}\to\mathrm{f}} = \frac{2\pi}{\hbar}\int \sum_j \mathrm{e}^{\mathrm{i}\omega t_j} b_j^*\langle\lambda_\mathrm{i}(0)|\,\mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\underline{\boldsymbol{R}}_j(t_j)}\mathrm{e}^{-\mathrm{i}\boldsymbol{k}_\mathrm{f}\cdot\underline{\boldsymbol{R}}_j(t_j)}|\lambda_\mathrm{f}(0)\rangle$$
$$\sum_k \mathrm{e}^{-\mathrm{i}\omega t_k} b_k\langle\lambda_\mathrm{f}(0)|\mathrm{e}^{\mathrm{i}\boldsymbol{k}_\mathrm{f}\cdot\underline{\boldsymbol{R}}_k(t_k)}\,\mathrm{e}^{-\mathrm{i}\boldsymbol{k}_\mathrm{i}\cdot\underline{\boldsymbol{R}}_k(t_k)}|\lambda_\mathrm{i}(0)\rangle\,\mathrm{d}t\ , \quad (3.38)$$

The integration is over all times, so we can redefine t as a difference between scattering times:

$$t \equiv t_j - t_k\ , \quad (3.39)$$

and likewise we define the scattering vector $\boldsymbol{Q}$:

$$Q \equiv \boldsymbol{k}_\mathrm{i} - \boldsymbol{k}_\mathrm{f}\ . \quad (3.40)$$

Substituting these differences into (3.38):

$$P_{\mathrm{i}\to\mathrm{f}} = \frac{2\pi}{\hbar}\int \mathrm{e}^{\mathrm{i}\omega t} \sum_j b_j^*\langle\lambda_\mathrm{i}(0)|\,\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_j(t)}|\lambda_\mathrm{f}(0)\rangle$$
$$\sum_k b_k\langle\lambda_\mathrm{f}(0)|\mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_k(0)}|\lambda_\mathrm{i}(0)\rangle\,\mathrm{d}t\ , \quad (3.41)$$

The next step is to consider changes in the state of the crystal, the $\{|\lambda(0)\rangle\}$. First consider the final states, $\{|\lambda_\mathrm{f}(0)\rangle\}$. We do not have control over which final state is obtained, and in prinicple all acceptable final states will occur over the duration of a long experiment. In effect, an experiment sums over final states, but this sum is special. Since the final states are assumed to form a complete set:

$$\sum_\mathrm{f} |\lambda_\mathrm{f}(0)\rangle\langle\lambda_\mathrm{f}(0)| = 1\ , \quad (3.42)$$

for each term in the double sum (3.41). The initial states $\{|\lambda_{\rm i}(0)\rangle\}$ can be controlled by thermodynamics. Temperature will alter the distribution of intial states by the appropriate thermodynamic distribution, i.e., a Bose–Einstein distribution for phonons. Instead of writing this distribution $n(\varepsilon)$ directly, we define the brackets $\langle\rangle$ to mean the thermodynamic average:

$$P_{\rm i\to f} = \frac{2\pi}{\hbar}\int {\rm e}^{{\rm i}\omega t}\sum_j\sum_k b_j^* b_k \left\langle {\rm e}^{{\rm i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_j(t)}\,{\rm e}^{-{\rm i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_k(0)}\right\rangle {\rm d}t\ , \tag{3.43}$$

$$P_{\rm i\to f} = \frac{2\pi}{\hbar}\int {\rm e}^{{\rm i}\omega t}\sum_j\sum_k b_j^* b_k \left\langle {\rm e}^{{\rm i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_j(t)}\,{\rm e}^{-{\rm i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_k(0)}\right\rangle {\rm d}t\ , \tag{3.44}$$

We make use of the prefactors described in Sect. 2.3.5 to convert the transition probability into a cross-section:

$$\frac{{\rm d}^2\sigma}{{\rm d}\Omega\,{\rm d}E} = \frac{k_{\rm f}}{k_{\rm i}}\frac{(2\pi)^3}{2V_0}\sum_j\sum_k b_j^* b_k \int {\rm e}^{{\rm i}\omega t}\left\langle {\rm e}^{{\rm i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_j(t)}\,{\rm e}^{-{\rm i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_k(0)}\right\rangle {\rm d}t\ . \tag{3.45}$$

Equation (3.45) is the most general result that can be obtained from Fermi's Golden Rule. At its heart is a thermodynamic average of phase factors from scattering by atoms j and k at their different positions in space and at different snapshots in time. Note that the space coordinates for the two atoms are generally not evaluated at the same time. If there were no time dependence to the atom positions, the Fourier transform would yield a delta function $\delta(\omega - 0)$, indicating pure elastic scattering. If there were no spatial periodicities, the thermodynamic average over all phase factors would not produce any constructive or destructive interferences at different angles, so there would be no structure of the cross section in solid angle, Ω. The topic of this book is inelastic scattering, so we assume motion of the scatterers, and we will be sensitive to the how the two phase factors change with time. The upcoming approximation will address the thermal spread of atom positions over time. Specifically we will assume this is a Gaussian function, or at least it is small. For the spatial periodicities of the scatterers, we will assume the translational periodicity of a crystal. This is a more restrictive assumption about the sample, and can be misleading in cases of disordered solids. It is therefore sometimes important to return to (3.45) for guidance on the scattering problem, since the only assumption is that the neutron is scattered one time.

3.3.2 Detailed Balance

The intensities of inelastic spectra depend on the ratio of energy transfer to temperature, at least in the usual case where the sample is in thermodynamic equilibrium before scattering. There is a detailed balance between the rates of two scattering processes, one with the creation of an excitation, and the other with the annihilation of the same excitation. Consider the temperature

dependence of the positive and negative energy transfers between the neutron and the specimen. One extreme is when the sample is at a very low temperature (in practice, where $k_{\mathrm{B}}T$ is much smaller than the energy resolution of the instrument). In this case there are no excitations present in the sample, so no scattering can occur with the annihilation of excitations. Excitations can be created by transfer of energy from the incident neutron, of course. At low temperatures the inelastic spectrum will have intensity on one side of the elastic peak, but no intensity on the other side. The other extreme is for very high temperatures (in practice, at temperatures where $k_{\mathrm{B}}T$ is much larger than largest energy transfer measured). Because the probability of creating or annihilating one additional excitation such as a phonon makes little difference to the energy of the sample, we expect the quantization of excitations to be less obvious. Scattering processes involving the creation or annihilation of excitations occur with similar probabilities when the sample is at high temperatures. Nevertheless, even at relatively high temperatures there is a measurable difference in the inelastic intensity on the loss and gain sides of the elastic peak. Each creation process has an inverse annihilation process, and in all cases these are in the ratio of a Boltzmann factor. The energy in this Boltzmann factor is the extra excitation energy required for the sample to be ready for the annihilation process. Since this Boltzmann factor ratio occurs in detail for each pair of creation and annihilation processes, this Boltzmann factor applies to all intensities across an inelastic spectrum.

The condition of detailed balance, (3.56) below, follows from two assumptions:

- The probability of the initial state of the sample, the $|\lambda_{\mathrm{i}}\rangle$ in (3.26), is as expected for thermodynamic equilibrium.
- The interaction operator for the transition probability, the $V(\boldsymbol{r}, t')$ of (3.25), is Hermitian. (This is certainly true for the delta function (2.60) for the Fermi pseudopotential.)

To show the essence of the derivation of the detailed balance condition, assume the initial state $|\lambda_1\rangle$ exists, and consider the probability, $W'_{1\to 2}$, for a transition to a final state $|\lambda_2\rangle$, and the probability for the reverse transition, $W'_{2\to 1}$ from a pre-existing state $|\lambda_2\rangle$:

$$W'_{1\to 2} = |\langle\lambda_2|V|\lambda_1\rangle|^2 \,, \tag{3.46}$$

$$W'_{2\to 1} = |\langle\lambda_1|V|\lambda_2\rangle|^2 \,, \tag{3.47}$$

which are both products of a number with its complex conjugate:

$$W'_{1\to 2} = \langle\lambda_2|V|\lambda_1\rangle \left(\langle\lambda_2|V|\lambda_1\rangle\right)^* \,, \tag{3.48}$$

$$W'_{2\to 1} = \left(\langle\lambda_1|V|\lambda_2\rangle\right)^* \langle\lambda_1|V|\lambda_2\rangle \,. \tag{3.49}$$

For a Hermitian operator, recall that $V = (V^{\mathrm{T}})^* \equiv V^\dagger$. We use the transpose to operate on the other side of V, for which we use the complex conjugates of the bras and kets:

$$W'_{1\to 2} = \langle\lambda_2|V|\lambda_1\rangle\ \langle\lambda_1|V|\lambda_2\rangle\ , \tag{3.50}$$

$$W'_{2\to 1} = \langle\lambda_2|V|\lambda_1\rangle\ \langle\lambda_1|V|\lambda_2\rangle\ , \tag{3.51}$$

so:

$$W'_{1\to 2} = W'_{2\to 1} \equiv W'\ . \tag{3.52}$$

This result (3.52) is true so long as V is Hermitian. Starting in the known states $|\lambda_1\rangle$ and $|\lambda_2\rangle$, the transition probabilites between these states are equal.

Now assume that the states $|\lambda_\mathrm{i}\rangle$ are populated in thermodynamic equilibrium, differing by a Boltzmann factor. The measured cross sections are proportional to $W_{1\to 2}$ and $W_{2\to 1}$:

$$W_{1\to 2} = \frac{\mathrm{e}^{-E_1/k_\mathrm{B}T}}{Z} W'_{1\to 2} = \frac{\mathrm{e}^{-E_1/k_\mathrm{B}T}}{Z} W'\ , \tag{3.53}$$

$$W_{2\to 1} = \frac{\mathrm{e}^{-E_2/k_\mathrm{B}T}}{Z} W'_{2\to 1} = \frac{\mathrm{e}^{-E_2/k_\mathrm{B}T}}{Z} W'\ , \tag{3.54}$$

where Z is the partition function. Now that we have taken into consideration the fact that the initial states have probabilities as expected from thermodynamic equilibrium, we can relate the observed intensity for the transition $1 \to 2$ to the observed intensity for its reverse transition $2 \to 1$:

$$W_{1\to 2} = \mathrm{e}^{(E_2-E_1)/k_\mathrm{B}T}\ W_{2\to 1}\ . \tag{3.55}$$

Suppose the state $|\lambda_2\rangle$ has an energy higher (more positive) than $|\lambda_1\rangle$, because the state $|\lambda_2\rangle$ has an extra excitation in the sample. The transition $1 \to 2$ is uphill energetically, and requires the neutron to transfer energy to create an excitation sample. Nevertheless, this transition is more intense experimentally because the initial state $|\lambda_1\rangle$ is more probable thermodynamically. Equation (3.55) shows that the intensity $W_{1\to 2} < W_{2\to 1}$ because $E_2 > E_1$ and the exponential is greater than 1. To clarify (3.55), we recognize that the difference in energy, $E = E_2 - E_1$, is the energy of the excitation in the solid. It is more traditional to write the condition of detailed balance as:

$$S(E) = \mathrm{e}^{E/k_\mathrm{B}T}\ S(-E)\ . \tag{3.56}$$

where E is the energy of the excition in the solid, and the argument $-E$ corresponding to $W_{2\to 1}$ signifies that $S(-E)$ is on the phonon annihilation side of the elastic peak in the spectrum. Detailed balance remains valid when a single scattering creates or annihilates multiple excitations – a detailed balance between forward and reverse processes still exists because the thermodynamic probabilities of the required initial states are set by E.

A practical use of detailed balance is to check the quality of experimental data. For example, in a spectrum measured at 300 K, equivalent to 25 meV, the intensities at ± 25 meV on the two sides of the elastic peak must be in the ratio of e^{-1}. If this were not true, we might suspect instrument artifacts, such as differences in sensitivity or resolution. A noise background could also

cause measured data to violate the condition of detailed balance, and perhaps detailed balance could be exploited to help subtract some sources of background from experimental data. We warn the reader, however, about such simple interpretations with data from a time-of-flight chopper spectrometer. The value of Q varies across the energy scale of data from a particular detector, and the relationship is not symmetric, i.e., $Q(E) \neq Q(-E)$. When multiphonon scattering is strong, and the Debye–Waller factor is significant, detailed balance will not be observed in the experimental data unless they are rebinned for constant Q.

3.3.3 Crystalline Periodicity

The translational periodicity of crystals allows the reduction of the double sum in (3.45) to a single sum. We separate the atom position operators, $\underline{\boldsymbol{R}}_j(t)$, into static, $\underline{\boldsymbol{x}}_{l,\kappa}$, and time-varying, $\underline{\boldsymbol{u}}_{l,\kappa}(t)$, components:

$$\underline{\boldsymbol{R}}_j(t) = \underline{\boldsymbol{x}}_l + \underline{\boldsymbol{x}}_\kappa + \underline{\boldsymbol{u}}_{l,\kappa}(t) \;, \tag{3.57}$$

where the static positions were broken into lattice vectors (index l) and basis vectors (index κ) in (3.57). The exponentials in (3.45) refer to pairs of atoms separated by a distance $\boldsymbol{x}_j - \boldsymbol{x}_k$. In an infinite crystal, where all unit cells are equivalent, these exponentials cannot depend on the absolute position of the unit cell, but only on the lattice translation vector $\boldsymbol{x}_l$. Any one of the N unit cells can be taken as the origin, and the terms from the l^{th} neighbor must be the same. This distance, $\boldsymbol{x}_l$, has no time dependence, and is a constant. It therefore commutes through the other distance operators, and we can write:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}} \frac{(2\pi)^3 N |b|^2}{2V_0} \sum_l \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l} \int \mathrm{e}^{\mathrm{i}\omega t} \left\langle \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{u}}_l(t)}\, \mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{u}}_0(0)} \right\rangle \mathrm{d}t \;. \tag{3.58}$$

Here, for simplicity we have also assumed all nuclei are the same so $b_j = b_k \equiv b$, and we have assumed a simple lattice without a basis. The result in (3.58) accounts for the interactions of waves scattered from pairs of nuclei, recognizing that their instantaneous displacements may differ by a phase factor. This difference in phase factor gives a constant prefactor $\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l}$. The other exponentials are not constants, however, and need to be handled with care. The temptation would be to combine the exponentials into a single factor such as $\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot(\underline{\boldsymbol{R}}_j(t) - \underline{\boldsymbol{R}}_j(0))}$. Unfortunately, this tempting step would be incorrect, except for classical systems. The quantum argument, described next, is subtle.

3.3.4 A Subtlety of Quantum Mechanics

Calculating the expectation value of an operator $\underline{\boldsymbol{A}}$ using the left- and right-hand sides of (3.28) and (3.29) gives:

$$\langle\psi(t)|\underline{\boldsymbol{A}}|\psi(t)\rangle = \langle\psi(t{=}0)|\mathrm{e}^{+\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}\underline{\boldsymbol{A}}\mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}|\psi(t{=}0)\rangle \;. \tag{3.59}$$

Evidently the time-dependence of the matrix element can be transferred from the state functions ψ to the operator $\underline{\boldsymbol{A}}$ by replacing the operator by $\mathrm{e}^{+\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}\underline{\boldsymbol{A}}\mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}$. This moves us into the "Heisenberg picture" of quantum mechanics where state functions are fixed, but the time-dependence is in the operator. The motivation for putting the dynamics into the operators is as follows. In passing from classical mechanics to quantum mechanics, we replace the position vector $\boldsymbol{R}(t)$ with an operator, denoted $\underline{\boldsymbol{R}}(t)$. In particular, we will alter soon work with phase factors that are a time-dependent operators:

$$\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{R}} \longrightarrow \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}} \,. \tag{3.60}$$

Changing $\boldsymbol{R}$ to the operator $\underline{\boldsymbol{R}}$ leads to a subtlety in calculating the intensity from the wave amplitude. It turns out that the operator $\underline{\boldsymbol{R}}(0)$ does not commute with the operator $\underline{\boldsymbol{R}}(t)$ for the same atom at a different time. The operators are related as:

$$\underline{\boldsymbol{R}}(t) = \mathrm{e}^{+\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}\underline{\boldsymbol{R}}(0)\mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar} \,, \tag{3.61}$$

$$\mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}\boldsymbol{R}(t) = \underline{\boldsymbol{R}}(0)\mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar} \,, \tag{3.62}$$

and likewise for the exponentiated operators:

$$\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}(t)} = \mathrm{e}^{+\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}(0)}\mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar} \,, \tag{3.63}$$

$$\mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar}\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}(t)} = \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}(0)}\mathrm{e}^{-\mathrm{i}\underline{\boldsymbol{H}}t/\hbar} \,. \tag{3.64}$$

We cannot interchange the order of $\underline{\boldsymbol{R}}$ and $\underline{\boldsymbol{H}}$ because $\underline{\boldsymbol{H}}$ includes the momentum operator.

To better understand how this works, we need a result about exponentiations of non-commuting operators. Operators in exponential functions are defined in terms of the power series expansion of the exponential. Consider two operators $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$:

$$\exp(\underline{\boldsymbol{A}}) = 1 + \underline{\boldsymbol{A}} - \frac{1}{2}\underline{\boldsymbol{A}}\,\underline{\boldsymbol{A}} + \dots \,, \tag{3.65}$$

$$\exp(\underline{\boldsymbol{B}}) = 1 + \underline{\boldsymbol{B}} - \frac{1}{2}\underline{\boldsymbol{B}}\,\underline{\boldsymbol{B}} + \dots \,, \tag{3.66}$$

Now take the product, including all terms to the second order:

$$\exp(\underline{\boldsymbol{A}})\exp(\underline{\boldsymbol{B}}) = \left[1 + \underline{\boldsymbol{A}} - \frac{1}{2}\underline{\boldsymbol{A}}\,\underline{\boldsymbol{A}} + \dots\right]\left[1 + \underline{\boldsymbol{B}} - \frac{1}{2}\underline{\boldsymbol{B}}\,\underline{\boldsymbol{B}} + \dots\right] \tag{3.67}$$

$$\begin{aligned}\exp(\underline{\boldsymbol{A}})\exp(\underline{\boldsymbol{B}}) = {} & 1 \quad + \quad \underline{\boldsymbol{B}} \quad - \quad \frac{1}{2}\underline{\boldsymbol{B}}\,\underline{\boldsymbol{B}} \\ & + \underline{\boldsymbol{A}} \quad - \quad \underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} \quad + \quad \mathcal{O}3 \\ & - \frac{1}{2}\underline{\boldsymbol{A}}\,\underline{\boldsymbol{A}} \; + \; \mathcal{O}3 \quad + \quad \mathcal{O}4 \end{aligned} \tag{3.68}$$

$$\begin{aligned}\exp(\underline{\boldsymbol{A}})\exp(\underline{\boldsymbol{B}}) = {} & 1 \quad + \quad \underline{\boldsymbol{A}} \quad + \quad \underline{\boldsymbol{B}} \\ & - \frac{1}{2}\Big[\underline{\boldsymbol{A}}\,\underline{\boldsymbol{A}} + 2\underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} + \underline{\boldsymbol{B}}\,\underline{\boldsymbol{B}}\Big] + \mathcal{O}3 \,, \end{aligned} \tag{3.69}$$

where we have been fastidious about keeping the operator $\underline{\boldsymbol{A}}$ to the left of the operator $\underline{\boldsymbol{B}}$, because they may not necessarily commute.

Now evaluate the exponentiation of the sum $\underline{\boldsymbol{A}}+\underline{\boldsymbol{B}}$, again to second order in the operators:

$$\exp(\underline{\boldsymbol{A}}+\underline{\boldsymbol{B}}) = 1 + (\underline{\boldsymbol{A}}+\underline{\boldsymbol{B}}) - \frac{1}{2}(\underline{\boldsymbol{A}}+\underline{\boldsymbol{B}})\,(\underline{\boldsymbol{A}}+\underline{\boldsymbol{B}}) + \mathcal{O}3\,, \tag{3.70}$$

$$\begin{aligned}\exp(\underline{\boldsymbol{A}}+\underline{\boldsymbol{B}}) = 1 \quad &+ \quad \underline{\boldsymbol{A}} \quad + \quad \underline{\boldsymbol{B}} \\ &- \frac{1}{2}\Big[\underline{\boldsymbol{A}}\,\underline{\boldsymbol{A}} + \underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} + \underline{\boldsymbol{B}}\,\underline{\boldsymbol{A}} + \underline{\boldsymbol{B}}\,\underline{\boldsymbol{B}}\Big] + \mathcal{O}3\,,\end{aligned} \tag{3.71}$$

Note that (3.69) and (3.71) are not equal unless $\underline{\boldsymbol{A}}$ commutes with $\underline{\boldsymbol{B}}$. In general, however:

$$\underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} \neq \frac{1}{2}\Big[\underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} + \underline{\boldsymbol{B}}\,\underline{\boldsymbol{A}}\Big] \quad \text{so :} \tag{3.72}$$

$$\exp(\underline{\boldsymbol{A}})\exp(\underline{\boldsymbol{B}}) \neq \exp(\underline{\boldsymbol{A}}+\underline{\boldsymbol{B}})\,. \tag{3.73}$$

In classical physics, of course we have $\exp(\boldsymbol{A})\exp(\boldsymbol{B}) = \exp\big[(\boldsymbol{A}+\boldsymbol{B})\big]$. We need to re-examine the results of Sect. 2.3.5 in light of the inequality of (3.73). Equation (3.72) shows that the difference between the two sides of (3.73) is approximately:

$$\underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} - \frac{1}{2}\Big[\underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} + \underline{\boldsymbol{B}}\,\underline{\boldsymbol{A}}\Big] = \frac{1}{2}\Big[\underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} - \underline{\boldsymbol{B}}\,\underline{\boldsymbol{A}}\Big]\,. \tag{3.74}$$

It turns out that the right side of (3.74) is the second term in a power series expansion of $\exp\Big(\frac{1}{2}\big[\underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} - \underline{\boldsymbol{B}}\,\underline{\boldsymbol{A}}\big]\Big)$, a factor that equates the two sides of (3.73). That is:

$$\exp(\underline{\boldsymbol{A}})\exp(\underline{\boldsymbol{B}}) = \exp(\underline{\boldsymbol{A}}+\underline{\boldsymbol{B}})\ \exp\Big(\frac{1}{2}\big[\underline{\boldsymbol{A}}\,\underline{\boldsymbol{B}} - \underline{\boldsymbol{B}}\,\underline{\boldsymbol{A}}\big]\Big)\,. \tag{3.75}$$

Proof of (3.75) is provided in Appendix I.1 of the excellent book by Squires, but here we have demonstrated its plausibility. An important point in the proof is that the commmutator $[\underline{\boldsymbol{A}}, \underline{\boldsymbol{B}}]$ is a pure number, rather than a differential operator. Fortunately this holds true when $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$ are atom displacements generated by phonons (obtained from the annihilation and creation operators, a and $a^\dagger$).

3.3.5 Gaussian Thermal Averages

Now that we have the expression (3.75), we can use it to rearrange (3.58) into a form that shows intensity contributions from different numbers of phonon excitations. To clarify the next steps, it is traditional to make the definitions:

$$\underline{\boldsymbol{U}} \equiv -\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{u}}_0(0)\,, \tag{3.76}$$

$$\underline{\boldsymbol{V}} \equiv \mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{u}}_l(t)\,. \tag{3.77}$$

Now we substitute (3.76) and (3.77) into (3.75):

$$\exp(\underline{\boldsymbol{U}})\exp(\underline{\boldsymbol{V}}) = \exp(\underline{\boldsymbol{U}}+\underline{\boldsymbol{V}})\ \exp\Big(\frac{1}{2}\big[\underline{\boldsymbol{U}}\,\underline{\boldsymbol{V}} - \underline{\boldsymbol{V}}\,\underline{\boldsymbol{U}}\big]\Big)\,. \tag{3.78}$$

At this point we need to take a thermal average (the $\langle\rangle$ in (3.58)). In doing so, we make the assumption that the vibrational atom displacements are distributed with a Gaussian spread. The Gaussian distribution function can be used to average a squared displacement, X^2:

$$\langle X^2\rangle = \int_{-\infty}^{\infty} X^2 \frac{1}{\sqrt{\pi\sigma^2}} \mathrm{e}^{-X^2/\sigma^2}\ \mathrm{d}X\ , \tag{3.79}$$

$$\langle X^2\rangle = \frac{1}{2}\sigma^2\ , \tag{3.80}$$

a standard result.

For comparison, we next average an exponential e^X:

$$\langle \mathrm{e}^X\rangle = \int_{-\infty}^{\infty} \mathrm{e}^X \frac{1}{\sqrt{\pi\sigma^2}} \mathrm{e}^{-X^2/\sigma^2}\ \mathrm{d}X\ , \tag{3.81}$$

by completing the square:

$$-\left(X/\sigma - \sigma/2\right)^2 = -\left(X^2/\sigma^2 - X + \sigma^2/4\right)\ , \tag{3.82}$$

so by adding and subtracting the last term of (3.82) in the exponential:

$$\langle \mathrm{e}^X\rangle = \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi\sigma^2}} \mathrm{e}^{-(X/\sigma-\sigma/2)^2} \mathrm{e}^{\sigma^2/4}\ \mathrm{d}X\ , \tag{3.83}$$

$$\langle \mathrm{e}^X\rangle = \mathrm{e}^{\sigma^2/4} \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi\sigma^2}} \mathrm{e}^{-(X/\sigma-\sigma/2)^2}\ \mathrm{d}X\ , \tag{3.84}$$

$$\langle \mathrm{e}^X\rangle = \mathrm{e}^{\sigma^2/4}\ , \tag{3.85}$$

where the last result was obtained by integrating the normalized Gaussian function. By comparing (3.80) and (3.85), we obtain:

$$\langle \mathrm{e}^X\rangle = \mathrm{e}^{X^2/2}\ . \tag{3.86}$$

Using (3.86), we obtain a factor with our operators $\underline{\boldsymbol{U}}$ and $\underline{\boldsymbol{V}}$:

$$\langle \exp(\underline{\boldsymbol{U}} + \underline{\boldsymbol{V}})\rangle = \exp\left(\frac{1}{2}\left(\underline{\boldsymbol{U}} + \underline{\boldsymbol{V}}\right)^2\right)\ . \tag{3.87}$$

We use this result to obtain the thermal average of (3.78):

$$\langle \exp \underline{\boldsymbol{U}} \exp \underline{\boldsymbol{V}}\rangle = \exp\left(\frac{1}{2}\left(\underline{\boldsymbol{U}} + \underline{\boldsymbol{V}}\right)^2\right)\ \exp\left(\frac{1}{2}\langle \underline{\boldsymbol{U}}\,\underline{\boldsymbol{V}} - \underline{\boldsymbol{V}}\,\underline{\boldsymbol{U}}\rangle\right)\ , \tag{3.88}$$

$$\langle \exp \underline{\boldsymbol{U}} \exp \underline{\boldsymbol{V}}\rangle = \exp\left(\frac{1}{2}\langle \underline{\boldsymbol{U}}^2 + \underline{\boldsymbol{V}}^2 + \underline{\boldsymbol{U}\boldsymbol{V}} + \underline{\boldsymbol{V}\boldsymbol{U}} + \underline{\boldsymbol{U}\boldsymbol{V}} - \underline{\boldsymbol{V}\boldsymbol{U}}\rangle\right) \tag{3.89}$$

$$\langle \exp \underline{\boldsymbol{U}} \exp \underline{\boldsymbol{V}}\rangle = \exp\left(\frac{1}{2}\langle \underline{\boldsymbol{U}}^2 + \underline{\boldsymbol{V}}^2\rangle\right)\ \exp\langle \underline{\boldsymbol{U}\boldsymbol{V}}\rangle\ . \tag{3.90}$$

By examining (3.76) and (3.77), we can see that:

$$\langle \underline{\boldsymbol{U}}^2 \rangle = \langle \underline{\boldsymbol{V}}^2 \rangle \ , \tag{3.91}$$

because the average vibrational amplitudes do not change over time, and all unit cells are equivalent in the crystal. This allows a final simplification of (3.90):

$$\langle \exp \underline{\boldsymbol{U}} \exp \underline{\boldsymbol{V}} \rangle = \exp\langle \underline{\boldsymbol{U}}^2 \rangle \ \exp\langle \underline{\boldsymbol{U}}\underline{\boldsymbol{V}} \rangle \ . \tag{3.92}$$

Equation (3.92) was obtained with the one assumption of a Gaussian thermal spread of atom positions. Even if this in not quite the case, (3.92) is expected to be valid when the atom displacements are small. By expanding both sides of (3.86), and recognizing that the thermal average of odd powers of X are zero, (3.86) seems a reasonable approximation.

Finally, we use (3.92) to rewrite (3.58) in a way that will later let us identify individual phonon scatterings. Making use of the definitions (3.76) and (3.77):

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}} \frac{(2\pi)^3 N |b|^2}{2V_0} \sum_l \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l} \int \mathrm{e}^{\mathrm{i}\omega t} \exp\langle \underline{\boldsymbol{U}}^2 \rangle \exp\langle \underline{\boldsymbol{U}}\underline{\boldsymbol{V}} \rangle \, \mathrm{d}t \ . \tag{3.93}$$

Taking the thermal averages has removed much of the time dependence in our factors. The only remaining time dependence is within the $\langle \underline{\boldsymbol{U}}\underline{\boldsymbol{V}} \rangle$ factor, so we rewrite:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}} \frac{(2\pi)^3 N |b|^2}{2V_0} \sum_l \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l} \exp\langle \underline{\boldsymbol{U}}^2 \rangle \int \mathrm{e}^{\mathrm{i}\omega t} \exp\langle \underline{\boldsymbol{U}}\underline{\boldsymbol{V}} \rangle \, \mathrm{d}t \ . \tag{3.94}$$

Using (3.76), the factor $\exp\langle \underline{\boldsymbol{U}}^2 \rangle$ in (3.94) becomes $\exp(-\langle [\boldsymbol{Q} \cdot \underline{\boldsymbol{u}}_l(0)]^2 \rangle)$. We recognize this as a Debye–Waller factor. To its right in (3.94) is a sum over pairs of atoms separated by $\boldsymbol{x}_l$. There is a phase associated with the atom separation in the pair. Suppose for the moment that there were no displacements of the atoms from their lattice sites. In this case the final exponential $\exp\langle \underline{\boldsymbol{U}}\underline{\boldsymbol{V}} \rangle$ would equal $\mathrm{e}^0 = 1$. This is the case of elastic scattering, and the sum of phase factors of (3.94) would reduce to to the elastic scattering result of (2.68). It is the final factor, the Fourier transform of $\exp\langle \underline{\boldsymbol{U}}\underline{\boldsymbol{V}} \rangle$, that makes (3.94) interesting, and it is in a convenient form for further development.

3.3.6 Impulse Approximation

It is relatively easy to adapt the general result (3.45) to the case where the energy of the incident neutron is much higher than the characteristic excitations in the solid. We therefore ignore the interatomic interactions, and consider the collision of the neutron with a single nucleus at $\boldsymbol{R}$. This approaches the problem of hitting a ball with a classical projectile, so we lose some features of wave mechanics.[4] We do have to account for momentum

[4] Another viewpoint is that we expect multiple excitations to occur in the solid. The effects of coherence are generally washed out when multiple excitations

and energy transfer, of course. The impulse approximation begins with the replacement of the operator:

$$\underline{\boldsymbol{R}}_j(t) \longrightarrow \underline{\boldsymbol{R}}_j(0) + t\underline{\boldsymbol{v}}_j \,, \tag{3.95}$$

$$\underline{\boldsymbol{R}}_j(t) \longrightarrow \underline{\boldsymbol{R}}_j(0) + \frac{t}{M_j}\underline{\boldsymbol{p}}_j \,, \tag{3.96}$$

where we expect t to be small since the neutron is moving fast. Because the incoherent character of the multiple excitations suppresses the phase relationships between different scatterers, and we consider terms $j = k$ in (3.45). It is tempting to substitute (3.96) directly into (3.45) to obtain:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} \;?=?\; \frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^3}{2V_0}\sum_j\sum_k b_j^* b_k \delta_{j,k} \times \int \mathrm{e}^{\mathrm{i}\omega t}\left\langle \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot(\underline{\boldsymbol{R}}_j(0)+t/M_j\underline{\boldsymbol{p}}_j)}\,\mathrm{e}^{-\mathrm{i}\boldsymbol{Q}\cdot\underline{\boldsymbol{R}}_k(0)}\right\rangle \mathrm{d}t \,, \tag{3.97}$$

and with $\underline{\boldsymbol{R}}_j(0) = \underline{\boldsymbol{R}}_k(0)$:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} \;?=?\; \frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^3}{2V_0}\sum_j |b_j|^2 \int \mathrm{e}^{\mathrm{i}\omega t}\left\langle \mathrm{e}^{\mathrm{i}t/M_j\boldsymbol{Q}\cdot\underline{\boldsymbol{p}}_j}\right\rangle \mathrm{d}t \,. \tag{3.98}$$

The missing piece in (3.98) is a phase factor associated with the energy gain of the scatterer. The energy gain is kinetic, $E_\mathrm{kin} = \frac{p^2}{2M_j} = \frac{\hbar^2 Q^2}{2M_j}$. Fermi's Golden Rule, which connects the wavefunctions before and after the scattering, is senstitive to the phases of the initial and final states of the scatterer. The transfer of energy causes a relative change in the phase of these two states by the factor: $\exp(\mathrm{i}E_\mathrm{kin}\hbar^{-1}\,t)$. This phase relationship for the total energy transfer leaves a minimal amount of quantum mechanics in the scattering problem (which we expect to go away at very high incident energies):

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^3}{2V_0}\sum_j |b_j|^2 \int \mathrm{e}^{\mathrm{i}\omega t}\mathrm{e}^{\mathrm{i}\left(\frac{\hbar^2Q^2}{2M_j}\right)\hbar^{-1}\,t}\left\langle \mathrm{e}^{\mathrm{i}t/M_j\boldsymbol{Q}\cdot\underline{\boldsymbol{p}}_j}\right\rangle \mathrm{d}t \,. \tag{3.99}$$

In Sect. 3.3.5 we calculated the thermal average, $\langle\rangle$, when the displacements of the scatterer had a Gaussian thermal spread. To apply this result to (3.99), we use the result: $\langle \mathrm{e}^{\mathrm{i}X}\rangle = \mathrm{e}^{-X^2/2}$:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^3}{2V_0}\sum_j |b_j|^2 \int \mathrm{e}^{\mathrm{i}\omega t}\mathrm{e}^{\mathrm{i}\left(\frac{\hbar^2Q^2}{2M_j}\right)\hbar^{-1}\,t}\mathrm{e}^{-(\langle\boldsymbol{Q}\cdot\underline{\boldsymbol{p}}_j\rangle t/M_j)^2/2}\,\mathrm{d}t \,. \tag{3.100}$$

We complete the square in the exponential, defining:

$$x^2 \equiv \left(\sqrt{a}t - \frac{b}{2\sqrt{a}}\right)^2 = at^2 - bt + \frac{b^2}{4a} \,, \tag{3.101}$$

occur. For incoherent inelastic scattering as in Sect. 3.1.4, we expect to analyze the scattering by considering only a single nucleus at a time. We expect the excitations to be incoherent.

and we obtain a result:

$$\int_{-\infty}^{\infty} \mathrm{e}^{-at^2+bt}\,\mathrm{d}t = \int_{-\infty}^{\infty} \frac{\mathrm{e}^{-x^2}}{\sqrt{a}}\,\mathrm{d}x = \sqrt{\frac{\pi}{a}}\;\mathrm{e}^{b^2/4a}\;. \tag{3.102}$$

With the associations for a and b:

$$a = \frac{1}{2}\left(\frac{\langle \boldsymbol{Q}\cdot\underline{\boldsymbol{p}}_j\rangle}{M_j}\right)^2\;, \tag{3.103}$$

$$b = \mathrm{i}\left(\omega + \frac{\hbar Q^2}{2M_j}\right)\;, \tag{3.104}$$

(3.100) becomes:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^{7/2}}{2V_0}\sum_j |b_j|^2\left(\frac{M_j}{\langle \boldsymbol{Q}\cdot\underline{\boldsymbol{p}}_j\rangle}\right)^2 \exp\left(-\frac{\left(\frac{\hbar Q^2}{2M_j}+\omega\right)^2}{2\left(\frac{\langle \boldsymbol{Q}\cdot\underline{\boldsymbol{p}}_j\rangle}{M_j}\right)^2}\right)\;. \tag{3.105}$$

The differential scattering cross section in (3.105) has the shape of a Gaussian function, centered at an energy $\hbar\omega$:

$$\hbar\omega = -\frac{\hbar^2 Q^2}{2M_j}\;. \tag{3.106}$$

The center of the Gaussian is simply the energy transfer from a scattering with a single-particle recoil. This result could be obtained by classical mechanics. The spread of this Gaussian is obtained from from the denoninator in the Gaussian of (3.105). This width increases with Q and with the mean-squared velocity of the scatterers. The ratio of shift to width grows larger with Q, however. In the classical limit of very large Q, the width is negligible, so the energy and momentum transfers are well-defined.

3.3.7 Multiphonon Expansion

We return to develop (3.94) with the "multiphonon expansion," which is obtained from the expansion of the exponential $\exp\langle\underline{\boldsymbol{U}\boldsymbol{V}}\rangle$. This is seen most easily in the incoherent approximation, where we replace:

$$\exp\langle\underline{\boldsymbol{U}\boldsymbol{V}}\rangle \longrightarrow \exp\langle\underline{\boldsymbol{U}}\rangle\langle\underline{\boldsymbol{V}}\rangle = \exp\langle\underline{\boldsymbol{U}}\rangle\langle\underline{\boldsymbol{U}}\rangle\;. \tag{3.107}$$

so that, treating $\underline{\boldsymbol{U}}$ as a displacement and not an operator:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^3 N|b|^2}{2V_0}\sum_l \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l}\exp\langle\underline{\boldsymbol{U}}^2\rangle$$
$$\times\int \mathrm{e}^{\mathrm{i}\omega t}\exp\left(-\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{u}\;\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{u}\right)\,\mathrm{d}t\;. \tag{3.108}$$

Following (2.73), the harmonic oscillator energy $M\omega^2 u_{\max}^2$, is assumed quantized in units of $\hbar\omega$. From $\hbar\omega \propto M\omega^2 u_{\max}^2$, we replace the displacement $u = \sqrt{\hbar/(2M\omega)}$, for which we expect a time-dependence:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^3 N|b|^2}{2V_0}\sum_l \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l}\exp\langle\underline{\boldsymbol{U}}^2\rangle \times\int \mathrm{e}^{\mathrm{i}\omega t}\exp\left(\frac{Q^2\hbar}{2M\omega}\Upsilon(t)\right)\mathrm{d}t\ , \tag{3.109}$$

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = \frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^3 N|b|^2}{2V_0}\sum_l \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l}\exp\langle\underline{\boldsymbol{U}}^2\rangle \times\int \mathrm{e}^{\mathrm{i}\omega t}\left(1+\frac{\frac{\hbar^2Q^2}{2M}}{\hbar\omega}\Upsilon(t)+\frac{1}{2}\left(\frac{\frac{\hbar^2Q^2}{2M}}{\hbar\omega}\Upsilon(t)\right)^2+\ldots\right)\mathrm{d}t \tag{3.110}$$

The terms in the expansion of (3.110) are recognized as a series in powers of $E_\mathrm{Recoil}/(\hbar\omega)$ — the recoil energy, $p^2/(2M)$, divided by the energy of the phonon, $\hbar\omega$. It is instructive to compare (3.110) to (2.67), which is re-written here. Note that it did not include a Debye–Waller factor:

$$\Psi_\mathrm{sc}(\boldsymbol{Q},\boldsymbol{r}) = -\frac{\mathrm{e}^{\mathrm{i}(\boldsymbol{k}_\mathrm{f}\cdot\boldsymbol{r}-\omega_0 t)}}{|\boldsymbol{r}|} \times\sum_{l,\kappa} b_\kappa\,\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_{l,\kappa}}\left(1+\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{u}_{l,\kappa}(t)-\frac{1}{2}(\boldsymbol{Q}\cdot\boldsymbol{u}_{l,\kappa}(t))^2+\ldots\right) \tag{3.111}$$

Analyzing the two series in (3.110) and (3.111), term-by-term, we find:

- The first term, the 1, provides a Fourier transform of a constant, which is a delta function, $\delta(\omega)$. Since the excitation energy is therefore zero, this is an elastic scattering process. Note that the sum of phase factors $\mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{x}_l}$ over lattice sites and the Debye–Waller factor $\mathrm{e}^{\langle\underline{\boldsymbol{U}}^2\rangle}$ are as expected for diffracted neutron wavefunctions. Elastic nuclear scattering has no Q dependence, except through the Debye–Waller factor.
- The second term, involving $\frac{\hbar^2Q^2}{2M}/(\hbar\omega)$, is the inelastic scattering that occurs by exciting one phonon. We found in (2.78) that its Fourier transform led to a delta function $\delta(\omega-\omega(\boldsymbol{q}))$ from the conservation of energy. A conservation of momentum led to a similar delta function, $\delta\big((\boldsymbol{q}-\boldsymbol{Q})-\boldsymbol{g}\big)$. Single phonon scattering increases with Q as Q^2.
- The third term, involving $\left(\frac{\frac{\hbar^2Q^2}{2M}}{\hbar\omega}\right)^2$, is the scattering that involves the excitation of two phonons. This occurs in one scattering event, not through the creation of two phonons by two different deflections of the neutron. (The latter is "multiple scattering.") When the time function is squared, such as $\cos^2(\omega t) = 1/2\big(1+\cos(2\omega t)\big)$, the frequency is doubled, and energy conservation provides the delta function $\delta(\omega-2\omega(\boldsymbol{q}))$. Two-phonon

scattering involves twice the energy transfer as a one-phonon process. Note how it increases rapidly with Q, going as Q^4.

- Higher order terms, involving $+\ldots$, involve the excitation of many phonons in one scattering of the neutron. These higher-order terms approach classical behavior. Typically the scattering of a particle with a large a large momentum transfer causes the excitation of many phonons, sometimes better described as the creation of heat.

We note that the analysis here has assumed ignored phase relationships in multiphonon scattering. This is typical of even more sophisticated treatments of the problem. Multiphonon scattering is usually analyzed in the incoherent approximation.

3.4 Magnetic Scattering

3.4.1 Magnetic Form Factor and Scattering Amplitude

Magnetic scattering originates with the interaction between the spin of the neutron and the spins of electrons and/or the motions of electrons. Magnetic scattering is inherently more complicated than nuclear scattering because the potentials have vector character. The magnetic forces are also long range.

The scattering potential can be written in the general form $V = -\boldsymbol{\mu}_\mathrm{n} \cdot \boldsymbol{B}$, where $\boldsymbol{B}$, which originates with the electrons, has a spin component $\boldsymbol{B}_\mathrm{S}$ and an orbital component $\boldsymbol{B}_\mathrm{L}$:

$$\boldsymbol{B} = \boldsymbol{B}_\mathrm{S} + \boldsymbol{B}_\mathrm{L} \; . \tag{3.112}$$

These components have different mathematical forms[5]:

$$\boldsymbol{B}_\mathrm{L} \propto \frac{\widehat{\boldsymbol{R}} \times \boldsymbol{p}}{R^2} \; , \tag{3.113}$$

$$\boldsymbol{B}_\mathrm{S} \propto \boldsymbol{\nabla} \times \left(\frac{\boldsymbol{s} \times \widehat{\boldsymbol{R}}}{R^2} \right) \; . \tag{3.114}$$

From (2.54) we obtain the scattering amplitude in the first Born approximation as the Fourier transform of the scattering potential $V = -\boldsymbol{\mu}_\mathrm{n} \cdot \boldsymbol{B}$

$$f_\mathrm{mag}(\boldsymbol{Q}, E) = \sqrt{\frac{k_\mathrm{f}}{k_\mathrm{i}}} \frac{2m}{\hbar^2} \frac{1}{4\pi} \int\limits_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}} \; (-\boldsymbol{\mu}_\mathrm{n} \cdot \boldsymbol{B}) \; \mathrm{d}\boldsymbol{r} \; . \tag{3.115}$$

$$f_\mathrm{mag}(\boldsymbol{Q}, E) = \sqrt{\frac{k_\mathrm{f}}{k_\mathrm{i}}} (-\gamma r_\mathrm{e}) \int\limits_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}} \; \boldsymbol{\sigma} \cdot \left(\frac{1}{\hbar} \frac{\widehat{\boldsymbol{R}} \times \boldsymbol{p}}{R^2} + \boldsymbol{\nabla} \times \left(\frac{\boldsymbol{s} \times \widehat{\boldsymbol{R}}}{R^2} \right) \right) \mathrm{d}\boldsymbol{r} \tag{3.116}$$

[5] Note that $\boldsymbol{B}_\mathrm{L}$ has the form of the Biot-Savart law for the electron current (with electron momentum $\boldsymbol{p}$) about the atom. The $\boldsymbol{B}_\mathrm{S}$ can be written as a curl of a vector potential, $\boldsymbol{B}_\mathrm{S} = \boldsymbol{\nabla} \times \boldsymbol{A}$, if $\boldsymbol{A} = \mu_\mathrm{e} \times \boldsymbol{R}/R^2$, and $\mu_\mathrm{e} = -e\hbar/m_\mathrm{e} = -2\mu_\mathrm{B}\boldsymbol{s}$.

where $\boldsymbol{\sigma}$ is the neutron spin, later to be the spin operator. Many constants were combined into the "classical electron radius," $r_{\rm e} = e^2/(m_{\rm e}c^2)$, and γ is the gyromagnetic ratio of the neutron, $\gamma = 1.913$.

The evaluation of (3.116) is most expedient with two mathematical tricks:

$$\int\limits_{-\infty}^{\infty} \frac{\widehat{\boldsymbol{R}}}{R^2}\, {\rm e}^{-{\rm i}\boldsymbol{Q}\cdot\boldsymbol{R}}\, {\rm d}\boldsymbol{R} = -4\pi{\rm i}\,\frac{\widehat{\boldsymbol{Q}}}{Q}\,, \tag{3.117}$$

$$\boldsymbol{\nabla}\times\left(\frac{\boldsymbol{s}\times\widehat{\boldsymbol{R}}}{R^2}\right) = \frac{1}{2\pi^2}\int\limits_{-\infty}^{\infty} \widehat{\boldsymbol{q}}\times(\boldsymbol{s}\times\widehat{\boldsymbol{q}})\,{\rm e}^{-{\rm i}\boldsymbol{q}\cdot\boldsymbol{R}}\,{\rm d}\boldsymbol{q}\,. \tag{3.118}$$

The derivation of (3.117) is not difficult – it involves transformation to spherical coordinates where the R^2 in the denominator is cancelled by an R^2 in the differential volume element.[6] Unfortunately, (3.118) is much more work to obtain unless one is rather clever with, or accepting of, vector identities. A sketch of its derivation is given in Appendix B.2 of Squires' book.

In using the tricks (3.117) and (3.118), the necessary steps are:

- With the neutron at $\boldsymbol{r}$ and the $i^{\rm th}$ electron at $\boldsymbol{r}_i$, the position for the magnetic field, $\boldsymbol{R}$ in (3.113) and (3.114), is the distance separating $\boldsymbol{r}$ and $\boldsymbol{r}_i$: $\boldsymbol{R} = \boldsymbol{r} - \boldsymbol{r}_i$.
- The Fourier transform of $\widehat{\boldsymbol{R}}\times\boldsymbol{p}/R^2$ first involves replacing the exponential ${\rm e}^{-{\rm i}\boldsymbol{Q}\cdot\boldsymbol{R}} = {\rm e}^{-{\rm i}\boldsymbol{Q}\cdot\boldsymbol{r}}\,{\rm e}^{+{\rm i}\boldsymbol{Q}\cdot\boldsymbol{r}_i}$, in order to use coordinates of the neutron and the $i^{\rm th}$ electron. The second phase factor ${\rm e}^{+{\rm i}\boldsymbol{Q}\cdot\boldsymbol{r}_i}$, is a constant, and is removed from the integration over all space. The remaining spatial integration becomes, from (3.117), $-{\rm i}4\pi\,\widehat{\boldsymbol{Q}}\times\boldsymbol{p}/Q$.
- Notice that the right-hand side of (3.118) has its only dependence on $\boldsymbol{R}$ in the exponential (which we again write as ${\rm e}^{-{\rm i}\boldsymbol{q}\cdot\boldsymbol{R}} = {\rm e}^{-{\rm i}\boldsymbol{q}\cdot\boldsymbol{r}}\,{\rm e}^{+{\rm i}\boldsymbol{q}\cdot\boldsymbol{r}_i}$). When is (3.118) substituted into (3.116), a double integral (over $\boldsymbol{r}$ and $\boldsymbol{q}$) is obtained. After the phase factor, ${\rm e}^{{\rm i}\boldsymbol{q}\cdot\boldsymbol{r}_i}$, is isolated, the $\boldsymbol{r}$-integral is of the form $\int {\rm e}^{{\rm i}(\boldsymbol{Q}-\boldsymbol{q})\cdot\boldsymbol{r}}{\rm d}\boldsymbol{r}$. The $\boldsymbol{r}$-integral therefore gives a three-dimensional δ-function, $(2\pi)^3\delta(\boldsymbol{Q}-\boldsymbol{q})$.

The result from these manipulations with (3.117) and (3.118) is:

$$\begin{aligned} f_{\rm mag}(\boldsymbol{Q},E) = \sqrt{\frac{k_{\rm f}}{k_{\rm i}}}(-\gamma r_0) \\ \times\boldsymbol{\sigma}\cdot\Bigg(\int\limits_{-\infty}^{\infty} {\rm e}^{{\rm i}\boldsymbol{q}\cdot\boldsymbol{r}_i}\,\frac{1}{2\pi^2}\widehat{\boldsymbol{q}}\times(\boldsymbol{s}\times\widehat{\boldsymbol{q}})\;(2\pi)^3\delta(\boldsymbol{Q}-\boldsymbol{q})\,{\rm d}\boldsymbol{q} \\ -\frac{{\rm i}4\pi}{\hbar}\,{\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{r}_i}\,\frac{\widehat{\boldsymbol{Q}}\times\boldsymbol{p}}{Q}\Bigg)\,. \end{aligned} \tag{3.119}$$

[6] Note that (3.117) is the inverse transformation of (A.27) in the Appendix, with the interchange of $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Integrating over the δ-function forces $\boldsymbol{q} \longrightarrow \boldsymbol{Q}$:

$$f_{\mathrm{mag}}(\boldsymbol{Q}, E) = \sqrt{\frac{k_{\mathrm{f}}}{k_{\mathrm{i}}}}(-\gamma r_0)\, \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}_i} \times 4\pi\boldsymbol{\sigma} \cdot \left(\widehat{\boldsymbol{Q}} \times (\boldsymbol{s} \times \widehat{\boldsymbol{Q}}) \; - \; \frac{\mathrm{i}}{\hbar} \frac{\widehat{\boldsymbol{Q}} \times \boldsymbol{p}}{Q} \right) . \tag{3.120}$$

With the definition of $\widetilde{\boldsymbol{M}}_{\perp}(\boldsymbol{Q})$, which involves a sum over all electrons in the sample:

$$\widetilde{\boldsymbol{M}}_{\perp}(\boldsymbol{Q}) \equiv \sum_{\boldsymbol{r}_i} \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}_i} \left(\widehat{\boldsymbol{Q}} \times (\boldsymbol{s} \times \widehat{\boldsymbol{Q}}) \; - \; \frac{\mathrm{i}}{\hbar} \frac{\widehat{\boldsymbol{Q}} \times \boldsymbol{p}}{Q} \right) , \tag{3.121}$$

$$f_{\mathrm{mag}}(\boldsymbol{Q}, E) = 4\pi \sqrt{\frac{k_{\mathrm{f}}}{k_{\mathrm{i}}}}(-\gamma r_0)\, \boldsymbol{\sigma} \cdot \widetilde{\boldsymbol{M}}_{\perp}(\widehat{\boldsymbol{Q}}) . \tag{3.122}$$

Equation (3.122) is a general expression for the magnetic scattering from the spin and orbital moment of the electrons. It includes a sum over the phase factors for electrons at all $\{\boldsymbol{r}_i\}$. Unfortunately, the spin and orbital terms in the large parentheses in (3.121) have different forms, and to see more cearly the vectorial character of magnetic scattering it is convenient to make them equivalent using the expression (proved with some effort in Squires Appendix H.1)

$$\widetilde{\boldsymbol{M}}_{\perp L} \equiv \frac{\mathrm{i}}{\hbar Q} \sum_{\boldsymbol{r}_i} \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}_i} (\widehat{\boldsymbol{Q}} \times \boldsymbol{p}) = \frac{1}{2\mu_{\mathrm{B}}} \widehat{\boldsymbol{Q}} \times \left(\widetilde{\boldsymbol{M}}_L(\boldsymbol{Q}) \times \widehat{\boldsymbol{Q}} \right) , \tag{3.123}$$

where the Fourier transform of the magnetic form factor from the spatial distribution of electron currents is:

$$\widetilde{\boldsymbol{M}}_L(\boldsymbol{Q}) \equiv \int \boldsymbol{M}_L(\boldsymbol{r})\, \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}_i} \mathrm{d}\boldsymbol{r} . \tag{3.124}$$

It is natural to write (3.121) as:

$$\widetilde{\boldsymbol{M}}_{\perp} \equiv \widetilde{\boldsymbol{M}}_{\perp s} + \widetilde{\boldsymbol{M}}_{\perp L} , \tag{3.125}$$

$$\widetilde{\boldsymbol{M}}_{\perp} = \frac{1}{2\mu_{\mathrm{B}}} \widehat{\boldsymbol{Q}} \times \left(\widetilde{\boldsymbol{M}}(\boldsymbol{Q}) \times \widehat{\boldsymbol{Q}} \right) , \tag{3.126}$$

where $\widetilde{\boldsymbol{M}}(\boldsymbol{Q})$ is the Fourier transform of the spatial distribution of all magnetization (as in (3.124)):

$$\widetilde{\boldsymbol{M}}(\boldsymbol{Q}) \equiv \int \boldsymbol{M}(\boldsymbol{r})\, \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}_i} \mathrm{d}\boldsymbol{r} , \tag{3.127}$$

We arrive at the cleaner expression for the magnetic scattering factor

$$f_{\mathrm{mag}}(\boldsymbol{Q}, E) = 4\pi \sqrt{\frac{k_{\mathrm{f}}}{k_{\mathrm{i}}}}(-\gamma r_0) \sum_{\boldsymbol{r}_i} \mathrm{e}^{\mathrm{i}\boldsymbol{Q}\cdot\boldsymbol{r}_i} \boldsymbol{\sigma} \cdot \left(\widehat{\boldsymbol{Q}} \times (\widetilde{\boldsymbol{M}} \times \widehat{\boldsymbol{Q}}) \right) , \tag{3.128}$$

3.4.2 Vector Orientations in Magnetic Scattering

Equation (3.128) shows that the magnetic scattering is proportional to the vector $\widehat{\boldsymbol{Q}} \times \widetilde{\boldsymbol{M}} \times \widehat{\boldsymbol{Q}}$. The maximum magnetic scattering therefore occurs when the direction of the spin, $\boldsymbol{S}$, or magnetization, $\boldsymbol{M}(\boldsymbol{r})$, is perpendicular to the scattering vector, $\boldsymbol{Q}$. This is illustrated in Fig. 3.4.2, which shows intensity contours measured about the forward beam in a small-angle scattering experiment. A magnetic field of 8 kG was applied to the specimen, perpendicular to the direction of the incident beam. This field should was sufficient to saturate the magnetic moment of the sample (a soft magnetic material), aligning all its spins. Notice that the contours are oriented perpendicularly to the direction of the applied magnetic field. The scattering along the direction of the magnetic field is non-zero, however, because Ni-Fe has strong nuclear scattering. By comparing intensities parallel and perpendicular to the applied magnetic field, it is possible to extract individual profiles for magnetic and nuclear scattering.

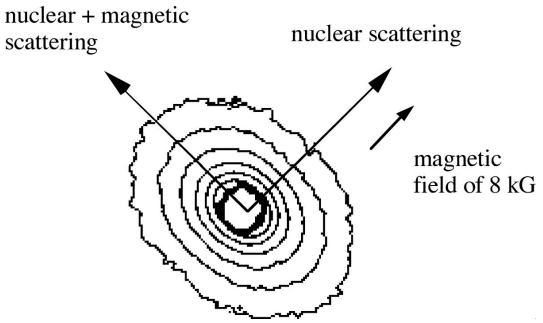


Fig. 3.6. Experimental intensity contours from small-angle neutron scattering (SANS) from fcc Ni-Fe in the presence of an 8 kG applied magnetic field. The forward beam was perpendicular to the plane of the paper. The intensity decreases with angle from the forward beam, but more rapidly in the direction of the applied magnetic field.

The relationship between the generalized magnetization, $\widetilde{\boldsymbol{M}}$, its projection, $\widetilde{\boldsymbol{M}}_\perp$, and the scattering vector, $\boldsymbol{Q}$, is illustrated with the help of Fig. 3.4.2 and its caption. By comparing the two parts of this figure, we find

$$\widehat{\boldsymbol{Q}} \times \widetilde{\boldsymbol{M}} \times \widehat{\boldsymbol{Q}} = \widetilde{\boldsymbol{M}} - \widehat{\boldsymbol{Q}}(\widetilde{\boldsymbol{M}} \cdot \widehat{\boldsymbol{Q}}) \; . \tag{3.129}$$

The cross-section for magnetic scattering is proportional to $|f_{\rm mag}|^2$. We need to take the product of $\widetilde{\boldsymbol{M}}_\perp$ with its Hermitian adjoint to obtain the intensity

$$\widetilde{\boldsymbol{M}}_\perp^\dagger \widetilde{\boldsymbol{M}}_\perp = \left(\widetilde{\boldsymbol{M}}^\dagger - \widehat{\boldsymbol{Q}}(\widetilde{\boldsymbol{M}}^\dagger \cdot \widehat{\boldsymbol{Q}})\right)\left(\widetilde{\boldsymbol{M}} - \widehat{\boldsymbol{Q}}(\widetilde{\boldsymbol{M}} \cdot \widehat{\boldsymbol{Q}})\right) \; . \tag{3.130}$$

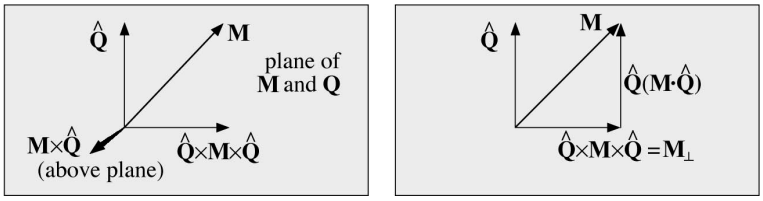


Fig. 3.7. Important vectors for magnetic scattering. (**left**) The gray plane, parallel to the paper, is defined by the vectors $\widetilde{\boldsymbol{M}}$ and $\widehat{\boldsymbol{Q}}$. The vector product $\widetilde{\boldsymbol{M}} \times \widehat{\boldsymbol{Q}}$ is perpendicular to the plane of the paper and $\widetilde{\boldsymbol{M}}_\perp \equiv \widehat{\boldsymbol{Q}} \times \widetilde{\boldsymbol{M}} \times \widehat{\boldsymbol{Q}}$ again lies in the plane. (**right**) The vector of length $\widetilde{\boldsymbol{M}} \cdot \widehat{\boldsymbol{Q}}$ along the direction $\widehat{\boldsymbol{Q}}$.

When distributing the product in (3.130), the two middle terms have the same form but opposite sign to the fourth term, so

$$\widetilde{\boldsymbol{M}}_\perp^\dagger \widetilde{\boldsymbol{M}}_\perp = \widetilde{\boldsymbol{M}}^\dagger \widetilde{\boldsymbol{M}} - (\widetilde{\boldsymbol{M}}^\dagger \cdot \widehat{\boldsymbol{Q}})(\widetilde{\boldsymbol{M}} \cdot \widehat{\boldsymbol{Q}}) \; . \tag{3.131}$$

We resolve $\widetilde{\boldsymbol{M}}$ and $\boldsymbol{Q}$ into Cartesian components

$$\widetilde{\boldsymbol{M}}_\perp^\dagger \widetilde{\boldsymbol{M}}_\perp = \sum_{\alpha,\beta} S_\alpha^\dagger S_\alpha - S_\alpha^\dagger \widehat{Q}_\alpha S_\beta \widehat{Q}_\beta \; , \tag{3.132}$$

$$\widetilde{\boldsymbol{M}}_\perp^\dagger \widetilde{\boldsymbol{M}}_\perp = \sum_{\alpha,\beta} \left(\delta_{\alpha\beta} - \widehat{Q}_\alpha \widehat{Q}_\beta \right) S_\alpha^\dagger S_\beta \; . \tag{3.133}$$

3.4.3 Averaging over Neutron Polarizations

To obtain an experimental cross-section from (3.128), we need to average over the spin orientations of the incident neutrons (orientations of $\boldsymbol{\sigma}$). We expect to write the cross-section as $|f_{\rm mag}|^2$

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = (\gamma r_{\rm e})^2 \frac{k_f}{k_i} \left| \langle \lambda_f, \sigma_f | \boldsymbol{\sigma} \cdot \widetilde{\boldsymbol{M}}_\perp | \lambda_i, \sigma_i \rangle \right|^2 \delta(E_f - E_i + \hbar\omega) \; . \tag{3.134}$$

Compared to nuclear scattering, (3.134) includes additional coordinates in the matrix element, σ_i and σ_f, to account for the change in spin of the neutron after scattering. Again, the λ_i and λ_f refer to states of the scatterer. For magnetic scattering, a change in λ may originate with the creation or annihilation of an excitation of the electron spins. Further progress with these coordinates will require a magnetic dynamics model. The total cross-section requires that we sum over all initial and final states of the neutron:

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = (\gamma r_{\rm e})^2 \frac{k_f}{k_i} \sum_i \sum_f \left| \langle \lambda_f, \sigma_f | \boldsymbol{\sigma} \cdot \widetilde{\boldsymbol{M}}_\perp | \lambda_i, \sigma_i \rangle \right|^2 \delta(E_f - E_i + \hbar\omega) \tag{3.135}$$

The coordinates σ_i and σ_f describe to the neutron spin, which we expect to be up or down (sometimes $\uparrow, \downarrow$). Equation (3.134) is the cross-section for one scattering process, but in an experiment we expect to average over the

spins of many neutrons. The total cross-section should include weighting functions such as p_{σ_i} to account for different probabilities of initial spin up and spin down neutrons, as for example with polarized beam experiments. The final spin states are assumed unbiased, however, and so have no associated weight function.

The operator in (3.135) can be resolved into its vector components:

$$\boldsymbol{\sigma} \cdot \widetilde{\boldsymbol{M}}_\perp = \sigma_x \widetilde{M}_{\perp x} + \sigma_y \widetilde{M}_{\perp y} + \sigma_z \widetilde{M}_{\perp z} \; . \tag{3.136}$$

The coordinates of σ_α pertain to the neutrons only, and the coordinates of $\widetilde{M}_{\perp\alpha}$ pertain only to the electrons. The products in (3.136) serve to group the factors involving electrons and neutrons, but they separate as for example:

$$\langle \lambda_f, \sigma_f | \sigma_x \widetilde{M}_{\perp x} | \lambda_i, \sigma_i \rangle = \langle \lambda_f | \widetilde{M}_{\perp x} | \lambda_i \rangle \, \langle \sigma_f | \sigma_x | \sigma_i \rangle \; . \tag{3.137}$$

The cross-section of (3.135) can be separated into nine different terms by use of (3.136) and (3.137) (note the sequencing of subscripts x and y):

$$\begin{aligned} \frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega \, \mathrm{d}E} = (\gamma r_\mathrm{e})^2 \frac{k_f}{k_i} \sum_i \sum_f \\ \Big(& \langle \lambda_i | \widetilde{M}_{\perp x} | \lambda_f \rangle \, \langle \sigma_i | \sigma_x | \sigma_f \rangle \langle \sigma_f | \sigma_x | \sigma_i \rangle \langle \lambda_f | \widetilde{M}_{\perp x} | \lambda_i \rangle \\ & + \langle \lambda_i | \widetilde{M}_{\perp x} | \lambda_f \rangle \, \langle \sigma_i | \sigma_x | \sigma_f \rangle \langle \sigma_f | \sigma_y | \sigma_i \rangle \langle \lambda_f | \widetilde{M}_{\perp y} | \lambda_i \rangle \\ & + \langle \lambda_i | \widetilde{M}_{\perp y} | \lambda_f \rangle \, \langle \sigma_i | \sigma_y | \sigma_f \rangle \langle \sigma_f | \sigma_x | \sigma_i \rangle \langle \lambda_f | \widetilde{M}_{\perp x} | \lambda_i \rangle + \ldots \Big) \\ & \delta(E_f - E_i + \hbar\omega) \; . \end{aligned} \tag{3.138}$$

By closure, $\sum_f |\sigma_f\rangle\langle\sigma_f| = 1$, there is simplification of the spin factors

$$\begin{aligned} \frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega \, \mathrm{d}E} = (\gamma r_\mathrm{e})^2 \frac{k_f}{k_i} \sum_i \sum_f \\ \Big(& \langle \lambda_i | \widetilde{M}_{\perp x} | \lambda_f \rangle \, \langle \sigma_i | \sigma_x \sigma_x | \sigma_i \rangle \langle \lambda_f | \widetilde{M}_{\perp x} | \lambda_i \rangle \\ & + \langle \lambda_i | \widetilde{M}_{\perp x} | \lambda_f \rangle \, \langle \sigma_i | \sigma_x \sigma_y | \sigma_i \rangle \langle \lambda_f | \widetilde{M}_{\perp y} | \lambda_i \rangle \\ & + \langle \lambda_i | \widetilde{M}_{\perp y} | \lambda_f \rangle \, \langle \sigma_i | \sigma_y \sigma_x | \sigma_i \rangle \langle \lambda_f | \widetilde{M}_{\perp x} | \lambda_i \rangle + \ldots \Big) \\ & \delta(E_f - E_i + \hbar\omega) \; . \end{aligned} \tag{3.139}$$

The spin operators have the properties[7]

$$\begin{aligned} & \sigma_x | \uparrow \rangle = | \downarrow \rangle \quad , \quad \sigma_y | \uparrow \rangle = +\mathrm{i} | \downarrow \rangle \quad , \quad \sigma_z | \uparrow \rangle = + | \uparrow \rangle \; , \\ & \sigma_x | \downarrow \rangle = | \uparrow \rangle \quad , \quad \sigma_y | \downarrow \rangle = -\mathrm{i} | \downarrow \rangle \quad , \quad \sigma_z | \downarrow \rangle = - | \downarrow \rangle \; . \end{aligned} \tag{3.140}$$

[7] These relations are obtained, for example, from explicit forms of the Pauli spin matrices.

Equation (3.139) includes terms with the factors $\langle\uparrow\ |\sigma_x\sigma_y|\ \uparrow\rangle$ and $\langle\uparrow\ |\sigma_y\sigma_x|\ \downarrow\rangle$. Evaluating them with (3.140) gives:

$$\langle\uparrow\ |\sigma_x\sigma_y|\ \uparrow\rangle = \langle\uparrow\ |\sigma_x(+\mathrm{i})|\ \downarrow\rangle = \langle\uparrow\ |+\mathrm{i}|\ \uparrow\rangle = +\mathrm{i}\langle\uparrow\ |\ \uparrow\rangle = +\mathrm{i}\ , \tag{3.141}$$

$$\langle\uparrow\ |\sigma_y\sigma_x|\ \uparrow\rangle = \langle\uparrow\ |\sigma_y|\ \downarrow\rangle = \langle\uparrow\ |-\mathrm{i}|\ \uparrow\rangle = -\mathrm{i}\langle\uparrow\ |\ \uparrow\rangle = -\mathrm{i}\ , \tag{3.142}$$

where the last result used the normalization $\langle\uparrow\ |\ \uparrow\rangle = 1$. It is not surprising that (3.141) and (3.142) give opposite results because the spin operators σ_x and σ_y do not commute. The consequence is that there is a pairwise cancellation of the six terms in (3.139) that involve the subscripts xy, yx, xz, zx, yz, zy.

For unpolarized neutrons with $|\uparrow\rangle$ and $|\downarrow\rangle$ of equal probabilities, we expect no bias for the three remaining terms of (3.139), which becomes

$$\begin{aligned}\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = (\gamma r_\mathrm{e})^2\frac{k_f}{k_i}\sum_i\sum_f \\ \Bigg(&\langle\lambda_i|\widetilde{M}_{\perp x}|\lambda_f\rangle\,\langle\sigma_i|\sigma_x^2|\sigma_i\rangle\langle\lambda_f|\widetilde{M}_{\perp x}|\lambda_i\rangle \\ &+\langle\lambda_i|\widetilde{M}_{\perp y}|\lambda_f\rangle\,\langle\sigma_i|\sigma_y^2|\sigma_i\rangle\langle\lambda_f|\widetilde{M}_{\perp y}|\lambda_i\rangle \\ &+\langle\lambda_i|\widetilde{M}_{\perp z}|\lambda_f\rangle\,\langle\sigma_i|\sigma_z^2|\sigma_i\rangle\langle\lambda_f|\widetilde{M}_{\perp z}|\lambda_i\rangle\Bigg)\,\delta(E_f-E_i+\hbar\omega)\ .\end{aligned} \tag{3.143}$$

The three remaining terms have factors $\langle\sigma_i|\sigma_x^2|\sigma_i\rangle$, $\langle\sigma_i|\sigma_y x^2|\sigma_i\rangle$, and $\langle\sigma_i|\sigma_z^2|\sigma_i\rangle$. These terms evaluate to 1 as for example

$$\langle\uparrow\ |\sigma_x^2|\ \uparrow\rangle = \langle\uparrow\ |\sigma_x|\ \downarrow\rangle = \langle\uparrow\ |\ \uparrow\rangle = 1\ . \tag{3.144}$$

simplifying (3.143)

$$\begin{aligned}\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = (\gamma r_\mathrm{e})^2\frac{k_f}{k_i}\sum_i\sum_f \\ \Bigg(&\langle\lambda_i|\widetilde{M}_{\perp x}|\lambda_f\rangle\,\langle\lambda_f|\widetilde{M}_{\perp x}|\lambda_i\rangle \\ &+\langle\lambda_i|\widetilde{M}_{\perp y}|\lambda_f\rangle\,\langle\lambda_f|\widetilde{M}_{\perp y}|\lambda_i\rangle \\ &+\langle\lambda_i|\widetilde{M}_{\perp z}|\lambda_f\rangle\,\langle\lambda_f|\widetilde{M}_{\perp z}|\lambda_i\rangle\Bigg)\,\delta(E_f-E_i+\hbar\omega)\ .\end{aligned} \tag{3.145}$$

In (3.145) we recognize that that the three terms in parentheses are the projections onto a set of final states of the three components of the vector $\widetilde{\boldsymbol{M}}_\perp$. This can be obtained as $\langle\lambda_f|\widetilde{\boldsymbol{M}}_\perp|\lambda_f\rangle\langle\lambda_f|\widetilde{\boldsymbol{M}}_\perp|\lambda_i\rangle$. We therefore make use of (3.133) to rewrite (3.145) as

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\,\mathrm{d}E} = (\gamma r_\mathrm{e})^2 \frac{k_f}{k_i} \sum_{\alpha,\beta} (\delta_{\alpha\beta} - \widetilde{M}_\alpha\, \widetilde{M}_\beta) \sum_i \sum_f \langle \lambda_i | \widetilde{M}_{\perp\alpha} | \lambda_f \rangle \, \langle \lambda_f | \widetilde{M}_{\perp\beta} | \lambda_i \rangle \, \delta(E_f - E_i + \hbar\omega) \; . \quad (3.146)$$

Further Reading

The contents of the following are described in the Bibliography.

Leonid V. Azároff: *Elements of X-Ray Crystallography*, (McGraw-Hill, New York 1968), reprinted by TechBooks, Fairfax, VA.

M. Born and K. Wang: *Dynamical Theory of Crystal Lattices* (Oxford Classic series, 1988).

B. Fultz and J. M. Howe: *Transmission Electron Microscopy and Diffractometry of Materials, Second Edn.* (Springer–Verlag, Heidelberg, 2002).

S. W. Lovesey: *Theory of Neutron Scattering from Condensed Matter, Vol. 1* (Oxford,1984).

S. W. Lovesey: *Theory of Neutron Scattering from Condensed Matter, Vol. 2* (Oxford,1984).

A. A. Maradudin, E. W. Montroll, G. H. Weiss and I. P. Ipatova: *The Theory of Lattice Dynamics in the Harmonic Approximation, Second Edn.* (Academic Press, New York, 1971)

V. F. Sears: *Neutron Optics* (Oxford, 1989).

G. L. Squires: *Introduction to the Theory of Thermal Neutron Scattering* (Dover, Mineola, New York 1996).

4. Dynamics of Materials and Condensed Matter

4.1 Lattice Dynamics

4.1.1 Atomic Force-Constants

Following the notation of Maradudin, et al. [1], we consider a crystal generated by the infinite repetition in space of a parallelepipedic unit cell defined by three noncoplanar vectors $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3\}$. The vectors $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{a}_3$ are the primitive lattice vectors of the crystal. We label each unit cell by a triplet l of integers (positive, negative or zero): $l = (l_1, l_2, l_3)$. The equilibrium position of the origin of the unit cell l is denoted

$$\boldsymbol{x}(l) = l_1\boldsymbol{a}_1 + l_2\boldsymbol{a}_2 + l_3\boldsymbol{a}_3 \ . \tag{4.1}$$

If there are $r > 1$ atoms per unit cell, we assign an index $\kappa = 1, 2...r$ to each atom of a unit cell and write its mass M_κ. We describe the atomic equilibrium positions with respect to the origin of a unit cell with so-called basis vectors $\{\boldsymbol{x}(\kappa), \kappa = 1, 2...r\}$ so that the equilibrium position of atom κ in cell l is then given by the sum of the lattice plus basis vectors

$$\boldsymbol{x}(l\kappa) = \boldsymbol{x}(l) + \boldsymbol{x}(\kappa) \ . \tag{4.2}$$

Thermal fluctuations induce displacements in the atomic positions; the vector $\boldsymbol{u}(l\kappa)$ is the displacement of the atom $(l\kappa)$ from its equilibrium position $\boldsymbol{x}(l\kappa)$, and the $u_\alpha(l\kappa)$ $(\alpha = x, y, z)$ are the corresponding cartesian components. The instantaneous position $\boldsymbol{R}(l\kappa)(t)$ of atom $(l\kappa)$ at time t is then

$$\boldsymbol{R}(l\kappa)(t) = \boldsymbol{x}(l\kappa) + \boldsymbol{u}(l\kappa)(t) \ . \tag{4.3}$$

The total potential energy Φ of the crystal is assumed to be a function of the instantaneous positions of all the atoms in the crystal

$$\Phi = \Phi(..., \boldsymbol{R}(l\kappa), ...\boldsymbol{R}(l'\kappa'), ...) \ . \tag{4.4}$$

and it can then be expanded in a Taylor series of the atomic displacements

$$\begin{aligned}\Phi = \Phi_0 &+ \sum_{l\kappa\alpha} \Phi_\alpha(l\kappa) u_\alpha(l\kappa) \\ &+ \frac{1}{2} \sum_{l\kappa\alpha} \sum_{l'\kappa'\alpha'} \Phi_{\alpha\beta}(l\kappa; l'\kappa') u_\alpha(l\kappa) u_\beta(l'\kappa') + ... \ .\end{aligned} \tag{4.5}$$

In the harmonic approximation of lattice dynamics, we keep only the terms of the series written explicitly in (4.5) – we neglect terms of order three and higher in the displacements. The coefficients of the Taylor series are the derivatives of the potential with respect to the displacements:

$$\Phi_\alpha(l\kappa) = \left(\frac{\partial \Phi}{\partial u_\alpha(l\kappa)}\right)_0 , \tag{4.6}$$

$$\Phi_{\alpha\beta}(l\kappa; l'\kappa') = \left(\frac{\partial^2 \Phi}{\partial u_\alpha(l\kappa)\partial u_\beta(l'\kappa')}\right)_0 , \tag{4.7}$$

where the subscript zero means that derivatives are evaluated in the equilibrium configuration (all displacements equal to zero) and Φ_0 is the static potential energy of the crystal. Because the force on any atom must vanish in the equilibrium configuration, we have [1]

$$\Phi_\alpha(l\kappa) = 0 \qquad \forall\ \alpha, l, \kappa\ . \tag{4.8}$$

The nuclear part of the classical Hamiltonian for the whole crystal is $H = T + \Phi$ where $T = \sum_{\kappa,l} \frac{\boldsymbol{p}^2_{\kappa.l}}{2M_\kappa}$ is the kinetic energy of the nuclei in the crystal. In the harmonic approximation we obtain

$$H = \sum_{\kappa,l} \frac{\boldsymbol{p}^2_{\kappa.l}}{2M_\kappa} + \Phi_0 + \frac{1}{2}\sum_{l\kappa\alpha}\sum_{l'\kappa'\alpha'} \Phi_{\alpha\beta}(l\kappa; l'\kappa') u_\alpha(l\kappa) u_\beta(l'\kappa')\ . \tag{4.9}$$

We can rewrite the Hamiltonian in matrix form

$$H = \sum_{\kappa,l} \frac{\boldsymbol{p}^2_{\kappa.l}}{2M_\kappa} + \Phi_0 + \frac{1}{2}\sum_{l\kappa}\sum_{l'\kappa'} \boldsymbol{u}^T(l\kappa)\Phi(l\kappa; l'\kappa')\boldsymbol{u}(l'\kappa')\ , \tag{4.10}$$

where we have defined the 3×3 force-constant matrix Φ for the atom pair $(l\kappa; l'\kappa')$ by

$$\Phi(l\kappa; l'\kappa') = [\Phi_{\alpha\beta}(l\kappa; l'\kappa')]\ . \tag{4.11}$$

If $(l, \kappa) \neq (l', \kappa')$, then the components of the force-constant matrix are given by the second-order derivatives of the potential of (4.7). On the other hand, if $(l, \kappa) = (l', \kappa')$,then Φ is a so-called "self-force constant", whose expression is derived from the requirement that there is no overall translation of the crystal:

$$\Phi(l\kappa; l\kappa) = -\sum_{(l',\kappa')\neq(l,\kappa)} \Phi(l\kappa; l'\kappa')\ . \tag{4.12}$$

Because $-\Phi(l\kappa; l'\kappa')\boldsymbol{u}(l'\kappa')$ is the force acting upon atom $(l\kappa)$ when atom $(l'\kappa')$ is displaced by $\boldsymbol{u}(l'\kappa')$, it follows that $\Phi(l\kappa; l'\kappa')$ must be a real symetric matrix:

$$\Phi(l\kappa; l'\kappa') = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}\ . \tag{4.13}$$

Because any crystal is invariant when translated by a lattice vector, the force-constant matrices must also have the following property:

$$\Phi(l\kappa; l'\kappa') = \Phi(0\kappa; (l'-l)\kappa') = \Phi((l-l')\kappa; 0\kappa') \tag{4.14}$$

4.1.2 Equations of Motion

From the crystal Hamiltonian, one can derive the equations of motion for all nuclei in the crystal. In the harmonic approximation, we obtain:

$$m_\kappa \ddot{\boldsymbol{u}}(l\kappa)^2 = -\sum_{l',\kappa'} \Phi(l\kappa; l'\kappa')\boldsymbol{u}(l'\kappa') \qquad \forall\ l, \kappa\ . \tag{4.15}$$

Up to this point, we have considered an ideal crystal with an infinite number of atoms and unit cells; in the previous equations, the indices l and l' were running over an infinite countable set. Recognizing that a real crystal can only be of finite size, we write N_x^{cell} (resp. N_y^{cell}, N_z^{cell}) for the number of unit cells in the crystal in the x (resp. y, z) direction, and $N^{cell} = N_x^{cell} N_y^{cell} N_z^{cell}$ is the total number of unit cells. According to 4.15, we thus have $3 \times r \times N^{cell}$ equations of motion to solve. It is then convenient to impose periodic boundary conditions on the crystal, by requiring that:

$$\boldsymbol{u}((l_x, l_y, l_z)\kappa) = \boldsymbol{u}((l_x + N_x^{cell}, l_y, l_z)\kappa) \tag{4.16}$$

$$= \boldsymbol{u}((l_x, l_y + N_y^{cell}, l_z)\kappa) \tag{4.17}$$

$$= \boldsymbol{u}((l_x, l_y, l_z + N_z^{cell})\kappa)\ . \tag{4.18}$$

We can seek the solutions having the form of plane waves of wavevector $\boldsymbol{q}$, angular frequency $\omega_{\boldsymbol{q},j}$, and polarization $\boldsymbol{e}(\boldsymbol{\kappa}, \boldsymbol{q}j)$:

$$\boldsymbol{u}_\pm(\kappa l, \boldsymbol{q}j) = \frac{1}{\sqrt{N m_\kappa}}\, \boldsymbol{e}(\kappa, \boldsymbol{q}j)\, e^{\mathrm{i}(\boldsymbol{q}\cdot\boldsymbol{r}_l \pm \omega_{\boldsymbol{q},j} t)}\ . \tag{4.19}$$

The periodic boundary conditions impose the condition that the set of possible wavevectors $\{\boldsymbol{q}\}$ are discrete, although the typically large number of unit cells in the crystal translates into a very fine mesh of $\boldsymbol{q}$-points in reciprocal space. The number of physically distinguishable wavevectors is equal to N^{cell}, and this is the number of wavevectors within the so-called first Brillouin zone in reciprocal space. A monoatomic crystal with one basis vector has a total of $3 \times N_x^{cell} \times N_y^{cell} \times N_z^{cell}$ vibrational modes. With r atoms in the basis of the unit cell, the crystal has $3 \times r \times N^{cell}$ vibrational modes, in agreement with its total number of mechanical degrees of freedom. Because there is a factor of $3r$ more modes than wavevectors, each wavevector $\boldsymbol{q}$ is associated *a-priori* with $3 \times r$ vibrational modes, identified by a "branch incex", j. Each of the $3r$ different modes corresponds to a different polarization vector $\boldsymbol{e}(\boldsymbol{\kappa}, \boldsymbol{q}j)$ and angular frequency $\omega_{\boldsymbol{q},j}$ $(1 \leq j \leq 3r)$, although degeneracies can be induced by symmetry, as discussed in section 4.3.

4.1.3 The Eigenvalue Problem for the Polarization Vector

The polarization vector, $\boldsymbol{e}(\boldsymbol{\kappa}, \boldsymbol{q}j)$, a characteristic of each mode $\boldsymbol{q}j$, contains informatio on the excursion of each atom κ in the unit cell. The vectors $\boldsymbol{e}(\boldsymbol{\kappa}, \boldsymbol{q}j)$ for all the atoms in the basis ($1 \leq \kappa \leq r$) and their associated angular frequencies $\omega_{\boldsymbol{q},j}$ can be calculated by diagonalizing the "dynamical matrix" $D(\boldsymbol{q})$. The dynamical matrix is obtained by substituting (4.19) into (4.15). It has the dimensions $(3N \times 3N)$ and is constructed from (3×3) submatrices $D(\kappa\kappa', \boldsymbol{q})$:

$$D(\boldsymbol{q}) = \begin{pmatrix} D(11, \boldsymbol{q}) & \dots & D(1N, \boldsymbol{q}) \\ \vdots & \ddots & \vdots \\ D(N1, \boldsymbol{q}) & \cdots & D(NN, \boldsymbol{q}) \end{pmatrix} . \tag{4.20}$$

Each sub-matrix $D(\kappa\kappa', \boldsymbol{q})$ is obtained by taking the Fourier transform of the force-constant matrix $\Phi(l\kappa, l'\kappa')$, considered as a function of $(l' - l)$:

$$D(\kappa\kappa', \boldsymbol{q}) = \frac{1}{\sqrt{m_\kappa m_{\kappa'}}} \sum_{l'} \Phi(0\kappa, l'\kappa')\, e^{i\boldsymbol{q}\cdot(\boldsymbol{r}_{l'} - \boldsymbol{r}_0)} , \tag{4.21}$$

where we took $l = 0$ since the summation is over all values of l' and the crystal is infinite periodic (i.e., the origin cell is arbitrary). By similarly collecting the polarization vectors into a vector of size $3 \times r$, we can rewrite our system of differential equations with the plane wave solutions (4.19) in the form of an eigenvalue problem:

$$D(\boldsymbol{q})\, \boldsymbol{e}(\boldsymbol{q}j) = \omega^2_{\boldsymbol{q},j}\, \boldsymbol{e}(\boldsymbol{q}j) , \tag{4.22}$$

where

$$\boldsymbol{e}(\boldsymbol{q}j) = \begin{pmatrix} e_x(1, \boldsymbol{q}j) \\ e_y(1, \boldsymbol{q}j) \\ e_z(1, \boldsymbol{q}j) \\ \vdots \\ e_z(N, \boldsymbol{q}j) \end{pmatrix} . \tag{4.23}$$

It can be shown that the $(3r \times 3r)$ dynamical matrix $D(\boldsymbol{q})$ is hermitian (for any value of $\boldsymbol{q}$), and thus is fully diagonalizable. The $3r$ eigenvectors and eigenvalues of the dynamical matrix evaluated at a particular wavevector $\boldsymbol{q}$ then correspond to the $3r$ eigenmodes of vibration of the crystal for that wavevector.

4.1.4 Calculation of the Phonon Density of States

In order to calculate the phonon density of states (DOS) of the crystal, we need to diagonalize the dynamical matrix at a large number of points in the first Brillouin zone of reciprocal space. The diagonalization of $D(\boldsymbol{q})$ at each $\boldsymbol{q}$ point returns $3r$ eigenvalues of angular frequency $\omega_{\boldsymbol{q},j}$ $(1 \leq j \leq 3r)$, which

are then binned into a DOS histogram. By carrying the procedure at a large number of wavevectors in reciprocal space, we can calculate the density of states.

Some description of procedures is needed to help connect to the actual computer codes.

4.1.5 Symmetry Constraints on the Force-Constant Matrices

4.1.6 References

[1] A. A. Maradudin, E.W. Montroll, G. H. Weiss and I. P. Ipatova: *The Theory of Lattice Dynamics in the Harmonic Approximation, Second Edn.* (Academic Press, New-York, 1971)
[2] G. Venkataraman, L. A. Feldkamp and V. Sahni: *Dynamics of Perfect Crystals* (MIT Press, Cambridge, 1975)
[3] M. Born and K. Huang: *Dynamical Theory of Crystal Lattices* (Oxford Classic series, 1988)
[4] P. Brüesch: *Phonons: Theory and Experiment I, Lattice Dynamics and Models of Atomic Forces* (Springer-Verlag, Berlin, 1982)
[5] D. C. Wallace: *Thermodynamics of Crystals* (Wiley, 1972)

4.2 Harmonic, Quasiharmonic and Anharmonic Phonons

4.2.1 Definitions

In mechanics, the words "harmonic oscillator" and "anharmonic oscillator" are well understood. A harmonic oscillator is composed of masses connected by generalized forces that are linear with a displacement of a generalized coordinate. This is Hooke's law. The word "anharmonic" describes any oscillator with forces that deviate from this linear response. In thermodynamics, the word "anharmonic" is further subdivided into "quasiharmonic" and "anharmonic," which is given a more restricted meaning. Unfortunately, there is no universal agreement on where this division occurs. Furthermore, the word "harmonic" is itself sometimes a bit negotiable.

For thermodynamics, we will use the following choices of words to explain how phonon frequencies change with the intensive variables temperature and pressure:

- Harmonic phonons undergo no change in frequency with T or P. The frequencies of harmonic phonons at all T and P are the same as at 0 K and 0 GPa pressure.
- Quasiharmonic phonons have frequencies that depend on volume only. At a fixed volume, however, they behave as harmonic oscillators. These frequencies can change with temperature, but only because thermal expansion

alters the volume of the solid. Their frequencies would not change with temperature if a controlled pressure were used to maintain a constant volume of the material.
- Anharmonic phonons change their frequencies for other reasons. For example, the interatomic potential may change with temperature because chemical bonding is altered owing to electronic excitations across the Fermi surface. Thermal expansion is not necessarily linked to this process, so there is a pure temperature dependence to the phonon frequencies, independent of volume effects.

4.2.2 Phonons in Thermodynamics

We can relate phonon changes with V and T to thermodynamics by considering how the energy, E, and entropy, S, vary with temperature (the independent variable). The Helmholtz free energy, $F = E - TS$, is expected to vary with temperature as:

$$\frac{\mathrm{d}F}{\mathrm{d}T} = \frac{\partial E}{\partial T}\bigg)_V + \frac{\partial E}{\partial V}\bigg)_T \frac{\mathrm{d}V}{\mathrm{d}T} - S - T\left[\frac{\partial S}{\partial T}\bigg)_V + \frac{\partial S}{\partial V}\bigg)_T \frac{\mathrm{d}V}{\mathrm{d}T}\right]. \qquad (4.24)$$

For the thermodynamics of electrons and phonons (neglecting magnetic entropy, for example), the physical origins of the five terms in Eq. (4.24) are:

- $\partial E/\partial T\big)_V$ includes the temperature-dependent occupancies of phonon and electron states.
- $\partial E/\partial V\big)_T \, \mathrm{d}V/\mathrm{d}T$ is the change in electronic energy that can be described as the work done against the bulk modulus owing to thermal expansion, $\mathrm{d}V/\mathrm{d}T$.
- S, evaluated at temperature T, is a large term that contains both electron and phonon parts, S^{el} and S^{ph}. It is often the only temperature-dependence in simple harmonic models for which $\mathrm{d}F/\mathrm{d}T \simeq \partial F/\partial T$. The big source of this temperature dependence of $S(T)$ is the Bose-Einstein occupancy factor of phonon states. At high temperatures, the energy in each phonon mode is $k_{\mathrm{B}}T$, and all phonon modes contribute equally to the heat capacity. For harmonic solids at high temperatures, differences in the phonon DOS will cause a difference in entropy for different phases, but this difference in entropy does not change with temperature. (S increases with T by the same amount for each phase.)
- $\partial S/\partial T\big)_V$ accounts for the temperature-dependence of the electron and phonon entropy, beyond what is expected from the Bose-Einstein occupancy factor. The temperature dependence of the phonon entropy involves changes in the energies of the phonon states (an anharmonic effect). It is commonly assumed that this term is zero (i.e., the energies of phonon states do not change with temperature if the volume is constant), but this assumption is often incorrect.

- $\partial S/\partial V\big)_T \, \mathrm{d}V/\mathrm{d}T$ includes both phonon and electron contributions. The phonon part is often parameterized by a Grüneisen parameter, but the electronic part can also be parameterized by an "electronic Grüneisen parameter."

We define harmonic, quasiharmonic, and anharmonic entropies in the terms of (4.24). For the phonon part of the entropy we have:

$$S_{\mathrm{harm}} = S \,, \tag{4.25}$$

$$S_{\mathrm{quasi}} = S + \Delta T \left[\frac{\partial S}{\partial V}\right)_T \frac{\mathrm{d}V}{\mathrm{d}T}\right] , \tag{4.26}$$

$$S_{\mathrm{anh}} = S + \Delta T \left[\frac{\partial S}{\partial V}\right)_T \frac{\mathrm{d}V}{\mathrm{d}T} + \frac{\partial S}{\partial T}\right)_V\Bigg] . \tag{4.27}$$

One view of the Grüneisen parameter, γ, is that it relates the change in entropy to the fractional change in volume of a solid:

$$S = S_0 \left(\frac{V}{V_0}\right)^{\gamma} , \tag{4.28}$$

$$\frac{\partial S}{\partial V}\Big)_T \simeq \gamma S \,. \tag{4.29}$$

Further recognizing that $\mathrm{d}V/\mathrm{d}T = \beta V$, where β is the volume coefficient of thermal expansion, we can use (4.29) to simplify (4.26) and (4.27):

$$S_{\mathrm{quasi}} = S\Big[1 + \Delta T \gamma \beta\Big] \,, \tag{4.30}$$

$$S_{\mathrm{anh}} = S\Big[1 + \Delta T \gamma \beta\Big] + \frac{\partial S}{\partial T}\Big)_V \,, \tag{4.31}$$

where we have switched to entropy per unit volume.

When electronic entropy is included, it is possible to use an analogous "electronic Grüneisen parameter" to express the volume dependence of the electronic entropy

$$\frac{\partial S^{\mathrm{el}}}{\partial V}\Big)_T = \gamma^{\mathrm{el}} S^{\mathrm{el}} \,. \tag{4.32}$$

Equation (4.32) can be used to extend the definition of the word quasiharmonic, but this convention has not yet been established. The temperature dependence of the electronic contributions to S caused by Fermi-Dirac thermal occupancy factors are quite conventional, and are perhaps appropriate for a "harmonic" models. Effects of electron-phonon interactions, however, are probably best considered as "anharmonic" effects.

4.2.3 Phonons and Heat Capacity

Suppose we have measured the phonon DOS at different temperatures, and found that the DOS changes with temperature. What are the effects of this temperature-dependence on the heat capacity of the material? We need to know how the heat goes into the material. Energy goes into phonon creation, but energy also goes into the expansion of a crystal against its bulk modulus, electronic excitations, magnetic excitations, and these excitations may interact with each other. For a harmonic solid without other interactions, it is acceptable to consider only the energy required to populate the phonon states with increasing temperature. Although energy goes into the phonon states, phonons also contribute to the entropy of the solid. In a general thermodynamic analysis, it is necessary to minimize a free energy function that includes phonon energy, phonon entropy, and the elastic energy of thermal expansion, for example. The latter can make a large contribution and change the heat capacity in a qualitative way.

Consider two analyses of phonon softening:

1. First ignore thermal expansion. The heat capacity is calculated as C_V for a distribution of harmonic oscillators. We can account for how the frequencies of these oscillators decrease with temperature, and C_V is obtained by accounting for this continuous decrease in frequency as a function of temperature. For a particular phonon DOS, shown as the inset in Fig. 4.1, C_V was computed at each temperature with the (softened) phonon DOS appropriate to that temperature. In this case all phonon softened by the same fraction of their energy to maximum of 6.5 % at 300 K. This fraction was assumed linear with temperature. At each temperature T, each mode of energy ε was assumed to contribute to the heat capacity the amount

$$C_{V,mode}(T) = k_\mathrm{B} \left(\frac{\varepsilon}{k_\mathrm{B}T} \right)^2 \frac{\exp(\varepsilon/k_\mathrm{B}T)}{(\exp(\varepsilon/k_\mathrm{B}T) - 1)^2} \ . \tag{4.33}$$

This expression can be obtained by differentiating the Planck distribution, $[\exp(\varepsilon/k_\mathrm{B}T) - 1]^{-1}$, and weighting by the energy of the mode, ε.

Notice in Fig. 4.1 that the total heat capacity curve differs little from the pure harmonic case with no phonon softening. The rise in heat capacity occurs at temperatures that are a bit lower, but the effect is essentially negligible. Additionally, there is no increase in the heat capacity above the Dulong-Petit limit of $3R$ at high temperature. In the high-temperature limit, with phonon occupancies of $k_\mathrm{B}T/\varepsilon$ per mode, the softening of the modes gives an increase in occupancy proportional to $1/\varepsilon$, but a decrease in energy proportional to ε. The product of these two factors produces a heat capacity C_V at high temperatures that is unaffected by mode softening. Even with phonon softening, C_V at high temperatures remains $3R$.

2. Now allow for thermal expansion. We have to consider a minimization of free energy where the phonon entropy compensates the elastic energy of ex-

pansion.[1] Here we first need to assess the entropy of the harmonic oscillators as a function of temperature

$$S_{osc}(T) = \frac{\varepsilon/T}{\exp(\varepsilon/k_{\mathrm{B}}T) - 1} - k_{\mathrm{B}} \ln[1 - \exp(-\varepsilon/k_{\mathrm{B}}T)] \,. \tag{4.34}$$

This expression can be obtained from the partition function of a harmonic oscillator, $\mathcal{Z} = \sum_n \exp(-n\varepsilon/k_{\mathrm{B}}T) = [1 - \exp(-\varepsilon/k_{\mathrm{B}}T)]^{-1}$. Obtain the free energy from this partition function as $F = -k_{\mathrm{B}}T \ln \mathcal{Z}$, and differentiate to obtain (4.34) as $S = -\partial F/\partial T$. Without thermal expansion, (4.34) is consistent with (4.33) if we integrate:

$$S_{V,osc}(T) = \int_0^T \frac{C_{V,mode}(T')}{T'} \, \mathrm{d}T' \,, \tag{4.35}$$

(an expression which can be remembered from $T \, \mathrm{d}S_{V,osc}(T) = C_{V,mode}(T) \, \mathrm{d}T$). The 6.5% shift of ε from 27 to 300 K was taken into account by making ε a linear function of T in the same way as in example 1. With the total entropy of the solid (assumed to be entirely phonon entropy), we can calculate the amount of heat going into the solid at constant pressure by using the definition of C_p

$$C_{p,mode}(T) = T \frac{dS}{dT}\bigg)_p \,. \tag{4.36}$$

The result is also shown in Fig. 4.1.

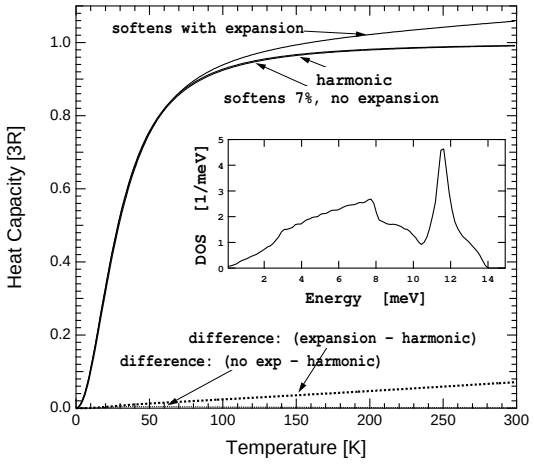


Fig. 4.1. Heat capacity versus temperature for two assumed physical models (examples 1 and 2 in text). Inset is the phonon DOS at 0 K – all modes were assumed to soften by 6.5 % at 300 K. A simple harmonic heat capacity with a phonon DOS unchanged with temperature is also shown.

[1] In general, other sources of entropy may compensate for this expansion, or other types of excitation can occur with temperature.

At high temperatures, $C_{p,mode}(T)$ can be shown to approach $3R$ if there is no thermal expansion. With thermal expansion, however, the difference between C_p and C_V is the famous classical result: $9Bv\alpha^2T$, where B, α, and v are bulk modulus, coefficient of linear thermal expansion, and specific volume. This exceeds $3R$ at elevated temperature, as shown in Fig. 4.1. The energy going into thermal expansion makes a big difference in the heat capacity. The phonon softening itself provides an entropy to allow this to occur, but the phonon contribution to the heat capacity from softening alone is not so large as the energy of the expanded solid.

4.3 Group Theory and Lattice Dynamics

4.3.1 Real Space

The paper by A. A. Maradudin and S. H. Vosko "Symmetry Properties of the Normal Modes of a Crystal", Reviews of Modern Physics **40**, 1-37 (1968), shows how group theory can be used to understand the normal modes of crystal vibrations. The focus of their substantial manuscript is on using the symmetry of the reciprocal lattice to find the eigenvectors of the dynamical matrix and classify them. The degeneracies of the different normal modes are addressed rigorously, and their association with the symmetry elements of the crystal are useful in labeling them. Projection operator methods offer the possibility of finding eigenvectors and eigenvalues without diagonalizing the dynamical matrix.

The paper by J. L. Warren "Further Considerations on the Symmetry Properties of the Normal Modes of a Crystal", Reviews of Modern Physics **40**, 38-76 (1968), also describes the point group of the bond. The bond means the interactions between an atom and its neighbors. The group of the bond refers to the real space symmetries of the interatomic interactions, i.e., how the force constants transform under the point group operations at a central atom.

Consider the space group operator in Seitz notation:

$$\underline{\mathcal{S}} \equiv \{\mathbf{S} \,|\, \boldsymbol{v}(S) + \boldsymbol{x}(m)\} \,, \tag{4.37}$$

where $\mathbf{S}$ is a rotation operation, $\boldsymbol{x}(m)$ is a lattice translation, and $\boldsymbol{v}(S)$ is an additional displacement that is required for non-symmorphic crystals (i.e., crystals with screw axes or glide planes).[2] Apply this operator to a displacement vector u (the displacement of an atom off its site during a vibration):

$$u'_\alpha(L,K) = \sum_\beta S_{\alpha\beta} u_\beta(l,\kappa) \,, \tag{4.38}$$

where the operation serves to mix the Cartesian components (denoted subscript α) of the displacement vector for the atom at l,κ, and translates the

[2] Note that this vector $\boldsymbol{v}(S)$ mixes the rotation and translation operations.

vector to the atom at L, K. The potential energy of the crystal depends quadratically on the u_α, times a force constant matrix involving the spatial derivatives of the potential along the Cartesian axes of the u_α. The potential energy of the crystal is invariant under symmetry operations of (4.37). This leads to the equality:

$$\Phi_{\mu\nu}(LK, L'K') = \sum_{\alpha\beta} S_{\mu\alpha} S_{\nu\beta} \Phi_{\alpha\beta}(l\kappa, l'\kappa') \; . \tag{4.39}$$

The matrix equation (4.39) must be true for all valid force constant matrices $\Phi_{\alpha\beta}(l\kappa, l'\kappa')$. It is instructive to set $\mathbf{S} = \mathbf{I}$, the identity, so there is no rotation, and only translations are imposed. This can be used to show that force constants can depend only on relative separations between atoms, not absolute positions.

A number of other tests of force constants can be performed with (4.39). Imposing rotations that mix the Cartesian axes can be used to demonstrate that some force constants are equal. Force constants that must equal their negative are found – this result means that the force constant is zero. Equation (4.39) could be handy for determining allowed force constants if a set of space group matricies, $\underline{\mathcal{S}}$, are available.

4.3.2 *k*-space

Quantum Mechanics. In quantum mechanics, group theory addresses the symmetry of the hamiltonian, H. If an operator, R, commutes with H,

$$RH = HR \; , \tag{4.40}$$

$$\sum_j R_{ij} H_{jk} = \sum_j H_{ij} R_{jk} \; , \tag{4.41}$$

it is convenient to select basis functions for which R is diagonal, so the single remaining terms in the sums are:

$$R_{ii} H_{ik} = H_{ik} R_{kk} \; , \tag{4.42}$$

$$(R_{ii} - R_{kk}) H_{ik} = 0 \; . \tag{4.43}$$

Equation (4.43) shows that when $i \neq k$, then the off-diagonal elements $H_{ik} = 0$. *The basis functions for which R is diagonal are the same for which H is diagonal.*

For example, consider the rotational symmetry of a hydrogen atom about the $\widehat{z}$-axis. The Abelian group of rotations has 1-dimensional representations in terms of functions $\mathrm{e}^{\mathrm{i}m\phi}$. Done. Obtaining the ϕ-dependence this way was much easier than solving the Schrödinger equation in spherical coordinates.

Lattice Dynamics. The analogous question for lattice dynamics is, "Can we use symmetry to get the eigenvectors of the dynamical matrix without solving the eigenvalue problem in detail?" The dynamical matrix is:

$$D_{\alpha\beta}(\kappa\kappa'|\boldsymbol{k}) = \frac{1}{\sqrt{M_\kappa M_{\kappa'}}} \sum_{l'} \Phi_{\alpha\beta}(l\kappa; l'\kappa') \exp[-\mathrm{i}\boldsymbol{k} \cdot (\boldsymbol{x}(l) - \boldsymbol{x}(l'))] \quad (4.44)$$

and the relevant eigenvalue equation is:

$$\sum_{\beta\kappa'} D_{\alpha\beta}(\kappa\kappa'|\boldsymbol{k}) u_\beta(\kappa') = \omega^2 u_\alpha(\kappa) \ . \quad (4.45)$$

All functions in (4.45) live in reciprocal space. The entire paper of Maradudin and Vosko is in reciprocal space, or k-space.

Table 4.1. Elementary Variables Used by Maradudin and Vosko

l, l'	indices of unit cells
κ, κ'	indices of atom in basis
r	number of atoms in basis
$\boldsymbol{x}_{l,\kappa}$	atom site in a crystal
$\boldsymbol{x}_l$	lattice site in a crystal
$\boldsymbol{x}_\kappa$	basis vector
M_κ	mass of atom in basis
α, β	Cartesian indicies $\{x, y, z\}$
$\boldsymbol{u}(\kappa)$ and $u_\alpha(\kappa)$, $u_\beta(\kappa)$	atom displacement in a phonon, and Cartesian components
$\Phi_{\alpha\beta}(l\kappa; l'\kappa')$	force constant between two atoms
$\boldsymbol{e}(\boldsymbol{k}, \sigma, \lambda)$	phonon polarization vector (with components for all κ in unit cell)
σ (or s)	denotes (or index for) a distinct value of ω_j^2
λ (or a)	denotes (or index for) independent eigenvector for each σ or ω_j^2
f_σ	number of eigenvectors for each degenerate energy ($1 \le a \le f_\sigma$)
$\underline{\mathcal{S}} \equiv \{\mathbf{S} \,\vert\, \boldsymbol{v}(S) + \boldsymbol{x}(m)\}$	Seitz space group operator
h	order of the group (number of elements)
$\mathbf{S}$	matrix of rotation or improper rotation
$\boldsymbol{v}(S)$	little displacement vector for a screw axis operation
$\boldsymbol{x}(m)$	lattice translation vector
$\boldsymbol{\Gamma}_{\underline{\mathcal{S}}}(\boldsymbol{k}; \{\mathbf{S} \,\vert\, \boldsymbol{v}(S) + \boldsymbol{x}(m)\})$	$3r \times 3r$ unitary matrix of symmetry operator (in recip. space)
$\mathbf{R}_i \equiv \{\mathbf{R} \,\vert\, \boldsymbol{v}(S) + \boldsymbol{x}(m)\}$	symmetry operator for a single wavevector $\boldsymbol{k}$

Consider the matrix $\boldsymbol{\Gamma}_{\underline{\mathcal{S}}}(\boldsymbol{k})$ that performs symmetry operations $\underline{\mathcal{S}}$ on the dynamical matrix. Doing two such operations is tricky, since the $\boldsymbol{k}$ used in constructing the matrix for the second operation depends on the result of the

first operation. Maradudin and Vosko choose instead to work with the space group of a single wavevector $\boldsymbol{k}$, and designate the symmetry operations by $\mathbf{R}_i = \{\mathbf{R} \,|\, \boldsymbol{v}(S)+\boldsymbol{x}(m)\}$. These symmetry operations on $\boldsymbol{k}$ produce equivalent wavevectors. In particular, the rotational part of $\mathbf{R}_i$ therefore must generate a $\boldsymbol{k}$ that differs only by a reciprocal lattice vector $\boldsymbol{g}$:

$$\mathbf{R}\boldsymbol{k} = \boldsymbol{k} - \boldsymbol{g} \; . \tag{4.46}$$

This is quite restrictive, and means that $\boldsymbol{g}$ is zero for most vectors $\boldsymbol{k}$. The situation is much like the allowed k-vectors for ordered structures in the Landau-Khachaturyan formalism of second-order phase transitions. The operations of (4.46) eliminate the equivalent vectors from the star of $\boldsymbol{k}$. Inequivalent $\boldsymbol{k}$ can exist only at special points such as the boundaries of Brillouin zones.

So although:

$$\mathbf{D}(\underline{\mathcal{S}}\boldsymbol{k}) = \boldsymbol{\Gamma}^{-1}_{\underline{\mathcal{S}}}(\boldsymbol{k})\mathbf{D}(\boldsymbol{k})\boldsymbol{\Gamma}_{\underline{\mathcal{S}}}(\boldsymbol{k}) \; , \tag{4.47}$$

and therefore in general:

$$\mathbf{D}(\boldsymbol{k}) \neq \boldsymbol{\Gamma}^{-1}_{\underline{\mathcal{S}}}(\boldsymbol{k})\mathbf{D}(\boldsymbol{k})\boldsymbol{\Gamma}_{\underline{\mathcal{S}}}(\boldsymbol{k}) \; , \tag{4.48}$$

with the restriction to the group of the wavevector $\boldsymbol{k}$, the unitary matricies $\boldsymbol{\Gamma}_{\mathbf{R}_i}$ commute with $\mathbf{D}$:

$$\mathbf{D}(\boldsymbol{k}) = \boldsymbol{\Gamma}^{-1}_{\mathbf{R}_i}(\boldsymbol{k})\mathbf{D}(\boldsymbol{k})\boldsymbol{\Gamma}_{\mathbf{R}_i}(\boldsymbol{k}) \; . \tag{4.49}$$

These $\{\boldsymbol{\Gamma}_{\mathbf{R}_i}(\boldsymbol{k})\}$ actually form a $3r$-dimensional unitary representation of the space group of the wavevector, $\boldsymbol{k}$. They could be used to develop the symmetry properties of the eigenvectors of the dynamical matrix, but this is not the approach of Maradudin and Vosko.

Multiplier Representation. Maradudin and Vosko define a new matrix $\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)$, which differs from the $\boldsymbol{\Gamma}_{\mathbf{R}_i}(\boldsymbol{k})$ by an exponential phase factor of modulus unity:

$$\begin{aligned}\mathbf{T}(\boldsymbol{k};\{\mathbf{R}_i \,|\, \boldsymbol{v}(S)+\boldsymbol{x}(m)\}) = \exp[\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{v}(S)+\boldsymbol{x}(m))] \\ \times\boldsymbol{\Gamma}_{\mathbf{R}_i}(\boldsymbol{k};\{\mathbf{R} \,|\, \boldsymbol{v}(S)+\boldsymbol{x}(m)\}) \; ,\end{aligned} \tag{4.50}$$

This choice of $\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)$ is somewhat unusual because these matricies do not form a group in the usual sense – the product of two of them is not an element of the group:

$$\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)\mathbf{T}(\boldsymbol{k};\mathbf{R}_j) = \exp[\mathrm{i}\boldsymbol{g}\cdot\boldsymbol{v}(\mathbf{R}_j)]\mathbf{T}(\boldsymbol{k};\mathbf{R}_i\mathbf{R}_j) \; . \tag{4.51}$$

(where the product of exponentials of two cases of (4.50) were simplified by noting the restrictive condition (4.46) on the allowed $\mathbf{R}$, giving the condition $\boldsymbol{g}\cdot\boldsymbol{x}(m) = 2\pi m$, and exponential phase factors of $+1$). The phase factor $\exp[\mathrm{i}\boldsymbol{g}\cdot\boldsymbol{v}(S)]$ can differ from $+1$, *although it differs from +1 only for crystals with screw axes AND special k-points.*[3]

[3] Is it important to study phonons (or magnons) at special k-points in crystals with screw axes (non-symmorphic crystals)?

Matrices $\{\mathbf{T}(\boldsymbol{k};\mathbf{R}_i\}$ that obey a multiplication rule of (4.51) are said to form a "multiplier representation" of the group of $\{\mathbf{R}_i\}$. The phase factor is the "multiplier." For most crystals this is +1, and we are back to ordinary group representations. Even if the multiplier is complex, however, if we can get the eigenvectors of these $\mathbf{T}$, we can get the eigenvectors of the dynamical matrix, $\mathbf{D}$.

We rewrite (4.45):

$$\mathbf{D}(\boldsymbol{k})\boldsymbol{e}(\boldsymbol{k}\sigma\lambda) = \omega_\sigma^2 \boldsymbol{e}(\boldsymbol{k}\sigma\lambda)\ , \quad \lambda = 1,2,3\ldots f_\sigma\ , \tag{4.52}$$

noting that:

$$\mathbf{D}(\boldsymbol{k})[\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)\boldsymbol{e}(\boldsymbol{k}\sigma\lambda)] = \omega_\sigma^2[\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)\boldsymbol{e}(\boldsymbol{k}\sigma\lambda)]\ . \tag{4.53}$$

The $\{\mathbf{T}\}$ mix the degenerate $\boldsymbol{e}(\boldsymbol{k}\sigma\lambda)$ (i.e., they mix those λ that go with the same σ). Showing this more clearly, we write:

$$\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)\boldsymbol{e}(\boldsymbol{k}\sigma\lambda) = \sum_{\lambda'}^{f_\sigma} \boldsymbol{\tau}^{(\sigma)}_{\lambda'\lambda}(\boldsymbol{k};\mathbf{R}_i)\boldsymbol{e}(\boldsymbol{k}\sigma\lambda')\ , \tag{4.54}$$

for every $\mathbf{R}_i$ in the group of the wavevector $\boldsymbol{k}$. The f_σ functions transform among themselves under $\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)$. Furthermore, from (4.49) we can show that $\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)$ commutes with $\mathbf{D}$. These conditions are sufficient to show that the $\{\boldsymbol{\tau}^{(\sigma)}(\boldsymbol{k};\mathbf{R}_i)\}$ provide an f_σ-dimensional *irreducible* multiplier representation of the point group of the wavevector $\boldsymbol{k}$.

Now make a vector of all eigenvectors:

$$\mathbf{e}(\boldsymbol{k}) = \Big[\boldsymbol{e}(\boldsymbol{k}\sigma_1\lambda_1), \boldsymbol{e}(\boldsymbol{k}\sigma_1\lambda_2), \ldots \boldsymbol{e}(\boldsymbol{k}\sigma_2\lambda_1), \boldsymbol{e}(\boldsymbol{k}\sigma_2\lambda_2), \ldots\Big]\ , \tag{4.55}$$

which transforms as:

$$\mathbf{T}(\boldsymbol{k};\mathbf{R}_i)\mathbf{e}(\boldsymbol{k}) = \mathbf{e}(\boldsymbol{k})\boldsymbol{\Delta}(\boldsymbol{k};\mathbf{R}_i)\ , \tag{4.56}$$

and the $3r \times 3r$ dimensional matrix $\boldsymbol{\Delta}(\boldsymbol{k};\mathbf{R}_i)$ has the block-diagonal form:

$$\boldsymbol{\Delta}(\boldsymbol{k};\mathbf{R}_i) = \begin{bmatrix} \boldsymbol{\tau}^{(1)}(\boldsymbol{k};\mathbf{R}_i) & 0 & 0 & \ldots \\ & \boldsymbol{\tau}^{(2)}(\boldsymbol{k};\mathbf{R}_i) & 0 & \ldots \\ 0 & 0 & \boldsymbol{\tau}^{(3)}(\boldsymbol{k};\mathbf{R}_i) & \ldots \\ . & & & \end{bmatrix}\ . \tag{4.57}$$

The unitary matrices $\boldsymbol{\tau}^{(\sigma)}(\boldsymbol{k};\mathbf{R}_i)$ are known for all 230 space groups. In a formal sense, the problem is solved.

Projection Operators. In a practical sense, we can generate the eigenfunctions by projection operator machinery. Maradudin and Vosko show that:

$$\underline{\mathcal{P}}^{(s)}_{\lambda\lambda'}(\boldsymbol{k}) = f_s/h \sum_{\mathbf{R}_i}^{h} \boldsymbol{\tau}^{(s)}_{\lambda'\lambda}(\boldsymbol{k};\mathbf{R}_i)^*\ \mathbf{T}(\boldsymbol{k};\mathbf{R}_i) \tag{4.58}$$

is a projection operator.

The projection operator method is considered standard practice, and is not developed further by Maradudin and Vosko. Perhaps it is useful to review it in the context of group theory in quantum mechanics.

Suppose a set of $\{\phi_\lambda^{(j)}\}$ are partner functions in the representation (j), and transform among themselves under operation of the symmetry operator $\underline{P}_{\mathbf{R}_i}$. Explicitly:

$$\underline{P}_{\mathbf{R}_i}\phi_\kappa^{(j)} = \sum_{\lambda=1}^{l_j} \phi_\lambda^{(j)} \boldsymbol{\Gamma}_{\lambda\kappa}^{(j)}(\mathbf{R}_i) \; . \tag{4.59}$$

Multiply by the complex conjugates of all representation matrices and sum over all elements in the group:

$$\sum_{\mathbf{R}_i}^{h} \boldsymbol{\Gamma}_{\lambda' k'}^{(i)}(\mathbf{R}_i)^* \underline{P}_{\mathbf{R}_i}\phi_\kappa^{(j)} = \sum_{\mathbf{R}_i}^{h}\sum_{\lambda=1}^{l_j} \boldsymbol{\Gamma}_{\lambda' k'}^{(i)}(\mathbf{R}_i)^* \boldsymbol{\Gamma}_{\lambda k}^{(j)}(\mathbf{R}_i)\phi_\kappa^{(j)} \; , \tag{4.60}$$

$$\sum_{\mathbf{R}_i}^{h} \boldsymbol{\Gamma}_{\lambda' k'}^{(i)}(\mathbf{R}_i)^* \underline{P}_{\mathbf{R}_i}\phi_\kappa^{(j)} = \delta_{ij}\delta_{\kappa\kappa'}\phi_{\lambda'}^{(j)} \; , \tag{4.61}$$

where the last line was obtained through the Great Orthogonality Theorem of group representation theory. Equation (4.61) shows how a projection operator, $\underline{\mathcal{P}}_{\lambda k}^{(j)}$:

$$\underline{\mathcal{P}}_{\lambda k}^{(j)} = l_j/h \sum_{\mathbf{R}_i} \boldsymbol{\Gamma}_{\lambda'\kappa'}^{(i)}(\mathbf{R}_i)^* \underline{P}_{\mathbf{R}_i} \; , \tag{4.62}$$

pulls out the function $\phi_{\lambda'}^{(j)}$.

For the eigenvectors of the dynamical matrix for a particular $\boldsymbol{k}$, the projection operator of (4.58) does the same thing. Since the matricies $\boldsymbol{\tau}^{(\sigma)}(\boldsymbol{k};\mathbf{R}_i)$ are available, one could use projection operator techniques to operate on an arbitrary vector and generate all eigenfunctions of the dynamical matrix. Labels can be assigned to various phonons based on group theory designations. Another step could be to substitute the $\boldsymbol{e}(\boldsymbol{k}\sigma\lambda)$ into (4.52) and obtain ω_σ^2 by matrix multiplication rather than by matrix inversion. Is this useful, or just elegant? Projection operator methods will reveal all degeneracies of normal modes for a particular $\boldsymbol{k}$. In such cases where a set of partner eigenvectors is found, it is not possible to obtain them directly. This requires diagonalizing the dynamical matrix to solve for the eigenvectors, or more accurately a block diagonal form of the dynamical matrix as in (4.57).

Compatibility Relations. The paper by J. L. Warren "Further Considerations on the Symmetry Properties of the Normal Modes of a Crystal", Reviews of Modern Physics **40**, 38-76 (1968), adds to the discussion of Maradudin and Vosko. Compatibility relationships are described in more detail in Warren's paper. Compatibility relationships refer to the new degeneracies that appear when special high symmetry directions in the Brillouin

zone come together at special points, such as at the point Γ at the center of the zone, or at the points {X, W, K, L} on the surface of the zone.

For example, when moving along a line in k-space along the (100) direction, an abrupt change in symmetry occurs when the edge of the zone (X) or center of the zone (Γ) is reached. The source of this new symmetry is the fact that at these special points, translational vectors of the reciprocal lattice cause the points to become equivalent (identical) to other points in the reciprocal lattice.[4] For example, at an arbitrary position along the (100) direction, the symmetry elements are $\{E, C_4^2, 2C_4, 2\sigma_v, 2\sigma_d\}$. This fourfold axis obviously does not have threefold symmetry elements, but the situation changes at the origin, point Γ. The (111) direction, which includes a threefold axis, also converges at Γ. Some representations along the (100) axis are expected to be compatible with some representations along the (111) axis when the two axes meet at the origin. The larger symmetry group at the orgin, point Γ, contains the elements of the threefold and fourfold axes as subgroups. Some representations of the groups from the threefold and fourfold axes are compatible at Γ, while some are not. Those that are compatible at Γ must have eigenvalues that are degenerate in energy. Their associated phonon branches rigorously converge in energy at Γ.

The way to test for compatibility is to inspect the characters, $\chi(\mathbf{R}_i) = \mathrm{Tr}[\Gamma(\mathbf{R}_i)]$, of the representations of the three groups. This can be done with the decomposition formula of group representation theory in exactly the same way as is done for analyzing the degeneracies in crystal field theory. (After all, we are considering only the point group of the wavevector.) The decomposition formula for the characters is:

$$a_i = \frac{1}{h} \sum_{\mathbf{R}_i} \chi(\mathbf{R}_i)^* \chi(\mathbf{R}) \ , \tag{4.63}$$

where a_i is the number of times that a particular representation of the subgroup (e.g., of the threefold axis) appears in the group representation of the special point (e.g., Γ).

Equation (4.63) is most easily used by inspection of character tables.[5] For example, if all characters of a representation along the fourfold axis match the characters of a representation at Γ, these representations are compatible. In a more complicated situation where there is only one $a_i = 1$, the sum of the characters from the representations along the fourfold axis must equal the character of a representation at Γ. For the simplest example, the characters of all symmetry elements in the Δ_1 representation of the group of the fourfold axis are +1. The same symmetry elements have characters +1 for the Γ point for the representation Γ_1. The threefold axis has a different set of symmetry

[4] These k-vectors are not identical when they are just slightly displaced off these special points, and related eigenvectors have to be considered independently.

[5] These are found in the classic paper by L. P. Bouckaert, R. Smoluchowski, and E. P. Wigner, Phys. Rev. **50**, 58 (1936), and an excellent description is provided in Chapter 8 of M. Tinkham's book on *Group Theory and Quantum Mechanics*.

elements than the fourfold axis, but these elements are also +1 for the Γ_1 representation, and are +1 only for the Λ_1 representation. We deduce that the Δ_1 representation of the (100) fourfold axis and the Λ_1 representation of the (111) threefold are compatible at the point Γ. These two representations have the same basis functions as Γ_1, but the symmetry operators at Γ interchange these basis functions because they are degenerate in energy. The significance that the basis functions of the Δ_1 and Λ_1 representations must have exactly the same energy at the point Γ in k-space. They are not degenerate just away from the Γ point (unless accidentally, or in the case of no crystal potential), but group theory says that they are rigorously degenerate at Γ. This is a result of potential use in the analysis of phonon data. We can determine the required conditions for the merging of dispersion curves at special point in k-space. (Incidentally, s-type functions are suitable basis functions for the symmetric representations in this simple example.)

4.3.3 Time-reversal symmetry in the dynamical matrices

Time reversal symmetry must be considered in magnetic systems. Consider the spin as a rotation in a particular direction. The spin is flipped upon time reversal. Time reversal symmetry must also be considered in certain cases for non-magnetic systems, and results in additional degeneracies. In most cases time-reversal symmetry does not result in additional degeneracies, however. (Perhaps we could consider additional degeneracies as "accidental," but get the software to flag them so we can inspect for this later.)

When the Hamiltonian is invariant by the time-reversal operation (the direction of momentum, or spin is inverted), the system is said to have the time-reversal symmetry. The system without the external magnetic field has generally the time-reversal symmetry. The time-reversal symmetry often gives the nontrivial physical significance like the additional degeneracy of the system. One of most well-known examples is the Kramers degeneracy, i.e. a system of an odd number of electrons has the extra degeneracy, but an even number does not. Of course, the present ionic lattice system has such symmetry, too. Therefore, it needs to be checked if there occur additional degeneracies or not due to the symmetry, which can be done by investigating the symmetry of the dynamical matrices.

Maradudin and Vosko begins the discussion by considering the additional rotational element $\mathbf{S\_}\boldsymbol{k} = -\boldsymbol{k}$ in the point group of the crystal. In the same way as $\mathbf{R}$ (in the the point group G_0), with each element $\mathbf{S\_R}$ (in the coset $\mathbf{S\_}G_0$) we associate a new matrix operator $T(\boldsymbol{k}; \mathbf{S\_R})$, which is defined anti-unitary as it should be. The analysis for $T(\boldsymbol{k}; \mathbf{R})$ in section 4.4.3 can be extended straightforwardly to include the anti-unitary matrix operator $T(\boldsymbol{k}; \mathbf{A})$ (the element $\mathbf{S\_R}$ will be denoted by $\mathbf{A}$ hereafter). It is highly desirable to derive the condition for the existence of extra degeneracies due to the time-reversal symmetry, by looking into the linear dependence of the eigenvectors

$\boldsymbol{e}(\boldsymbol{k}sa\lambda)$ and $T(\boldsymbol{k};\mathbf{A})\boldsymbol{e}(\boldsymbol{k}sa\lambda)$ (it is clear both should be eigenvectors because the system has the time-reversal symmetry). We define

$$\bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda) = T(\boldsymbol{k};\mathbf{A})\boldsymbol{e}(\boldsymbol{k}sa\lambda) \tag{4.64}$$

and consider and compare the transformation properties of $\boldsymbol{e}(\boldsymbol{k}sa\lambda)$ and $\bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda)$ under $T(\boldsymbol{k};\mathbf{R})$

$$T(\boldsymbol{k};\mathbf{R})\boldsymbol{e}(\boldsymbol{k}sa\lambda) = \sum_{\lambda'}^{f_s} \tau^{(s)}_{\lambda'\lambda}(\boldsymbol{k};\mathbf{R})\boldsymbol{e}(\boldsymbol{k}sa\lambda') \ , \tag{4.65}$$

$$T(\boldsymbol{k};\mathbf{R})\bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda) = \sum_{\lambda'}^{f_s} \bar{\tau}^{(s)}_{\lambda'\lambda}(\boldsymbol{k};\mathbf{R})\bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda') \ , \tag{4.66}$$

where it is found that the irreducible multiplier representations $\{\bar{\tau}^{(s)}(\boldsymbol{k};\mathbf{R})\}$ and $\{\tau^{(s)}(\boldsymbol{k};\mathbf{R})\}$ belong to the same factor system. They can be either equivalent or inequivalent. In particular, they can be separated into three types depending on the relationship between two irreducible multiplier representations.

First, we think of a case the irreducible multiplier representations $\bar{\tau}^{(s)}$ and $\tau^{(s)}$ are inequivalent, which will be referred to as of the *third* type. In this case, the eigenvectors $\boldsymbol{e}(\boldsymbol{k}sa\lambda)$ and $\bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda)$ are orthogonal and the f_s-fold degeneracy in the dynamical matrix is doubled by $2f_s$ by the time-reversal symmetry.

On the other hand, it is a bit complicated for an alternative situation $\bar{\tau}^{(s)}$ and $\tau^{(s)}$ are equivalent. In the case, two representations are related by a similarity transformation, $\bar{\tau}^{(s)}(\boldsymbol{k};\mathbf{R}) = \beta^{-1}\tau^{(s)}(\boldsymbol{k};\mathbf{R})\beta$, where β is a unitary matrix. Thanks to the irreducibility of the matrices $\{\tau^{(s)}(\boldsymbol{k};\mathbf{R})\}$, the uniqueness up to a phase factor is followed for the matrix β. By Schur's Lemmas, the matrix $\beta\beta^*\tau^{(s)}(\boldsymbol{k};\mathbf{A}_0^{-2})$ commuting with all the matrices $\{\tau^{(s)}(\boldsymbol{k};\mathbf{R})\}$ of an irreducible multiplier representation of the point group must be proportional to the unit matrix. This results in two cases;

$$\beta\beta^* = \phi(\boldsymbol{k};\mathbf{A}_0;\mathbf{A}_0)\tau^{(s)}(\boldsymbol{k};\mathbf{A}_0^2) \ , \tag{4.67}$$

$$\beta\beta^* = -\phi(\boldsymbol{k};\mathbf{A}_0;\mathbf{A}_0)\tau^{(s)}(\boldsymbol{k};\mathbf{A}_0^2) \ , \tag{4.68}$$

where $\phi(\boldsymbol{k};\mathbf{A}_0;\mathbf{A}_0)$ is a "multiplier". Further, as is often the case, $\beta\beta^* = \pm 1$ for a suitable element $\mathbf{A}_0$ in $\mathbf{S}\_G_0$. Now we can turn our attention to the problem of liner independence of $\boldsymbol{e}(\boldsymbol{k}sa\lambda)$ and $\bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda)$. For the purpose, it would be easy to investigate the scalar product of two eigenvectors. For the necessary algebra, it is important to note the scalar products of a unitary and an anti-unitary transformation on vectors are distinguished as

$$\langle T(\boldsymbol{k};\mathbf{R})\varphi, T(\boldsymbol{k};\mathbf{R})\psi\rangle = \langle\varphi,\psi\rangle \ ,$$

$$\langle T(\boldsymbol{k};\mathbf{A}_0)\varphi, T(\boldsymbol{k};\mathbf{A}_0)\psi\rangle = \langle\psi,\varphi\rangle \ ,$$

and also the properties for the successive transformations under $T(\boldsymbol{k};\mathbf{R})$ and $T(\boldsymbol{k};\mathbf{A}_0)$ on the eigenvectors $\boldsymbol{e}(\boldsymbol{k}sa\lambda)$ and $\bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda)$. After a little algebra, it is finally found that, adopting $\beta\beta^* = \pm 1$,

$$\langle \bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda'), \boldsymbol{e}(\boldsymbol{k}sa\lambda)\rangle = \pm\langle \bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda'), \boldsymbol{e}(\boldsymbol{k}sa\lambda)\rangle \ , \tag{4.69}$$

where the upper plus sign is corresponding to a case in Eq.(4) and the lower minus sign to a case in Eq.(5), respectively. Clearly, if $\beta\beta^*$ satisfies Eq.(5), then

$$\langle \bar{\boldsymbol{e}}(\boldsymbol{k}sa\lambda'), \boldsymbol{e}(\boldsymbol{k}sa\lambda)\rangle = 0 \ .$$

This means that two eigenvectors are linearly independent and the minimum dimension of the subspace that is invariant under $\{T(\boldsymbol{k};\mathbf{R})\}$ is $2f_s$. In the case, the set of eigenvectors transforms according to an irreducible multiplier representation of the *second* type under the symmetry operations. The other case where $\beta\beta^*$ satisfies Eq.(4) cannot say anything about the linear independence of two eigenvectors. If they are linearly dependent, they can differ by at most an arbitrary phase factor and there occurs no additional degeneracy. On the other hand, if they are linearly independent, this can be referred to as an accidental degeneracy. This is a case under the irreducible multiplier representations called the *first* type.

The above discusses the criteria for establishing the type of representation by dealing with the relation between $\{\bar{\tau}^{(s)}(\boldsymbol{k};\mathbf{R})\}$ and $\{\tau^{(s)}(\boldsymbol{k};\mathbf{R})\}$. Before closing this section, it may be heuristic to introduce the analogous criterion for irreducible multiplier representations. Starting from the orthogonality theorem,

$$\begin{aligned}\sum_{\mathbf{R}} \bar{\tau}^{(s)}_{\mu\mu'}(\boldsymbol{k};\mathbf{R})^* \tau^{(s)}_{\nu\nu'}(\boldsymbol{k};\mathbf{R}) &= (h/f_s)\beta^{-1^*}_{\mu\nu}\beta^*_{\nu'\mu'} \ , \qquad \text{first or second type,}\\ &= 0 \ , \qquad\qquad\qquad\quad \text{third type,}\end{aligned}$$

where h is the order of the group G_0. Using the relation between $\bar{\tau}^{(s)}_{\mu\mu'}(\boldsymbol{k};\mathbf{R})$ and $\tau^{(s)}_{\mu\mu'}(\boldsymbol{k};\mathbf{A}_0^{-1}\mathbf{R}\mathbf{A}_0)$ and Eqs.(4) and (5), we arrive at another interesting criterion

$$\begin{aligned}\sum_{\mathbf{R}} \phi(\boldsymbol{k};\mathbf{A}_0\mathbf{R},\mathbf{A}_0\mathbf{R})\tau^{(s)}_{\lambda\lambda'}(\boldsymbol{k};\mathbf{A}_0\mathbf{R}\mathbf{A}_0\mathbf{R}) &= +(h/f_s)\delta_{\lambda\lambda'} \ , \qquad \text{first type,}\\ &= -(h/f_s)\delta_{\lambda\lambda'} \ , \qquad \text{second type,}\\ &= 0 \ , \qquad\qquad\qquad\quad \text{third type.}\end{aligned}$$

It is often customary to express the criterion in terms of the characters χ^s, i.e. $\chi^s(\boldsymbol{k};\mathbf{R}) = \mathrm{Tr}\tau^s(\boldsymbol{k};\mathbf{R})$,

$$\begin{aligned}\sum_{\mathbf{R}} \phi(\boldsymbol{k};\mathbf{A}_0\mathbf{R},\mathbf{A}_0\mathbf{R})\chi^{(s)}(\boldsymbol{k};\mathbf{A}_0\mathbf{R}\mathbf{A}_0\mathbf{R}) &= +h \ , \qquad \text{first type,}\\ &= -h \ , \qquad \text{second type,}\\ &= 0 \ , \qquad \text{third type.}\end{aligned}$$

The criterion can be also expressed as a sum over the elements $\mathbf{A}_0$ of the coset $\mathbf{S}\_G_0$ instead of $\mathbf{R}$ of the point group G_0

$$\begin{aligned}\sum_{\mathbf{A}_0} \phi(\boldsymbol{k};\mathbf{A}_0,\mathbf{A}_0)\chi^s(\boldsymbol{k};\mathbf{A}_0^2) &= +h\ , \qquad \text{first type,}\\ &= -h\ , \qquad \text{second type,}\\ &= 0\ , \qquad \text{third type.}\end{aligned}$$

4.3.4 Implementation in *DANSE*

Some results from group theory seem both useful and practical to implement in software. Others are not such good value:

- The point group of the bond can be used to fill out the force constant matrix, to check for inconsistencies of the force constants, and to help in generating them. This should, in principle, be facilitated by the matrix operations that are available as Pythonized routines in *Computational Crystallographic Toolbox* from the Lawrence Berkeley National Lab [?].
- The group of the wavevector can be assesed. If the Brillouin zone has been assessed previously by group theory, we can use the compatibility relationships to determine which dispersions are degenerate in energy at special points. This is important information when assessing the behavior of fuzzy data. Knowing that a convergence of curves is expected puts constraints on data, and allows a larger number of counts to be analyzed simultaneously, improving reliability.
- Time reversal symmetry might be useful to consider in magnetic systems. It seems plausible that one could assess the merging of magnon dispersions at special points, but we need to see how well this has been worked out by theorists. It would probably be too much to actually calculate character decompositions on the computer for individual cases.
- I think that actually reducing the dynamical matrix by symmetry is too much to do. The procedure may be elegant to implement, but the net gain does not seem large enough when compared to brute force diagonalization, which works rather well so far.

4.4 Spin Dynamics in Solids

4.4.1 Spin as a Source of Magnetism

Electrons are the source of magnetism in materials. Although the atomic nuclei also have spins and magnetic moments, nuclear moments are smaller by a factor of 1,000. The electron contributes to magnetism in two separate ways. One is from orbital motion – the classical origin of magnetism. The Hamiltonian for electron motion in an electromagnetic field is

$$\mathcal{H} = \frac{1}{2m}\left[\boldsymbol{p} + \frac{e}{c}\boldsymbol{A}(\boldsymbol{r})\right]^2 . \tag{4.70}$$

Here we assume that the magnetic field $\boldsymbol{H}$ is uniform and take the Coulomb gauge of $\nabla \cdot \boldsymbol{A} = 0$, consistent with $\boldsymbol{A}(\boldsymbol{r}) = \frac{1}{2}(\boldsymbol{H} \times \boldsymbol{r})$:

The first order term with respect to $\boldsymbol{A}(\boldsymbol{r})$ gives

$$\mathcal{H}^{(1)} = \frac{e}{2mc}\boldsymbol{H} \cdot (\boldsymbol{r} \times \boldsymbol{p}) \tag{4.71}$$

$$= \mu_B \boldsymbol{H} \cdot \boldsymbol{L} , \tag{4.72}$$

where $\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{p}$ (orbital angular momentum) and $\mu_B = e\hbar/2mc$ (Bohr magneton). Incidentally, the second order term in (4.70) is:

$$\mathcal{H}^{(2)} = \frac{e^2}{8mc^2}(\boldsymbol{r} \times \boldsymbol{H})^2 , \tag{4.73}$$

which is simply the magnetic induction, i.e. Lenz's law. The first order term (4.72) gives the paramagnetism leading to the magnetization parallel to the magnetic field, while the second order term (4.73) gives the diamagnetism leading to the magnetization antiparallel to the magnetic field. For atoms with closed shells (i.e. $\boldsymbol{L} = 0$) or superconducting metals, diamagnetism plays the dominant role. Except for such cases, however, paramagnetism is usually more important than diamagnetism.

There is another important contribution to magnetism from the electron besides its orbital motion. Electron spin angular momentum is the essential ingredient of magnetism in most systems having interesting magnetic properties. In metallic systems where the electron wavefunction is well approximated as a plane wave, it can be shown easily that $\boldsymbol{L} = 0$. The angular momentum also vanishes in insulators when the force the electron feels in a solid deviates much from the spherical symmetry the electron would feel in an isolated atom. In these cases, an orbital motion of the electron cannot contribute to the magnetism.

Electron spins can be localized or itinerant, and these two cases provide different types of magnetism. Usually, localized spins are found at ions in magnetic insulators and itinerant spins are on conduction electrons in metallic systems, although there are other important magnetic systems with both itinerant and localized spins that show unusual properties. The following sections describe the characteristic magnetic properties of localized spins, itinerant spins, localized spins immersed in itinerant spins, and strongly correlated electrons.

4.4.2 Localized Spins

Consider a system of localized spins, where the spins are arranged periodically on a crystal, and interact with each other through an exchange interaction. To understand effects of the spin-spin exchange interaction in the system, several

kinds of spin models are available. These textbook models have a long history. Depending on the spin dimensionality, there are the Heisenberg model, the XY model, and the Ising model or n-state Potts model. The commonly-used Heisenberg model treats three-dimensional spins, the XY model treats two-dimensional spins, and the Ising model or n-state Potts model treats one-dimensional spins. This section discusses magnetic properties of localized spins within the ferromagnetic Heisenberg model.

For an external magnetic field $h\hat{\boldsymbol{z}}$, we have the spin interaction Hamiltonian ($J > 0$) on the square lattice[6]

$$\mathcal{H} = -J \sum_{i \neq j} \boldsymbol{S}_i \cdot \boldsymbol{S}_j - g\mu_B \sum_i h S_i^z \ , \tag{4.74}$$

where the first term gives the spin-spin interaction intrinsic to the Heisenberg model. Now introduce the "mean field" (or "molecular field") approximation. Details of interactions between neighboring spins are ignored, and the Hamiltonian $\mathcal{H}$ is replaced by:

$$\mathcal{H} = -J \sum_{i \neq j} \boldsymbol{S}_i \cdot \langle \boldsymbol{S}_j \rangle - g\mu_B h \sum_i S_i^z \ . \tag{4.75}$$

The magnetization $\langle M_z \rangle = -g\mu_B \sum_i \langle S_{iz} \rangle = -g\mu_B N \langle S_z \rangle$ can then be obtained through $\langle S_z \rangle = \sum S_z \mathrm{e}^{-\beta\mathcal{H}}/Z$, where $Z = \sum \mathrm{e}^{-\beta\mathcal{H}}$. The evaluation of $\langle S_z \rangle$ is simple because $\mathcal{H}$ now includes only scalar quantities. Following the definition of the susceptibility $\chi = \lim_{h \to 0} \langle M_z \rangle / h$,

$$\chi = \frac{C}{T - T_{\mathrm{C}}} \ , \tag{4.76}$$

where T_{C} and C are called the Curie temperature and the Curie constant, respectively. The temperature-dependence of χ in (4.76) is characteristic of a localized spin system.

For understanding dynamical properties of a magnetic system, a starting point is its elementary excitations. For the case of three-dimensional spins in the Heisenberg model, the elementary excitation is known as a spin wave (or magnon). We consider the operator $\mathcal{O}_{\boldsymbol{q}}$ that creates a specific excitation in the system,

$$[\mathcal{H}, \mathcal{O}_{\boldsymbol{q}}] = \omega_{\boldsymbol{q}} \mathcal{O}_{\boldsymbol{q}} \ , \tag{4.77}$$

where $\omega_{\boldsymbol{q}}$ is the energy of the excitation. The operator $\mathcal{O}_{\boldsymbol{q}}$ raises the spin wave and provides the spin wave energy. Introducing the Fourier transformation of spin operators, we reexpress the Heisenberg Hamiltonian as:

$$\mathcal{H} = -\sum_{\boldsymbol{q}} J(\boldsymbol{q}) \boldsymbol{S}(\boldsymbol{q}) \cdot \boldsymbol{S}(-\boldsymbol{q}) \tag{4.78}$$

$$= -\sum_{\boldsymbol{q}} J(\boldsymbol{q}) \left[S_z(\boldsymbol{q}) S_z(-\boldsymbol{q}) + \frac{1}{2} \Big(S_+(\boldsymbol{q}) S_-(-\boldsymbol{q}) + S_-(\boldsymbol{q}) S_+(-\boldsymbol{q}) \Big) \right] \ , \tag{4.79}$$

[6] For geometrical reasons, on some lattices the spins cannot have long-range order.

where we introduce $\boldsymbol{S}(\boldsymbol{q}) = \sum_i \boldsymbol{S}_i e^{i\boldsymbol{q}\cdot\boldsymbol{R}_i}$ and $S_\pm(\boldsymbol{q}) = \sum_i S^i_\pm e^{i\boldsymbol{q}\cdot\boldsymbol{R}_i}$ ($S^i_\pm = S^i_x \pm i S^i_y$). Assuming the ground state is ferromagnetic, the low energy excited states should correspond to states with spins slightly deviated from perfect alignment. $S_-(\boldsymbol{q})$ is an operator that performs this role – it creates the spin wave excitation (magnon). The excitation energy of magnon is determined from (4.77):

$$[\mathcal{H}, S_-(\boldsymbol{q})] = \omega_{\boldsymbol{q}} S_-(\boldsymbol{q}) \ , \tag{4.80}$$

which is not, however, exactly solvable and requires an approximation. The most immediate approach is the mean field approximation for $\mathcal{H}$, which yields:

$$\omega_{\boldsymbol{q}} = 2\langle S_z\rangle[J(0) - J(\boldsymbol{q})] \tag{4.81}$$

$$\approx 2(J/N)\langle S_z\rangle a^2 \boldsymbol{q}^2 \ , \tag{4.82}$$

where for the three-dimensional cubic lattice:

$$J(\boldsymbol{q}) = 2(J/N)[\cos(q_x a) + \cos(q_y a) + \cos(q_z a)] \ . \tag{4.83}$$

At low temperatures, $\langle S_z\rangle \approx NS$ and the magnon energy is

$$\omega_{\boldsymbol{q}} = 2JSa^2\boldsymbol{q}^2. \tag{4.84}$$

The magnon energy spectrum is gapless, i.e., $\omega_{\boldsymbol{q}} \to 0$ as $\boldsymbol{q} \to 0$. The zero temperature magnetization (M_0) then decreases with increasing T as magnons are created. At small T:

$$M(T) = M_0 - g\mu_B \sum_{\boldsymbol{q}} n_{\boldsymbol{q}} \ , \tag{4.85}$$

where $n_{\boldsymbol{q}}$ is the Bose distribution function for magnons. A very famous result is the decrease of magnetization at low T is:

$$\frac{M(T)}{M_0 - 1} \propto T^{3/2} \ , \tag{4.86}$$

which is obtained by a simple integration of (4.85).

4.4.3 Itinerant Spins

Both itinerant electrons in metals and localized spins in insulators exhibit interesting magnetism. The magnetization of the system, $\langle M_z\rangle$, in an external magnetic field h is:

$$\langle M_z\rangle = -\mu_B \sum_{\boldsymbol{k}} \left[\langle c^\dagger_{\boldsymbol{k}\uparrow} c_{\boldsymbol{k}\uparrow}\rangle - \langle c^\dagger_{\boldsymbol{k}\downarrow} c_{\boldsymbol{k}\downarrow}\rangle\right] \ , \tag{4.87}$$

where $c^\dagger_{\boldsymbol{k}\uparrow}$ ($c_{\boldsymbol{k}\uparrow}$) is an electron creation (annihilation) operator with a momentum $\boldsymbol{k}$ and spin up. In the noninteracting electron gas, the magnetic susceptibility (called the Pauli susceptibility $\chi_{\rm P}$) is:

$$\chi_{\rm P} = 2\mu_B^2 N(\epsilon_F)[1 + \mathcal{O}(T/\epsilon_F)^2] \,, \tag{4.88}$$

where $N(\epsilon_F)$ is the electron density of states at the Fermi level and ϵ_F is the Fermi energy. $\chi_{\rm P}$ is almost constant at low temperatures and has a very weak temperature dependence.

Now consider a ferromagnetic metal. In ferromagnetic metals, the exchange interaction between electrons can raise the ferromagnetic order by overcoming the competition between kinetic energy and exchange energy. The susceptibility, χ_S, depends on the electron exchange interaction $\bar{V}$, called the Stoner susceptibility. It can be obtained by extension of $\chi_{\rm P}$ as:

$$\chi_S = \frac{\chi_{\rm P}}{1 - \frac{1}{2\mu_B^2}\bar{V}\chi_{\rm P}}. \tag{4.89}$$

Noting that $\chi_{\rm P} \approx 2\mu_B^2 N(\epsilon_F)$ at low temperatures, χ_S becomes:

$$\chi_S = 2\mu_B^2 \frac{N(\epsilon_F)}{1 - \bar{V}N(\epsilon_F)} \,. \tag{4.90}$$

From (4.90), $1 - \bar{V}N(\epsilon_F) \geq 0$ is a ferromagnetic instability condition at $T = 0$, called the "Stoner condition." For transition metals, the d-band is narrow and $N(\epsilon_F)$ can be large, so the Stoner condition is often satisfied (as for Fe, Co, Ni).

The temperature dependence of χ_S is very weak at room temperature, especially compared to the Curie–Weiss susceptibility. This can be easily understood because only electrons near the Fermi level can participate in thermally-driven magnetic excitations. Nevertheless, the temperature dependence of the Stoner susceptibility provides context for a longstanding puzzle of the ferromagnetism observed in Fe ($T_{\rm C}$ =1044 K), Ni (627 K), and Co (1388 K). Putting $\chi_{\rm P} = 2\mu_B^2 N(\epsilon_F)[1 - aT^2]$, and $\chi_S(T_{\rm C}) = \infty$ at the transition temperature $T_{\rm C}$, we obtain for $\chi_S(T)$:

$$\chi_S = \frac{2\mu_B^2/\bar{V}a}{T^2 - T_{\rm C}^2} = \frac{2\mu_B^2/\bar{V}a}{(T + T_{\rm C})(T - T_{\rm C})} \,, \tag{4.91}$$

and, in particular, near $T_{\rm C}$:

$$\chi_S \approx \frac{\mu_B^2/\bar{V}aT_{\rm C}}{T - T_{\rm C}} \,. \tag{4.92}$$

The Stoner susceptibility therefore shows a Curie–Weiss behavior near $T_{\rm C}$. However, it is well known that ferromagnetic metals like Fe, Ni, and Co show Curie–Weiss behavior over much wider temperature ranges. The observed Curie–Weiss behavior has been one of the most important sources of controversy over whether spins in ferromagnetic transition metals are itinerant or localized. Improving the Stoner spin susceptibility within a picture of itinerant spins remains an open problem.

Dynamical responses of spins in metals are also very different from those in localized spin systems. For conduction electrons, magnetic responses of their spins are directly related to details of the electron band structure. In the noninteracting electron gas, $\chi_0(\boldsymbol{q},\omega)$ governing the magnetic responses of the system is

$$\chi_0(\boldsymbol{q},\omega) = 2\mu_B^2 \sum_{\boldsymbol{k}} \frac{n_{\boldsymbol{k}} - n_{\boldsymbol{k}+\boldsymbol{q}}}{\varepsilon_{\boldsymbol{k}+\boldsymbol{q}} - \varepsilon_{\boldsymbol{k}} - \omega} \ , \tag{4.93}$$

where $n_{\boldsymbol{k}}$ is the Fermi distribution function and $\varepsilon_{\boldsymbol{k}}$ is the band energy at $\boldsymbol{k}$. We note that the Pauli susceptibility $\chi_{\rm P}$ corresponds to $\chi_0(\boldsymbol{q},\omega)$ at $\boldsymbol{q} = 0$ and $\omega = 0$. In the same way as was found for non-interacting systems, for interacting electron systems in the paramagnetic state, $\chi(\boldsymbol{q},\omega)$ is:

$$\chi(\boldsymbol{q},\omega) = 2\mu_B^2 \frac{F(\boldsymbol{q},\omega)}{1 - \bar{V}(\boldsymbol{q})F(\boldsymbol{q},\omega)} \ . \tag{4.94}$$

The susceptibility $\chi(\boldsymbol{q},\omega)$ is directly related to the neutron scattering cross section as:

$$S(\boldsymbol{q},\omega) = \frac{2}{1 - e^{-\beta\omega}} \ {\rm Im}\chi(\boldsymbol{q},\omega) \ . \tag{4.95}$$

It is curious that for temperatures well above $T_{\rm C}$, neutrons are still scattered magnetically by metallic spins. A further consideration makes this even more curious. In thermal neutron scattering experiments, the neutron velocity v_n is typically smaller than the electron velocity $v_{\rm F}$ by a factor of 10^{-3}. This means that a neutron interacts with approximately 10^3 electrons as it moves across the magnetic moment of an atom. The average spin of 10^3 electrons will be zero at $T > T_{\rm C}$. The answer to this puzzle is provided by further analysis of (4.94). Neutrons are scattered by collective motions of electron spins (called spin-fluctuations)[7], not by individual electron spins. More precisely, the paramagnetic scattering of neutrons is caused by fluctuations of the magnetic force on neutrons exerted by spins of itinerant electrons. On the other hand, in the ferromagnetic state ($T < T_{\rm C}$), we need to consider effects from the spin polarization from the splitting of the electron band states into up and down spin states. In this case of itinerant band electrons, the elementary excitations are like the spin wave and Stoner excitation in the ferromagnetic states, obtained from the condition:

$$1 = \bar{V}(\boldsymbol{q}) \sum_{\boldsymbol{k}} \frac{n_{\boldsymbol{k}\uparrow} - n_{\boldsymbol{k}+\boldsymbol{q}\downarrow}}{\varepsilon_{\boldsymbol{k}+\boldsymbol{q}\downarrow} - \varepsilon_{\boldsymbol{k}\uparrow} - \omega} \ . \tag{4.96}$$

Even though the spin wave is usually discussed in the context of systems with localized spins, it is interesting that spin waves are also excited in the

[7] The original discussion of spin-fluctuations is provided by T. Izuyama, D.J. Kim, and R. Kubo, J. Phys. Soc. Jpn. **18**, 1025 (1963).

ferromagnetic ordered state of itinerant spins. Like the insulating case, we find $M(T)/M_0 - 1 \propto T^{3/2}$, as in (4.86) for ferromagnetic metals. This experimental finding strongly implies the existence of spin waves in metallic magnetism.

Returning to the problem of the ferromagnetic transition metals Fe, Co, and Ni, we ask again, "*Are these spins itinerant?*" or "*Can Stoner theory properly explain their magnetism?*" The answers are still extremely unclear. There have been attempts to incorporate the electron-phonon interaction within the itinerant picture. These have improved the Stoner suceptibility somewhat and have led to a Curie–Weiss-like behavior with a smaller T_{C} than obtained from band theory, more consistent with experiment. Nevertheless, many features consistent with local moments are observed experimentally. The observed change in the specific heat of Fe at T_{C}, for example, is associated with an entropy of order $k_{\mathrm{B}} \ln 2$, as expected for localized electrons. Spin-resolved photoemission experiments show that the exchange splitting does not vanish at $T > T_{\mathrm{C}}$, strong experimental indication of local moments in ferromagnetic transition metals. These controversies demonstrate that a better theory is needed to properly account for electron correlations.[8]

4.4.4 Localized Spins Embedded in Itinerant Spins

A mixed magnetic system with localized spins (or magnetic impurities) embedded in itinerant spins provides many exotic magnetic properties that differ from the properties of from systems having localized or itinerant spins, but showing features of both. Consider two cases:

- Mn atoms in Cu metal, where Mn $(3d)^5$ electrons play the role of a magnetic impurity of $S = 5/2$, and Cu gives a sea of conduction electrons,
- An f-electron system, for example Ce which has a localized f-level and a $5d$ conduction band.

The governing Hamiltonian[9] for both cases can be written as

$$\mathcal{H} = t \sum_{ij} c^{\dagger}_{i\sigma} c_{j\sigma} + J \sum_{il} \boldsymbol{S}_l \cdot \sigma_i, \tag{4.97}$$

where σ is the spin of conduction electrons and $\boldsymbol{S}$ is the localized spin. Although both cases can be explained by $\mathcal{H}$, they correspond to different limits of $\mathcal{H}$.

First, for Mn impurities in Cu, we set $t \gg J$, so the Hamiltonian expresses an effective indirect interaction of localized spins mediated by conduction

[8] A compromise model that includes both itinerant and localized features with "correlated delocalized electrons" is available, where the magnetic moments at different sites fluctuate in magnitude and direction at finite temperatures.

[9] This is called the "Kondo lattice model" (KLM). Recently, the ferromagnetic KLM ($J < 0$) has been frequently adopted to describe the colossal magnetoresistance in manganese oxides, where J is the Hund coupling.

electrons. This interaction is called the RKKY (Ruderman-Kittel-Kasuya-Yosida) interaction:

$$\mathcal{H}_{\mathrm{RKKY}} = J(k_{\mathrm{F}}|\boldsymbol{r}_i - \boldsymbol{r}_j|)\ \boldsymbol{S}_i \cdot \boldsymbol{S}_j\ , \tag{4.98}$$

and $J(x)$ has the functional form:

$$J(x) \propto -\frac{\cos x}{x^3}\ +\ \frac{\sin x}{x^4}\ . \tag{4.99}$$

The RKKY interaction is also observed in magnetic multilayer systems composed of layers of Fe/Cu, for example, where Cu plays the role of a nonmagnetic metallic spacer. Here it is found that the exchange coupling constant oscillates and has both positive and negative values depending on the thickness of the Cu layer, consistent with the k_{F} for the conduction electron sea in Cu.

The second case is the opposite limit of $t \ll J$, and approaches the Kondo model for heavy fermion systems. In the past decades, very unusual low temperature behaviors have been observed in rare earth metals (e.g., Ce) and actinides (e.g., U). The linear specific heat at low temperatures shows an unusually high coefficient, γ, of order 1 J/mol K^2, in contrast to a 1 mJ/mol K^2 typical of ordinary metals. High values of γ are typical of "heavy fermion" systems.[10] Furthermore, these heavy fermion systems have an electrical resistivity of $\rho_0 + AT^2$ at low T with huge values of A of order $10\mu\Omega\mathrm{cmK}^{-2}$, whereas A is of order $10^{-5}\mu\Omega\mathrm{cmK}^{-2}$ or less for ordinary metals. Heavy fermion systems exhibit another basic and universal magnetic property. Below a characteristic temperature T^*,[11] heavy fermion systems show Fermi liquid behavior with a huge effective mass, and then constant (but very high) Pauli susceptibility. Above T^*, they show a Curie–Weiss susceptibility that originates from the localized f electrons. At T^*, the quasi-particles are screened by conduction electrons and a singlet (nonmagnetic) state is formed, owing to strong electron correlations. Although heavy fermion systems show differences in their properties, the disappearance of a magnetic moment is a common feature.

Another unusual property originating with magnetic scattering is the Kondo effect. The Kondo effect is a logarithmic increase of the electrical resistivity of the Kondo system when the temperature is reduced. It is caused by spin flip scatterings of conduction electrons at magnetic impurities.

[10] Values of γ for UPt_3, $CeAl_3$, $CeCu_2Si_2$, UBe_{13} are 0.45 J/mol K^2, 1.6 J/mol K^2, 1.1 J/mol K^2, and 1.1 J/mol K^2, respectively.

[11] T^* is not necessarily same as T_K, which is the usual Kondo temperature for a single Kondo ion, and in most cases we see $T^* < T_K$ owing to the condensation of Kondo ions.

4.4.5 Strongly Correlated Electrons

Since the discovery of high T_c superconductors,[12] enormous theoretical effort has been made to understand the two-dimensional Hubbard model, which was originally introduced to understand the metal-insulator transition in transition metals. The Hubbard Hamiltonian is:

$$\mathcal{H} = t \sum_{ij} c^\dagger_{i\sigma} c_{j\sigma} + U \sum_i n_{i\uparrow} n_{i\downarrow} \ . \tag{4.100}$$

In high T_c superconductors, $c^\dagger_{i\sigma}(c_{i\sigma})$ is the creation (annihilation) operator of the electron (or the hole) at the highest antibonding orbital (predominantly $d_{x^2-y^2}$) of Cu^{2+} positioned at i in CuO_2 plane. Approproximately, $t \sim 0.1$ eV and $U \sim 1$ eV. It is immediately clear that the energy scale of the Coulomb correlation energy, U, is too high compared to the interesting energy scales, which are around 10 meV for critical temperature of 100 K. The low energy excitation relevant to superconductivity is therefore believed to originate with spins, not with charge, and effort has been made to derive an effective spin Hamiltonian that depends on the hole concentration (doping).

In the undoped case (half-filling), there is one electron at each site and the hopping into the nearest neighbor costs the energy U. In the limit of $t \ll U$, the electrons look localized at each site because of the high energy barrier U. The Hubbard model can be transformed into the Heisenberg spin Hamiltonian:

$$\mathcal{H} = J \sum_{ij} \boldsymbol{S}_i \cdot \boldsymbol{S}_j \ , \tag{4.101}$$

$$J = 4t^2/U \ . \tag{4.102}$$

In this limit of $t \ll U$, the system acts as an antiferromagnetic insulator.

By introducing a small fraction, x, of holes, the superconducting phase appears. Typical fractions for $Bi_2Sr_2CaCu_2O_{8+x}$ (Bi2212) are $x \sim 0.1$ (this may be underdoped, but overdoping occurs for $x \sim 0.2$). The effective Hamiltonian for a small number of holes has also been reduced from the Hubbard model, and is called the $t-J$ model:

$$\mathcal{H}_{t-J} = t \sum_{ij} a^\dagger_{i\sigma} a_{j\sigma} + J \sum_{ij} \boldsymbol{S}_i \cdot \boldsymbol{S}_j \ , \tag{4.103}$$

where $a^\dagger_{i\sigma} = c^\dagger_{i\sigma}(1 - n_{i-\sigma})$ and $a_{i\sigma} = c_{i\sigma}(1 - n_{i-\sigma})$. In the limit of half-filling, $\mathcal{H}_{t-J}$ is the Hamiltonian of a Heisenberg antiferromagnet. Research interests are in small deviations from half-filling, where holes move in the antiferromagnetic lattice and counteract antiferromagnetic long-range order. This leads to

[12] Examples of some high T_c materials are $La_{2-x}Sr_xCuO_4$ ($T_c \sim 40$ K), $YBa_2Cu_3O_7$ ($T_c = 92$ K), $Bi_2Sr_2Ca_2Cu_3O_{10}$ ($T_c = 110$ K), and $Tl_2Sr_2Ca_2Cu_3O_{10}$ ($T_c \sim 125$ K).

antiferromagnetic spin-fluctuations peaked at a momentum near $\boldsymbol{Q} = (\pi, \pi)$. In the last decades, spin fluctuations have been found to play fundamental roles in high T_c superconductors. In the antiferromagnetic Fermi liquid model, many key properties of superconducting cuprates have been understood by a strong interaction between quasi-particles and spin-fluctuations. For example, the spin-fluctuation model has been successful in explaining transport properties such as electrical resistivity ($\rho \propto T$), the magnitude of T_c, results from nuclear magnetic resonance (NMR) experiments, angle-resolved photoemission spectroscopy (ARPES) experiments, and other experimental results. The detailed phenomenological propagator for spin-fluctuations is:

$$\chi_{\mathrm{sf}}(\boldsymbol{q}, \omega) = \frac{\chi_Q}{1 + \xi^2(\boldsymbol{q} - \boldsymbol{Q})^2 - \mathrm{i}\omega/\omega_{\mathrm{sf}}} \,, \tag{4.104}$$

which has proved applicable for interpreting NMR measurements and neutron scattering experiments. Here ω_{sf} is the spin-fluctuation energy and ξ is the antiferromagnetic correlation length. Through the value of ξ, the spin-fluctuation depends on the hole concentration. In the limit of $\xi \to \infty$, the system in the pure magnon region, i.e. the undoped antiferromagnetic insulator with long-range order. An interaction between quasi-particles and spin-fluctuations is especially strong in underdoped materials. It has been shown recently that strong anisotropies in ARPES data for underdoped Bi2212 originate with a strong coupling between quasi-particles and spin-fluctuations. Incidentally, for other superconducting materials, i.e., $La_{2-x}Sr_xCuO_4$ and $YBa_2Cu_3O_{7-x}$, data from inelastic neutron scattering and measurements of the spin-lattice relaxation rate by NMR indicate that the spin-fluctuations induce an opening of the gap in the spin excitation[13] in the CuO_2 plane, with an energy comparable to the BCS gap. The apparent gap develops well above T_c.

4.5 N/A Simulations of Lattice Dynamics

4.6 Simulations of Spin Dynamics

4.6.1 Monte Carlo Method

Section 4.4 introduced different spin systems and their excitations. Several approaches are available for calculating the states and state evolution of these spin systems. Perhaps the most conventional approach, although not a simulation *per se*, is the diagrammatic expansion of the Green's function. This formal approach is not practical for many complex systems, however. Instead, one may use the local density approximation (LDA), or LDA+U to

[13] The spin gap corresponds to the pseudogap of a small energy scale.

study band magnetism, or the small-size exact diagonalization for more localized spin systems. However, for more complete descriptions of the dynamics of spins, the most suitable and reliable method is often a Monte Carlo simulation. There are many variants of Monte Carlo simulations developed and optimized for specific problems. This section explains the Monte Carlo approach for calculating the states and dynamics of classical lattice spin systems. Monte Carlo simulations of classical systems are easier to understand than quantum Monte Carlo simulations, which are mentioned at the end of this section.

A Monte Carlo simulation is a Markovian process. Such processes will reach a steady state of a system that is independent of the initial configuration. Unfortunately, especially at low temperatures, this final state of equilibrium may require a very long time to achieve. If one knows the ground state *a priori*, at low temperatures it may be appropriate to start with the system in a ground state configuration, and allow temperature to produce disorder in this configuration. The ground state is often unknown, however, and this is typical of more complicated systems. It may be possible to start with several candidate ground state configurations, and identify the true ground state structure as the one that does not evolve with time.

An alternative approach is sometimes called "simulated annealing," where the simulation begins with the system in a fully random state characteristic of infinite temperature. Equilibrium at lower temperatures is achieved by gradually reducing the temperature, and allowing the system to relax under the spin-spin interactions. Some delicacy is required for balancing the slowness of cooling with the need to minimize the time of the simulation.

Finally, systems that undergo spontaneous symmetry breaking may pose special problems. Starting from the random spin configuration at $T = \infty$ in the isotropic ferromagnet, one may find that the ordering direction would rotate without any preferred direction even below T_{C}. In such a case, therefore, one may include an infinitesimal anisotropy in the Hamiltonian to induce the symmetry breaking, or one may start with a symmetry-broken ground state configuration[14] at $T = 0$ and raising the temperature. Without an appled biad, the formation of local domains, each with its own direction of spin alignment, is a common feature. Elimination of the domain boundaries is favorable energetically, but may take a long time in practice.

4.6.2 Spin Updates in Monte Carlo Simulations

Metropolis Algorithm. In the Metropolis algorithm, a new configuration is generated from an existing one by using a transition probability that depends on the energy difference between the two configurations. The state of thermodynamic equilibrium satisfies the detailed balance between two states n and m,

[14] For example, by initially aligning all the spins along a certain direction.

$$P_n W_{n\to m} = P_m W_{m\to n} \,, \tag{4.105}$$

where P_n is the probability of the system being in the state n and $W_{n\to m}$ is the transition rate from $n \to m$. The Metropolis algorithm selects the simplest choice of the transition rate that is consistent with detailed balance

$$W_{n\to m} = \begin{cases} \exp(-\Delta E/k_B T) & \Delta E > 0 \\ 1 & \Delta E < 0 \,, \end{cases} \tag{4.106}$$

where $\Delta E = E_n - E_m$. The Metropolis algorithm updates one spin at a time in a given configuration at temperature T:

1. select the spin at site i,
2. evaluate ΔE by updating the spin at i,
3. generate a random number η, where $0 < \eta < 1$,
4. accept the updated configuration if $\eta < \mathrm{e}^{-\Delta E/k_\mathrm{B}T}$, or reject otherwise,
5. return to step 1 for a different spin at site $i + 1$.

For the model of continuous spins of $|\mathbf{S}_i| = 1$, oriented in three dimensions, one may update the spin at site i by generating two random numbers η_1 and η_2, such that $\zeta^2 = \zeta_1^2 + \zeta_2^2 < 1$ such that $\zeta_1 = 1 - \eta_1$ and $\zeta_2 = 1 - \eta_2$ (where $0 < \eta_1, \eta_2 < 1$)

$$S_x = 2\zeta_1 \sqrt{1 - \zeta^2} \,, \tag{4.107}$$

$$S_y = 2\zeta_2 \sqrt{1 - \zeta^2} \,, \tag{4.108}$$

$$S_z = 1 - 2\zeta^2 \,. \tag{4.109}$$

The Metropolis algorithm ensures that the steady state of the system is the actual state of thermodynamic equilibrium. This is proved by assuming that the system is in thermodynamic equilibrium, and then showing that the Metropolis algorithm has the transition rates needed to keep it there. In thermodynamic equilibrium, the probability P_n is the Boltzmann factor normalized by the partition function, $\mathrm{e}^{-E_n/k_\mathrm{B}T}/Z$, a central result of statistical physics. Substituting into (4.105)

$$\mathrm{e}^{-E_n/k_\mathrm{B}T}\, W_{n\to m} = \mathrm{e}^{-E_m/k_\mathrm{B}T}\, W_{m\to n} \,, \tag{4.110}$$

$$\mathrm{e}^{-(E_n - E_m)/k_\mathrm{B}T} = \frac{W_{m\to n}}{W_{n\to m}} \tag{4.111}$$

Note that $W_{n\to m} = 1$ (since the transition $n \to m$ is downhill energetically, it will always occur according to (4.106)). Equation (4.111) becomes:

$$W_{m\to n} = \mathrm{e}^{-(\Delta E)/k_\mathrm{B}T} \,, \tag{4.112}$$

which is the rate used by the Metropolis algorithm in step 4 or (4.106).

All Markovian processes converge to a steady state[15], and the Metropolis algorithm assures us that thermodynamic equilibrium will be achieved between all pairs of spins in the system. It does not ensure that equilibration will occur in a reasonable time, however.

[15] Or a cyclic state in anomalous cases.

4.6.3 Low Temperatures

At low temperatures, a more sophisticated update algorithm is necessary. Most random updates cause a large energy exponent $\Delta E/k_{\rm B}T$, so the Metropolis algorithm will reject most changes to the spin configuration. The equilibration procedure can then become far too slow to be practical. To speed things up, the randomly-selected changes in spin can be made smaller, for example. This can be done by attenuating the random changes of spin by a factor δ ($0 \le \delta \le 1$), i.e. $\Delta\mathbf{S}$ should be replaced by $\delta\Delta\mathbf{S}$. The factor δ can be adjusted so that the acceptance rate is around 50% on the average.

An actual implementation could be based on a parameterized temperature-dependent solid angle, $\Omega(T)$, of spin i with respect to its original orientation $\mathbf{S}_i$. This is begun by defining $\alpha = \tan^{-1} S_i^y/S_i^x$ and $\beta = \cos^{-1} S_i^z$, from which we define the rotation matrices $R_z(\alpha)$ and $R_y(\beta)$:

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} , \tag{4.113}$$

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix} . \tag{4.114}$$

The updated orientation of spin i, $\mathbf{S}_i'$ is

$$\mathbf{S}_i' = R_z^{-1}(\alpha)\, R_y^{-1}(\beta)\, \mathbf{S}_i'' , \tag{4.115}$$

where

$$\mathbf{S}_i'' = (\sin\theta\cos\phi,\ \sin\theta\sin\phi,\ \cos\theta) , \tag{4.116}$$

$$\theta = \Omega(T)\ \eta_1 , \tag{4.117}$$

$$\phi = 2\pi\ \eta_2 . \tag{4.118}$$

A useful form[16] of $\Omega(T)$ may be

$$\Omega(T) = \pi \tanh(\xi T) . \tag{4.119}$$

With this form of $\Omega(T)$, we can adjust ξ so that the acceptance rate is around 50%. Note the two extreme limits: $\Omega(T) \to 0$ as $T \to 0$ and $\Omega(T) \to \pi$ as $T \to \infty$.

Overrelaxation Technique. The overrelaxation technique is used in combination with the Metropolis algorithm for improving the rate of relaxation to the equilibrium configuration, especially at low temperatures. Let us assume that we are treating a system of isotropic continuous spins in the Heisenberg model, with the Hamiltonian $\mathcal{H} = J\sum_{ij} \mathbf{S}_i \cdot \mathbf{S}_j$. In the overrelaxation method, a spin is precessed about the full interaction field, which it can do

[16] The form of $\Omega(T)$ given here is just the simplest example. One may introduce a more complicated form to give a more desirable performance.

without any change of energy. When the angle of precession, θ, is as large as π, this alteration of the spin structure can promote quicker changes in the orientation of other spins during subsequent Monte Carlo steps.

At the site i, the full interaction field for $\mathbf{S}_i$ is $\mathbf{S_{nn}} \equiv J\sum_{j\neq i}\mathbf{S}_j$. The overrelaxed spin, $\mathbf{S}_i'$, is evaluated by successive rotations using rotations by Euler angles. One needs two angles of α and β to define the direction of $\mathbf{S_{nn}}$:

$$\alpha = \tan^{-1} S^y_{\mathbf{nn}}/S^x_{\mathbf{nn}}\,, \tag{4.120}$$

$$\beta = \cos^{-1} S^z_{\mathbf{nn}}/|\mathbf{S_{nn}}|\,, \tag{4.121}$$

$$\mathbf{S}_i' = R_z^{-1}(\alpha)\; R_y^{-1}(\beta)\; R_z(\pi)\; R_y(\beta)\; R_z(\alpha)\; \mathbf{S}_i. \tag{4.122}$$

In an actual simulation, a single "hybrid" Monte Carlo update of the spin configuration could include a Metropolis sweep and two overrelaxation sweeps.

Equilibration (Thermalization). Repeating the hybrid Monte Carlo steps, spins are updated by sweeping the whole spin lattice, eventually reaching the equilibrium configuration at a given T. It is an important problem to minimize the number of Monte Carlo steps needed to obtain the fully relaxed equilibrium state. The optimization differs, however, depending on the specific problem or the simulation conditions such as temperature and lattice size, for example. It is essential to perform calculations of simple quantities like susceptibilities or magnetization, and test if the system is in the equilibrium state. In more advanced validations, one may check the autocorrelation time. Substantial gains in efficiency are often found for hybrid Monte Carlo simulations. For the square lattice Heisenberg model, the nonhybrid Monte Carlo method typically requires at least $\mathcal{O}(10^4)$ steps at the temperature range of $\mathcal{O}(0.1J)$, whereas the hybrid method requires $\mathcal{O}(10^3)$ steps.

4.6.4 Time Evolution of Spins

For a system of spins in the Heisenberg model, the equation of motion for the spin degrees of freedom is

$$\frac{\partial \mathbf{S}_i}{\partial \tau} = -\mathbf{S}_i \times \frac{\partial \mathcal{H}}{\partial \mathbf{S}_i} = -J\mathbf{S}_i \times \sum_{j\neq i}\mathbf{S}_j\,. \tag{4.123}$$

The N spins are coupled to each other, and the time-evolution of the spins is obtained by solving the coupled differential equations. There are many integration algorithms to solve the differential equations. Here we introduce the Suzuki-Trotter decomposition, which is especially suitable for the lattice problem. Taking the example of the simple square lattice with the nearest-neighbor spin coupling, one can divide the lattice into two sublattices $\mathcal{A}$ and $\mathcal{B}$ by checkerboard decomposition. Employing the Suzuki-Trotter decomposition up to $\mathcal{O}(\mathrm{d}\tau^5)$,

$$\{\mathbf{S}_i(\tau+\mathrm{d}\tau)\} = \mathrm{e}^{(\mathbf{A}+\mathbf{B})\mathrm{d}\tau}\{\mathbf{S}_i(\tau)\}\,, \tag{4.124}$$

and $\mathrm{e}^{(\mathbf{A}+\mathbf{B})\mathrm{d}\tau}$ is decomposed using

$$\mathrm{e}^{(\mathbf{A}+\mathbf{B})\mathrm{d}\tau} = \prod_{i=1}^{5} \mathrm{e}^{p_i \mathbf{A}\mathrm{d}\tau/2}\ \mathrm{e}^{p_i \mathbf{B}\tau}\ \mathrm{e}^{p_i \mathbf{A}\mathrm{d}\tau/2} + \mathcal{O}(\mathrm{d}\tau^5)\ , \tag{4.125}$$

with

$$p_1 = p_2 = p_4 = p_5 = p = 1/(4 - 4^{1/3}) \tag{4.126}$$

and

$$p_3 = 1 - 4p\ . \tag{4.127}$$

Here $\mathbf{A}$ and $\mathbf{B}$ are the rotation generators of the sublattices $\mathcal{A}$ and $\mathcal{B}$, with fixed $\{\mathbf{S}_{i\mathcal{B}}\}$ and $\{\mathbf{S}_{i\mathcal{A}}\}$, respectively.

The following algorithmic explanation may be more clear.

$\mathrm{e}^{\mathbf{A}\delta\tau}\{\mathbf{S}_i\}$: time evolution of sublattice $\mathcal{A}$

for $i = 1$ to N
when $i \in \mathcal{A}$
$\mathbf{S_{nn}} = J \sum_{j \neq i (j \in \mathcal{B})} \mathbf{S}_j$
$\mathrm{e}^{\mathbf{A}\delta\tau}\{\mathbf{S}_i\} = \left[(\mathbf{S_{nn}} \cdot \mathbf{S}_i)\mathbf{S_{nn}}/|\mathbf{S_{nn}}|^2\right](1 - \cos(|\mathbf{S_{nn}}|\delta\tau))$
$+\ \mathbf{S}_i \cos(|\mathbf{S_{nn}}|\delta\tau) + [(\mathbf{S_{nn}} \times \mathbf{S}_i)/|\mathbf{S_{nn}}|] \sin(|\mathbf{S_{nn}}|\delta\tau)$

$\mathrm{e}^{\mathbf{B}\delta\tau}\{\mathbf{S}_i\}$: time evolution of sublattice $\mathcal{B}$

for $i = 1$ to N
when $i \in \mathcal{B}$
$\mathbf{S_{nn}} = J \sum_{j \neq i (j \in \mathcal{A})} \mathbf{S}_j$
$\mathrm{e}^{\mathbf{B}\delta\tau}\{\mathbf{S}_i\} = \left[(\mathbf{S_{nn}} \cdot \mathbf{S}_i)\mathbf{S_{nn}}/|\mathbf{S_{nn}}|^2\right](1 - \cos(|\mathbf{S_{nn}}|\delta\tau))$
$+\ \mathbf{S}_i \cos(|\mathbf{S_{nn}}|\delta\tau) + [(\mathbf{S_{nn}} \times \mathbf{S}_i)/|\mathbf{S_{nn}}|] \sin(|\mathbf{S_{nn}}|\delta\tau)$

In this algorithm, the time evolution of spins is performed by combinations of $\mathrm{e}^{-\mathbf{A}\delta\tau}$ and $\mathrm{e}^{-\mathbf{B}\delta\tau}$, just as given by the Suzuki-Trotter decomposition.

4.6.5 Observables

Inelastic neutron scattering, especially experiments on single crystal samples, can probe directly the spin-spin correlation function $S(\mathbf{q}, \omega)$ that describes the dynamics of spins

$$S(\mathbf{q}, \omega) = \frac{1}{2\pi} \frac{1}{N^2} \sum_{ij} \mathrm{e}^{\mathrm{i}\mathbf{q}\cdot(\mathbf{r}_i - \mathbf{r}_j)} \int_0^{\tau_{\max}} \mathrm{e}^{\mathrm{i}\omega\tau} \langle \mathbf{S}_i(\tau) \cdot \mathbf{S}_j(0) \rangle\ \mathrm{d}\tau\ . \tag{4.128}$$

This $S(\mathbf{q}, \omega)$ can be calculated by Monte Carlo simulation. $S(\mathbf{q}, \omega)$ is obtained by a simple Fourier transformation of the time-dependent correlation function

$$C_{ij}(\tau) = \langle \mathbf{S}_i(\tau) \cdot \mathbf{S}_j(0) \rangle\ , \tag{4.129}$$

which is readily evaluated in a Monte Carlo simulation. Pole structures of $S(\mathbf{q},\omega)$, $\omega(\mathbf{q})$, can provide information on magnetic excitations and relaxations, which are fundamental to understanding the spin systems. Incidentally, the quasi-elastic response $S(\mathbf{q},0)$ tells us the kinds of magnetic fluctuations (correlations) that dominate at a given T. For instance, if $S(\mathbf{Q},0) \gg S(\mathbf{0},0)$ with say $\mathbf{Q} = (\pi,\pi)$, we conclude that antiferromagnetic exchange interactions are dominant over ferromagnetic ones at the given T.

For powder or polycrystal samples, the measured spectra are in the angle-integrated form

$$S(|\mathbf{q}|,\omega) = \frac{2}{N^2}\sum_{ij}\frac{\sin(|\mathbf{q}||\mathbf{r}_i-\mathbf{r}_j|)}{|\mathbf{q}||\mathbf{r}_i-\mathbf{r}_j|}\int_0^{\tau_{\max}} \mathrm{e}^{\mathrm{i}\omega\tau}\langle \mathbf{S}_i(\tau)\cdot\mathbf{S}_j(0)\rangle \,\mathrm{d}\tau \ . \quad (4.130)$$

In some experiments, or in some stages of data analysis, scattered neutrons with a rather wide distribution of momenta are collected into a single energy bin. For such data, the resulting spectra $S(\omega)$ probes the local responses of spins as

$$S(\omega) = \int S(\mathbf{q},\omega)\,\mathrm{d}\mathbf{q} = \frac{1}{2\pi}\frac{1}{N}\sum_i \int_0^{\tau_{\max}} \mathrm{e}^{\mathrm{i}\omega\tau}\langle \mathbf{S}_i(\tau)\cdot\mathbf{S}_i(0)\rangle \,\mathrm{d}\tau \ . \quad (4.131)$$

The quasi-elastic local response, $S(0)$, is also an interesting quantity. It is directly related to the spin-lattice relaxation rate T_1 of a local probe as in nuclear magnetic resonance (NMR) or Mössbauer spectrometry.

4.6.6 Comments on Quantum Monte Carlo Simulations

An essential difference between classical and quantum Monte Carlo simulations is in how the spin configuration is updated, that is, how the Metropolis algorithm is implemented. The classical Monte Carlo method is easier in that the equilibrium probability has the proportionality

$$P(\{\mathbf{S}_i\}) \propto \mathrm{e}^{-\beta\mathcal{H}(\{\mathbf{S}_i\})} \quad (4.132)$$

for the particular configuration $\{\mathbf{S}_i\}$, that is: $\mathcal{H}(\{\mathbf{S}_i\}) = E(\{\mathbf{S}_i\})$. On the other hand, for the quantum Heisenberg model, for which

$$\mathcal{H} = \mathcal{H}_0 + \mathcal{V} \quad (4.133)$$

with

$$\mathcal{H}_0 = J\sum_{ij} S_i^z\ S_j^z \ , \quad (4.134)$$

$$\mathcal{V} = J\sum_{ij}(S_i^x\ S_j^x + S_i^y\ S_j^y) \ , \quad (4.135)$$

$$\big[\mathcal{H}_0,\mathcal{V}\big] \neq 0 \ , \quad (4.136)$$

we apply the Trotter formula[17]

$$\mathrm{e}^{-\beta\mathcal{H}} \approx [\mathrm{e}^{-\beta\mathcal{H}_0/m}\mathrm{e}^{-\beta\mathcal{V}/m}]^m \ , \tag{4.137}$$

$$Z = \sum_{\alpha}\langle\alpha|\mathrm{e}^{-\beta\mathcal{H}}|\alpha\rangle \ , \tag{4.138}$$

$$Z = \sum_{\{\alpha_k\}}\prod_{k=1}^{m}\langle\alpha_k|\mathrm{e}^{-\beta\mathcal{H}_0/m}|\alpha'_k\rangle\langle\alpha'_k|\mathrm{e}^{-\beta\mathcal{V}/m}|\alpha_{k+1}\rangle \ , \tag{4.139}$$

where $|\{\alpha\}\rangle$ can be the eigenstate of $\mathcal{H}_0$ and $|\alpha_1\rangle = |\alpha_{m+1}\rangle$. Then the matrix element $\langle\alpha|\mathrm{e}^{-\beta\mathcal{V}/m}|\alpha'\rangle$ is evaluated classically, leading to $\mathrm{e}^{-\beta\mathcal{V}(\alpha,\alpha')/m}$. A d-dimensional quantum spin problem therefore always corresponds to an effective $(d+1)$-dimensional problem.[18]

Besides the quantum (localized) spin model, another important problem is that of electrons with itineracy. One may introduce the Hubbard model as the simplest example,

$$\mathcal{H} = t\sum_{ij}\sum_{\sigma}c^{\dagger}_{i\sigma}c_{j\sigma} + U\sum_{i}n_{i\uparrow}n_{i\downarrow} \ . \tag{4.140}$$

For this problem, we need to apply another kind of quantum Monte Carlo method that incorporates the path integral formalism. In this method, the itinerant degrees of freedom of electrons are completely integrated out by the path integral formalism[19] and the remaining problem is then cast as an Ising spin problem in $(d+1)$-dimensions.

Further Reading

The contents of the following are described in the Bibliography.

Phillip R. Bevington: *Data Reduction and Error Analysis for the Physical Sciences* (McGraw-Hill, New York, 1969). See especially Ch. 4.

L. P. Bouckaert, R. Smoluchowski, and E. P. Wigner: Phys. Rev. **50**, 58 (1936).

A. A. Maradudin and S. H. Vosko: 'Symmetry Properties of the Normal Modes of a Crystal', Reviews of Modern Physics **40**, 1-37 (1968).

M Tinkham: *Group Theory and Quantum Mechanics* (McGraw-Hill, New York, 1964) Chapter 8.

[17] cf. Suzuki-Trotter decomposition for the time evolution integrator

[18] One additional dimension comes from the Trotter decomposition.

[19] The basic idea is to transform the Hubbard model into a quadratic form by introducing the additional Ising-type bosonic field through the Trotter decomposition and the Hubbard-Stratonovich transformation. Fields of the quadratic action can be always integrated out, that is, $\mathrm{Tr}\big[\mathrm{e}^{-\sum_{ij}c^{\dagger}_{i}A_{ij}c_j}\big] = \det(1+\mathrm{e}^{-A})$, where A carries the Ising-type auxiliary field.

G. L. Squires: *Introduction to the Theory of Thermal Neutron Scattering* (Dover, Mineola, New York 1996).

J. L. Warren: 'Further Considerations on the Symmetry Properties of the Normal Modes of a Crystal', Reviews of Modern Physics **40**, 38-76 (1968).

5. Instruments

5.1 Chopper Spectrometers

5.1.1 Concept of a Chopper Spectrometer

Enrico Fermi was unusual. It is a remarkable and profound honor that the name "fermion," with a lower case f, is standard terminology. The field of experimental neutron scattering offers him a smaller honor with the term "Fermi chopper" (Fig. 5.1). A generic Fermi chopper is depicted in Figure 5.2. The ARCS instrument is in fact a Wide Angular Range Direct Geometry Fermi Chopper Spectrometer.[1]

Fig. 5.1. Enrico Fermi works with an electronic control for a neutron chopper during his Argonne days.

The Fermi chopper is discussed at length in Section 5.1.4. For now we note that it works as a fast shutter. By aligning its slot at the right instant after a neutron burst leaves the moderator, it selects a bunch of neutrons that have a particular velocity. We know the time of the neutron burst and the

[1] Unfortunately for Enrico Fermi, the acronym ARFCS is less harmonious than ARCS.

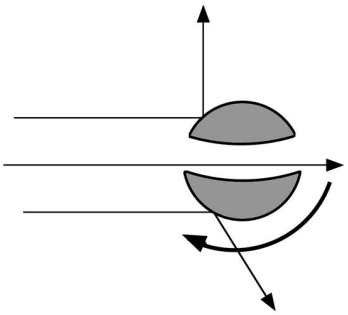


Fig. 5.2. Fermi chopper, comprising a spinning cylinder with a slot in it. Neutrons are transmitted only when the slot is aligned properly along the path of the beam. The chopper works by scattering neutrons out of the forward beam, or by absorbing them by the inner surfaces of the slot. The ARCS choppers work primarily by absorption.

distance between the neutron source and the Fermi chopper, so we therefore know the neutron energy, $E = \frac{1}{2}mv^2$. The idea is best illustrated with the time and distance diagram in Figure 5.3.

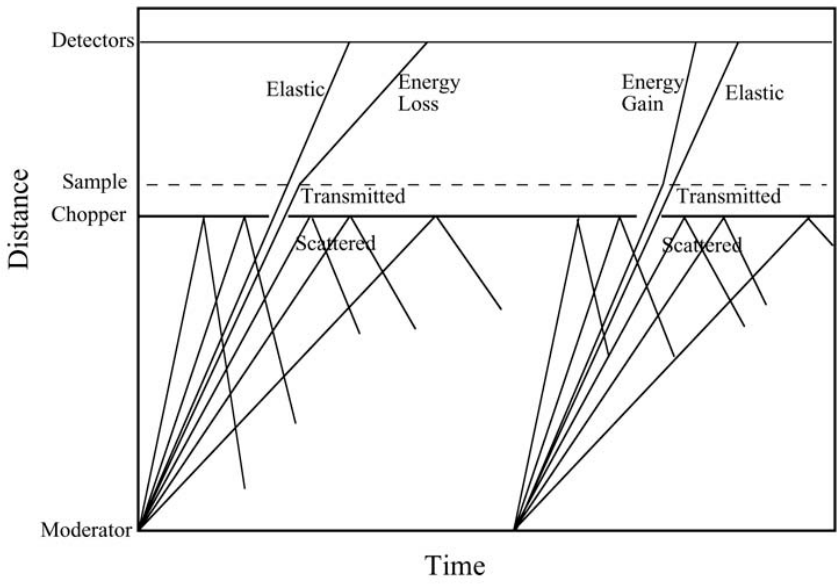


Fig. 5.3. Distance–time diagram for inelastic scattering by a direct geometry chopper spectrometer. Positions of the moderator, chopper, sample, and detectors are marked on the vertical axis. Two moderator pulses occur at points on the x-axis. (Real pulses are not instantaneous, and this issue is discussed below.

The slopes of the lines in Figure 5.3 are velocities of neutrons. Neutrons of many velocities are emitted from the moderator. The figure depicts two neutron pulses from the moderator, separated in time by perhaps 1/60 sec. The Fermi chopper selects a narrow range of velocities, corresponding to a

time window of tens of microseconds. Of course the narrower this window, the more neutrons are blocked by the chopper, and the lower the flux of neutrons on the sample. On the other hand, the narrower this window, the more precisely selected are the velocities, and hence better energy resolution is achieved. The Fermi chopper therefore controls the incident energy, the intensity, and the energy resolution of the neutrons incident on the sample.

The sample usually transmits most of the incident neutrons without energy change, and most of the scattered neutrons are scattered elastically. There is a strong tendency for the lines in Figure 5.3 to have unchanged slopes through the sample. Those neutrons that are scattered inelastically, however, have lines with kinks at the sample, and have slopes after the sample that are either steeper or shallower than for the incident beam.

It is straightforward to know the total distance from the neutron source to the sample, and from the sample to the detector tube. Given the velocity selected by the Fermi chopper, we can figure out the time of arrival of the elastic neutrons at the detectors. Experimentally, we observe an intense elastic peak in the time spectrum at any of the detectors. This is typically quite close to the predicted arrival time, but the experimental time is used to identify the elastic scattering. The neutrons that arrive earlier have gained energy from the sample, and those that arrive later have lost energy to the sample. The inelastic spectrum is obtained from the histogram of neutron arrival times. A Fermi chopper spectrometer works entirely by timing.

5.1.2 Neutron Sources

Spallation. It is a challenge to produce "free neutrons," meaning neutrons that are "free of the nucleus." "Spallation" is one way to make them. The "spallation" process got its name by analogy to using a hammer to "chip" pieces off a heavy stone. Here the the hammer is the particle beam, the chips are neutrons, and the stone is the nucleus. Other things come out of the nucleus besides neutrons, especially γ radiation. High-energy protons (of order 1 GeV) are preferred for the particle beam because protons produce more neutrons, and less heat and photons than are produced by electron beams. The major component of a spallation neutron source is therefore a high-energy, high-current proton accelerator. It often includes a linear accelerator followed by a buncher ring to compress the proton pulse into a short burst in time. Neutron yields are largest for nuclei of high atomic number, since these are neutron-rich.[2] Tungsten and mercury are good choices in practice. Uranium is even better, but it tends to give problems at higher power levels.

[2] An excess of neutron over protons is required for stability. Too many protons means too much Coulombic repulsion, but neutrons can help overcome this instability and keep the nucleus together. Nevertheless, too many neutrons will "drip" out of the nucleus, and neutron-rich isotopes near the neutron "drip line" are good candidates for the target material.

Moderation and Moderator Spectrum. The neutrons fresh from a spallation reaction have energies of order MeV, but the neutrons used in inelastic scattering have energies of order 100 meV. The excessive energies of the spalled neutrons need to be "moderated." The "moderator" makes the transition between the neutron target and the neutron instrument by delivering a useful spectrum of neutrons to the instrument.

How does the moderator reduce the neutron energy by a huge factor of 10^7? By inelastic collisions with nuclei in the moderator. In an inelastic collision between a neutron and a nucleus of the same mass, hydrogen of course, up to half of the kinetic energy of the collision can be transferred to the hydrogen nuclei. The hydrogen in the moderator can be in various chemical forms such as liquid hydrogen, water, or solid methane, depending on the desired temperature and density, for example. The number of collisions required for moderation, n, is

$$2^n \simeq 10^7 \,, \tag{5.1}$$

$$n \simeq 23 \,. \tag{5.2}$$

For the moderator of the ARCS spectrometer, the process of moderation therefore involves a relatively small number of collisions, and is therefore a statistical process. As an approximation it is often assumed that the neutrons leaving the moderator consist of two components. The first is a fully moderated spectrum of thermal neutrons. These neutrons have the Maxwell-Boltzmann spectrum, with probabilities based on Boltzmann factors of $\exp[(-m_{\mathrm{n}}v^2)/(kT)]$, where T is the temperature of the moderator. The second subspectrum is called the "epithermal" neutron spectrum. It is a broad spectrum with a tail that goes to very high energies. Epithermal neutrons have not undergone enough interactions with the moderator to acquire a thermal distribution. This approximation of two subspectra has some semblance of the truth. A better moderator spectrum can be calculated by Monte Carlo neutron transport codes, such as the one that produced the spectra of Figure 5.4a. The cryogenic hydrogen moderator has its maximum intensity at a lower energy than the ambient water moderator for ARCS.

Decoupled, Poisoned Moderator. An important consideration for moderator design is "coupling," meaning the connection between the target (hit by the proton beam) and the material of the moderator. After slowing down in the moderator, the slower neutrons can be absorbed by materials such as Cd and Gd. Putting a layer of Cd or Gd between the moderator and the target can suppress the transmission of the slower neutrons back into the target. This is a "decoupled" moderator. (On the other hand, the fast neutrons from the target are not absorbed as they enter the moderator.) In contrast, a "coupled" moderator allows transmission of all neutrons between the target and the moderator, and less absorption. A coupled moderator produces more neutrons, but it has a disadvantage in the time structure of its neutron pulse. The slow neutrons traveling around the coupled moderator take some time

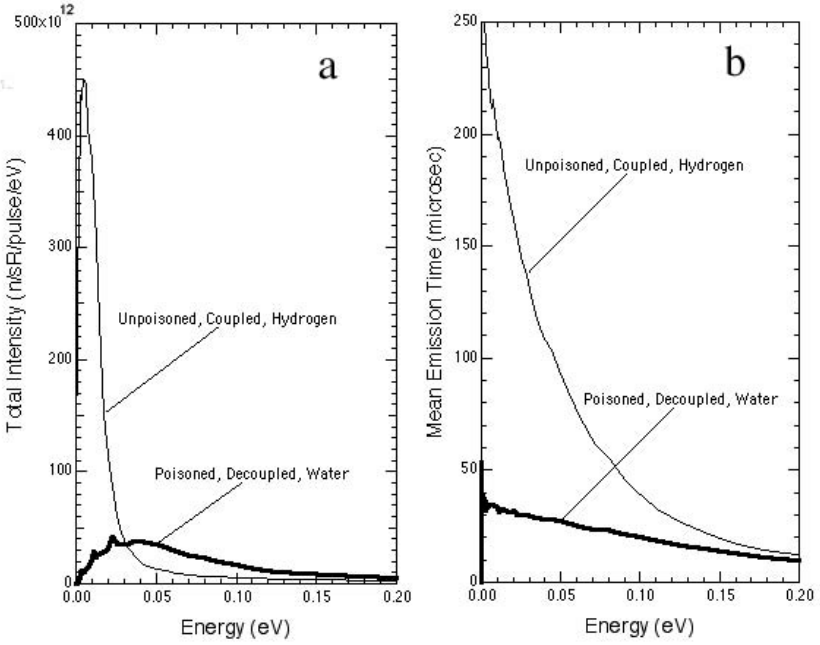


Fig. 5.4. Monte Carlo simulations of intensity spectra and time spectra for two SNS moderators. ARCS is on the decoupled, poisoned water moderator. For comparison a cryogenic hydrogen moderator is also shown. (**a**) Total intensity. (**b**) Mean emission time.

before entering the neutron instrument. This time delay degrades the energy resolution of the Fermi chopper spectrometer, which works by timing.

Another trick to producing short neutron bursts from the moderator is to put a neutron absorber such as Cd or Gd at some depth inside the moderator itself. Again, the idea is to absorb the slow neutrons that haven't found an efficient path out of the moderator. This is called "poisoning" the moderator. A "poisoned, decoupled moderator" generates neutron bursts that are short in time, and this is the moderator of choice for the ARCS instrument, for example. On the other hand, an "unpoisoned, coupled moderator" generates more neutrons in each burst, although these neutrons are emitted over a longer time.

An additional complexity is that the neutrons of different energies are emitted from the moderator at different times. Of course the highest energy neutrons, which do not undergo enough collisions with the nuclei in the moderator, are emitted in the shortest times after the proton pulse hits the target. In general, the lower-energy neutrons leave the moderator at later times, with a broader spread in their emission times. The details of this time-energy correlation are not simple, however, and are best understood by Monte Carlo simulations and experimental measurements. Nevertheless, the neutron emis-

sion times affect the energy resolution of the spectrometer, as discussed in Section 5.1.4. Figure 5.4b shows this energy dependence of the pulse emission times for two SNS moderators. It also shows clearly that the emission time is shorter for the decoupled and poisoned ARCS moderator than for the coupled, unpoisoned hydrogen moderator.

5.1.3 Neutron Guides

Geometrical Optics. Here we develop the scattering of neutrons at interfaces between two homogeneous media. In a homogeneous potential, a neutron wavefunction propagates forward without deflection, but it has a wavelength that depends on the potential. This is much like the propagation of light through glass, for example. The only deflections occur at interfaces, or at changes in the "density." With geometrical optics we can readily utilize the familiar constructions of light optics, scaled appropriately for neutrons. In the present analysis we justify the "geometrical optics" approach to analyzing neutron scattering from macroscopic objects, especially mirrors.

Recall from (2.53) and (2.54), or (2.59) the integral form of the Schrödinger equation in the Born approximation:

$$\Psi_{\rm sc}(\boldsymbol{Q}) = -\frac{m}{2\pi\hbar^2}\int V(\boldsymbol{r}')\,{\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{r}'}{\rm d}^3\boldsymbol{r}'\ , \tag{5.3}$$

where we have ignored the standard form of the outgoing spherical wave that properly multiplies (5.3) (and sets the relationship between $f(\boldsymbol{Q})$ and $\Psi_{\rm sc}(\boldsymbol{Q},\boldsymbol{r})$). For nuclear scattering we use the "Fermi pseudopotential" of (2.60), which places all the scattering potential at a point nucleus at $\boldsymbol{r}$:

$$V_{\rm nuc}(\boldsymbol{r}) = 4\pi\frac{\hbar^2}{2m}\,b\,\delta(\boldsymbol{r})\ , \tag{5.4}$$

where b is a simple constant (perhaps a complex number). The next step is to place numerous Fermi pseudopotentials at the positions of all N nuclei in the material, $\{\boldsymbol{r}_i\}$:

$$V(\boldsymbol{r}) = 4\pi\frac{\hbar^2}{2m}\,\bar{b}\,\sum_{\boldsymbol{r}_i}^{N}\delta(\boldsymbol{r}-\boldsymbol{r}_i)\ , \tag{5.5}$$

where $\bar{b}$ is the average scattering length per nucleus (assuming a mix of isotopes or elements). Substituting (5.5) into (5.3), we notice the handy cancellation of many constant prefactors:

$$\Psi_{\rm sc}(\boldsymbol{Q}) = -\bar{b}\int\sum_{\boldsymbol{r}_i}^{N}\delta(\boldsymbol{r}'-\boldsymbol{r}_i)\,{\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{r}'}{\rm d}^3\boldsymbol{r}'\ . \tag{5.6}$$

In a homogeneous medium we can consider wave motion only in the forward direction, for which $\boldsymbol{Q}=0$, In the forward direction, ${\rm e}^{{\rm i}\boldsymbol{Q}\cdot\boldsymbol{r}'}=1$, simplifying (5.6):

$$\Psi_{\rm sc}(\boldsymbol{Q}) = -\bar{b} \int \sum_{\boldsymbol{r}_i}^{N} \delta(\boldsymbol{r}' - \boldsymbol{r}_i)\, \mathrm{d}^3\boldsymbol{r}' \,, \tag{5.7}$$

$$\Psi_{\rm sc}(\boldsymbol{Q}) = -N\bar{b} \,. \tag{5.8}$$

Note how the details of the positions $\{\boldsymbol{r}_i\}$ are lost for forward scattering. We can obtain the same equation (5.8) if, instead of placing δ-functions at all nuclei, we we use a homogeneous potential throughout the entire material. Instead of using (5.5), equation (5.8) can be obtained by using the following homogeneous potential for $V(\boldsymbol{r}')$ in (5.3):

$$v_0(\boldsymbol{r}) = 4\pi \frac{\hbar^2}{2m}\, \bar{b}\rho \,, \tag{5.9}$$

where ρ is the number of nuclei per unit volume.

This is an important observation. Using (5.9) instead of (5.5), we make the transition from individual scatterings by atomistic Fermi pseudopotentials to geometrical optics. The neutron wave is now considered to travel through a homogeneous potential. The neutron is treated as propagating without scattering, although with a different wavevector depending on the "density," $\bar{b}\rho$.

Total Reflection. When a neutron of energy E enters a region where the potential energy $v_0(\boldsymbol{r})$ is positive, its kinetic energy is reduced and its wavelength is increased (see Fig. 5.5). The kinetic energy cannot go below zero, of course, and a consequence is that some neutrons may not have enough energy to enter a material having a positive $v_0(\boldsymbol{r})$ (except for some surface penetration). Substituting typical numbers into (5.9), we find that $v_0(\boldsymbol{r}) = 3\times10^{-4}\,\mathrm{meV}$, corresponding to a neutron wavelength of 500 Å. Reflection will be total, since the neutron cannot penetrate into the solid, but the neutron is conserved. This result of 500 Åpertains to neutrons arriving normal to a surface. Neutrons of wavelength longer than this critical wavelength will be reflected by the surface. A critical wavevector is shown in Fig. 5.6.

It is possible to get total reflection of more energetic neutrons if they arrive at oblique angles to the surface. The effect is analogous to the case of total internal reflection of light, where a light ray moving in a medium of high index of refraction can be totally reflected at an interface with a medium of low index of refraction, if it reaches this interface at an angle exceeding a critical angle. Such a case is shown in Fig. 5.6. The neutron travels a bit slower in the Ni than in the vacuum. In the Ni, the neutron has a lower kinetic energy (and higher potential energy) and a longer wavelength. Notice the matching of the wave crests across the interface. This continuity of the neutron wavefunction forces a change in direction of the wavevector across the interface – this allows for differences in wavelengths in the vacuum and in the Ni.

The relationship between the neutron wavelength and the critical angle for a homogeneous potential $v_0(\boldsymbol{r})$ can be derived from the Schrödinger equation

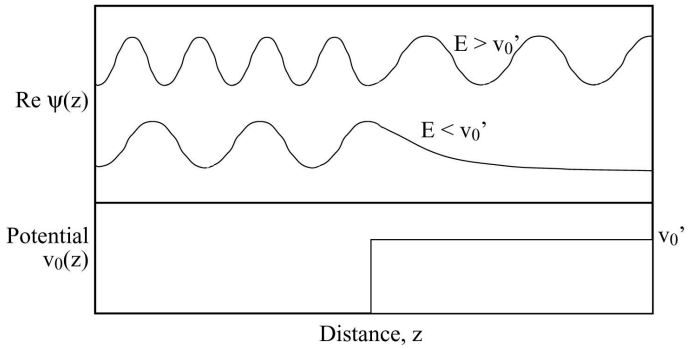


Fig. 5.5. Snapshots of neutron wavefunctions near an interface, which has a step in its potential at the transition between the vacuum (left) and the material (right). The total energies of the wavefunctions are either larger or smaller than the homogeneous potential inside the material.

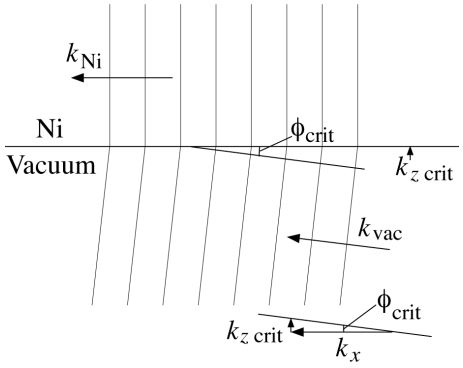


Fig. 5.6. Wavevectors and wave crests at a Ni/vacuum interface. The critical condition has the wavevector in the Ni layer parallel to the interface as shown.

(5.14) by separation of variables. Start with the neutron wavefunction for a neutron moving as a plane wave in the x-z plane, where $\hat{\boldsymbol{z}}$ is normal to the interface between, say vacuum and nickel metal. (For neutrons, the nickel metal has the lower index of refraction than the vacuum.)

$$\Psi_{\rm sc}(\boldsymbol{k},\boldsymbol{r}) = {\rm e}^{{\rm i}\boldsymbol{k}\cdot\boldsymbol{r}} \,, \tag{5.10}$$

$$\Psi_{\rm sc}(\boldsymbol{k},x,z) = {\rm e}^{{\rm i}(k_x\hat{\boldsymbol{x}}+k_z\hat{\boldsymbol{z}})\cdot(x\hat{\boldsymbol{x}}+z\hat{\boldsymbol{z}})} \,, \tag{5.11}$$

$$\Psi_{\rm sc}(\boldsymbol{k},x,z) = {\rm e}^{{\rm i}k_x x}\ {\rm e}^{{\rm i}k_z z} \,, \tag{5.12}$$

$$\Psi_{\rm sc}(\boldsymbol{k},x,z) \equiv \psi_x(x)\ \psi_z(z) \,. \tag{5.13}$$

Substitute (5.13) into the Schrödinger equation (2.32):

$$-\frac{\hbar^2}{2m}\Big(\frac{\partial^2\psi_x}{\partial x^2}+\frac{\partial^2\psi_z}{\partial z^2}\Big)+V(z)\,\psi_x(x)\psi_z(z)=E\,\psi_x(x)\psi_z(z) \,, \tag{5.14}$$

where we have assumed the potential varies only along $\boldsymbol{z}$, changing only at the interface. Dividing through by $\psi_x(x)\psi_z(z)$ and rearranging:

$$-\frac{\hbar^2}{2m}\frac{\frac{\partial^2\psi_z}{\partial z^2}}{\psi_z} + V(z) - E = +\frac{\hbar^2}{2m}\frac{\frac{\partial^2\psi_x}{\partial x^2}}{\psi_x} \,, \tag{5.15}$$

$$= -\varepsilon \,. \tag{5.16}$$

We have separated the z-dependence from the x-dependence. The left-hand side of (5.15) depends only on z, the right only on x, but both z and x can change independently. This means that both sides can only be equal to a constant, which we denoted $-\varepsilon$ in (5.16). The two equations for $\psi_x(x)$ and $\psi_z(z)$ become:

$$\frac{\hbar^2}{2m}\frac{\partial^2\psi_z}{\partial z^2} = \big[\varepsilon - E + V(z)\big]\,\psi_z \,, \tag{5.17}$$

$$\frac{\hbar^2}{2m}\frac{\partial^2\psi_x}{\partial x^2} = -\varepsilon\,\psi_x \,. \tag{5.18}$$

To obtain the required a plane wave solution for a propagating neutron, $\psi_x = \mathrm{e}^{\pm \mathrm{i}(\sqrt{2m\varepsilon})x/\hbar}$, we see that ε must be positive. Our neutron is moving mostly along the x-direction at a glancing angle to the surface, with kinetic energy E when it is in the vacuum. We therefore know that $\varepsilon - E$, although negative, must be rather small. It is therefore possible for a small positive $V(z)$ to switch the sign of the right-hand side of (5.17) from negative to positive. The consequence is interesting. The solution for $\psi_z(z)$ changes from a propagating wave to a damped exponential function. When $V(z)$ is sufficiently positive, the neutron wavefunction therefore does not propagate into the nickel. It is instead reflected from the interface. Of course the $V(z)$ for nickel is fixed, but we can alter the incident angle to get the same effect as shown in Fig. 5.5. By reference to (5.12) and (5.13), for example, we can see that the change in sign of (5.17) occurs when we select k_z so that:

$$k_{z\,\mathrm{crit}} = \frac{\sqrt{2mv_0}}{\hbar} = \sqrt{4\pi\,\bar{b}\rho} \,. \tag{5.19}$$

Critical Angle. The critical angle for total reflection, ϕ_{crit}, is the ratio of $k_{z\,\mathrm{crit}}$ to the wavevector along x, k_x, (the magnitude of k_x is essentially the same as the magnitude of the incident wavevector k):

$$\phi_{\mathrm{crit}} = \frac{\sqrt{4\pi\,\bar{b}\rho}}{k} = \frac{\lambda}{2\pi}\sqrt{4\pi\,\bar{b}\rho} \,, \tag{5.20}$$

$$\phi_{\mathrm{crit}} = \lambda\sqrt{\frac{\bar{b}\rho}{\pi}} \,. \tag{5.21}$$

The critical angle increases in proportion to the wavelength of the neutron. Lower-energy neutrons can be reflected from a surface at higher angles than higher-energy neutrons. For natural Ni metal, the evaluation of (5.21) gives the wavelength-dependence of the critical angle, ϕ_{crit}, here converted from radians to degrees:

$$\phi^{\mathrm{Ni}}_{\mathrm{crit}}\ [^\circ] = 0.0991\ \lambda\ [\mathring{A}] \,. \tag{5.22}$$

Finally, neutron reflectivity can be used for making spin-polarized neutron beams. The coherent scattering lengths are different when neutrons of up spin or down spin are scattered by a magnetic moment having a component along $\hat{\boldsymbol{z}}$. For Co, in fact, the scattering lengths, $b_{\mathrm{Co}}^{\uparrow}$ ad $b_{\mathrm{Co}}^{\downarrow}$, are of opposite sign. A polarizer can be built by choosing incident angles where one of the neutron polarizations is transmitted into the magnetic material, and the other reflected.

Guide Design. For neutrons incident on a surface below the critical angle, simple ray diagrams can be used to develop conceptual designs of neutron guides. Their characteristics are analogous to cylindrical optical fibers for light, but neutron guides are usually made of four long mirrors with a rectangular cross section of order 0.1×0.1 m^2. A section of the ARCS guide is shown in Fig. 5.7. The four mirrors are seen end-on in the drawing at the top left.

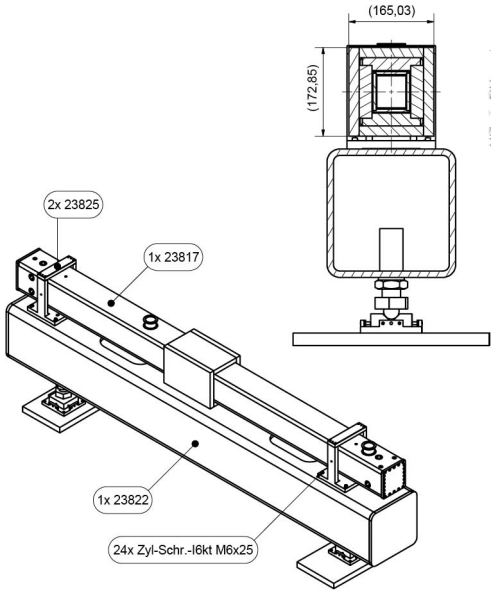


Fig. 5.7. End view and overview of the second section of the main neutron guide for the ARCS spectrometer. Rigid mounting and mechanical adjustment screws are required for ensuring that the long glass mirrors are aligned precisely.

The critical angle for Ni metal is a reference standard, but today a "supermirror" can be prepared from multiple layers of metals, giving higher critical angles by a factor m, where $\phi_{\mathrm{crit}} = m\phi_{\mathrm{crit}}^{\mathrm{Ni}}$. Today the upper limit to m is a bit less than 4. Since these angles are quite small for neutrons of meV energy, guides tend to be quite long. Nevertheless, focusing guides can be designed with parabolic or elliptical surfaces, for example, and these are efficient for transporting neutrons over long distances.

Efficient neutron transport is usually the main function of a guide. With a guide in its incident flight path, the instrument can be placed a good distance

away from the moderator. This allows more space around the instrument, which can be important when instruments are crowded around a small neutron target. The separation from the moderator also allows the instrument to be placed in a location with lower levels of background radiation. Two other issues are useful for understanding the use of guides for neutron transport.

- By giving the mirrors of the guide a slight curvature along the beam direction, the direction of the neutron beam can be bent away from its original straight-line path out of the moderator. This makes it possible for the specimen to be out of a line-of-sight path from the moderator, and therefore less subject to background from fast neutrons.
- Most importantly, the guide reduces the usual attenuation of neutron flux with distance. Instead of the r^{-2} fall-off of intensity, a good guide should cause minimal loss of neutrons that enter the guide below the critical angle. (Note that those entering at higher angles would often be blocked by collimators before the specimen anyhow.) A simple straight mirror guide essentially takes the numbers and divergences of neutrons entering the guide, and translates this distribution to the exit of the guide.

Brightness. Inelastic scattering measurements require beams on samples that have:

- High neutron current [neutrons/sec]
- Small size [cm^2]
- High flux [neutrons cm^{-2} s^{-1}]
- Low divergence [α, radians] (This requirement is unimportant if the Q-dependence is not of interest.)

Even if the neutron guide optics were perfect, and did not absorb neutrons or cause unnecessary spread in divergence, for example, compromises will always be required for meeting all these criteria of a good incident beam. These compromises are quantified in the present section.

A fundamental problem with neutron optics is the size of the moderator. With a cross section of order 0.1×0.1 m^2, the moderator is far from being a point source of neutron emission. This gives a fundamental limitation to the "brightness," β, which is depicted with the three sources shown at the top of Fig. 5.8. All three moderators in Fig. 5.8 emit the same neutron current, and they send the same flux (current density) into the guides, which focus the rays on the sample below. The sources to the left have the higher brightness, and sources with higher brightness are better for making the smallest neutron beams on the sample. The focused spot on the specimen is, in fact, an image of the source itself, so it should be easiest to form a small spot when the source is small.

The source of Fig. 5.8c has the lowest brightness. Nevertheless, the focused beams in Figs. 5.8b and 5.8c have the same size. To make a small spot on the specimen with the low brightness source of Fig. 5.8c, however, the guide in

Fig. 5.8c must provide stronger focusing, i.e., a larger angle of convergence. (Good focusing with a large angle of convergence requires higher quality optics.)

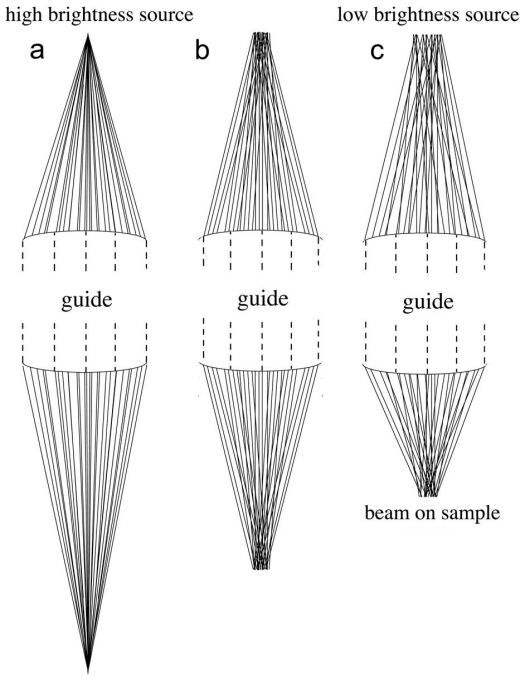


Fig. 5.8. a–c. Formation of focused beams with sources of differing brightness. For all 3 moderators (at top) the neutron currents (number of lines) are the same, and the fluxes at the guide entrances are the same. The brightness of the sources decreases from left to right, owing to a larger area of the source.

More quantitatively, the source brightness, β, is defined as the flux per solid angle $\left[\text{neutrons}/\left(\text{s cm}^2\ \text{sr}\right)\right]$, measured at the source of the neutrons. Brightness is a valuable concept because brightness is a conserved quantity when the subsequent optics are ideal. For example, after a guide focuses the beam as in Fig. 5.8c, the width of the focused neutron beam is reduced by a factor of two compared to the source, but the angle of convergence is increased by a factor of 2. In other words, the flux has increased by a factor of 4, and the solid angle has increased by a factor of 4, leaving unchanged the flux per solid angle (the brightness is conserved). Where the focused beam hits the specimen:

$$\beta = \frac{j_0}{\alpha_x \alpha_y} \ . \tag{5.23}$$

Here j_0 is the flux (neutrons/cm^2) in the beam on the specimen, α_x and α_y are the the angles of beam convergence in the x- and y-planes. We can relate the beam size to the brightness of the source and the convergence angles of the guide, assuming perfect guide optics. The beam width, d_0, is related to

the total neutron current, I_p, by the relationship between current and flux:

$$I_\mathrm{p} = d_x d_y \, j_0 \ . \tag{5.24}$$

For simplicity, assume square cross-sections for the guide and moderator, so $d_x = d_y = d$ and $\alpha_x = \alpha_y = \alpha$. Substituting (5.23) into (5.24) and solving for d_0:

$$d_0 = \frac{\sqrt{\frac{I_\mathrm{p}}{\beta}}}{\alpha} \ . \tag{5.25}$$

Equation 5.25 shows that the beam width d_0 improves (becomes smaller) in proportion to the product $\alpha\sqrt{\beta}$, as suggested by the previous discussion of Fig. 5.8.

The compromises needed in guide design and experimental setups are therefore clear. We have to balance beam width against divergence. A high divergence impairs the Q-resolution of the instrument, but for a moderator of fixed brightness, a high divergence is required for a high neutron flux on a small sample. The trade-off is one of inverse proportionality. If Q-resolution is not an issue, however, small samples become more appropriate.

- To Do: plots of I(E) for ARCS

5.1.4 Fermi Choppers

A first glance at Figure 5.1 shows electronic equipment that looks a bit complicated, and in fact it is. The electromechanical control of a Fermi chopper is not simple, and requires specialized technology. The energy resolution of a Fermi chopper neutron spectrometer depends on timing, and much timing precision is demanded from the Fermi chopper. It must be open at a precise time after the proton pulse hits the neutron target. Microsecond precisions are needed for Fermi chopper timings, as we now show from an elementary calculation.

A neutron of 200.0 meV energy has a velocity of 6,185 m/s. It travels down a 11.6 m flight path in 1,876 μsec, where it encounters the ARCS Fermi chopper. If the energy resolution is to be 1 %, the velocity resolution needs to be 2 %. The Fermi chopper should therefore be open to the beam for 37 μsec. To ensure that all neutrons passing through the Fermi chopper have energies of 200 meV, this opening must occur at a reproducible time delay after each proton pulse hits the neutron target. Variations in this time delay correspond directly to an energy broadening (with no gain in intensity). The electromechanical control system for the Fermi chopper should ensure that the chopper is at the same angle of rotation after the required time delay, here 1,876 μsec. This "phasing accuracy" needs to be 2 to 4 μsec to be negligibly smaller than opening window of 37 μsec in our example.

This is a stringent demand – the slot in a rotating cylinder must be at the same orientation after each neutron pulse, with only a 2 μsec margin for error. For the ARCS spectrometer, the SNS proton pulse on target has a pulse frequency of 60 Hz, but has some drift over time associated with drift in the Tennessee power grid. An electromechanical feedback control system ensures synchronization of the Fermi chopper rotor with the proton pulse. Today these electronic units are somewhat smaller than the one being caressed by Enrico Fermi in Figure 5.1, but they are far more precise and reliable than his.

Another stringent demand can be understood from the opening of the slot in the rotor. The slot has an effective width that corresponds to perhaps 5° of the rotor circumference. How fast does the rotor need to spin for the opening time of 37 μsec in our example? This rotational frequency is $(5/360)/(48\ \mu\text{sec}) = 375\,\text{Hz}$. Such fast rotations present mechanical challenges for bearings and heat dissipation.[3] In practice, Fermi choppers have magnetic bearings, and their rotors spin in a partial vacuum with some helium gas for heat transport. An obvious issue with our calculation of a 375 Hz rotational frequency is that it is not an integral multiple of the 60 Hz proton pulse frequency of the Spallation Neutron Source. In practice, we might select a 360 Hz rotational frequency so that the chopper can be open at a fixed time delay for every proton pulse. Note that the chopper will be open six times for each neutron pulse.[4]

There are different opinions on what to do with the other five pulses that could pass through the Fermi chopper for each proton pulse on target. The relationship between neutron energy and neutron velocity is:

$$E_n = 5.2276 \times 10^{-6}\ v_n^2\ . \tag{5.26}$$

For our hypothetical rotor spinning at 360 Hz (six rotations per proton pulse at 60 Hz), with a phase delay of 1,876 μsec to select a 200.0 meV energy, chopper openings will occur at time delays, τ, of:

$$\tau = 1,876 + n \times 2,778\,[\mu\text{sec}]\ , \tag{5.27}$$

where $n = \{0, 1, 2, 3, 4, 5\}$, and 2,778 μsec is one-sixth of 1/60 sec. For these opening times and a 11.6 m flight path, the five incident energies are 200.0, 25.9, 9.6, 5.0, 3.0 and 2.0 meV. It might be possible to acquire additional inelastic scattering spectra from these five additional incident energies. This practice is called "repetition rate multiplication." It should be noted that in

[3] Disk choppers are an alternative to cylindrical Fermi choppers. For a neutron beam of moderate width, the disk needs to be large, and centrifugal stresses become excessive at 375 Hz.

[4] There are actually twelve openings per proton pulse if you count the half-rotations where the back side of the rotor is facing the incident beam. For these cases, however, the rotor is upside-down in the figures below, and the curvature of the slats impedes the neutron transmission.

the simple Fermi chopper configuration proposed here, the energy resolution becomes impractically narrow for the lower incident energies, however.

The ARCS spectrometer includes a second chopper, a "T_0-chopper," located closer to the moderator than the Fermi chopper. This T_0-chopper is much like a Fermi chopper, and is phased to block the incident neutrons that would arrive at the Fermi chopper during our five other openings of the Fermi chopper. The T_0-chopper also serves to block the flash of γ-radiation and fast neutrons that are emitted by the moderator when the proton pulse hits the target.

The slot through a Fermi chopper has a gradual curvature to accommodate the time for neutron passage through the chopper rotor. This idea is shown in Figure 5.9 for a neutron depicted as moving from left to right. The positions of the neutron and the chopper are shown for three snapshots in time, with $t_1 < t_2 < t_3$.

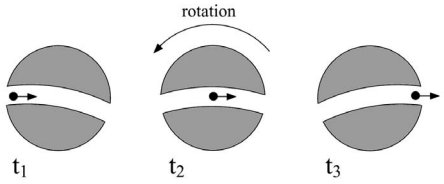


Fig. 5.9. Fermi chopper, rotating counterclockwise, as a neutron moves from left to right. Compare the rotation direction and slot curvature to that of Fig. 5.2.

The slot through the rotor of a Fermi chopper contains a stack of slits and slats, which allow the Fermi chopper to have good energy resolution while passing a wide neutron beam. The structure is shown in Figure 5.10 for a simple case with two "slats." The slats typically contain boron, a strong absorber of neutrons. The neutrons traversing the Fermi chopper must travel through the narrow gaps called "slits." In practice these slits are plates of aluminum with many holes in them. The aluminum serves to maintain a precise spacing between the slats, a challenge at high rotational frequencies. In practice, a Fermi chopper may contain 10 to 40 slats. With many slats, it is possible for the Fermi chopper to pass a beam of 5 to 6 cm in width while maintaining good energy resolution. Not surprisingly, with narrower slits there is a loss of intensity that accompanies the improvement in energy resolution.

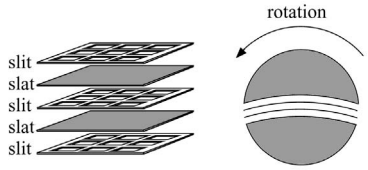


Fig. 5.10. Fermi chopper rotor with two slats. The structure at left is an exploded view of the layered slits and slats that fill the curved slot through the rotor.

A drawing of the Fermi chopper used in the ARCS instrument is shown in Fig. 5.11. The neutron beam passes through the circular opening at the center of the housing. The electric motor, sensors, and magnetic bearings are above and below the rotor assembly. A corresponding pair of photographs is shown in Fig. 5.12. In the ARCS instrument, two identical chopper systems are mounted on a translation table so either can be moved into the neutron beam. The choppers can be moved while they are spinning, allowing quick changes to the energy of the incident beam of the ARCS instrument.

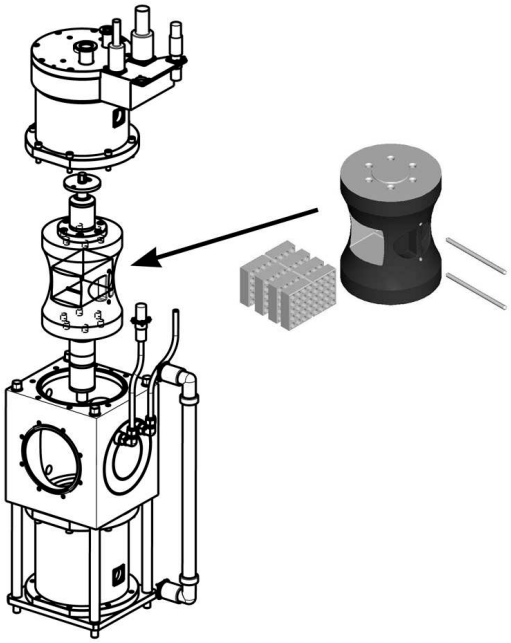


Fig. 5.11. Exploded view of ARCS Fermi chopper, showing housing and rotor at left. The slit packages are held into the rotor with two pins, as shown at right.

Rotor assemblies with curved slots, especially those with many slits and slats, are optimized to work at particular combinations of rotational frequencies, neutron velocities, and energy resolutions. These need to be selected in the planning stage of an inelastic neutron scattering experiment.

Finally, it is important to remember that Fermi choppers scatter neutrons out of the incident beam, generating a high intensity of scattered neutrons around a Fermi chopper. Good shielding is required, both for radiation protection for the experimenters, and for minimizing background in the instrument itself. The ARCS T_0 chopper helps alleviate this problem by minimizing the number of neutrons that are scattered out of the beam by the Fermi chopper.

Fig. 5.12. Left: photograph of ARCS Fermi chopper, showing housing, windows, and water cooling. Right: photograph of the slit package, with aluminum slits seen end-on.

5.1.5 Detectors

- short: no user adjustments.
  - n capture
  - ionization
  - gas gain
  - position sensitivity
  - problems
  - calibration

5.1.6 Energy Resolution

Energy resolution is an important figure-of-merit for the design of direct geometry chopper spectrometers, and can be the most important figure-of-merit for incoherent scattering. The primary flight paths at SNS are quite long, and the moderator pulses are short on flight paths 17 and 18, promoting good energy resolution. In evaluating instrument configurations, the secondary flight path is the parameter for adjusting energy resolution. When selecting a distribution of detector positions around the specimen, a spherical locus of detector positions provides energy resolution that is uniform at all angles around the sample.

Basic considerations for energy resolution were presented in the discussion of Fig. 5.3. Here we consider the issue further, starting with the moderator. The moderator does not produce an instantaneous pulse of neutrons, but the pulse has a time spread of several microseconds. The important consequence is a lower bound on the energy resolution. Even by reducing the opening time

for the Fermi chopper to nearly zero seconds, a minimum energy resolution is caused by the spread of moderator emission times. Figure 5.3 presents a graphical explanation of the issue. In comparison to Fig. 5.3, note the spread in emission times. Even though the Fermi chopper has a minimal open time, the neutrons passing through the chopper have a spread in slopes in Fig. 5.3, a spread in velocities, and hence a spread in energy.

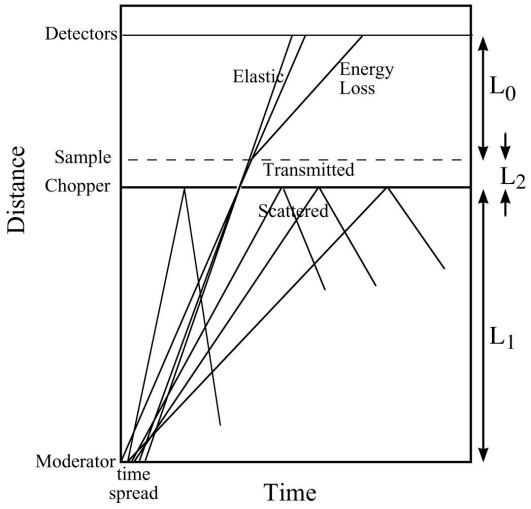


Fig. 5.13. Distance–time diagram for a direct geometry chopper spectrometer. The figure is much the same as Fig. 5.3, but there is a spread in times when the neutrons leave the moderator. Positions of the moderator, chopper, sample, and detectors are marked on the vertical axis. Here the Fermi Chopper is open for only a minimal time.

The energy resolution of a Fermi chopper spectrometer is often analyzed in terms of the moderator performance and the chopper performance. For analytical calculations, it is typical to assume that both the moderator and the chopper cause time smearing that has a Gaussian shape about an average time. The advantage to this approach is that one can convolute the different effects of time smearing to get another Gaussian function, whose width is obtained by adding in quadrature the widths from independent broadenings. The resolution is then obtained in terms of the following variables for time, energy, and distance:

- L_0...distance from the moderator to the Fermi chopper
- L_1...distance from the sample to detector
- L_2...distance from the Fermi chopper to the sample
- t_r...time spread of neutrons passing through the Fermi chopper
- t_m...time spread of neutrons from the moderator
- δ_m...distance the neutrons travel in time t_m

- E_0...incident energy selected by Fermi chopper
- E_1...energy of the neutron leaving the sample

It is typical to express the energy resolution as a fraction of the incident energy, $\Delta E/E$. The Gaussian analysis performed by Windsor [C.G. Windsor] gives:

$$\frac{\Delta E}{E} = \frac{2\delta_m}{L_0}\left[\left(1+\left(\frac{E_1}{E_0}\right)^{3/2}\frac{L_2}{L_1}\right)^2 + \left(\frac{t_r}{t_m}\right)^2\left(1+\left(\frac{E_1}{E_0}\right)^{3/2}\frac{L_0}{L_1}\left(1+\frac{L_2}{L_0}\right)\right)^2\right], \tag{5.28}$$

Energy resolution can be presented in various ways, and it is important to be careful when comparing plots for different instruments. Perhaps the most important criterion is the neutron flux on the sample for a given energy resolution, calculated for conditions that optimize the intensity. Figures 5.14 and 5.15 show such plots for the ARCS instrument. The first figure shows the performance for incident neutrons of 63 meV energy, approximately at the peak of the moderator brightness. These neutrons have a longer time spread for emission from the moderator, however, and this limits the ultimate energy resolution to about 2.5% of the incident energy. Figure 5.15 shows the same plot for neutrons of 250 meV incident energy. The ultimate resolution is better, perhaps 2% of the incident energy, in large part owing to moderator performance. The intensity is lower at 250 meV for the modest resolution of 4%, however. The neutron guide is not so effective at 250 meV, and the moderator spectrum is weaker at this energy.

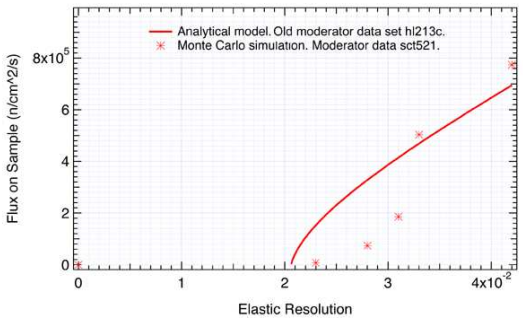


Fig. 5.14. Flux on sample for various energy resolutions of the ARCS spectrometer for 63 meV incident neutrons.

5.1.7 *Q* Resolution

Optimization is quite different for Q resolution, however. This has been less of a focus of the ARCS instrument design, however, because historical trends

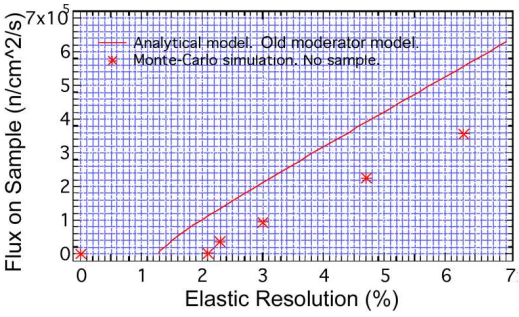


Fig. 5.15. Flux on sample for various energy resolutions of the ARCS spectrometer for 250 meV incident neutrons.

have emphasized the design of instruments for work with polycrystalline samples. Measuring dispersive excitations in single crystals forces the consideration of Q resolution. Specifically, it is suggested that the instrument be optimized for fractional Q resolution. The figure-of-merit, $\Delta Q/Q$, is analogous to the figure-of-merit, $\Delta E/E$ for E resolution.

The parameter space for optimization is broad, since various types of excitations must be considered. The extreme cases are treated below as: 1) elastic scattering, 2) highly dispersive inelastic scattering and 3) dispersionless inelastic scattering.

Summary of Q Broadening. Four sources of Q broadening are considered:

- The finite size of the moderator, even over a long primary flight path, provides a significant angular divergence of neutrons on the specimen. This divergence is about 0.4° without a guide, and will be larger for thermal neutrons that have been reflected along the guide. If the excitations in the specimen do not select among these incident $\boldsymbol{k}_i$, the resolution of the experiment will be dominated by incident divergence.
- A 0.2° mosaic spread of a good crystal will cause an angular spread of 3.5 cm at 10 m. This is comparable to detector pixelation along a diagonal direction, which is approximately $2.5\sqrt{2} = 3.5$ cm.
- The energy resolution leads to a coupled smearing in Q. Since the energy resolution is quite good for all flight paths, this source of Q smearing is not the biggest problem for single crystal work, especially in the forward direction.
- The $\sin\theta$-dependence of Q conspires with finite detector pixel size to give a divergent l_3 in the the forward direction. A R_Q of 1.0 % is not possible for scattering angles, 2θ, smaller than 14° if a 10 m secondary flight path is used with 2.54 cm detectors. The problem is proportionately worse for shorter flight paths.

5.1.8 Optimization for $\Delta Q/Q$ in Elastic Scattering

It is easiest to first analyze the Q resolution for elastic scattering, and some of the issues for elastic scattering pertain directly to inelastic scattering.

Incident Divergence. The finite size of the moderator provides an angular divergence of incident neutrons on the sample. This is approximately 10 cm over 1350 cm, or about 0.4°. This incident divergence may or may not contribute to the divergence in scattered beams, depending on the process involved. For elastic Bragg scattering, for example, the incident divergence will allow intense Bragg peaks from all parts of a sample misoriented to within about 0.4°. Although incident divergence is a potentially serious problem, a perfect crystal will still provide sharp Bragg diffractions. This incident divergence can be the dominant effect on Q-resolution for some types of inelastic scattering, however.

Mosaic Spread. It is important to consider the limit to Q resolution caused by the sample itself – the instrument should need not be much better than the intrinsic resolution of the sample. One issue is that a finite sample subtends a non-zero angle over the secondary flight path, l_3. This can be ignored for a typical 1 cm sample, which is much smaller than the expected size of detector pixels.

The "mosaic spread," denoting the mean mutual misorientations of the different subvolumes of a crystal, is an important figure-of-merit for single crystal samples. A good single crystal sample may have a mosaic spread of 0.2 °, for which the angular blurring of an elastic beam at a detector located 10 m away from the sample is 3.4 cm. This is only slightly larger than typical detector pixel resolutions. Better instrument resolution would be a reasonable request, except for the fact that a 10 m sphere of detectors around the sample is prohibited by constraints of cost and space. Nevertheless, we note that a long secondary flight path, $l_3 = 10\,\mathrm{m}$, could be justified for single crystals of high quality in cases where the incident divergence does not dominate the Q resolution.

Locus of Constant $\Delta Q/Q$. We seek the locus of detectors that provides a constant $\Delta Q/Q$, defined as the resolution, R_Q:

$$R_Q \equiv \frac{\Delta Q}{Q} \,. \tag{5.29}$$

By definition

$$Q \equiv 4\pi \frac{\sin(\theta)}{\lambda} \,, \text{ so} \tag{5.30}$$

$$\Delta Q = 4\pi \frac{\cos(\theta)}{\lambda} \Delta\theta \,, \tag{5.31}$$

giving

$$R_Q = \cot(\theta)\Delta\theta \,. \tag{5.32}$$

When R_Q is plotted in polar coordinates, the detector locus is defined. This is shown in Fig. 5.16. This shape is quite incompatible with the spherical

solution for constant $\Delta E/E$. At low angles, the detector placements required for constant $\Delta E/E$ and constant $\Delta Q/Q$ are exactly orthogonal.

The resolution R_Q diverges at small angles, because at small angles $Q \rightarrow 0$, so very small ΔQ is needed to maintain a constant $\Delta Q/Q$. A small angular spread for ΔQ is achieved only when the detector pixels are very small or are placed at very large l_3. At the other extreme, the detector distance for constant $\Delta Q/Q$ is infinitesimal when the scattering angle, 2θ, is $180\,^\circ$, since small variations in angle have no effect on Q for direct backscattering.

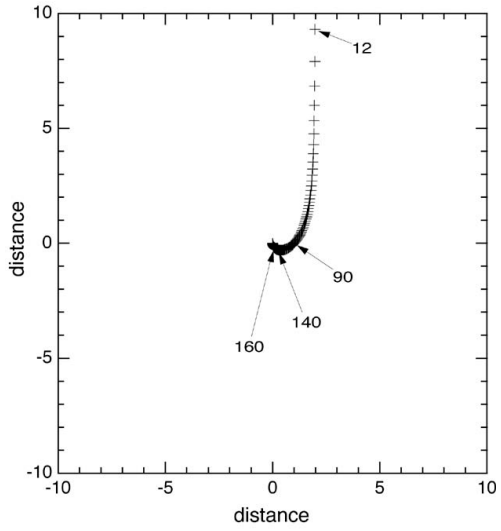


Fig. 5.16. Detector locus for constant $\Delta Q/Q$. Crosses are spaced at $2\,^\circ$ increments of 2θ . Labels indicate specific 2θ angles. Sample would be located at (0,0). Distance units are arbitrary, but they are equal for the two axes.

Coupling of Q Resolution to E Resolution. One obvious incompatibility of the $\Delta Q/Q$ optimization and the $\Delta E/E$ optimization is that the secondary flight path, l_3, becomes vanishingly small at $\theta = 90°$. (This is strictly true only for the elastic scattering. For a fixed energy transfer, however, there will be a particular angle of inelastic scattering for which $\Delta Q = 0$ in Eq. (5.32).)

Poor energy resolution will cause a smearing in Q, increasing R_Q. For elastic scattering:

$$Q = k_f - k_i = \frac{\sqrt{2mE}}{\hbar} 2\sin\theta \,, \tag{5.33}$$

$$\frac{\mathrm{d}Q}{\mathrm{d}E} = \frac{\sqrt{2m}}{\hbar\sqrt{E}} \sin\theta \,, \tag{5.34}$$

$$\frac{\mathrm{d}Q}{\mathrm{d}E} = \frac{Q}{2E} \sin\theta \,, \tag{5.35}$$

$$\frac{\mathrm{d}Q}{Q} = \frac{\mathrm{d}E}{E} \frac{\sin\theta}{2} \,, \tag{5.36}$$

so $\Delta Q/Q$ and $\Delta E/E$ are comparable for modest angles.

The obvious solution is to use a spherical detector locus for high scattering angles. The 3.0 m flight path of ARCS gives energy resolutions as small as $\Delta E/E = 1\,\%$. At $2\theta = 40^\circ$ the contribution to $\Delta Q/Q$ is less than 0.2 %, and this is even smaller at lower angles.

5.1.9 Optimization of $\Delta Q/Q$ for Inelastic Scattering

Ewald Spheres and Incident Divergence. Three important cases are depicted in Fig. 5.2. Case 1, elastic scattering, was the topic of the previous Sect. 5.1.8. Case 2, dispersive inelastic scattering, is considered first. This problem is presented in two parts, the case for the first Brillouin zone, and the case for higher Brillouin zones. Case 3 of Fig. 5.2 is presented last, since it is more straightforward.

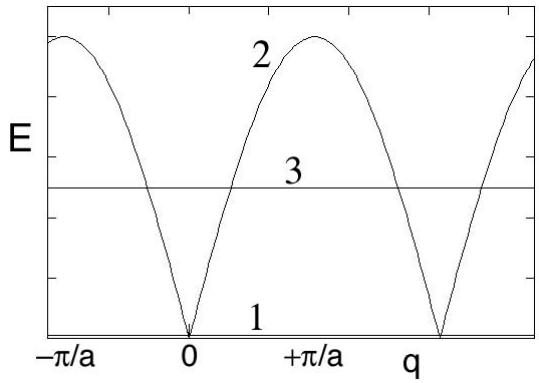


Fig. 5.2. Three types of scattering processes (1) is elastic scattering, (2) is dispersive inelastic scattering, and (3) is non-dispersive inelastic scattering.

For a fixed energy loss, the value of $|\boldsymbol{k}_f|$ is constant, although $\boldsymbol{k}_f$ may take different orientations. The kinematics of this scattering are understood most conveniently by use of the Ewald sphere construction, which is shown for elastic scattering in Fig. 5.3a. The Ewald sphere construction is useful for

analysis of diffraction conditions because it identifies the relationship between the wavevector transfer, Q, and the reciprocal lattice vector, τ, showing when the Laue condition, $\boldsymbol{Q} = \boldsymbol{\tau}$, is satisfied for diffraction. In inelastic scattering the momentum of the excitation, $\boldsymbol{q}$, plays a similar role to $\boldsymbol{\tau}$, especially in the first Brillouin zone where τ is zero.

For the inelastic scattering process shown in Fig. 5.3b, all $\{\boldsymbol{k}_f\}$ have the same length (as for the elastic case of Fig. 5.3a), since the energy E_f is identical for all orientations of $\boldsymbol{k}_f$. In both Figs. 5.3a and 5.3b, the allowed $\boldsymbol{k}_f$ make a sphere, which when placed at the tail of $\boldsymbol{k}_i$, provide the allowed $\{\boldsymbol{Q}\}$ as shown in the figure (where as usual $\boldsymbol{Q} \equiv \boldsymbol{k}_f - \boldsymbol{k}_i$).

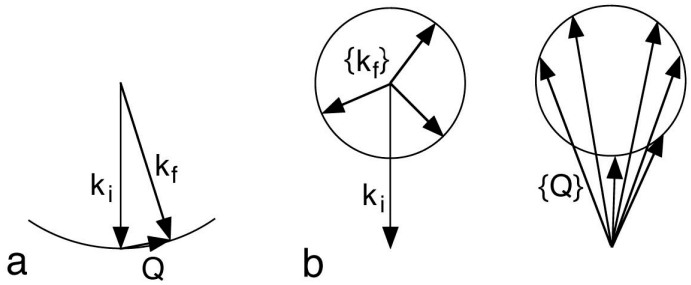


Fig. 5.3. (**a**) Ewald sphere construction for elastic scattering, showing allowed $\boldsymbol{Q}$ for Bragg diffraction. (**b**) Ewald sphere construction for inelastic scattering, showing allowed $\{\boldsymbol{Q}\}$.

The advantage of the Ewald sphere constructions of Fig. 5.3 is in analyzing tilts of the incident beam, as occurs for incident divergence, for example. The cases of Fig. 5.4 show conditions for elastic scattering before and after tilt. It is evident that the condition for momentum conservation (the Laue condition)

$$\boldsymbol{Q} - \boldsymbol{\tau} = 0 \tag{5.37}$$

is generally violated by tilting, because tilts of $\boldsymbol{k}_i$ by the angle ϕ cause $\boldsymbol{Q}$ to tilt by this same angle ϕ. The consequence of this violation of momentum conservation is that coherent elastic scattering is altered, and Bragg diffractions may be eliminated, for example. Those neutron trajectories with improper $\boldsymbol{k}_i$ will not contribute to the Bragg diffraction.

First Brillouin Zone – Soft Dispersions. One case for inelastic scattering is shown in Fig. 5.5a. This case is a scattering from the first Brillouin zone, where the reciprocal lattice vector, $\boldsymbol{\tau} = 0$. The diameters of the two circles are set by the energy and momentum transfer to the excitation. In Fig. 5.5a, $\left|\boldsymbol{k}_f\right| \simeq \sqrt{2}\left|\boldsymbol{k}_i\right|$, indicating that the neutron has lost half its energy to the solid. For the present case where the excitation is of energy $E_i/2$, we further assume a relatively soft dispersion where the momentum of the excitation q

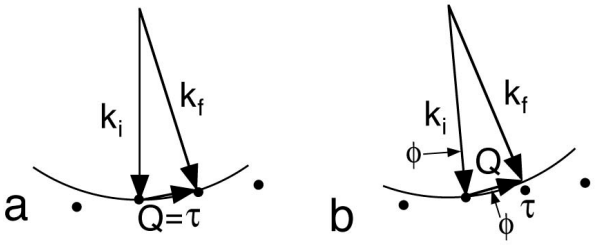


Fig. 5.4. Ewald sphere construction for elastic scattering (**a**) before, and (**b**) after tilt of $\boldsymbol{k}_i$. Notice that $\boldsymbol{Q}$ no longer touches the reciprocal lattice vector $\boldsymbol{\tau}$ after tilt.

is assumed the same as the momentum of the neutron, k_f. A circle is used for the locus of acceptable $\boldsymbol{q}$ (a circle is not expected for anisotropic crystals, of course). This circle has a large radius, as may be expected if a dispersive excitation has a soft dispersion relation.

The effect of tilt on the scattering condition is shown in Fig. 5.5b. It is evident by geometry that the tilt of $\boldsymbol{k}_i$ and $\boldsymbol{q}$ are by the same angle for this case of $\boldsymbol{q}$ lying in the first Brillouin zone. This could affect the ability of the neutron to excite the dispersive mode, at least if the dispersive mode is as sharp in $\boldsymbol{q}$ as a Bragg diffraction. Conditions where highly anisotropic dispersions have a delicate contact with the locus of $\boldsymbol{Q}$ can also be envisioned.

Higher Brillouin Zones – Stiff Dispersions. The case of a high energy excitation is shown in Fig. 5.6. In this case the dispersion is very steep, and the energy is so high that the energy of the excitation is large, even when q is small. In this particular case, there are no excitations in the first Brillouin zone because $q < Q_{min}$.[5] The excitation can occur in a higher Brillouin zone with the help of a reciprocal lattice vector, $\boldsymbol{\tau}$ so that

$$\boldsymbol{Q} - \boldsymbol{q} - \boldsymbol{\tau} = 0 \,. \tag{5.38}$$

This case of a high-energy excitation is a higher Brillouin zone is interesting because a tilt of $\boldsymbol{k}_i$ does not induce a tilt of $\boldsymbol{Q}$ by the same angle. Figure 5.6 shows that small tilts have big effects on the orientation of $\boldsymbol{q}$. In essence, the reciprocal lattice vector can amplify the tilt of the incident beam on the rotation of $\boldsymbol{q}$. In this case it seems plausible that for sharp, stiff excitations, only a $\boldsymbol{k}_i$ of the correct orientation can generate the excitation. The mosaic spread of the sample will be able to pick these acceptable $\boldsymbol{k}_i$ from the incident beam, and the Q resolution of the experiment will originate with the mosaic spread of the sample and not the incident divergence.

[5] Experimentally, it might be prudent to use a larger E_i and k_f so that excitations in the first zone are allowed. In this case the result of Sect. 5.1.9 is recovered, that is the tilt angle of $\boldsymbol{k}_i$ equals the tilt angle of $\boldsymbol{q}$. The present results from higher Brillouin zones remain valid, however.

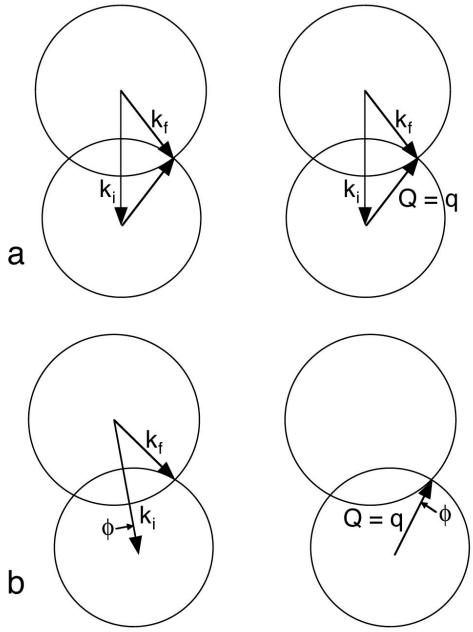


Fig. 5.5. Ewald sphere construction for inelastic scattering (**a**) before, and (**b**) after tilt of $\boldsymbol{k}_i$. Notice that although $\boldsymbol{Q}$ has the same magnitude, it changes orientation by the same tilt angle, ϕ, as the incident beam.

Non-Dispersive Excitations. The case of non-dispersive excitations in Fig. 5.2 is straightforward to analyze with Figs. 5.5 and 5.6. The point is that the circle of q can be of arbitrary radius, since the energy of the excitation does not depend on q. For this reason, at a fixed energy transfer equal to that of the excitation, there will always be an excitation with an appropriate q to satisfy momentum conservation. All orientations of $\boldsymbol{k}_i$ will be useful for generating the excitation, and the incident divergence will dominate over the crystal mosaic spread in setting the Q resolution.

A similar result pertains to inelastic incoherent scattering.

5.1.10 Background

- Important section...as long as possible. This is a real issue.
  - forward beam scattering
  - double scattering/triple scattering
  - neutron gas theory

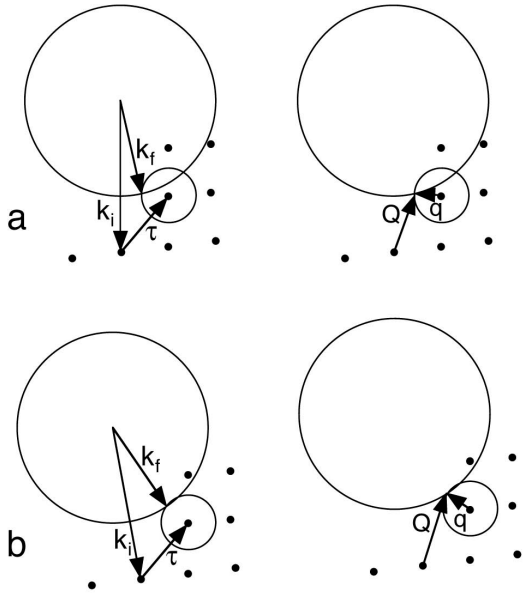


Fig. 5.6. Ewald sphere construction for inelastic scattering (**a**) before, and (**b**) after tilt of $\boldsymbol{k}_i$. In this case the excitation has a large E for a small q. Notice that although $\boldsymbol{Q}$ has the same magnitude, it changes orientation strongly after tilt of the incident beam.

5.1.11 Sample Design

Inelastic neutron scattering experiments require careful attention to the sample itself. Of course the sample must be in the correct state, satisfying basic criteria such as magnetic polarization, chemical composition, crystal structure, crystallographic quality or texture, temperature and pressure. Criteria for these sample states must not be compromised to the detriment of the science. The in-situ control of temperature, pressure, and magnetic field often leads to some compromises in experiment design, and expertise with the available "sample environment equipment" is essential in planning an experiment.

A most common problem for inelastic scattering experiments is a sample that is too small. There is always a background in the spectrum from stray neutrons in the instrument, and inelastic neutron scattering experiments will suffer from poor signal-to-noise ratio whenever this background has an intensity comparable to the scattering from the sample itself. Robust scattering from the sample usually means having a large quantity of sample. Experimenters are often dismayed that inelastic scattering measurements typically require one or even two orders-of-magnitude more sample than is typical for neutron diffraction measurements. The ideal sample for inelastic scattering will be as wide as possible so that it fills the cross section of the incident

beam. With a wide sample, all neutrons in the beam will be candidates for scattering, and no parts of the beam will pass directly into the beamstop or generate unnecessary background.

The thickness of the sample in the direction of the beam needs to be chosen with care. It is possible to have a sample that is too thick. There are two limits that set upper bounds on the thickness of samples, and we consider each in turn below. The sample should be as thick as possible, provided it does not exceed limits imposed by:

- Multiple scattering. If the scattering cross-section is large and the sample is thick, too many neutrons will be scattered multiple times inside the specimen. Each scattering has its own energy spectrum, and the convolution of these spectra is difficult to sort out in the measured data.
- Absorption. Some nuclei are strong absorbers of neutrons. For thick samples of strong absorbers, few neutrons can leave the samples and enter the detectors.

Scattering and Attenuation. As the neutron beam passes through a material, there is a reduction in the number of neutrons traveling in the incident beam. At the depth x, the increment of thickness of a material, $\mathrm{d}x$, scatters a number of neutrons, $\mathrm{d}I$, removing them from the beam. The number of scattered neutrons, $-\mathrm{d}I(x)$, equals the product of 1) the increment of thickness, $\mathrm{d}x$, 2) the number of neutrons present at x, $I(x)$, and 3) a material coefficient, s:

$$-\mathrm{d}I(x) = \mathsf{s}\, I(x)\, \mathrm{d}x \;, \tag{5.39}$$

$$\frac{\mathrm{d}I(x)}{\mathrm{d}x} = -\mathsf{s}\, I(x) \;, \tag{5.40}$$

$$I(x) = I_0\, \mathrm{e}^{-\mathsf{s}x} \;. \tag{5.41}$$

If scattering is the only process that removes neutrons from the incident beam as it traverses a sample of thickness t, the loss from the beam equals the intensity of the scattering:

$$I_{\mathrm{scat}}(t) = I_0 \left(1 - \mathrm{e}^{-\mathsf{s}t}\right) \;. \tag{5.42}$$

The product in the exponent, $\mathsf{s}x$ or $\mathsf{s}t$, must be dimensionless, so s has dimensions of $[\mathrm{cm}^{-1}]$. When $\mathsf{s}x$ is small, it equals the fraction of neutrons removed from the incident beam. From Fig. 2.3 we know that this fraction also equals $N\sigma/A$, so:

$$\mathsf{s} = \frac{N\sigma}{Ax} = \frac{N}{V}\sigma \;, \tag{5.43}$$

where N/V has units [atoms cm^{-3}] and σ is the scattering cross-section with units $[\mathrm{cm}^2]$.[6]

[6] Since density varies with the type of material, "mass attenuation coefficients" are often normalized as ratios s/ρ. Here the density, ρ, has units $[\mathrm{g\,cm}^{-3}]$, so the coefficients s/ρ have units $[\mathrm{cm}^{-1}]/[\mathrm{g\,cm}^{-3}]$=$[\mathrm{cm}^2\ \mathrm{g}^{-1}]$. Exponents in (5.41) are products $(\mathsf{s}/\rho) \times \rho \times x$, and are, of course, dimensionless.

It is straightforward to calculate the composite mass attenuation coefficient for a compound or an alloy. The point to remember is that the total neutron scattering depends on the number and types of atoms in the path of the beam. The composite attenuation coefficient is obtained from the attenuation coefficients, s_i, for the different elements, i, weighted by their atomic fractions in the material, f_i:

$$\langle \mathsf{s} \rangle = \sum_i f_i \, \mathsf{s}_i \ . \tag{5.44}$$

Multiple Scattering Criterion. It may seem curious that for well-planned experiments, most of the neutrons are transmitted through the sample without scattering.[7] Inverting this heirarchy with very thick specimens can cause the data to be uninterpretable, as we now show.

Consider the probability of inelastic scattering, p_i, and elastic scattering, p_e, through a thin layer of material. We set $p_\mathrm{e} + p_\mathrm{i} = p$, where p is the total probability of scattering in the layer. For thin samples of n layers, each of thickness x, we have the relationship between numbers of layers and the bulk scattering coefficient, s:

$$\langle \mathsf{s} \rangle x = n \ p \ . \tag{5.45}$$

For considerations of multiple scattering, it is most convenient to work with the numbers of layers and scattering probabilities.

A challenge for multiple scattering calculations is the three-dimensionality of the sample. Neutrons can be scattered initially upwards, followed by a second scattering to the side, and a third scattering out to the detectors. A computer program such as MSCATT is required for accounting for such possibilities. Here we provide an approximate analysis in one dimension.

We assume that all neutrons entering the layered sample eventually leave the back of the specimen and are observed. The probability, p' for a neutron being scattered j times is the product of j of the layers doing a scattering, and $n - j$ of the layers doing no scattering:

$$p' = p^j (1-p)^{n-j} \ . \tag{5.46}$$

We are not keeping track of which particular layer has done the scattering. We keep track of the fractions of neutrons that are scattered j times in our n layers when the scattering probability in each layer is p. The number of ways of arranging the j scatterings over the n layers is therefore the binomial coefficient. The total fraction of neutrons that are observed in our example is 1, and should equal the sum of all scatterings, including $j = 0$ for no scattering:

$$1 = \sum_{j=0}^{n} \frac{n!}{(n-j)! \ j!} \ p^j (1-p)^{n-j} \ . \tag{5.47}$$

[7] Often the next largest fraction of neutrons are scattered elastically, and the smallest fraction are scattered inelastically.

We have implicitly assumed $p \ll 1$, so the number of scatterings $j \ll n$, and each layer scatters at most once. (This also ensures that the terms for which $j \sim n$ are negligible in (5.47).) We can always satisfy this assumption $p \ll 1$ by dividing our sample into finer and finer layers, because (5.45) shows us that for a fixed sample, np is a constant. Write out the first few terms of (5.47) in the binomial expansion:

$$1 = (1-p)^n + np(1-p)^{n-1} + \frac{n(n-1)}{2}\, p^2(1-p)^{n-2} + \\ + \frac{n(n-1)(n-2)}{6}\, p^3(1-p)^{n-3} + ... \tag{5.48}$$

We simplify by using our condition of large n and small p,

$$1 = (1-p)^n + (np)^1(1-p)^{n-1} + \frac{1}{2}\, (np)^2(1-p)^{n-2} + \\ + \frac{1}{6}\, (np)^3(1-p)^{n-3} + ... \tag{5.49}$$

The first term in (5.49) is the the probability of zero scattering, the second is the probability of one scattering, the third – two scatterings, etc. We are interested in knowing the ratios of the different terms in this series. Since $1 - p \simeq 1$, and using (5.45), the ratios are:

$$1 \ : \ \mathsf{s}x \ : \ \frac{1}{2}(\mathsf{s}x)^2 \ : \ \frac{1}{6}(\mathsf{s}x)^3 \ ... \tag{5.50}$$

The parameter $\mathsf{s}x$ is dimensionless, and for experiment design we seek $\mathsf{s}x \simeq 0.10$. In other words, we seek a sample that scatters 10% of the incident neutrons. For this particular case, (5.50) gives the ratio of double scattering to single scattering of $1/2\mathsf{s}x = 0.05$. Consider the effect of this double scattering on the inelastic spectrum. The scattering probability is the sum of an inelastic and elastic probability:

$$p = p_{\mathrm{e}} \ + \ p_{\mathrm{i}} \ , \tag{5.51}$$

$$np = np_{\mathrm{e}} \ + \ np_{\mathrm{i}} \ , \tag{5.52}$$

$$\langle \mathsf{s} \rangle x = \langle \mathsf{s}_{\mathrm{e}} \rangle x \ + \ \langle \mathsf{s}_{\mathrm{i}} \rangle x \ . \tag{5.53}$$

For this sample that scatters 10% of the incident neutrons, the ratio of the amount of inelastic single scattering to the amount of inelastic double scattering will be 5% – the same ratio as for the total scattering.

The effect of multiple scattering is to smear out the measured inelastic energy spectrum, and spread it out over twice the energy range of the single-scattering spectrum. Suppose the first scattering has the double-differential cross section of (2.78). Suppose a particular inelastic scattering causes an energy loss $\epsilon = \hbar\omega$. If this scattered neutron with altered energy and wavevector now undergoes a second inelastic scattering, the process may involve another double-differential cross section of (2.78), but with somewhat different parameters. An energy spectrum associated with this second scattering will be

associated with every energy loss from the first scattering. Approximately, the energy spectrum for double scattering is the convolution of the single-scattering energy spectrum with itself. A similar type of argument applies to momentum distribution, but this is complicated by the vectorial aspects of the momentum transfer. Today the accepted practice is to ignore multiple scattering by assuming it is a weak background underneath the measured data, and does not contain significant structure. So long as it represents only 5% of the spectral area, this shouldn't be a problem, should it?

Absorption Criterion. When the sample contains a strong absorber of neutrons, the analysis is a bit more complicated than for (5.41). The loss of neutrons from the incident beam is now

$$-\mathrm{d}I(x) = (\mathsf{s}+\mathsf{a})\, I(x)\, \mathrm{d}x \;, \tag{5.54}$$

$$I(x) = I_0\, \mathrm{e}^{-(\mathsf{s}+\mathsf{a})x} \;. \tag{5.55}$$

where the parameter a is the absorption coefficient, causing neutrons to "disappear," rather than scatter. The $I(x)$ in (5.55) is the number of neutrons at depth x into the specimen, but it is not equal to the number of scattered neutrons. Furthermore, the scattered neutrons can be absorbed on their way out of the sample. With a one-dimensional model, for a sample of thickness t, the path out has a length $t-x$, so an exponential function with this argument attenuates the outgoing beam. For the scattered neutron intensity, I_{scat} from the increment $\mathrm{d}x$

$$\mathrm{d}I_{\mathrm{scat}} = I(x)\, \mathrm{e}^{-\mathsf{a}(t-x)}\, \mathsf{s}\, \mathrm{d}x \;, \tag{5.56}$$

Substituting (5.55) into (5.56), rearranging, and integrating $\mathrm{d}I_{\mathrm{scat}}$:

$$I_{\mathrm{scat}} = I_0\, \mathsf{s}\, \mathrm{e}^{-\mathsf{a}t} \int_0^t \mathrm{e}^{-\mathsf{s}x}\, \mathrm{d}x \;, \tag{5.57}$$

$$I_{\mathrm{scat}} = I_0\, \mathrm{e}^{-\mathsf{a}t}\, (1 - \mathrm{e}^{-\mathsf{s}t}) \;. \tag{5.58}$$

We can use (5.58) to calculate the scattering for any combination of absorption coefficient (a) and scattering coefficient (s). Suppose that $\mathsf{a} = 0$, and there is no absorption. For a sample that scatters 10% of the incident neutrons, we obtain (5.42). Incidentally, for such a thin scatterer we can expand the exponential in (5.42) to obtain $I_{\mathrm{scat}} = I_0\, \mathsf{s}\, t$, which is perhaps more intuitive. It is perhaps more interesting to consider the other limit where absorption is strong. A sample that is too thick will have no scattered intensity, since all neutrons are absorbed. There is an optimal thickness t' for maximum scattering that we find by the analysis:

$$\left.\frac{I_{\mathrm{scat}}}{\mathrm{d}t}\right|_{t'} = 0 \tag{5.59}$$

$$t' = \frac{1}{\mathsf{s}}\, \ln\left(\frac{\mathsf{s}}{\mathsf{a}} + 1\right) \;. \tag{5.60}$$

For strong absorbers, $\mathsf{a} \gg \mathsf{s}$, and we obtain from (5.60):

$$t' = \frac{1}{\mathsf{a}} \quad \text{optimal for strong absorbers.} \tag{5.61}$$

Hydrogen Criterion. Finally, we consider the inelastic scattering from hydrogen, a unique element. Phonon scattering cross sections are proportional to σ/m. Hydrogen has a uniquely large σ and an uniquely low m, making it a stronger inelastic scatterer than Zr, by a factor of 1100. A trace of hydrogen in Zr would contribute a large amount of the inelastic scattering, and Zr has a tendency to absorb hydrogen and retain it in a modest vacuum. In many cases the spectral contributions from hydrogen are at high frequencies, and can perhaps be separated from the modes from Zr, for example. Nevertheless, it is important to know the hydrogen concentration in a sample for inelastic scattering experiments, and the best practice is to eliminate hydrogen from the sample.

5.1.12 Sample Design: Worked Example of $LiFePO_4$

Here are some sample calculations that were used for obtaining a thickness and mass of a typical sample for inelastic scattering. The neutronic properties of the elements of lithium iron phosphate are listed in Table 5.1.

Table 5.1. Neutronics of $LiFePO_4$

	units	Li	Fe	P	O	O×4
σ_{scat}	10^{-24} cm^2	1.37	11.62	3.312	4.232	19.93
σ_{abs}	10^{-24} cm^2	70.5	2.56	0.172	0.0	0.0
σ_{scat}/m	10^{-24} cm^2/A	0.196	0.208	0.107	0.265	1.06

The first step is to calculate the mass, M, of a sample that is a 10% scatterer. We do the calculation for $1\,\text{cm}^2$ of sample area, so the total cross section should be $0.1\,\text{cm}^2$

$$\begin{aligned}\sigma_{\text{tot}} = 0.1\,\text{cm}^2 &= [1.37 + 11.62 + 3.31 + 16.93] \\ &\quad \times [10^{-24}\,\text{cm}^2] \left[\frac{1\text{ mole}}{158\,g}\frac{6.02\times 10^{23}\,\text{atoms}}{\text{mole}}\right] M\ . \end{aligned} \tag{5.62}$$

$$M = 0.79\,\text{g}\ . \tag{5.63}$$

To fill the 5×5cm beam of the ARCS instrument, the sample mass should be 19 g.

The second check is for absorption. The total absorption for a $1\,\text{cm}^2$ sample is:

$$\sigma_{\mathrm{abs\,tot}} = [70.5 + 2.56 + 0.176] \times [10^{-24}\,\mathrm{cm}^2] \left[\frac{1\ \mathrm{mole}}{158\,g}\frac{6.02 \times 10^{23}\,\mathrm{atoms}}{\mathrm{mole}}\right] 0.79\,\mathrm{g}\ . \tag{5.64}$$

$$\sigma_{\mathrm{abs\,tot}} = 0.219\ . \tag{5.65}$$

This is small enough to be ignored, but if it were much larger we would have to resize the sample to be consistent with (5.61), for example.

Finally, we consider the risk from some residual water in the sample. One H_2O molecule has a ratio of $\sigma_{\mathrm{scat}}/m = 170$ for its hydrogen atoms. The σ_{scat}/m in Table 5.1 are nearly 1000 times smaller. We should seek a mole fraction of water that is 10^{-4} or less. Perhaps some water could be removed by heating the sample slightly in the vacuum of the ARCS spectrometer.

Through data analysis procedures described in Chapter 6, it is possible to convert a measured inelastic spectrum to a representation of the phonon DOS. The problem of neutron weighting in $LiFePO_4$ can be seen from the last line in Table 5.1 in the values of σ_{scat}/m. For this material, the vibrational modes involving large amplitudes of motion for Li atoms will be much more prominent in the extracted DOS than the modes involving large amplitudes for O atoms. This neutron weight problem is one that can be addressed by lattice dynamics calculations, or at least it should be in the near future.

Further Reading

The contents of the following are described in the Bibliography.

Varley F. Sears: *Neutron Optics*, (Oxford University Press, New York and Oxford 1989).

G. Shirane, S. M. Shapiro, and J. M. Tranquada: *Neutron Scattering with a Triple-Axis Spectrometer*, (Cambridge University Press, Cambridge 2002).

G. L. Squires: *Introduction to the Theory of Thermal Neutron Scattering*, (Dover, Mineola NY 1978).

C. G. Windsor: *Pulsed Neutron Scattering*, (Taylor and Francis, London 1981).

6. Essential Data Processing

This chapter explains procedures for the reduction of data from a time-of-flight inelastic neutron spectrometer. To date the practice of reducing inelastic scattering data from time-of-flight instruments has been poorly-documented, dull, tedious, and error-prone. We hope this chapter transcends these traditional shortcomings.

The present chapter explains the rationale behind the data reduction modules of *DANSE* (Chapter 7 better explains their use). After an overview of a typical data reduction process, the individual steps are explained in detail. Approximately, analysis steps that are further from the raw data are more closely-related to the dynamic processes in the sample, and tend to be more computationally-intensive. These steps (such as corrections for absorption, multiple scattering, and multiphonon scattering) are described later in this chapter. Although the data analysis steps described first are somewhat independent of the previous chapters, understanding some of the later steps (such as corrections for multiphonon scattering) may be easier after reading Chapters 2 and 3.

The treatment in this chapter presupposes a "direct geometry" configuration (monochromation before the sample) and a two-dimensional, pixellated detector system, composed of an array of linear, position-sensitive detectors. This configuration is used in the Pharos and ARCS spectrometers, for example.

The first task is to take a raw set of data from an inelastic neutron scattering experiment (with a powder sample), and convert it into physical units such as meV and Å^{-1} for energy and momentum transfer. Such basic steps are needed before the data can be compared to predictions from theory, or even compared to results from other inelastic spectrometers. Although the specific steps may depend in part on the scientific issues being studied, some data processing steps are common for nearly all experimental work, and are required to make an inelastic neutron scattering data set useful. There may be others steps as well, but most experiments follow most of the steps presented here.

Some notation is listed in Table 6.1. Fig. 6.1 shows the position of the detector tubes with respect to the sample, S, in the plane of the spectrometer, Fig. 6.2 the position perpendicular to the plane.

Table 6.1. Notation

k_i (k_f)	modulus of the initial (final) neutron wavevector
$\boldsymbol{Q} = \boldsymbol{k}_\mathrm{i} - \boldsymbol{k}_\mathrm{f}$	momentum transferred from the neutron to the sample
ϕ	scattering angle ($Q = \sqrt{k_\mathrm{i}^2 + k_\mathrm{f}^2 - 2k_\mathrm{i}k_\mathrm{f}\cos\phi}$)
E_i (E_f)	incident (final) neutron energy
$\hbar\omega = \Delta E = E_\mathrm{i} - E_\mathrm{f}$	Energy transferred from neutron to sample
l_2	distance from the sample to the center of a detector
d	labels a specific detector
h	the height of a pixel in a detector

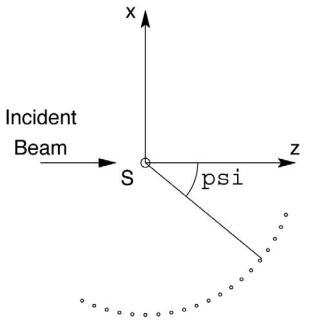


Fig. 6.1. Layout of the secondary flight path of a direct geometry time-of-flight spectrometer. The detectors are arrayed in a circle of radius l_2 around the sample (at S). The beam is incident on the sample from the left. The x and z axes show the coordinate system used in the discussion of binning into rings (Sect. 6.1.2), the y-axis of that system points out of the page. ψ is the angle from the z-axis to the center of a given detector (in the x-z plane).

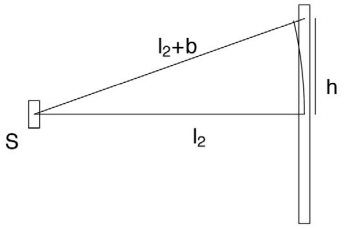


Fig. 6.2. Labeling some of the distances in the secondary flight path of a direct geometry time-of-flight spectrometer. The center of the detector is l_2 from the sample (at S), and a given pixel is at height h from the center. The total distance from the sample to a pixel is $l_2 + b(h) = (l_2^2 + h^2)^{1/2}$.

Typically one wants to know the dynamic structure factor, or scattering law, $S(Q,\omega)$. The coherent nuclear scattering of (3.94) can be summarized in simplified form:

$$\left(\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\mathrm{d}E}\right)_\mathrm{coh} = \frac{\sigma_\mathrm{coh}}{4\pi}\frac{k_\mathrm{f}}{k_\mathrm{i}} N S_\mathrm{coh}(\boldsymbol{Q},\omega)\ , \tag{6.1}$$

with similar expressions for incoherent scattering or magnetic scattering. Owing to several complications originating with how the measurement is made, the measured data are actually:

$$\left(\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\mathrm{d}t}\right)_{\mathrm{coh}} = \Phi\Big[\frac{\sigma_{\mathrm{coh}}}{4\pi}\frac{k_{\mathrm{f}}}{k_{\mathrm{i}}}NS_{\mathrm{coh}}(d,h,t)\times T(d,h,t)\times\epsilon(d,t) + S_{\mathrm{nuis}}(d,h,t)+b(d,h,t)\Big]\,. \tag{6.2}$$

Here Φ is the incident flux, and often must be known by an independent measurement with a beam monitor, for example. The data are initially histogrammed as a function of neutron time-of-flight (TOF) rather than neutron energy,[1] where the two are related by $E = 1/2m_nv^2 = 5.227(l/t)^2$ (the second equality holds for speeds in mm/μs or equivalent). Background b and nuisance scattering from the sample, S_{nuis}, pollute the interesting signal.[2]

Other factors modify the interesting term involving S_{coh}. The sample may absorb neutrons, so only a fraction T of neutrons are transmitted through the sample, and this varies with scattering angle and neutron energy. The efficiency ϵ of the detectors varies with energy and from one detector to another. The jobs ahead are:

- Account for incident flux.
- Remove background.
- Compute incident energy.
- Convert from time to energy.
- Correct for detector efficiency.
- Correct for absorption.
- Subtract similarly treated empty can data.
- Bin into rings of constant scattering angle ϕ.
- Subtract additional nuisance scattering (multiple scattering and so on).
- Convert from angle to momentum.

Errors must be propagated. This is usually done according to a few simple rules, such as when data sets are added or subtracted, the errors add in quadrature; the error for the raw histogram is the square root of the counts in any given bin [81].

While many experiments could be treated by a similar process, not all of them can. In the later case, we have to know when departures from the routine require intervention in the reduction process. The sequence in which these tasks are presented here is that in which they are usually performed. As a rule, these operations do not commute, and we have to figure out what needs to be adjusted when re-ordering things. For instance, if the background is known as a function of time, either it should be subtracted before the data are

[1] The introduction of new data acquisition hardware and software might alleviate this by histogramming directly into energy and momentum. That may simplify some tasks, for example eliminating the conversions from TOF to energy and from TOF and angle to momentum. However, this may complicate other tasks such as background subtraction.

[2] This list includes anything that does not interest the experimenter. For example, in a phonon measurement this may include all magnetic scattering, multiphonon and multiple scattering effects.

transformed into energy, or the background must be transformed into energy as well. The sequence for other tasks, such as conversion to momentum units, is motivated by efficiency (one needs to think about energy when thinking about momentum, but not vice versa, so one might as well do the energy conversion first, followed by the momentum conversion).

6.1 Steps to Transforming Data into a Function of Energy and Momentum

6.1.1 Operations and Data Structures

The basic data structure is a histogram. This implies sets of arrays: the array to store the histogram values *per se*, a similar array for errors, and associated arrays to store the axis values for each dimension.

Operations can be divided into those that change the structure of some of the arrays, and those that change only the contents of the arrays. Those that change the structure include energy rebin (§ 6.1.2), sum into rings (§ 6.1.2), and momentum rebin (§ 6.1.2); those that don't change the array structures include subtracting various artifacts (§ 6.1.2, 6.1.2) and multiplicative corrections (§ 6.1.2, 6.1.2). The structure-changing operations require the creation of one or more new arrays; the rest can be done in place (assuming one is confident of the correction or if the correction can be easily undone). In the sections ahead, changes to the structure of the data will be noted.

6.1.2 A Closer Look at Each Task

Initial Data. The starting point is an array representing $I_0(d, h, t)$, the raw counts in each detector, at each detector position, for each TOF bin. The error is assumed to be the square root of the intensity: $\sigma_0 = \sqrt{I_0(d, h, t)}$. The TOF bins in this histogram usually have a constant time width, so all the bins are specified by some initial time t_0, the bin-width Δt, and the number N of bins.

Initially there are five major arrays representing:

- The actual data. Three dimensional array; if there are T time bins, D detectors, and H positions per tube, the array dimensions are $D \times H \times T$.
- The error. Same dimensions as the data.
- The time of flight. A one dimensional array of length T.
- The detector angles. A one dimensional array of length D.
- The pixel heights. A one dimensional array,[3] length H.

[3] One might allow for the possibility that the pixel heights vary from one tube to another. The pixel position array becomes a two dimensional array, with size $D \times H$.

Those arrays are stored in a histogram.

Some additional parameters are needed. These include the incident neutron energy, E_i, and the geometry of the instrument, especially the length l_2 from the sample to each detector, and the angle of each detector. We assume these additional parameters are known.

Normalize by Incident Flux. Typically one divides by the total counts m in a peak in a beam monitor, which gives a proportional measure of the incident flux. Other measures may also be considered in unusual circumstances, for example, microamp-hours of proton beam delivered to the target. Regardless of the source, one still divides each element of the input array by some scalar:

$$I_1(d_j, h_k, t_i) = I_0(d_j, h_k, t_i)/m \ , \tag{6.3}$$

for all $\{i, j, k\}$, and similarly for the error.

Subtract Background. Physically, the background is another complex subject; computationally, it is not. It can have different origins, from cosmic rays hitting the detectors (probably a small contribution) and neutrons from other experiments, to more serious causes, such as fast neutrons thermalizing in the instrument shielding, or slow neutrons scattering from poorly-masked beamline components. At times it can be difficult to determine what is background and what isn't, especially when looking for diffuse scattering.

A common way to estimate the background is to subtract a constant (in time) background determined from the data. This may have some justification for removing the background that originates with radiations from neighboring instruments, or neutrons from far enough away that any time structure is lost. Another approach subtracts the scattering measured with an empty can. This is better done at a later stage, and is described below.

Because the background could vary with time and detector position, we denote it as $b(d, h, t)$. Thus, we have

$$I_2(d_j, h_k, t_i) = I_1(d_j, h_k, t_i) - b(d_j, h_k, t_i) \ . \tag{6.4}$$

For each element of I_2 one must know about one element of I_1 and one element of b, and the same for the error.

Convert from Time to Energy. This step changes the structure of the data arrays. Prior to this step, the structures are the same as the initial arrays. There are still some more corrections to be made, but those corrections can be done more easily in terms of energy. This is computationally more intricate than the previous steps. In fact, it may be done in two steps.

First, one creates a new set of histogram bins whose boundaries are determined by t_2, the sample-to-detector TOF, and l_2, the distance from sample to detector, via the classical relation

$$\hbar\omega = E_\mathrm{i} - E_\mathrm{f} = 5.227\Big[v_\mathrm{i}^2 - \Big(\frac{l_2}{t_2}\Big)^2\Big] \ . \tag{6.5}$$

Note that pixels at different positions along the length of the detector tube are at different distances from the sample.[4] The effect is to coarsen the energy resolution. An extra 3 cm is a little more than the diameter of the detector tube, so this makes a contribution to the resolution similar to the size of the detector, or the sample size. For 35 meV neutrons and $l_2 = 4$ m the difference in energy is about 1.5%. One way to account for this is to write a separate set of histogram energy bin boundaries for each height in the detector tube:

$$\hbar\omega = E_\mathrm{i} - E_\mathrm{f} = 5.227\Big[v_\mathrm{i}^2 - \Big(\frac{l_2^2 + h^2}{t_2^2}\Big)\Big] \,. \tag{6.6}$$

In making this transition, we need to multiply by the time bin-width and divide by the energy bin-width:[5]

$$I_3(d, \tilde{h}, \omega(h)) = I_2(d, h, t)\Big|\frac{dt}{d\omega}\Big| \,. \tag{6.7}$$

At this step, the array describing the time bins has been replaced by a two-dimensional array describing the energy bins. Here is a list of the arrays at this stage of data reduction:

- Data array. Still $D \times H \times T$.
- Error array. Still $D \times H \times T$.
- Intermediate energy bin boundaries array. Two dimensional, $H \times T$ or so.
- Detector angles array, one-dimensional, length H. Same as before.
- Pixel position array. Same as before.

The energy binwidth is changing from bin to bin, which makes it hard to think about in plotting, fitting, etc. Also, the energy of any given bin is a function of the position of the pixel height. This is why we usually take the second step, rebinning into constant energy bins. Rebinning will also remove the artifact of each detector position having a unique set of energy bin boundaries.

For the second step one creates another set of histogram bin boundaries, spaced uniformly in energy. One then assigns counts from the old bins to the new bins by prorating them. The rules are simple: if an old bin lies entirely within a new bin, one puts all of the old counts into the new bin; in doing so, multiply by the old bin width and divide by the new bin width. If an old bin overlaps two new bins, assign counts to the first new bin based on the fraction of old bin overlapped by the first, and assign counts to the second new bin according to the fraction of the old bin overlapped by the second bin (and multiply by the old bin width while dividing by the new). The same

[4] For a nominal l_2 of 4 m (Pharos), this discrepancy gets as large as about 0.03 m (generally, the extra path length is $b = l_2[(1 + h^2/l_2^2)^{1/2} - 1] \approx h^2/2l_2^2$).

[5] This is because the histogram reflects counts per some unit (μs or meV); we have the constraint that the *integral* of the cross section be independent of the unit. If the histogram contains counts instead of counter-per-some-unit, then this factor is not necessary.

holds true if the old bin overlaps many new bins. One can of course think of this the other way around: each new bin asks "what does each old bin owe me".

One could write:

$$I_3(d,h,\omega) = \sum_i \mathbf{M}(\omega,\tilde{\omega}_i(h)) \times I_3(d,h,\tilde{\omega}_i(h)) \times \left|\frac{d\tilde{\omega}_i}{d\omega}\right| , \tag{6.8}$$

where $\mathbf{M}(\omega,\tilde{\omega}(h))$ is a matrix with the overlaps between old bins and the bins. One could get clever and drop the $|dt/d\tilde{\omega}|$ in Eq. 6.7 if one remembers to use $|dt/d\omega|$ instead of $|d\tilde{\omega}/d\omega|$ in Eq. 6.8. One typically needs to know something about several (but not all) elements of the old array before he can learn something about one element of the new array.

At the end of this step, three arrays have changed:[6]

- Data array. Now $D \times H \times N_E$.
- Error array. Now $D \times H \times N_E$.
- Energy bin values array. One dimensional, N_E.
- Detector angles array, one-dimensional, length H. Same as before.
- Pixel position array. Same as before.

Detector Efficiency. Detector efficiency varies with energy. These data are provided by the manufacturer. One simply divides by a pre-existing array of numbers:

$$I_4(d_j,h_k,\omega) = \frac{I_3(d_j,h_k,\omega)}{\epsilon(d_j,\omega)} . \tag{6.9}$$

The efficiencies are typically known as a function of final neutron energy, so an additional step is required to convert $\epsilon(E_\mathrm{f})$ into $\epsilon(\omega)$.

Absorption correction. The absorption by the sample varies with both neutron energy and mean path length through the sample. For a single energy, the transmission through a material with unit cell volume V and absorption cross section per unit cell σ_{abs}, the transmission probability is $T = \mathrm{e}^{-l\sigma_{abs}/V}$. For a wise choice of units for V and σ_{abs}, the path length l will be in cm. The mean path length through the sample varies with angle, so one needs to compute $T(\phi,\omega)$.

In a real experiment, the length through the sample depends on where in the sample the neutron scatters, to which direction it scatters, and the change in the neutron energy (recall that the absorption cross section per unit cell varies as inversely with neutron speed, so when the speed changes, the probability of absorption per unit length changes). It does not depend on momentum transfer. One must calculate this dependence; let us assume this has been done. Then one divides each element of the old array by one element the array representing $T(\phi,\omega)$

[6] N_E is the number of new energy bins

$$I_7(\phi_i, \omega_j) = I_6(\phi_i, \omega_j)/T(\phi_i, \omega_j) \; . \tag{6.10}$$

Section 6.3 of this chapter takes a more detailed look at the absorption correction.

Subtract empty can data. Having accounted for absorption by the sample, it now makes sense to subtract the scattering measured from the empty can, if the data from the empty can be brought through a similar treatment chain to this stage[7].

Bin into Rings. For a powder sample, the useful spatial information is the scattering angle ϕ. That is to say, for a given Q, E_i, and ω, a powder scatters into a cone of angle ϕ . So the two labels d and h can be condensed into the scattering angle ϕ.

The first step is to identify the ϕ for each pixel. A conventional instrument coordinate system has the sample at the origin and the transmitted beam forming the z-axis. The x-axis runs horizontally from the sample (for Pharos, that's toward FP-15) and perpendicular to the beam, and the y-axis, determined by right-handedness, points upward. (See Fig. 6.1.) It is natural to think of the detector as lying on a cylinder, with the axis of the cylinder coincident with the y-axis just described. So consider two vectors: $\boldsymbol{p}$, which runs from the sample to the pixel, and $\boldsymbol{a}$, with length l_2 running from the sample in the direction of the transmitted beam. In the instrument coordinate system, $\boldsymbol{a} = l_2\hat{z}$, and $\boldsymbol{p} = -l_2\sin(\psi)\hat{x} + h\hat{y} + l_2\cos(\psi)\hat{z}$. ψ is the angle between the tranmitted beam and the center of the detector; it lies in the x-z plane. The angle between $\boldsymbol{a}$ and $\boldsymbol{p}$ is given by $\boldsymbol{p}\cdot\boldsymbol{a} = |a||p|\cos\phi = a_z p_z$:

$$\begin{aligned}\phi &= \cos^{-1}\left(\frac{a_z p_z}{|a||p|}\right) = \cos^{-1}\left(\frac{p_z}{|p|}\right) = \cos^{-1}\left(\frac{l_2\cos(\psi)}{\sqrt{l_2^2+h^2}}\right) \\ &= \cos^{-1}\left(\frac{\cos(\psi)}{\sqrt{1+h^2/l_2^2}}\right) \; . \end{aligned} \tag{6.11}$$

Having found the angle ϕ for each pixel, an efficient method to combine pixels with similar angles is desirable. The simplest method uses three nested `for` loops: one over the new angles, and two over the old detector indices. More abstractly:

$$I_5(\phi_k, \omega) = \sum_{i,j} \mathbf{N}(\phi_k, d_i, h_j) \times I_4(d_i, h_j, \omega) \; , \tag{6.12}$$

where $\mathbf{N}$ is a matrix whose elements are 1 or 0 depending on whether ϕ for the i^{th}, j^{th} pixel is within the bounds defined for the k^{th} element of the new array. It may not be necessary to know all of the points of the old array in order to find one element of the new array.

This operation depends on the instrument configuration (where the detectors are) and the set of angular rings which can be defined by users. After

[7] Not an identical chain: hopefully your empty can didn't need an absorption correction. (If it was made out of vanadium, it might need that!)

this step, all but one of the arrays have changed. Here is a list of the arrays after this step of data reduction:

- Data array: $N_\phi \times N_E$.
- Error array: $N_\phi \times N_E$.
- Energy bin values array: N_E.
- Phi-bin values array, N_ϕ

Subtract Nuisance Scattering. Other nuisance scattering includes all scattering from the sample that is either non-interesting or confusing. This usually includes higher-order scattering processes, and is generally difficult to compute (especially since one has now applied a number of multiplicative factors which must be accounted for). Computationally, this is simply a matter of subtracting two arrays, element-by-element, which brings us to:

$$I_6(\phi, \omega) = I_5(\phi, \omega) - S_{\text{nuis}}(Q, \omega) \tag{6.13}$$

Convert to Momentum Transfer. Ideally, there would be a prorating scheme like that used in going between time and energy. Another approach frequently used is interpolation. Essentially one recognizes that the regularly-spaced angles at which one has measured are an irregularly-spaced array of momentum transfers, Q. This momentum rebinning is typically performed with a convenient interpolation algorithm in, for example, IDL. Like the energy rebinning, one could write this process as a matrix operator acting on a vector:

$$S(Q, \omega_j) = \sum_i \mathbf{M}(Q, \phi_i, \omega_j) I_7(\phi_i, \omega_j) \; . \tag{6.14}$$

The arrays have changed again. After this step in data reduction, the arrays are:

- Data array: 2D, $N_E \times N_q$.
- Error array: 2D, $N_E \times N_q$.
- Energy bin values array: 1D, N_E.
- Q-bin values array: 1D, N_q.

At this point we have isolated the scattering of interest, and converted it into real physical units. This was considered a real achievement in the 1990's. For some experimental work the data analysis can be declared complete, and graphs of the results prepared for publication.

6.2 Transformations and Information

Information is lost in the course of data reduction – the word "reduction" itself implies a loss. Some loss of information is both inevitable and necessary

with data from a direct geometry chopper spectrometer – with 10^5 detector pixels of 10^4 time bins each, a data set of 10^9 elements defies human comprehension. As described in the previous sections, the full experimental information on the arrival of each neutron at a specific detector pixel at a specific time is put through a number of transformations to improve comprehension, but also to improve counting statistics by summing events that are expected to be physically equivalent. It should also be noted that many data elements are usually uninteresting, or hold minimal counts, but even this information can be useful for removing extraneous scattering in the "good" data elements.

This section discusses the different physical scattering processes that can be measured experimentally, the possible transformations that can be performed on them, and whether these transformations are warranted, given the information required to do the transformation properly. For example, if the Q-resolution is lost by doing experiments on an incoherent scatterer like hydrogen, by performing measurements on a filter-difference spectrometer, or by rebinning all the Q-dependent data into energy, there is no path back to Q information by direct analysis of data alone. Fortunately, there may be a route back to the Q information through the use of a theoretical model or a simulation, and this is an opportunity provided by `DANSE` software.

Theory and computation play a bigger role when the experimental data require more supplementary information to enable the transformation. For data containing minimal information, it is possible to obtain detailed elementary excitations with theory alone – this is of course what has been done for many years by computational condensed-matter physicists. This dominance of theory over experiment was not the original intent of `DANSE`, but since `DANSE` provides such tools it will be interesting to see the path taken by future users of the `DANSE` system. Some community standards will have to be considered, but it is already clear that some transformations are better described as computational science rather than experimental science, even if experimental data is part of the effort. For example, consider data on elastic incoherent scattering from a polycrystalline sample. Very little information is contained in such data on excitations in solids, but these data could be augmented by ab-initio theory to make predictions of the inelastic coherent scattering from a monocrystal. In this example most of the work would be done by theory, and the experimental data would add little value. For this reason, the `DANSE` software does not support this transformation.[8] The present section lists all possible data transformations, and explains why approximately half of them are supported by `DANSE`.

[8] Such analysis is possible by the ab-initio tools in `DANSE`, but `DANSE` does not offer this as a transformation of experimental data.

6.2.1 Categorization of Transformations and Information

Table 6.2 lists the fundamental possibilities for a neutron scattering measurement. A real experiment may include combinations of coherent and inelastic scattering processes, but we consider these independently in what follows, because this is the approach used in data analysis. The categories of energy, momentum, and sample are orthogonal, and make the natural diagram of Fig. 6.3. Figure 6.3 is structured so that it the data of maximum detail is near the origin in the box labeled "inelastic coherent mono." (Note: with the word "inelastic," elastic scattering is also assumed included.) This inner cube contains enough information so that reprocessing allows one to move in any of the other three directions using experimental data alone (assuming the data are complete, of course).

As examples, consider the reprocessing of data from the inner box labeled "inelastic coherent mono":

1. Averaging the scattering over all crystal orientations (from $\boldsymbol{Q}$ to Q) allows reprocessing into the box above it, "inelastic coherent poly,"
2. Averaging over Q allows transformation to incoherent scattering, to the box in back, "inelastic incoherent mono,"
3. Discarding the inelastic contribution moves to the box at right "elastic coherent mono," which is useful for single-crystal diffraction patterns.

These three statements are discussed further below, as are other transformation paths. Some are less obvious than others – statement 2 above is less obvious than 1 and 3, and it has been discussed in the literature. Nevertheless, it should be almost intuitive that considerable information is lost in going to the remote cube, "elastic incoherent poly." Returning from such data to the origin is impossible.

Table 6.2. Scattering Processes

Energy	Momentum	Sample
inelastic	coherent	monocrystal
elastic	incoherent	polycrystal

6.2.2 Coherent – Incoherent

Forward. An energy spectrum can be obtained by integrating over the Q coordinate of $S(Q, E)$. With some care to account for thermal factors, multiple scattering, and multiphonon scattering, often before integrating over Q, this can be converted to a phonon energy spectrum or a density of states. This approach is rigorous if all values of Q are accounted for, even if the

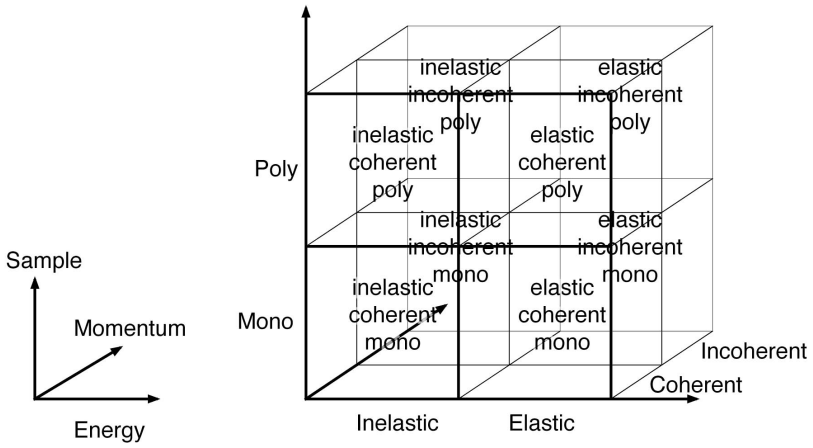


Fig. 6.3. Eight possibilities for scattering processes in a neutron experiment. Each of the eight can be calculated independently in a simulation or modeling process. Some can be obtained from another by transformation. In general, information is lost as one moves away from the origin, as explained in the text.

sample is a coherent scatterer with sharp dispersion information. An integration over all Q accounts for all phonons, at least in principle. Some of the issues concerning the incoherent averaging over coherent scattering have been discussed in the literature:

- V.S. Oskotskii, "Measurement of the Phonon Distribution Function in Polycrystalline Mateirals using Coherent Scattering of Slow Neutrons into a Solid Angle" Sov. Phys. Solid State **9**, 420 (1967).
- F. de Wette and A. Rahman, "Inelastic Scattering of Neutrons by Polycrystals" Phys. Rev. **176**, 784 (1968).
- M.M. Bredov, B.A. Kotov, N.M. Okuneva, V.S. Oskotskii and A. L. Shakh-Budagov, "Possibility of Measuring the Thermal Vibration Spectrum $g(\omega)$ using Coherent Inelastic Neutron Scattering from a Polycrystalline Sample," Sov. Phys. Solid State **9**, 214 (1967).

Nevertheless, the forward transformation from "inelastic coherent" to "inelastic incoherent" can be done reliably, especially for data from direct geometry chopper spectrometers that provide a wide range of Q. This transformation is offered by the `DANSE` software.

It should be mentioned that for many years, phonon DOS information has been obtained from phonon dispersion curves measured in special crystallographic directions on a triple-axis spectrometer. These dispersions are not good averages over Q, and energies are not measured for all phonons in the solid. Nevertheless, after fitting the dispersion curves to a lattice dynamics model, the energies of all other phonons can be calculated. This approach has never been challenged for measurements on pure elements – it seems to

work well because the high amount of information in the dispersion curves cause the interatomic force constants to be well determined. This method is not appropriate for disordered alloys, at least when the virtual crystal approximation is employed.

Reverse. In the same sense that it takes a lattice dynamics model to go from coherent inelastic scattering data from a triple-axis instrument to a good sampling of the incoherent inelastic scattering over all phonons, a model is always required for the reverse transformation. The positions of the Van Hove singularities in the phonon DOS are effective in constraining the dispersion curves, but there are only three of them in a phonon DOS of a pure element. It is not yet clear how reliably the reverse transformation can be performed, but it is certainly a useful capability that is offered in the `DANSE` software.

6.2.3 Monocrystal – Polycrystal

Forward. Transforming a complete data set from a monocrystalline sample to a polycrystalline average is simple in principle, and has the most physical interpretation of all the transformations in this section. In essence, the single crystal data need to be averaged over all crystallographic orientations by "rotating" the monocrystalline data to produce a polycrystalline average. This amounts to transforming $\boldsymbol{Q}$ to Q. The results can be similar to the transformation of coherent to incoherent scattering, and in fact proves identical when the crystals have isotropic scattering. `DANSE` supports these transformations.

Reverse. Except in special cases, the reverse transformation from polycrystal data to monocrystal data requires a model. The question of importance is how reliably can the polycrystalline average of, for example phonon dispersions, be used to define the single crystal phonon dispersions along specific crystallographic directions. The answer is not fully known today, but in the case of crystals such as tungsten, which are elastically isotropic, it is expected that the amount of information is essentially the same for data from polycrystalline and monocrystalline scattering experiments. In the case of anisotropic crystals it is expected that the dispersive information in $S(Q, E)$ will have intensity variations from anisotropic effects, and perhaps important information can be obtained from the intensity variations within the dispersion curves if the data are of good statistical quality. `DANSE` supports these transformations, using both phenomenological and ab-initio models. It is hoped that the neutron scattering community will eagerly assess these capabilities of `DANSE`, because they free the experimenter from acquiring single crystals.

6.2.4 Inelastic – Elastic

Forward. A good diffraction pattern should be possible to acquire using white-beam mode on a direct-geometry chopper spectrometer, and the interpretation of the data would be the same as for a total scattering diffractometer. Here, however, we consider operation with a monochromatic beam

as in the normal operation of a direct-geometry chopper spectrometer such as ARCS. The coherent elastic scattering is measured as part of the spectrum, so ignoring the scattering with energy transfers greater than the elastic peak width is essentially the same as performing a diffraction experiment. There are some defects in this approach – the spread of incident wavelengths broadens the diffraction peaks in Q. Nevertheless, useful information about the sample can often be obtained by simply ignoring the inelastic part of the data. Transformations to do this are provided by `DANSE`.

Reverse. Taking elastic data and transforming it to inelastic data is not impossible. Unfortunately, direct transformations are not robust. Robust approaches require substantial additional information. Phonon effects are evident in pair distribution functions, but care must be taken to separate these from atomic size effect diffuse scattering, which can be of the same order. There have been some efforts to obtain phonon dispersions from diffraction patterns, and the reader is welcome to explore these:

- D.A. Dimitrov, D. Louca, H. Roder, "Phonons from neutron powder diffraction," Phys. Rev. B **60**, 6204 (1999).
- W. Reichardt, L. Pintschovius, "Influence of phonons on the pair distribution function deduced from neutron powder diffraction," Phys. Rev. B **63**, 174302 (2001).
- A.L. Goodwin, M.G. Tucker, M.T. Dove, D.A. Keen, "Phonons from powder diffraction: A quantitative model-independent evaluation," Phys. Rev Lett. **93** 075502 (2004), *Ibid.* **95**, 119901 (2005).

At the inception of the `DANSE` project, the methods in these reference were placed outside the scope of the effort.

Nevertheless, `DANSE` does offer methods for reverse transformation. These are robust in that they give stable answers, but they require substantial computational effort. The approach is to obtain the lattice dynamics for the structure. From the lattice dynamics the inelastic scattering can be obtained. If the crystal structure or molecular structure is obtained from a diffraction measurement, for example, phonon dynamics can be calculated in one of two ways.

- A model of the lattice dynamics for the structure can be used to calculate the phonons. This requires input on interatomic force constants. They may be guessed, or obtained from other sources. From the dynamics, the inelastic scattering can then be calculated.
- Using the unit cell obtained from a diffraction pattern, or a molecular structure obtained by other means, an ab-initio electronic structure calculation can be performed to optimize the positions of the atoms in the structure. These may not be the real atom positions, but relaxed positions are needed for the following computational step. Next, the atom positions are displaced by a controlling software package such as PHON, and the ab-initio code is run again for the distorted unit cell. This permits calculation

of tensorial force constants that are used in the lattice dynamics. From the dynamics, the inelastic scattering can be calculated.

Without such theoretical input, however, attempting a reverse transformation from elastic to inelastic would not be robust, and with the DANSE software it is an error to do so. Note that the transformation of structural information into inelastic inelastic scattering information requires perhaps the most sophisticated tools offered by DANSE. An increasing loss of experimental information requires an compensating sophistication in the theoretical tools to reverse the situation.

6.2.5 All Specific Cases

The previous subsections discussed general issues when transforming data forward or backwards between elastic and inelastic, coherent and incoherent, and monocrystal and polycrystal. For each of these three cases, and six general transformations, there are four specific cases. For example, Fig. 6.3 shows that between the monocrystal and polycrystal samples (up in the graph), there are a total of eight forward and reverse transformations:

- inelastic coherent (mono$\leftrightarrows$poly),
- inelastic incoherent (mono$\leftrightarrows$poly),
- elastic coherent (mono$\leftrightarrows$poly),
- elastic incoherent (mono$\leftrightarrows$poly).

Similar forward and reverse transformations between adjacent cubes are also possible along the elastic$\leftrightarrows$inelastic and coherent$\leftrightarrows$incoherent directions of Fig. 6.3, giving a total of 24 such transformations between adjacent cubes in the figure (along the $\langle 100 \rangle$ directions). In addition, a total of 12 transformations are possible between scattering data in cubes in Fig. 6.3 that are not immediately adjacent (along the $\langle 110 \rangle$ directions), and a total of 8 possibilities for transformations between cubes located diagonally in the figure (along $\langle 111 \rangle$ directions). Tables 6.3 – 6.5 list all 44 possible transformations, and identify the specific transformations supported by the DANSE software. As of early 2007, the plan for DANSE supports 24 of these possible transformations.

Sections 6.2.2 - 6.2.4 provided general reasons for the why transformations along the main axes of Fig. 6.3 are not provided by DANSE. Eight transformations are possible along each of these axes, however. Nearly all cases in Table 6.3 can be understood with the rules of Sects. 6.2.2 - 6.2.4, with the following exceptions:

- Transformations from elastic coherent polycrystal data (powder diffraction patterns) to elastic coherent monocrystal data (single crystal diffraction data) are not supported in the rebinning operation in the first release of DANSE. These capabilities are possible with Rietveld refinement methods

(or PDF modeling), which are available as methods outside the scattering kernel.

- Elastic incoherent scattering pertains to the isotopic incoherence of neutron scattering, for example, and not to incoherence from atomic disorder (which is treated in methods for elastic coherent scattering). `DANSE` therefore does not support well transformations involving elastic incoherent scattering from either monocrystals or polycrystals.

The more complex transformations of two or three data characteristics, listed in Tables 6.4 and 6.5, are generally consistent with the previous rules and individual transformations. More transformations are possible by chaining individual transformations from Table 6.3 than are listed in these Tables 6.4 and 6.5. These latter tables list algorithms that do both transformations in an integrated way. The emphasis is on data from inelastic scattering, since this field was first to exploit this method of direct experiment simulation.

Table 6.3. `DANSE` transformations involving a change in one data characteristic

Scattering Data	Transf.	Scattering Data	Support?	
inelastic coherent *monocrystal*	→	inelastic coherent *polycrystal*	Yes	rotational rebin
inelastic coherent *monocrystal*	←	inelastic coherent *polycrystal*	Yes	B-vK model invert
inelastic incoherent *monocrystal*	→	inelastic incoherent *polycrystal*	Yes	isotropic rescale, rotational rebin
inelastic incoherent *monocrystal*	←	inelastic incoherent *polycrystal*	Yes	isotropic rescale, B-vK model invert
elastic coherent *monocrystal*	→	elastic coherent *polycrystal*	Yes	rotational rebin
elastic coherent *monocrystal*	←	elastic coherent *polycrystal*	No	none
elastic incoherent *monocrystal*	→	elastic incoherent *polycrystal*	Yes	isotropic rescale
elastic incoherent *monocrystal*	←	elastic incoherent *polycrystal*	No	none
inelastic *coherent* monocrystal	→	inelastic *incoherent* monocrystal	Yes	Q rebin
inelastic *coherent* monocrystal	←	inelastic *incoherent* monocrystal	Yes	B-vK model invert
inelastic *coherent* polycrystal	→	inelastic *incoherent* polycrystal	Yes	Q rebin
inelastic *coherent* polycrystal	←	inelastic *incoherent* polycrystal	Yes	B-vK model invert
elastic *coherent* monocrystal	→	elastic *incoherent* monocrystal	Yes	Q rebin
elastic *coherent* monocrystal	←	elastic *incoherent* monocrystal	No	none
elastic *coherent* polycrystal	→	elastic *incoherent* polycrystal	Yes	Q rebin
elastic *coherent* polycrystal	←	elastic *incoherent* polycrystal	No	none
inelastic coherent monocrystal	→	*elastic* coherent monocrystal	Yes	energy slice, intensity threshold
inelastic coherent monocrystal	←	*elastic* coherent monocrystal	No	none
inelastic coherent polycrystal	→	*elastic* coherent polycrystal	Yes	energy slice, intensity threshold
inelastic coherent polycrystal	←	*elastic* coherent polycrystal	No	none
inelastic incoherent monocrystal	→	*elastic* incoherent monocrystal	Yes	energy slice, intensity threshold
inelastic incoherent monocrystal	←	*elastic* incoherent monocrystal	No	none
inelastic incoherent polycrystal	→	*elastic* incoherent polycrystal	Yes	energy slice, intensity threshold
inelastic incoherent polycrystal	←	*elastic* incoherent polycrystal	No	none

Table 6.4. `DANSE` transformations involving changes in two data characteristics

Scattering Data	Transf.	Scattering Data	Support?
inelastic *coherent monocrystal*	→	inelastic *incoherent polycrystal*	Yes
inelastic *coherent monocrystal*	←	inelastic *incoherent polycrystal*	Yes
elastic *coherent monocrystal*	→	elastic *incoherent polycrystal*	Yes
elastic *coherent monocrystal*	←	elastic *incoherent polycrystal*	No
inelastic coherent *monocrystal*	→	*elastic* coherent *polycrystal*	Yes
inelastic coherent *monocrystal*	←	*elastic* coherent *polycrystal*	No
inelastic incoherent *monocrystal*	→	*elastic* incoherent *polycrystal*	No
inelastic incoherent *monocrystal*	←	*elastic* incoherent *polycrystal*	No
inelastic coherent monocrystal	→	*elastic incoherent* monocrystal	Yes
inelastic coherent monocrystal	←	*elastic incoherent* monocrystal	No
inelastic coherent polycrystal	→	*elastic incoherent* polycrystal	Yes
inelastic coherent polycrystal	←	*elastic incoherent* polycrystal	No

Table 6.5. `DANSE` transformations involving changes in three data characteristics

Scattering Data	Transf.	Scattering Data	Support?
inelastic coherent monocrystal	→	*elastic incoherent polycrystal*	Yes
inelastic coherent monocrystal	←	*elastic incoherent polycrystal*	No
inelastic coherent polycrystal	→	*elastic incoherent monocrystal*	Yes
inelastic coherent polycrystal	←	*elastic incoherent monocrystal*	No
elastic coherent monocrystal	→	*inelastic incoherent polycrystal*	No
elastic coherent monocrystal	←	*inelastic incoherent polycrystal*	No
elastic coherent polycrystal	→	*inelastic incoherent monocrystal*	No
elastic coherent polycrystal	←	*inelastic incoherent monocrystal*	No

6.3 Absorption

With an absorbing sample, one observes $\mathrm{d}^2\sigma/\mathrm{d}\Omega\mathrm{d}\omega \times T(\Omega,\omega)$ rather than just $\mathrm{d}^2\sigma/\mathrm{d}\Omega\mathrm{d}\omega$, where $T(\Omega,\omega)$ is the probability of a neutron being transmitted through the sample in a given direction (described by the solid angle Ω) and with a given final energy (final energy = incident energy − energy transfer, or $E_f = E_i - \hbar\omega$). All that needs to be done is to calculate the transmission for each final energy and for each detector, and then divide the observed scattering by the transmission probability. For an experiment in which all the observed neutrons scatter once, the calculation is straightforward, though somewhat computationally demanding.

We suppose that the sample is divided into a number of cells of equal volume; then a neutron observed in any given detector with a given final energy was scattered in any cell with equal probability. We compute the transmission probability for each scattering cell, then average over probabilities for all cells. The reason we have to be so careful is that the transmission probability depends on the distance the neutron travelled through the sample, and this depends on where in the sample the scattering occurred, and the outgoing direction of the neutron[9].

For a neutron travelling across an absorbing medium, the probability of transmission is [10]

$$T = \frac{N}{N_0} = \exp(-l\sigma_{abs}\rho). \tag{6.15}$$

Here, l is the length of the path through the sample, σ_a is the absorption cross section, and ρ is the number density of absorbers. If the neutron changes direction (scatters) while travelling through the sample, the probability becomes

$$T = \exp(-l_{inc}\sigma_{abs}\rho - l_f\sigma_{abs}\rho) \tag{6.16}$$

where l_{inc} is the path length from where the neutron enters the sample to the scattering spot, and l_f is the path length from the scattering spot to where the neutron exits the sample. There's one more difficulty: the absorption cross section depends on the neutron energy, so if the scattering is inelastic, there will be two absorption cross sections, $\sigma(\omega_{inc})$ and $\sigma(\omega_f)$, and we have the transmission probability as

$$T = \exp(-l_{inc}\sigma_{abs}(\omega_{inc})\rho - l_f\sigma_{abs}(\omega_f)\rho). \tag{6.17}$$

In what follows, we assume that the scattering is purely horizontal. This is an approximation that greatly reduces complexity of the computation. This is justifiable for several existing instruments including LRMECS and PHAROS.

[9] We assume the incoming neutrons are perfectly monochromatic and collimated.
[10] Think $\mathrm{d}N/\mathrm{d}x = -\sigma\rho$...

In fact, any horizontal scattering lies within about $\pm 5^\circ$ of the horizontal on an instrument like Pharos,[11] and is therefore negligible.[12] This means each scattering cell needs two labels.

In considering a number of cells, the incoming path length depends on where the cell is located. The outgoing path length depends on where the cell is, and in what direction the neutron is heading. This direction can be determined if we know the detector d that the neutron is hitting. So we need four labels for the transmission from a given scattering cell to a given detector with a given final energy:

$$T(d, \omega_f, i, j) = \exp(-l_{inc}(i,j)\sigma_{abs}(\omega_{inc})\rho - l_f(i,j)\sigma_{abs}(\omega_f)\rho). \tag{6.18}$$

In the last step, we're going to average over all N cells for each detector and energy:

$$T(d, \omega_f, i, j) = \frac{1}{N}\sum_{i,j} \exp\Big(- l_{inc}(i,j)\sigma_{abs}(\omega_{inc})\rho - l_f(i,j)\sigma_{abs}(\omega_f)\rho\Big). \tag{6.19}$$

It does not get much easier than that. Now all we need is a program that computes Eq. 6.19 for a number of interesting geometries.

6.4 N/A Multiple Scattering Correction

6.5 Calculation of Multiphonon Scattering

The multiphonon expansion was developed with some rigor in Section 3.3.7. Here we explain how it works in practice. The total measured spectrum, $S(E)$, is the sum of components, $\sum_{n=0}^{\infty} S_n(E)$, from neutrons scattered after creating different numbers, n, of phonons in the sample.[13] Only the 1-phonon scattering is useful for obtaining a phonon DOS, so it is important to have an understanding of the higher-order terms for planning an experiment, or if one seeks quantitative corrections of experimental data. Performing a multiphonon expansion is done in two steps.

[11] the detectors are at most 0.5 m above the scattering plane, while the middle of the detector is 4 m from the sample, so the maximum angle is about $\pm 7^\circ$. However, the top and bottom 10 cm or so of the tubes are thrown out...

[12] This assumption has not been sufficiently examined. Unless the effect is very small ($\ll 1\%$), it should be included in the calculation when that becomes computationally affordable. On Pharos, the extra path length is about 0.5%. Is this important? For instruments with larger vertical divergence (ARCS), this will be necessary anyway, so might as well get used to it.

[13] Phonon annihilation is handled by extending the range of E to negative numbers for each $S_n(E)$.

- First, the weights of the different n-components are calculated independently with input information on Q and atom displacements u at the temperature T.
- Second, the spectral shape of each component $S_n(E)$ is obtained by sequentially convoluting the 1-phonon profile with itself a total of $n-1$ times.

Finally, for a chopper spectrometer, detectors at different scattering angles ϕ provide energy spectra $S_\phi(E)$ where Q varies with E across the spectrum. The multiphonon corrections must take into account the kinematical relation of Q vs. E at each scattering angle ϕ if data are analyzed from each detector bank. The constant-Q multiphonon analysis is simpler to understand, and is applicable for data from triple-axis spectrometers, or for data from chopper instruments that have been reduced to $S(Q,E)$.

Intensities of n-Phonon Spectral Components. Start with the Debye–Waller factor, which attenuates the elastic scattering S_0, to a fraction of the total scattering S:

$$S_0 = S\exp(-2W)\ , \tag{6.20}$$

which we rearrange and expand:

$$S = S_0\exp(+2W)\ , \tag{6.21}$$

$$S = S_0\sum_{n=0}^{\infty}\frac{(2W)^n}{n!}\ . \tag{6.22}$$

We now have a series of terms in an expansion, but the next step of substituting (6.20) into (6.22) amounts to nothing more than writing $\exp(-2W)\exp(+2W) = 1$:

$$S = S\exp(-2W)\sum_{n=0}^{\infty}\frac{(2W)^n}{n!}\ , \tag{6.23}$$

$$1 = \sum_{n=0}^{\infty}\frac{(2W)^n}{n!}\exp(-2W)\ . \tag{6.24}$$

The terms in (6.24) are associated as 0 for elastic scattering, 1 for 1-phonon scattering, 2 for 2-phonon scattering, etc. What is special about (6.24) is that the n^{th} term in the series is exactly the fraction of the n-phonon spectral component in the total scattering (elastic plus inelastic).

To calculate the fraction of any multiple phonon term (e.g., the amount of 2-phonon scattering), all we need is $2W$. Recall the physical origin of $2W$:

$$2W = \langle Q^2u^2\rangle\ , \tag{6.25}$$

where $\langle u^2\rangle$ is the mean-squared atom displacement. Equation (6.25) shows that $\exp(-2W)$ becomes much less than 1 when the atom displacement becomes comparable to $1/Q \sim \lambda/2\pi$, the number of wavelengths associated

with the scattering angle. This is consistent with $2W$ originating from the destructive interference of scattered wavelets. The Q^2 and u^2 are related to the energy of recoil, and the thermal energy, respectively. The recoil energy, E_R is

$$E_R = \frac{\hbar^2 Q^2}{2M} \ , \tag{6.26}$$

where M is the mass of the atom that is scattering, in units of the neutron mass. (The units of M are very similar to the atomic weight.) The thermal energy is:

$$k_{\rm B}T = \frac{1}{2} M\omega^2 \langle u^2 \rangle \tag{6.27}$$

for one mode of frequency ω, and is a quantity that depends on the phonon DOS for a real material. Sadly, an exact evaluation of (6.27) is not simple, in part because the phonon states are not fully occupied at modest temperatures, but also because of the commutation relationships discussed in Section 3.3.4. The result is summarized as γ_0, where:

$$\gamma_0 = \int_0^\infty \coth(E/2k_{\rm B}T) \frac{g(E)}{E} dE \ . \tag{6.28}$$

In the limit of high temperatures, where $\coth(E/2k_{\rm B}T) \to 1$, it is clear from (6.28) that γ_0 increases as $1/E$ (because the amplitudes of atom motions become larger in low-energy modes as predicted by (6.27)). The low-energy modes are even more important at low temperatures, where they have a larger phonon occupancy. The Debye–Waller factor of (6.25) is, rigorously:

$$2W = \gamma_0 E_R \ , \tag{6.29}$$

In performing a multiphonon expansion for one T and one phonon DOS $g(E)$, the value of γ_0 is computed once with (6.28). The $2W$ from (6.29) for appropriate Q is used in (6.37) to get the fractions of all n-phonon scatterings. We therefore rewrite (6.24) as

$$S(Q) = \sum_{n=0}^{\infty} S_n(Q) \ , \quad \text{where :} \tag{6.30}$$

$$S_n(Q) = \frac{\Big(2W(Q)\Big)^n}{n!} \ \exp\big(-2W(Q)\big) \ . \tag{6.31}$$

Shapes of *n*-Phonon Spectral Components. It remains to get the spectral shape of each order of the multiphonon scattering. The energy spectrum for 1-phonon scattering was discussed in Sections 2.3 and 3.3.7. The spectrum for one-phonon scattering weights more heavily the low-energy modes because they have larger amplitudes of motion, providing a factor of $g(E)/E$. The number of phonons is the Planck distribution $1/[\exp(E/k_{\rm B}T) - 1]$, so the two factors provide the shape of the 1-phonon profile, $A_1(E)$:

$$A_1(E) = \frac{g(E)}{E\gamma_0} \; \frac{1}{\mathrm{e}^{E/k_\mathrm{B}T} - 1} \; . \tag{6.32}$$

Each phonon created has the profile $A_1(E)$ of (6.32). When two phonons are created simultaneously, the total spectrum of energies is the convolution of the 1-phonon profile with the 1-phonon profile.[14] The 2-phonon spectrum is:

$$A_2(E) = A_1 * A_1 = \int_{-\infty}^{\infty} A_1(E - E') \; A_1(E') \; \mathrm{d}E' \; , \tag{6.33}$$

and the n-phonon profile is the convolution of another 1-phonon profile with the $(n-1)$-phonon profile:

$$A_n(E) = A_1 * A_{n-1} = \int_{-\infty}^{\infty} A_1(E - E') \; A_{n-1}(E') \; \mathrm{d}E' \; . \tag{6.34}$$

Starting with A_1, we can generate the spectral shapes of all the orders of multiphonon scattering by the systematic application of (6.34). The total scattering is the sum of these spectral profiles, weighted by the corresponding terms of (6.30):

$$S(Q,E) = \sum_{n=0}^{\infty} S_n(Q) \; A_n(E) \; , \tag{6.35}$$

$$S(Q,E) = \sum_{n=0}^{\infty} \frac{(2W)^n}{n!} \exp(-2W) \; A_n(E) \; . \tag{6.36}$$

If we define each term in the sum as $S_n(E)$:

$$S(E) = \sum_{n=0}^{\infty} S_n(E) \; . \tag{6.37}$$

Examples of Multiphonon Scattering from a Chopper Spectrometer. The analysis presented here is for inelastic energy spectra obtained from a group of detectors centered about the scattering angle, ϕ. The energy transfer, E, is obtained simply and reliably from the arrival times of the neutrons at the detector. On the other hand, for each detector bank at ϕ, the value of Q varies with E and the energy of the incident neutron E_inc as:

$$Q[\mathring{A}^{-1}] = 0.6947\sqrt{2E_\mathrm{inc} - E - 2\sqrt{E_\mathrm{inc}(E_\mathrm{inc} - E)}\cos\phi} \; , \tag{6.38}$$

where E and E_inc have units [meV]. Curves showing relationships $Q(E)$ are presented in Figure 6.4. The curves cannot extend to the right beyond E_inc because the incident neutron cannot lose a greater amount of energy to the sample. Notice that the curves are generally asymmetrical in $\pm E$.

[14] Consider each phonon excitation to be a random variable with a probability distribution of A_1. The sum of two random variables has a distribution that is the convolution of the probability distributions $A_1 * A_1$.

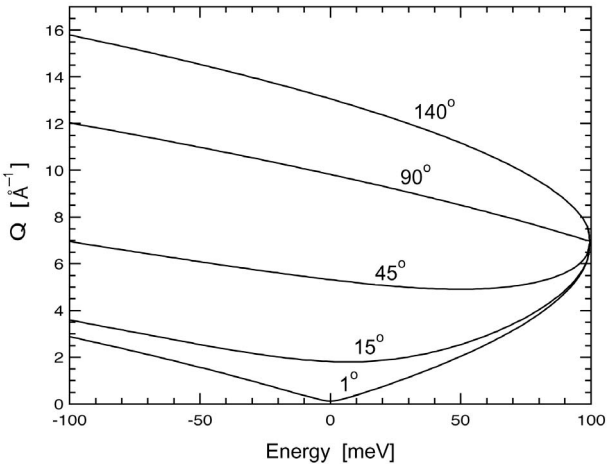


Fig. 6.4. Graphs of Q vs. E for a time-of-flight chopper spectrometer, calculated using (6.38) for five values of scattering angle ϕ (labels on curves). Incident energy was $E_{\rm inc} = 100$ meV. Positive E correspond to phonon creation, negative to phnon annihilation.

For calculating the multiphonon scattering at a particular scattering angle ϕ, it is necessary to evaluate the $2W$ and the Debye–Waller factor at each energy of the spectrum, because Q varies with E as in (6.38) and as shown in Figure 6.4. Examples of such calculations are shown in Figure 6.5. Figure 6.5a shows the phonon DOS curve from fcc nickel metal from which the subsequent curves were calculated. Figure 6.5b shows the total inelastic scattering calculated for a temperature of 500 K.[15] The 1-phonon scattering is confined to the range ± 37 meV, since this is the maximum phonon energy. With the excitation of two phonons, the energy range can be extended to $2 \times \pm 37 = \pm 54$ meV, and this is the energy range of the 2-phonon scattering. In Figure 6.5b, the intensity at 40–50 meV is almost entirely 2-phonon scattering.

Notice the general trend in Figure 6.5b showing that the inelastic scattering to increases with the scattering angle, owing to the factor of $2W^n$ in (6.31). On the other hand, the scattering is suppressed by the Debye–Waller factor, $\exp(-2W)$. At high Q or high T the Debye–Waller factor becomes increasingly important, and effects of this are seen in Figure 6.5c. First notice that although there are about twice as many phonons at 1000 K as at 500 K, the inelastic scattering of Figure 6.5c is nowhere near twice as large as in Figure 6.5b. Also notice the change in symmetry of the spectra in Figure 6.5c with detector angle. For a detector angle of 60°, Figure 6.5c shows an

[15] An experimental spectrum would contain a large peak at $E = 0$ from the elastic scattering. In principle, the area of this elastic peak would be the factor $\exp(-2W)$ of the total integrated scattering (elastic plus inelastic).

inelastic scattering that is approximately symmetrical in $\pm E$. This is consistent with the gradual variation of Q with E in the kinematics as shown in Figure 6.4. On the other hand, for the larger detector angles of 90° and 140°, the value of Q is larger for $-E$ than $+E$. This asymmetry causes the Debye–Waller factor to attenuate the scattering at $-E$ more severely than at $+E$. When these effects are significant, it is obviously naive to interpret the asymmetry in E of the inelastic scattering in terms of detailed balance.

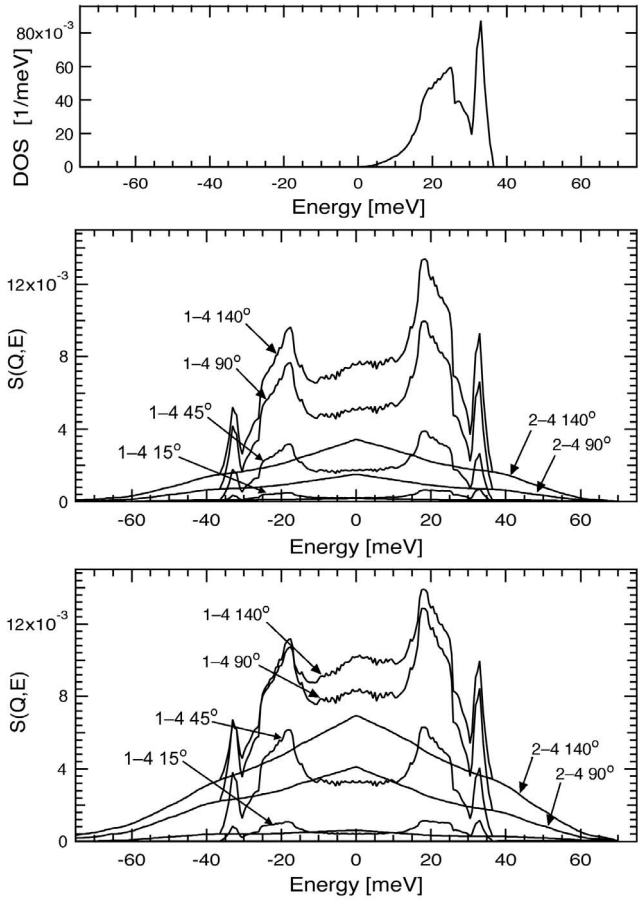


Fig. 6.5. (**a**) Phonon DOS of fcc Ni metal. Inelastic scattering from four angle banks of a time-of-flight chopper spectrometer with incident neutron energy of 70 meV are shown in: (**b**) for 500 K and (**c**) for 1000 K. In parts b and c, the four curves with detailed features are the total scattering, summing the 1-phonon through 4-phonon terms. The broad curves, which are the full intensity above 37 meV, are the mulitphonon scattering from 2-phonon through 4-phonon contributions.

6.5.1 Multiphonon Correction – Iterative

The present section describes a correction procedure that can be used to remove higher-order multiphonon scattering from an experimental inelastic scattering spectrum, so the spectrum can the be used to deduce a 1-phonon profile and a phonon DOS. It is based on the incoherent approximation, but this procedure has been used for many years with coherent multiphonon scattering data, and it is often expected to be reliable in such cases, at least for polycrystalline samples. (The idea is that averaging over crystalline orientations provides a broad sampling of reciprocal space, performing a good average over phonon modes. The incoherent approximation is adequate when all modes are sampled this way. It is unlikely that this approach could be reliable for the analysis of coherent scattering from single crystals, of course.)

An example of multiphonon calculations and an extraction of 1-phonon scattering is shown in Figure 6.6. Cerium is a coherent scatterer, and the data were acquired at several values of Q to average over the different phonon modes. An inelastic spectrum for one value of $Q = 3.924\,\mathring{A}^{-1}$ is presented in the figure.[16] Steps of an iterative procedure to refine the phonon DOS are shown in Figure 6.6. The first step of the iteration used a very crude approximation for the multiphonon scattering plus random background – a simple constant function of approximately 70 counts. After this constant was subtracted from the experimental data, the DOS curve, $g(E)$ labeled "1" in Figure 6.6b was obtained from the 1-phonon profile from (6.32) with a Debye–Waller factor of (6.31). The 2–5 phonon scattering was calculated from this first iteration of a phonon DOS, scaled in height (plus a constant background was also fit to the data), and a second iteration of the DOS was generated by the same procedure. Notice that the second and third iterations of the phonon DOS in Figure 6.6b are rather similar. Even for this difficult case where the 2–5 phonon scattering is a large fraction of the total inelastic scattering, convergence is fairly quick because the 2–5 phonon scattering does not have much structure.

The multiphonon expansion assumes that the scattering is incoherent. Unfortunately, a better approximation would require a detailed simulation of the lattice dynamics to account for the $\boldsymbol{Q}$-dependence. To our knowledge, such a calculation has not been done as of 2004. Another case where the assumptions of this procedure are stretched is the application of the incoherent multiphonon correction procedure to inelastic spectra from alloys. One concern about alloys is that the Debye–Waller factors differ for the different atoms in the alloy, causing some phonons to be weighted more than others. Although the multiphonon correction procedure has been used without adapting to the different $\langle U^2 \rangle$ of the atoms in the alloy, comparisons with

[16] A spectrum at one value of Q can provide a piece of the dynamical structure factor intensity, which we loosely call a "phonon DOS," since the result would be a phonon DOS if the scattering were incoherent. Data were acquired from the triple-axis spectrometer HB3 at HFIR.

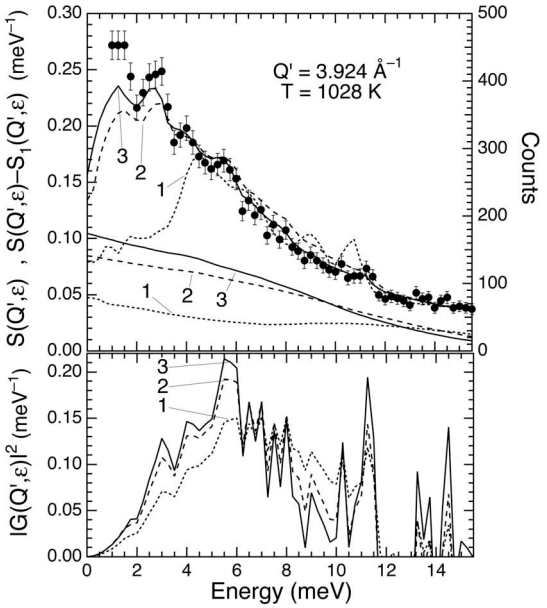


Fig. 6.6. (**a**) points: experimental inelastic scattering from cerium metal at 1028 K and $Q = 3.924\,\mathring{A}^{-1}$. The elastic peak at low energy extends an order-of-magnitude beyond the top of the graph. Calculations of the 1–5 phonon scattering plus a constant background were scaled to fit these data, and are labeled 1, 2, 3, denoting the order of iteration. The 2–5 phonon scattering is shown with the same scaling, again with labels 1, 2, 3 for the iteration. By subtacting the 2–5 phonon scattering from the experimental data, the difference was converted into a "phonon DOS", $g(E)$, by correcting for the thermal factor of (6.32) and Debye–Waller factor of (6.31). (**b**) The "phonon DOS" obtained from the experimental data of part a at different iterations. The first iteration, labeled 1, was obtained by subtraction of a constant background from the experimental data, and correcting by the thermal factor of (6.32). This "DOS" was used for generating the multiphonon corrections for the next iteration of the procedure.

other measurements have shown that the procedure is often acceptable, perhaps because the multiphonon scattering is essentially featureless.

6.5.2 Multiphonon Correction – Fourier Log

Another way to correct for multiphonon scattering is possible for energy spectra at constant Q. Following the notation of Section 6.5, for fixed values of Q and temperature, the energy dependence of the various phonon contributions, $S_n(Q, E)$, is

$$S(Q, E) = \frac{(2W)^n}{n!} \mathrm{e}^{-2W} A_n(E) \,, \tag{6.39}$$

The measured scattering intensity, $S(Q,E)$, is the sum of intensities from all phonon processes:

$$S(Q,E) = \sum_n S_n(Q,E) \ , \tag{6.40}$$

Our problem is to use the measured $S(Q,E)$ to isolate the single scattering profile, $A_1(E)$.

An approach to extracting the phonon partial DOS from experimental data is called the "Fourier-log deconvolution method." It was used for analysis of plasmon scattering measurements in electron energy-loss spectrometry [Spence]. This method is also used for incoherent inelastic phonon scattering [M. Y. Hu, W. Sturhahn, et al.]. It has the additional feature of being able to correct for the broadening of the elastic line, but only if the energy resolution is constant across the energy spectrum (not the case in TOF spectrometers). The zero-loss peak, $Z(E)$, is convoluted with the scattering from the specimen as:

$$S(Q,E) = Z(E) * \left(\mathrm{e}^{-2W} \delta(E) + \sum_n S_n(Q,E) \right) \ , \tag{6.41}$$

where the first term in (6.41) is the zero-loss peak, reduced by the Debye–Waller factor. Taking the Fourier transform, $F[]$, of (6.41) simplifies the convolutions of (6.34), which become multiplications in Fourier space:

$$F[S(Q,E)] = F[Z(E)]\mathrm{e}^{-2W} \left(1 + \sum_n \frac{(2W)^n}{n!} \left(F[A_1(E)] \right)^n \right) \ . \tag{6.42}$$

The term in parentheses () on the right side of (6.42) is recognized as the expansion of an exponential function:

$$F[S(Q,E)] = F[Z(E)]\mathrm{e}^{-2W} \left(\mathrm{e}^{2W F[A_1(E)]} \right) \ . \tag{6.43}$$

Taking the logarithm of (6.43) and rearranging:

$$F[A_1(E)] = \frac{1}{2W} \ln \left(\frac{F[S(E)]}{F[Z(E)]} \right) + 2W \ . \tag{6.44}$$

The inverse Fourier transformation, $F^{-1}[]$, provides the single scattering profile, $A_1(E)$:

$$A_1(E) = \frac{1}{2W} F^{-1} \left[\ln \left(\frac{F[S(E)]}{F[Z(E)]} \right) \right] + 2W\delta(E) \ . \tag{6.45}$$

If we do not care about the normalization of the single scattering profile, and if we delete the zero-loss peak $Z(E)$ from the data (it is suppressed by the thermal correction anyway), we obtain:

$$A_1(E) = \frac{1}{2W} F^{-1} \left[\ln \left(\frac{F[S(E)]}{F[Z(E)]} \right) \right] \ . \tag{6.46}$$

6.5.3 Neutron Weighting

Strictly speaking, the phonon DOS arrived at using the procedure outlined above is not the true phonon DOS, but rather the neutron-weighted DOS. The neutron-weighted DOS is identical with the phonon DOS for an elemental scatterer. This is not the case for a sample that contains more than one type of atom, or scatterer. The neutron-weighted phonon DOS is rigorously defined as:

$$g_{\mathrm{NW}}(E) \propto \sum_d g_d(E) \exp(-2W_d) \exp(2W) \frac{\sigma_d}{m_d} \tag{6.47}$$

where $\exp(-2W_d)$, σ_d and m_d are the Debye-Waller factor, total scattering cross-section and mass of atom d. The Debye-Waller factor is an explicit function of $g_d(E)$. The term $\exp(2W)$ is the average Debye-Waller correction; this is calculated from the self-consistent neutron-weighted DOS. The factor $\exp[2(W - W_d)]$ is approximately unity. For the case where σ_d/m_d is the same for all species d, $g_{\mathrm{NW}}(E) \approx g(E)$.

Obtaining the true phonon DOS from the neutron-weighted phonon DOS requires a full analysis of the lattice dynamics. This can be performed by simulational procedures described in a later chapter. The neutron-weight correction as well as other approximations involved in the data analysis can be overcome by fitting a dynamics model to the neutron-scattering data directly. Although this approach is both scientifically and computationally demanding, we foresee no better method for extracting the vibrational dynamics from inelastic neutron scattering measurements.

6.5.4 N/A Coherent Case

6.5.5 Simultaneous Multiphonon and Multiple Scattering Corrections

Corrections for multiple scattering have been performed in many ways, from subtracting a constant from the data [1], to full Monte-Carlo simulations [2]. At high temperatures, the former does not account for the slope of the scattering past the cutoff energy. The latter can be computationally intensive, and requires detailed information about the shape of the sample. Here we take an approach of intermediate complexity, reported previously [3]. For both multiple scattering and multiphonon scattering, a two-scattering profile involves a convolution of two single-scattering profiles. In either case, the idea is that an n-phonon-scattering profile, $P^n(E)$, is related to the 1-phonon-scattering profile, $P^1(E)$, through the recursion relation:

$$P^n(E) = \int_{-\infty}^{\infty} P^{n-1}(E')\, P^1(E - E')\, \mathrm{d}E' \,. \tag{6.48}$$

For multiple scattering processes, the n-phonon probability function has additional position and momentum dependencies, that do not appear for

multiphonon scattering processes. Sears, *et al.* [1], argue that the integrals for multiple scattering are related to those for the multiphonon scattering through slowly varying functions of Q and E. Here we take these functions to be constants, a_n. In essence, we make the approximation that the position and momentum dependencies can be factored out. Thus,

$$I(Q,E) = N' \left[\sum_{n=1}^{\infty} (1+a_n) S^n(Q,E) \right] , \tag{6.49}$$

where $I(Q,E)$ is the experimentally-determined total scattering (including multiple scattering), $S^n(Q,E)$ is the n-phonon scattering (both creation and annihilation), and N' is a normalization constant. Note that $I(Q,E)$ is distinct from the scattering function, $S(Q,E)$, which does not include multiple scattering [4]. (When we stripped the elastic peak from the data, the dominant multiple elastic scattering is removed, so the index n in Eq. 6.49 starts at 1 rather than 0.)

Consistent with this factoring of Q and E dependencies, we make the incoherent approximation [5]:

$$S^n_{\rm coh}(Q,E) = \frac{\sigma_{\rm coh}}{\sigma_{\rm inc}} S^n_{\rm inc}(Q,E) , \tag{6.50}$$

where we apply this equation to the 1-phonon terms as well as all higher orders. The last step in our procedure will be to assess any error this has introduced into our analysis.

Our next assumption is that $a_n = C'_{\rm ms}$ for all $n \geq 2$, where $C'_{\rm ms}$ is a single constant that relates the multiple scattering to the multiphonon scattering. Since the multiphonon scattering drops off rapidly with increasing n, this approximation will only have a small effect on our results. The final normalization is performed with the total scattering, so the factor $1 + a_1$ is included in the normalization constant. We find:

$$I(Q,E) = N \left[S^1_{\rm inc}(Q,E) + (1+C_{\rm ms}) S^{2+}_{\rm inc}(Q,E) \right] , \tag{6.51}$$

where $N = N'(1+a_1)\,(1+\sigma_{\rm coh}/\sigma_{\rm inc})$ is the normalization constant, and $1 + C_{\rm ms} \equiv (1+C'_{\rm ms})/(1+a_1)$. Also, for notational convenience,

$$S^{j+}(Q,E) \equiv \sum_{n=j}^{\infty} S^n(Q,E) . \tag{6.52}$$

For a cubic crystal, and a fixed value of $C_{\rm ms}$, we can now find the DOS by solving Eq. 6.51 in the manner described by Sears, *et al.* [1].

Since we do not know the value of $C_{\rm ms}$ a-priori, we generate a list of possible values, and solve for the DOS at each one. In the current study, values of $C_{\rm ms}$ between 0.0 and 2.0 were tested. It then remains to select the "best" DOS from those generated with the different $C_{\rm ms}$. This was done by minimizing a penalty function constructed to find the DOS that produced $S(E)$ that best satisfied the following conditions:

(1) $$\frac{I(E)}{N} = S_{\rm inc}1(E) + (1 + C_{\rm ms})S^{2+}_{\rm inc}(E)\ , \tag{6.53}$$

where the implied sum over Q allows us to compare the partially coherent scattering on the left with the totally incoherent scattering on the right.
(2) The experimental noise at energy transfers near the incident energy oscillates about $(1 + C_{\rm ms})S^{2+}_{\rm inc}(E)$.
(3) At energy transfers near the incident energy, the slope of a linear fit to the experimental noise matches the slope of a linear fit to $(1 + C_{\rm ms})S^{2+}_{\rm inc}(E)$.

These three criteria are correlated, but are not identical. For nickel at 300 K, these three contributions and their sum are shown in Fig. 6.7. Figure 6.8 shows the best fit to the normalized scattering, $I(E)/N$ for nickel at 300 K, which had $C_{\rm ms} = 0.6$.

The DOS curves obtained this way were fit with a Born–von Kármán model, from which all phonon contributions to the scattering, both coherent and incoherent, were calculated. With these results, and with the final value for $C_{\rm ms}$, the calculation was checked against the measured scattering. It was our experience that this procedure worked well for the present case of nickel, and also works for cases of other BCC and FCC materials.

[1] V.F. Sears, E.C. Svensson, and B.M. Powell, "Phonon density of states in vanadium," Can. J. Phys. **73**, 726 (1995).

[2] E. Johnson, and L. Robinson, "Neutron multiple scattering and absorption factors," Rev. Sci. Instrum. **60**, 3447 (1989).

[3] M. Kresch, O. Delaire, R. Stevens, J.Y.Y. Lin, and B. Fultz "Neutron scattering measurements of phonons in nickel at elevated temperature," Phys. Rev. B **75**, 104301 (2007).

[4] G. Placzek and L. Van Hove, "Crystal Dynamics and Inelastic Scattering of Neutrons," Phys. Rev. **93**, 1207 (1954).

[5] G.L. Squires, *Introduction to the Theory of Thermal Neutron Scattering*, (Dover Publications, New York, 1997), Chap. 3, Sec. 10, p. 57.

See also: V.F. Sears, "Slow-neutron multiple scattering," Adv. Phys. **24**, 1 (1975).

Further Reading

The contents of the following are described in the Bibliography.

Mark Lutz and David Ascher: *Learning Python* (O'Reilly & Associates, Inc. 1999).

Stephen W. Lovesey: *Theory of Neutron Scattering from Condensed Matter Vol. 1*, (Clarendon Press, Oxford 1984).

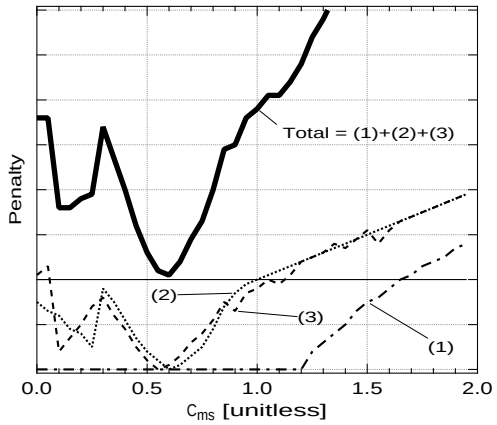


Fig. 6.7. Penalty functions for nickel at 300 K, as defined in the text. The dash-dotted line (1) relates to the overall fit, the dotted line (2) relates to the noise near the incident energy, and the dashed line (3) relates to the slope near the incident energy. The solid line is the sum of these three contributions (offset).

A. A. Maradudin, E. W. Montroll, G. H. Weiss and I. P. Ipatova: *The Theory of Lattice Dynamics in the Harmonic Approximation, Second Edn.* (Academic Press, New York, 1971)

Varley F. Sears: *Neutron Optics*, (Oxford University Press, New York and Oxford 1989).

G. Shirane, S. M. Shapiro, and J. M. Tranquada: *Neutron Scattering with a Triple-Axis Spectrometer*, (Cambridge University Press, Cambridge 2002).

G. L. Squires: *Introduction to the Theory of Thermal Neutron Scattering*, (Dover, Mineola NY 1978).

C. G. Windsor: *Pulsed Neutron Scattering*, (Taylor and Francis, London 1981).

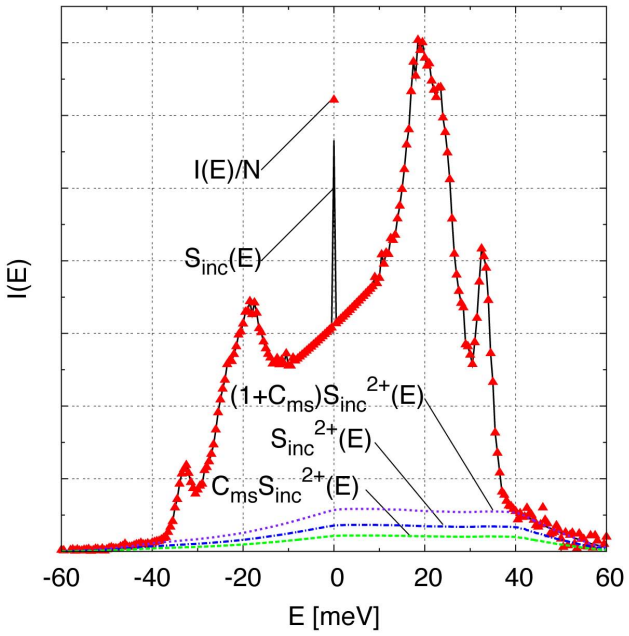


Fig. 6.8. Best fit to scattering for nickel at 300 K. The triangles are the normalized experimental scattering, $I(E)/N$. The solid line shows the fit, $S^1_{\rm inc}(E) + (1 + C_{\rm ms})S^{2+}_{\rm inc}(E)$. The dashed line is the multiple scattering, $C_{\rm ms}S^{2+}_{\rm inc}(E)$. The dash-dotted line is the multiphonon scattering, $S^{2+}_{\rm inc}(E)$. The dotted line is the sum, $(1 + C_{\rm ms})S^{2+}_{\rm inc}(E)$.

7. Software Reference

7.1 `reduction`

DANSE `Reduction software` is a set of orthogonal python packages for reducing neutron scattering data. Those components can be used as building blocks of a new reduction application if a developer follow the procedure in Section 7.1.10.

First let us make a distinction between two words we are going to use in this chapter: `reduction software` and `reduction package`.

- reduction software:a collection of packages that we created for the purpose of reducing data;
- reduction package:one specific package in the `reduction software`.

7.1.1 Introduction

Reduction is a procedure to transform measured raw data to a form which is more easily understandable for scientists. This procedure usually consists of transformations [1] of a histogram measured in a dimension to another dimension that is more physically meaningful (an example of which would be converting time-of-flight to neutron energy) and/or conversion of multidimensional data to lower dimension (an example of which is intensity measured in an area detector sometimes can be binned into "rings" in case of powder sample). In conclusion, a reduction procedure consists mainly of transformations from input histograms to output histograms.

Thus, it is pretty obvious that our fundamental data structure is **histogram**. Section 7.1.2 has more details about the `histogram` package.

In reduction, transformations from input histograms to output histograms require knowledge of the instrument in which measurements are taken. For example, in direct-geometry time-of-flight spectrometer, to convert time of flight to neutron energy, we need to know the length of the flight path, and the time at which neutron hit the sample. The positions of sample and detectors are clearly necessary inputs for reduction procedure. Therefore, we need find a way to store information of the instrument in which the to-be-reduced

[1] For inelastic neutron scattering, such transformations are described in *Ch. 6*.

histogram is measured. The `instrument` package deals with this problem, and details of that package can be found in Section 7.1.3.

Another major piece of our reduction software is the `measurement` package. The `measurement` package is a layer of classes between input data files and reduction engine. Section 7.1.4 has more details.

Finally, the transformations from input histograms to output histograms are implemented in the `reduction` package, which queries the `measurement` package for input histograms, calls reduction engines with parameters derived from the `instrument` package, and produces output histograms.

In the following sections, we will go deeper into each packages in Sections 7.1.2, 7.1.3, 7.1.4, and 7.1.5. We can then get a feeling of how they interact with each other in Section 7.1.7. Developers who want to create a reduction application using the available building blocks should read on to Section 7.1.10.

7.1.2 Histogram

To further this discussion, we need to first clearly define what do we mean here by "histogram". The result of any measurement is actually a histogram, by which we mean we have data in some bins. For example, if we measure a spectrum with x-axis being time-of-flight, we will get an array of counts, while each element in that array represents the number of counts measured in a predefined time slot (bin). This array of counts can be approximated by

$$\frac{dI}{dx}(x)\Delta x \tag{7.1}$$

where $\frac{dI}{dx}$ is a density function and Δx is bin size. This observation forms the base of our design of histogram classes.

It is now clear that there are two pieces of critical information in a histogram: the data, and the axis (or axes). Some time we need to know the context in which the histogram is, and that brings us meta-data. Following are more rigorous definitions:

A histogram consists of axes, datasets, and meta-data related to the histogram:

- dataset: a dense array of numbers which may have many dimensions. Data and errors are all represented by datasets.
- axis: a one dimensional dataset whose elements are the bin boundaries of one dimension of a histogram.
- meta-data: data which provides context for other data: data about data.
- histogram: it contains
  - (1) at least one dataset whose elements represent the number of counts in some range of axis or axes values;
  - (2) optionally one dataset for error bars [2]

[2] actually the squares of error bars are stored to improve computation efficiency

– (3) a set of associations concerning a histogram and potentially everything that can be known about it: axes, history, etc. (meta-data).

Design. The design of the histogram package is not too complex. We need a way to deal with representations of arrays in low-level language, and we have an abstract inteface NdArray to handle that. By introducing the NdArray layer, we isolate histogram from any particular c/c++ array implementations. We also need a way to keep meta-data, and this is handled by AttributeContBase class.

Figure 7.1 is the class diagram of histogram package.

7.1.3 Instrument

Classes in the `instrument` package are used to represent an instrument, independently of any particular measurement. It could be used in several places, such as parsing files derived from engineering plans of an instrument (for `reduction` package), creating a 3d visualization of an instrument, or perhaps in setting up a Monte Carlo simulation.

An instrument consists of several instrument elements and form a hierarchial structure. All instrument elements are in subpackage instrument.elements. In our design, all instrument elements are derived from class instrument.elements.Element.Element. An instrument object can be built out of those elements and follow the real instrument as closely as possible. A diagram of an example instrument is shown in Figure 7.2.

Currently this package is mostly concerned with inelastic direct geometry time-of-flight instrument. To support more instruments, we just need to find out those elements that are not yet supported and add them to the element library.

geometers. Geometers are responsible to measure distances, angles, and other geometric quantities. Each instrument should have an geometer (or several geometers) associated. There are some Geometers in subpackage instrument.geometers.

factories. Three instruments are now supported: LRMECS, PHAROS, and ARCS. Factory methods are created for those instruments and a factory method can create a representation of a particular instrument by reading a text configuration file (for older instruments not using nexus) or a nexus file (for ARCS).

Note: Since the nexus format for ARCS is not settled yet, so the support for ARCS may not be compatible with nexus standard.

For more information, try

http://wiki.cacr.caltech.edu/danse/index.php/Instrument_and_related_classes wiki page

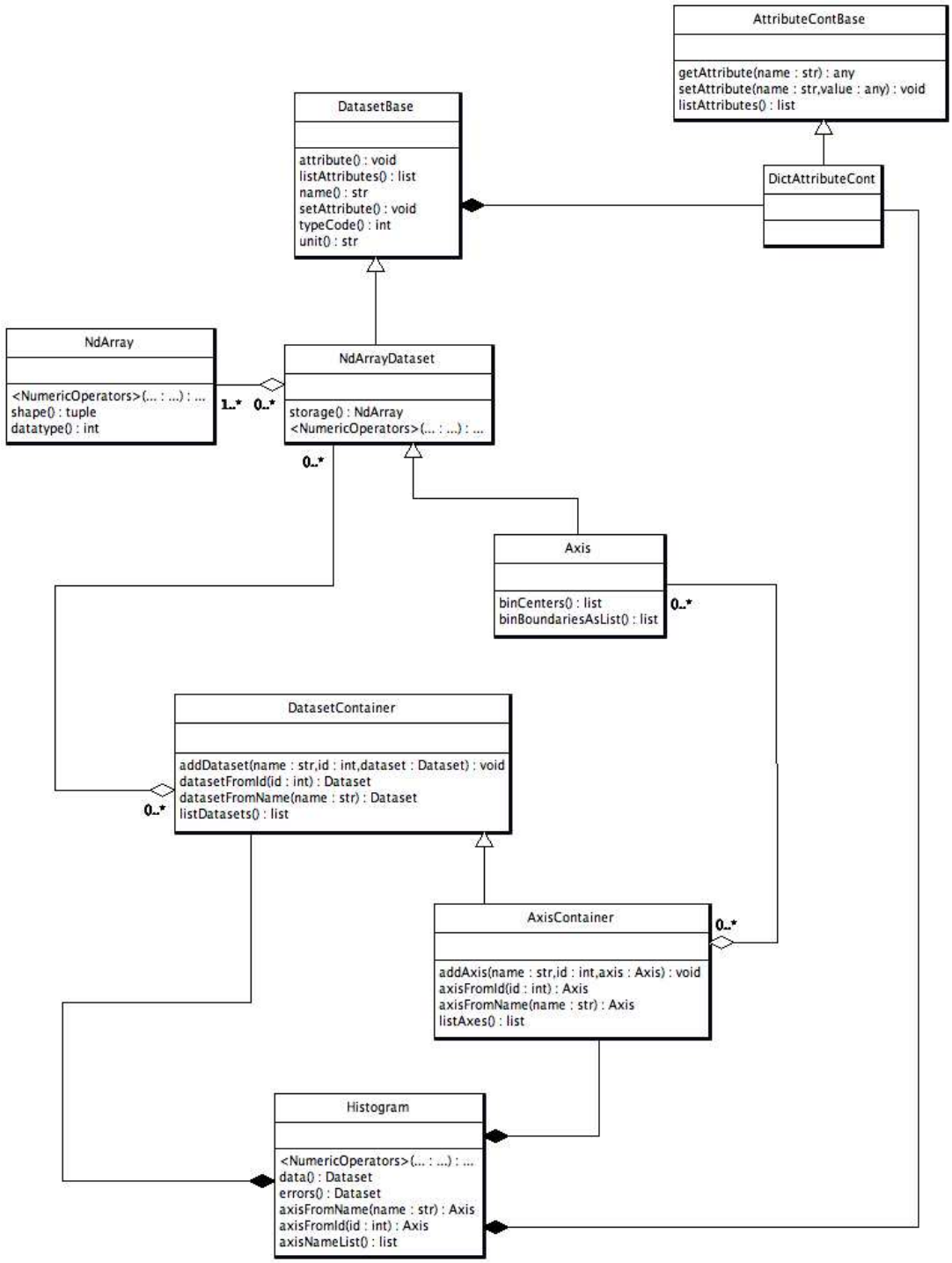


Fig. 7.1. Class diagram for histogram package

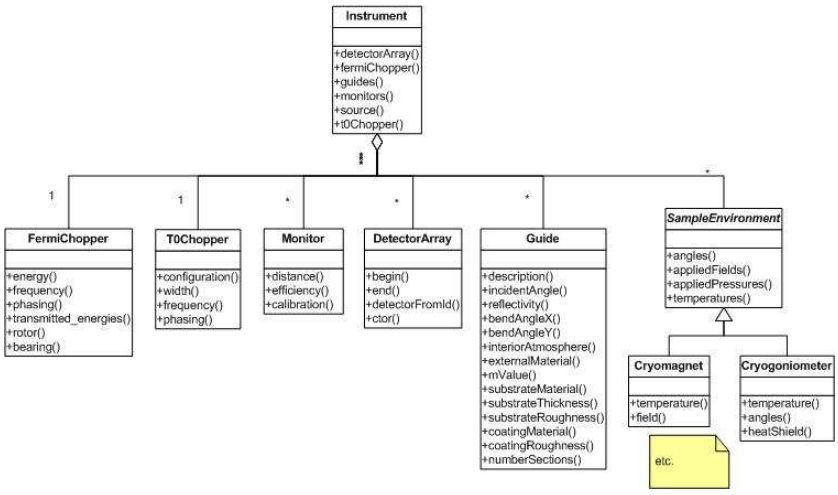


Fig. 7.2. Hierarchy of instrument elements in an example instrument

7.1.4 Measurement

Now we introduce another important piece in reduction software, the `measurement` package. It is a layer of classes between input data files and reduction engine. It converts input data files into input histograms.

In reality, a successful measurement usually consists of several runs:

- a run for the sample you are interested in
- a run for the empty sample can
- a run to calibrate detector efficiency

Each run results in one or more histograms (one main histogram, plus other accompanying histograms, like beam monitor histogram).

The structure of `measurement` package is simple. It consists of measurment classes for every instrument we are supporting, and their common base class. Each instance of a measurement class is a container of "run"s. Each "run" has methods to extract histograms from input data files.

7.1.5 Reduction Package

This section is about the `reduction` package inside the `reduction software`.

The `reduction` package is carefully separated to several layers to ensure a clean design.

- "c/c++" layer is responsible for intensive computations only feasible to be implemented in low level language. For example, this layer includes a class ERebinAllInOne to rebin data in tof bins to data in evenly-spaced energy bins.

- "python vector compatible" layer vectorCompat is the joint point between c++ and python. All c++ codes are implemented to deal with "vector"-like objects, e.g., energy bins. The vectorCompat python package accepts vector arguments and call the corresponding c++ methods to do the real work. This layer separate other python layers from c++ codes and python bindings. It forms a bridge between the reduction operators in the histogram-compatible layer and the low-level language implementations of reduction operators.
- "python histogram compatible" layer histCompat allows developers to deal with objects with more physics meanings. This layer is built on top of the vectorCompat layer. A histogram is an object consisting of axes and datasets and meta data. In the histCompat layer, histograms are our focus. Classes in this layer take histograms instead of vectors as arguments, and implementations of those classes decompose histograms to vectors and call the corresponding methods in the vectorCompat layer.
- "pyre vector compatible" layer pyreVC wraps classes in vectorCompat layer to pyre components. There should not be many of this kind of components because we should be dealing more with histograms in the pyre layer
- "pyre histogram compatible" layer pyreHC wraps classes in histCompat layer to pyre components.
- "pyre" layer reduction.pyre makes use of pyre components in pyreVC and pyreHC layers, and implement classes that are more high-level. The pyreVC and pyreHC layers are more concerned with lower-level operations like "rebin to evenly-spaced energy bins" and "fit a curve to gaussian and find the center". The pyre layer is more concerned with "calculate calibration constants out of calibration data" and "reduce I(det, pix, tof) to S(phi,E).

This layered structure may be illustrated in Figure 7.3 (it is far from complete, and only shows a part of the whole structure).

7.1.6 Reduction Applications

Reduction components are assembled together to create reduction applications. Details of them can be found in

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/reduction/reduction/html/namespacereduction_1_1applications.html

7.1.7 Package Design

As mentioned in the 7.1.1, several packages are involved in our reduction software. There are four "central" packages:

- histogram: fundamental data structure
- instrument: instrument information
- measurement: provide histograms

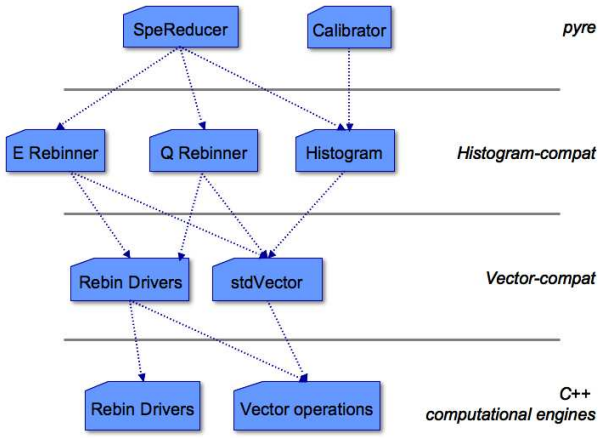


Fig. 7.3. Layers in reduction package

- reduction: reduction methods

And there are other supporting packages:

- nx5: "nexus" readers/(writers) based on hdf5fs
- hdf5fs: a tool to create/read/write hdf5 files by treating a hdf5 file as a filesystem with trees of directories and files
- stdVector: python binding to c++ std::vector template
- array_kluge: tools to manipulate c arrays

Relationship among those packages can be illustrated in Figure 7.4.

7.1.8 Miscellaneous Design issues

Error propagation. Error propagtion happens in all histogram numerical operatorions like + - * /

Reduction components always propagate errors.

IMPORTANT POINT: The error arrays in histograms are always assumed to contain the squares of errors. The reason is that constantly squaring and square-rooting data is not only redundant, it will thrash your numerical precision. It is up to the user (which may be other programmers) to take square roots as appropriate (for instance, when plotting an intermediate result, or writing a final result to disk).

7.1.9 Doxygen documentations

More details can be found in

http://www.cacr.caltech.edu/projects/ARCS/autogen-arcs-docs/reduction/reduction/html/ doxygen documentations

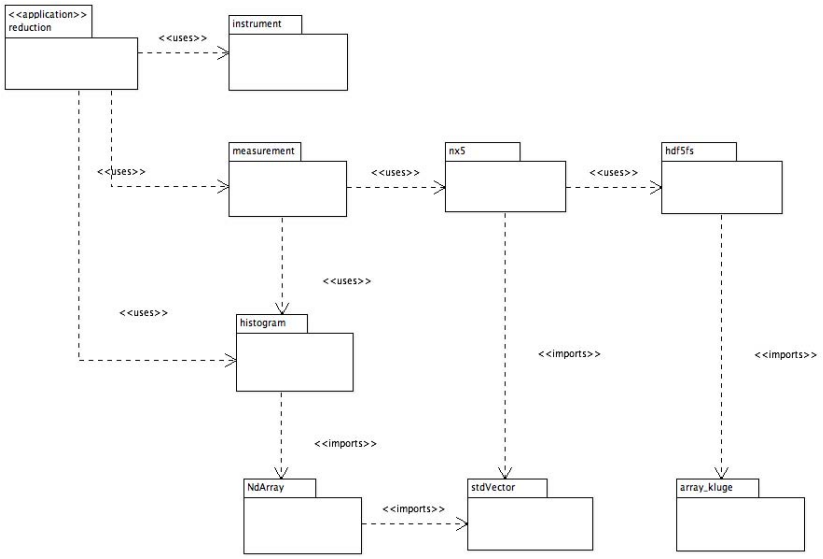


Fig. 7.4. Reduction software package diagram

7.1.10 Procedure to Create Reduction Application

This tutorial explains how to create a reduction application using buiding blocks in DANSE reduction packages within the DANSE reduction framework.

Powder reduction application for direct-geometry time-of-flight spectrometer is used as the example.

Instrument representation. Our understanding of reduction is that reduction is a transformation between input-histogram and output-histogram. Instrumentation information are needed to do those transformations. So the first step is to create a representation of the instrument where data are taken.

An instrument representation should be created with a hierarchial structure which mimics the real instrument structure. It can be done by using instrument elments available in instrument.elements python package. If an instrument element does not exist in that package, you will need to implement a new element.

For an quick example, here we create an instrument without introducing new instrument elements:

```
>>> from instrument.elements import *
>>> instrument = Instrument.Instrument( 'instr_name', version = '0.1' )

>>> moderator = Moderator.Moderator( instrument.getUniqueID(), instrument.guid(),
                                     100, 100, 100 )
>>> instrument.addModerator( moderator )
```

```
>>> monitor = Monitor( instrument.getUniqueID(), instrument.guid(),
                       xLength=30, yLength=0., zLength=50,
                       monitorNumber=1,
                       name = "monitor1")
>>> instrument.addMonitor( monitor, 1 )
```

Things to note:

- The first step is to create an instrument by calling Instrument constructor.
- A instrument-wide unique ID is needed for any instrument element except instrument itself. This unique ID can be generated by calling instrument's method "getUniqueID".
- Sizes are in units mm

In the simple example above, we did not introduce the geometer. A geometer is always needed before an instrument becomes useful. Positions and orientations of any instrument element must be registered by the instrument's geometer. Following is a more complete example of an instrument, this time with a geometer.

```
>>> from instrument.elements import *
>>> instrument = Instrument.Instrument( "instr_name", version = "0.1" )
>>> from instrument.geometers.ARCSGeometer import Geometer
>>> geometer = Geometer()
>>> moderator = Moderator.Moderator( instrument.getUniqueID(), instrument.guid(),
                                     100, 100, 100 )

>>> instrument.addModerator( moderator )
>>> position = [20000.0, 90.0, 180.0]
>>> orientation = [0.0, 0.0, 0.0]
>>> geometer.register( moderator, position, orientation)

>>> monitor = Monitor( instrument.getUniqueID(), instrument.guid(),
                       xLength=30, yLength=0., zLength=50,
                       monitorNumber=1,
                       name = "monitor1")
>>> position = [2300.0, 90.0, 180.0]
>>> orientation = [0.0, 0.0, 0.0]
>>> geometer.register( moderator, position, orientation)
```

Things to note here:

- The coordinate system used by ARCSGeometer is a spherical one centered at sample position. Position is specified hence by a 3-tuple (ρ, θ, ϕ)

You might want to implement your own geometer if the ARCSGeometer does not fit your needs.

Examples of instrument factories and geometers can be found in python package instrument.factories and instrument.geometers.

Histogram. In this section we identify and implement histogram classes that are specific for the target reduction task.

For direct-geometry TOF spectrometer, following histograms are possible inputs and outputs:

- input histograms
  - I(det, pix, tof)
  - I(det, tof)
  - I(tof) for monitors
- output histograms
  - S(phi,E)
  - S(Q,E)

To implement those histogram classes, we simply inherit a new class from the Histogram base class in python package `histogram`, and implement new interface that is more convenient.

Measurement. In this section we explain how to implement measurement classes and measurement pyre components.

As mentioned in Section 7.1.4, a measurement usually consists of several "run"s. First we would like to implement a class "PharosRun".

The "PharosRun" class is a subclass of the measurement.Run.Run class. We implemented several methods to extract input histograms we have identified in Section 7.1.10 from data files. So it is quite clear that we should have following interface for PharosRun:

- ctor constructor takes the data file name as input
- getDetPixTOFData Extract I(det,pix,tof) histogram from data file
- getDetTOFData Extract I(det,tof) histogram from data file
- getMonitorData Extract monitor I(tof) histogram from data file

Now we implement PharosMeasurement class. It will be a subclass of measurement.Measurement.Measurement class. It should be a container of several "run"s which are instances of PharosRun. Those "run"s are

- main data for main sample
- calib data for calibration sample
- mt data for empty sample can

The last step is to create a pyre component for the PharosMeasurement class. This pyre component will be very thin because most of the real work are already done in class PharosMeasurement.

Transformation and rebinning. In this section we implement the transformation in low-level language and export those transformations up to python and pyre layer.

Pyre components. In this section we carefully separate reduction procedure into small steps and implement pyre componenents for each step.

Pyre application. In this step we combin pyre components constructed in the previous section and create a reduction pyre application.

User Interface. In this step we uses and customizes the univeral pyre GUI (prototype) to create GUI for the pyre application created in the previous section

7.1.11 Status

Currently (November, 2006), `reduction` has four essential layers: pure C++, and Python-C++ integration (written in C++), pure Python, and pyre. In `reduction`'s precursor, `topline`, all scientific functionality resided in the C++ layer; Python bindings (exposing a few specialized pieces of C++ code) only made it easier to run. `reduction` blurs that relationship: now Python is the "primary" layer, with most of what happens in C++ accessible to Python and a number of high-level services, such as rebin drivers, implemented purely in Python. For simplicity, scientific content is forbidden in the integration layer (Hopefully, this has been enforced). The growth in the code base from `topline` to `reduction`-1.0 reflects these changes: `topline` had about 2000 lines of C++ in a dozen or so classes; about 900 lines of C++ code provided Python bindings to six entry points for high-level routines. Version 1.0 of `reduction` has about 1500 lines of C++ code, and about 2100 lines of Python bindings providing roughly 60 entry points for much finer-grained access to the C++ layer. The `topline` Python layer was an after thought, weighing in with 100 or so lines of Python; `reduction`'s pure Python layer has over 21000 lines.

7.1.12 Build/Install

Please download the ARCS 1.0 reduction source from

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/Download_source.html

and follow instructions there to build and install.

7.1.13 Performance

Under linux (gcc 4.0, etc.) a 2.0 GHz AMD Opteron box with 4 GB RAM can reduce a dataset to $S(Q, E)$ in about 27 sec—about how long it took to do the same task under these conditions with `topline`. On a Windows XP box (using Microsoft Visual C++ v7.1), it takes about 60 sec; this is a disappointment, as the `topline` code executed in about 6 sec. Some of the Windows bottleneck seems to be caused by memory swapping. Other factors might include file I/O of hdf5 file, and expensive looping in python layer.

7.1.14 To Do

- `instrument`: More instrument elements should be added to instrument.elements package. Geometers might deserve better implementation. Currently the way geometer deals with coordination system is a little clumsy.
- `reduction`:
  1. Energy rebin driver in c++ layer has a generic alogrithm to rebin 1-D data, but is not implemented generic enough. We can create a generic rebinner so that it can be used by many rebin tasks.
  2. Reduction pacakge make explicit uses of STL vectors; many of these uses can be replaced by iterators, which would broaden their reusability while insulating them from changes to the underlying containers. For instance, rebinning procedures should be willing and able to operate on any contiguous chunk of data, whether it belongs to an STL vector, an ordinary C-array, NumArray, etc.
  3. Will need to incorporate Jae Dong Lee's single crystal codes.

7.2 Module Documentation

This section provides web access to the online ARCS software documentation. You will need to have an internet connection to make proper use of the present section.

The following include links to the developer's API-level socumentation for all available ARCS 1.0-release packages. You may need to alter your Acrobat preferences to open a web browser or capture the web pages in Acrobat, as you prefer.

As a test of your connection and browser interface, please check some of the top pages of the ARCS and DANSE resources:

DANSE homepage http://wiki.cacr.caltech.edu/danse/

ARCS homepage http://www.cacr.caltech.edu/projects/ARCS/

PYRE homepage http://www.cacr.caltech.edu/projects/pyre/

The top-level web page for the ARCS software, including installation and release information, is:

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/Software.html

The top-level documentation index for the reduction software is:

http://wiki.cacr.caltech.edu/danse/index.php/DANSE_Reduction_Documentation_Index

The top-level web page for all software module documentation can be accessed at:

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/Modules.html

7.2.1 distutils_adpt

Distutils was extended to more easily generate distributions for packages that include both python codes and low-level c/fortran codes. Official python distutils don't build c/c++/fortran libraries, but only python extensions. With distutils_adpt, one can build and install libraries and executables like normal software in a scheme with "bin", "lib", "include", and "python" folders. Some improvements include recursive inclusion of python subdirectories, the option to use dynamically linked libraries, and the ability to find and modify PATH variables (from DANSE's grid services light package).

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/distutils_adpt/distutils_adpt/html/index.html
distutils_adpt python package

7.2.2 config headers

Config is a cross-platform build procedure. For developer, one configuration file ".tools" is what he all needs to start enjoying developing without worries

on things like how to create a dynamic library on darwin. Pakcage config-headers only contains header files from Config.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/config/headers/html/index.html
distutils_adpt python package

7.2.3 journal

journal allows component and core programmers to communicate information to subsequent users on an unlimited number of channels that can be switched on at will, with severities including debug, error, info, warning, and firewall. journal is available in both C++ and Python, with C++ journals fully integrated with their Python counterparts. journal devices are abstracted, making them adaptable to environments such as massively parallel platforms or geographically dispersed distributed computing.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/journal/journal/html/index.html
journal python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/journal/libjournal/html/index.html
journal C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/journal/journalmodule/html/index.html
journal python bindings

7.2.4 pyre

http://www.cacr.caltech.edu/projects/pyre

pyre is the software framework for high performance computing. Written in Python, pyre is an extensible, object-oriented framework for specifying and staging complex, multi-physics simulations. It includes support for remote monitoring and visualization. With pyre, you don't need to be fluent in FORTRAN or C to harness the power of massively parallel computational resources. Pyre, which has the ability to utilize code written in other languages, provides a level of transparency that allows scientists to focus on their research rather than being distracted by computational details. However, while pyre will enable the average user, it was also designed not to hinder expert users, offering powerful, flexible development features.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/pyre/pyre/html/index.html
pyre python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/pyre/libpyre/html/index.html
pyre C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/pyre/pyremodule/html/index.html
pyre python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/pyre/applications/html/index.html
examples/applications

7.2.5 array_kluge

http://wiki.cacr.caltech.edu/danse/index.php/array_kluge

Dynamically allocate C arrrays of numbers and maintain them correctly. Limited array to and from list conversions are offered. array_kluge has been largely replaced by stdVector.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/array_kluge/array_kluge/html/index.html
array_kluge python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/array_kluge/libarray_kluge/html/index.html
array_kluge C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/array_kluge/array_klugemodule/html/index.html
array_kluge python bindings

7.2.6 stdVector

http://wiki.cacr.caltech.edu/danse/index.php/StdVector

StdVector allows Python programmers to create C++ standard vector objects in a variety of numerical types. Limited interaction with those vectors is supported through methods like size, assign, begin, end, plusEquals, average, etc. Through the asNumarray() method, StdVectors may also be manipulated using the Numarray multidimensional indexing interface, and passed to functions that expect a numarray object, without potentially costly copying. Through the asList method, a StdVector may be converted to a Python list. Thus, StdVector objects are compatible with a wide variety of Python, C, and C++ interfaces.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/stdVector/stdVector/html/index.html
stdVector python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/stdVector/libstdVector/html/index.html
stdVector C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/stdVector/stdVectormodule/html/index.html
stdVector python bindings

7.2.7 hdf5_cpp

http://wiki.cacr.caltech.edu/danse/index.php/hdf5_cpp

Recent releases of NCSA's C++ interface have not supported the building of shared objects. For those who do not have access to a properly prepared HDF5 distribution, this package provides a limited remedy by distributing the NCSA HDF5 C++ interface, version 1.6.4, along with Caltech build system or Python distutils instructions to build as a shared object. It requires the user to have the NCSA HDF5 version 1.6.4 of the C library already built.

7.2.8 hdf5fs

http://wiki.cacr.caltech.edu/danse/index.php/Hdf5fs

hdf5fs provides Python bindings to part of the HDF5 C++ library. It offers the user a UNIX filesystem abstraction, implementing function like 'open', 'stat', etc. Written primarily by Maciek Brodowicz of CACR, it has since been modified by Tim Kelley. This is the primary ARCS interface to HDF5 files.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/hdf5fs/hdf5fs/html/index.html
hdf5fs python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/hdf5fs/hdf5fsmodule/html/index.html
hdf5fs python bindings

7.2.9 nx5

http://wiki.cacr.caltech.edu/danse/index.php/nx5

Our primary file format is HDF5, structured according to NeXus. nx5 is a layer above HDF5 (via hdf5fs, though one could use the nexus API) that represents the structure of files and the corresponding semantic content.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/nx5/nx5/html/index.html
nx5 python package

7.2.10 histogram

http://wiki.cacr.caltech.edu/danse/index.php/ARCS_Alpha_subpackage_histogram

The histogram package supplies classes that combine arrays (storage) with attributes to model various complex ideas about datasets for neutron scattering.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/histogram/histogram/html/index.html
histogram python package

7.2.11 measurement

http://wiki.cacr.caltech.edu/danse/index.php/measurement

The measurement package combines nx5 and histogram to provide a convenient layer for applications programmers through functions such as "getPixelData", "getMonitorData", etc.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/measurement/measurement/html/index.html
measurement python package

7.2.12 instrument

http://wiki.cacr.caltech.edu/danse/index.php/ARCS_Alpha_subpackage_instrument

The instrument package consists of classes to represent neutron scattering instrument information. An instrument is represented by an instance of the Instrument class, which is the root node of a graph of objects (Elements). The coordinates of the elements of an instrument are maintained by a separate class, Geometer. This creates a generic way to represent and serve information about instrument configuration and properties and sample configuration and properties. Factories construct core representations from both hard-coded sources (such as ARCSBootstrap) and from nx5 file graphs (NX5ToARCS family); additional factories render instrument graphs to file graphs (ARCSToNX5 family) or to visual models (VTKInstrumentViewer).

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/instrument/instrument/html/index.html
instrument python package

7.2.13 reduction

http://wiki.cacr.caltech.edu/danse/index.php/ARCS_alpha_subpackage_reduction

The reduction subpackage contains the classes and components that are responsible for actually reducing data from a time-of-flight direct geometry spectrometer. Reduction transforms data from I(TOF, detectorLabels) to something approximating (or resembling) $S(|\boldsymbol{Q}|, E)$. These classes are mostly built up out of pieces from other ARCS subpackages.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/reduction/reduction/html/index.html
reduction python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/reduction/libreduction/html/index.html
reduction C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/reduction/reductionmodule/html/index.html
reduction python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/reduction/examples/index.html
examples/applications

7.2.14 bvk

http://wiki.cacr.caltech.edu/danse/index.php/bvk

bvk provides basic Born-von Karman (lattice dynamics) models of phonons from perfect crystals. While nothing in the code precludes the use of a completely arbitrary crystal structure, presently the relation between bonds and force constants must be specified by the user by overriding the ForceConstantTensor class. Present outputs include density of states histograms; plans are in place to include neutron scattering cross sections. The code is structured to allow arbitrary iteration of Q-space; random sampling from a user-specified cube is implemented.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/phonons/bvk/bvk/html/index.html
bvk python package
http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/phonons/bvk/libbvk/html/index.html
bvk C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/phonons/bvk/bvkmodule/html/index.html

bvk python bindings

7.2.15 sam

Sam provides python bindings to Matlab, and is written and maintained by Patrick Hung. Sam does NOT provide a pyre component; however, it is both a standalone package and is required by 'graphics.Matlab'.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/sam/sam/html/index.html
sam python package
http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/sam/libsam/html/index.html
sam C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/sam/sammodule/html/index.html

sam python bindings head

7.2.16 pyIDL

pyIDL provides python bindings to RSI's IDL, and is based on Andrew McMurray's python–IDL. pyIDL provides an IDL session emulator where IDL commmands can be used directly to manipulate data. pyIDL is also NOT a pyre component; however, it is both a standalone package and is required by 'graphics.IDL'.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/pyIDL/pyIDL/html/index.html
pyIDL python package
http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/pyIDL/libpyIDL/html/index.html
pyIDL C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/pyIDL/pyIDLmodule/html/index.html

pyIDL python bindings head

7.2.17 graphics

This package is a collection of bindings to several graphics and visualization enviroments. Packages currently include: Matlab, IDL, Matplotlib, gnuplot, and grace. The gnuplot component uses gnuplot-py, while the grace component uses both Michael Haggerty's grace_np and Nathaniel Gray's gracePlot.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/graphics/gnuplot/html/index.html
graphics.gnuplot
python subpackage
http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/graphics/grace/html/index.html
graphics.grace
python subpackage
http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/graphics/IDL/html/index.html
graphics.IDL python subpackage
http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/graphics/Matlab/html/index.html
graphics.Matlab python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/graphics/Matplotlib/html/index.html

graphics.Matplotlib python subpackage

7.2.18 cctbx_adpt

This package extends the cctbx python bindings to the pyre framework. The boost python library was integrated to speed up large matrix manipulations.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/cctbx_adpt/cctbx_adpt/html/index.html
cctbx_adpt
python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/cctbx_adpt/boost_ext/boost_ext/html/index.html
boost_ext
python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/cctbx_adpt/boost_ext/libboost_ext/html/index.html
boost_ext C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/cctbx_adpt/boost_ext/boost_extmodule/html/index.html

boost_ext python bindings

7.2.19 simulation

http://wiki.cacr.caltech.edu/danse/index.php/Instrument_Simulation:_simulation#package_simulation

The simulation package provides a framework allowing flexible integration of virtual neutron instruments for simulation. A base class for constructing a simulation application can be conveniently subclassed for a neutron instrument. Currently most neutron components that can be used in such a simulation application are provided by package "mcstas." This simulation package, however, has a collection of generic components, which provides a scheme for flexible and extensible simulation of samples and detectors.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/simulation/simulation/html/index.html
simulation python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/common/common/html/index.html
simulation.common python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/common/libcommon/html/index.html
simulation.common C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/common/commonmodule/html/index.html
simulation.common python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/geometry/geometry/html/index.html
simulation.geometry python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/geometry/libgeometry/html/index.html
simulation.geometry C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/geometry/geometrymodule/html/index.html
simulation.geometry python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/instruments/html/index.html
simulation.instruments python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/neutron_buffer/neutron_buffer/html/index.html
simulation.neutron_buffer python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/neutron_buffer/libneutron_buffer/html/index.html
simulation.neutron_buffer C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/neutron_buffer/neutron_buffermodule/html/index.html
simulation.neutron_buffer python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/neutron_comp/neutron_comp/html/index.html
simulation.neutron_comp python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/neutron_comp/libneutron_comp/html/index.html
simulation.neutron_comp C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/neutron_comp/neutron_compmodule/html/index.html
simulation.neutron_comp python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/nexus/nexus/html/index.html
simulation.nexus python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/phonon/phonon/html/index.html
simulation.phonon python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/phonon/libphonon/html/index.html
simulation.phonon C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/phonon/phononmodule/html/index.html
simulation.phonon python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/stdVector/stdVector/html/index.html
simulation.stdVector python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/stdVector/libstdVector/html/index.html
simulation.stdVector C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/stdVector/stdVectormodule/html/index.html
simulation.stdVector python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/xtal/xtal/html/index.html
simulation.xtal python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/simulation/xtal/lib/html/index.html

simulation.xtal C++ library

7.2.20 mcstas

http://wiki.cacr.caltech.edu/danse/index.php/Instrument_Simulation:_simulation#package_mcstas

The mcstas package provides Python bindings to the distribution of McStas from Riso. Bindings are available for building neutron scattering instrument components and launching simulations of experiments. In the Riso McStas, an instrument is constructed from an ordered list of components described by a dedicated meta-language. Neutrons go through each component one by one. The state of a neutron is modified (scattered or absorbed) by the C-code of each component. The statistical weight of the neutron is adjusted if, for example, a specific target solid angle for scattering is provided, in order to reduce the simulation time. The code is generated by McStas, which acts as a compiler that compiles the meta language in an instrument description file to one big C file. In pyre-mcstas, the instrument components are pyre components, and can be scripted together. In essence, the meta-language is Python. Another difference is that a buffer of many neutrons is passed from component to component in pyre-mcstas.

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/mcstas/html/index.html
mcstas python package

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/mcstas_lib/html/index.html
mcstas C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/mcstas_wrapper/html/index.html
mcstas.mcstas_wrapper python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/components/html/index.html
mcstas.components python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/utils/html/index.html
mcstas.utils python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/array_conversion/array_conversion/html/index.html
mcstas.array_conversion python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/array_conversion/lib/html/index.html
mcstas.array_conversion C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/array_conversion/module/html/index.html
mcstas.array_conversion python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/bin_file_io/bin_file_io/html/index.html
mcstas.bin_file_io python subpackage

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/bin_file_io/libbin_file_io/html/index.html
mcstas.bin_file_io C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/bin_file_io/bin_file_iomodule/html/index.html
mcstas.bin_file_io python bindings

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/generic/libgeneric/html/index.html
mcstas.generic C++ library

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/mcstas/generic/genericmodule/html/index.html

mcstas.generic python bindings

7.2.21 pyIO

PyIO is a 'generic' file reader/writer package for N-column text or Python-nested list formats. PyIO reads some plain text dialects, can read/write data asRows or asColumns, and can extract simple headers from text files (if matches dialect format)

http://www.cacr.caltech.edu/projects/ARCS/arcs-1.0/web/autogen/pyIO/pyIO/html/index.html

pyIO python package

7.2.22 Multiphonon

http://wiki.cacr.caltech.edu/danse/index.php/Multiphonon_py

Multiphonon performs a multiphonon expansion to simulate experimental inelastic scattering spectra. Input to the program is a disk file of a phonon DOS in terms of energy, intensity, and error, plus user input on the temperature, atom mass, energy of the incident neutron, request for analysis of data with constant-Q (must supply Q) or TOF angle bank (must supply angle, ϕ). Multiphonon calculates terms in the multiphonon expansion as explained in Section 6.5, using the theory of Section 3.3.7, and sums spectral contributions from the user-specified multiphonon processes. The names of the variables in the code are matched to those of the theory, as developed by Varley Sears and used in Section 6.5. An iterative procedure for performing this correction is explained in Section 6.5.1.

Further Reading

The contents of the following are described in the Bibliography.

Tim Kelley, Mike McKerns, Jiao Lin, Michael Aivazis and Brent Fultz: *DANSE wiki web site*,
`http://wiki.cacr.caltech.edu/danse/index.php/Main_Page`

Tim Kelley, Mike McKerns, Jiao Lin, Michael Aivazis and Brent Fultz: *ARCS software web site*,
`http://www.cacr.caltech.edu/projects/ARCS/Software.html`

Mark Lutz and David Ascher: *Learning Python* (O'Reilly & Associates, Inc. 1999).

C. G. Windsor: *Pulsed Neutron Scattering*, (Taylor and Francis, London 1981).

8. N/A Structure of Computer Programs

8.1 N/A Abstractions

8.1.1 N/A Abstractions of Procedures

8.1.2 N/A Abstractions of Data

8.2 N/A Functions, Classes, Modules, and All That

8.3 N/A Data Flow and Streams

8.4 N/A Computer Graphics

Further Reading

The contents of the following are described in the Bibliography.

H. A. Abelson and G. J. Sussman: *Structure and Interpretation of Computer Programs* (MIT Press, Cambridge Mass, 2001).

Mark Lutz and David Ascher: *Learning Python* (O'Reilly & Associates, Inc. 1999).

9. *DANSE* Architecture and Engineering

9.1 Software for Inelastic Scattering

9.1.1 Overview of Capabilities

Data from inelastic chopper spectrometers can reveal the fundamental dynamical processes in materials or condensed matter, but not without substantial data analysis to produce even a basic graph of $S(\boldsymbol{Q}, E)$, the intensity as a function of momentum and energy transfers. The elementary excitations in solids are a substantial topic in themselves, and are covered in the earlier chapters of this book. Making comparisons between the experimental $S(\boldsymbol{Q}, E)$ and the underlying theory requires another level of software sophistication. The it DANSE architecture was designed to make basic data reductions, theoretical computations, and comparisons between theory and experiment as convenient as possible. A useful software architecture must be flexible, allowing users to rearrange the steps of data analysis, for example. For longevity this system must allow users to extend its capabilities beyond what those of today.

This chapter describes the essential features of the it DANSE architecture, although it does not document the individual software components. The first section provides an overview of the scientific capabilities of it DANSE, explaining some of the design philosophy. The second section describes the essential features of the architecture – components acting on data streams. A later section explains how to extend the capabilities of it DANSE by developing new components from C++ code.

9.1.2 Data Reduction

A schematic of the *DANSE* software for inelastic scattering is presented in Fig. 9.1. This figure, first presented as a roadmap in Sept. 2001, shows three paths for extracting scientific results from the raw data in the upper left corner.

The first path, horizontally across the top of Fig. 9.1, is the traditional approach to data reduction and visualization. The goal of this analysis is to obtain the intensity as a function of momentum transfer and energy transfer,

$S(\boldsymbol{Q}, E)$. To do so, the data arrays of counts acquired in terms of instrument parameters such as detector pixel and arrival time must be converted into normalized intensities with physical units such as Å and meV. Instrument backgrounds and other distortions must also be removed. It is often necessary to correct for other distortions caused by, for example, multiple scattering or multiple excitations. By implementing all software components in the interpreted Python language, the *DANSE* architecture provides a set of components that are continuous across all data analysis steps. Furthermore, components in the chain can be replaced or rearranged to test different processing algorithms. It is also important that data streams can be piped into visualization windows for inspecting the data after the different steps of data processing. An example of a component for energy rebinning of data from a chopper spectrometer is described in the context of the data stream architecture in Sect. 9.2.2.

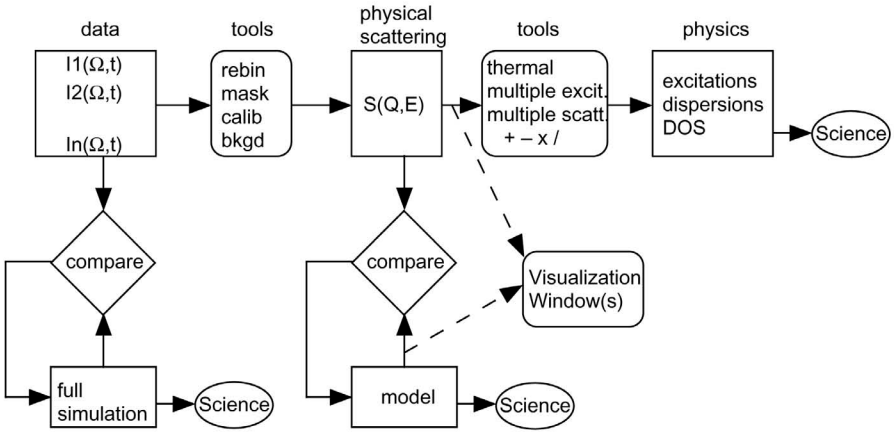


Fig. 9.1. Schematic of *DANSE* software for inelastic scattering.

9.1.3 Modeling

The second and third paths for extracting scientific results from experimental data are designed to connect experimental data to theory or analytical models.

The second path, the vertical chain in the center of Fig. 9.1, is for comparing experimental results to models of sample dynamics. This path is especially appropriate for analytical models with adjustable parameters. Consider its use for the "neutron-weighting problem" in phonon dynamics, which originates as follows. A measured $S(Q, E)$ from a polycrystalline sample of a pure element can often be converted into a phonon density-of-states using

a thermal correction procedure. On the other hand, a phonon DOS from a binary compound cannot be otained from the measured $S(Q, E)$ because the different elements in the compound do not scatter neutrons with equal efficiencies, causing a "neutron-weighting" of the experimental spectra. Without knowing the lattice dynamics of the compound, it is impossible to know the distortions of an experimental DOS obtained from a measured $S(Q, E)$ after a thermal correction procedure.

The phonon scattering efficiencies of the different atoms are well known, so a lattice dynamics model can be used to calculate an experimental spectrum. We do so with an iterative procedure where the force constants in the dynamics model are varied to obtain the best fit to the experimental data. We therefore can use a lattice dynamics model plus a "neutron weighting" correction to obtain the true phonon density-of-states from the measured $S(Q, E)$ of the compound. The force constants are obtained as parameters that give the best fit between calculation and experiment. This is now a routine procedure for us, and new experiments can be designed around this capability.

Besides phonon dynamics in ordered compounds, there are many dynamics models that can be compared to experimental data. Applications include data corrections, the determination of parameters such as force constants or exchange stiffnesses, or testing if the model is in fact consistent with the measured $S(Q, E)$. Four standard types of dynamics models were included in the baseline design for the inelastic software project:

1. lattice dynamics with a Born–von Kármán model (periodic structure)
2. spinwave dynamics with a Heisenberg hamiltonian on a periodic structure
3. lattice dynamics on a disordered structure, using a moments analysis of the dynamical matrix
4. spin dynamics in a paramagnetic model

9.1.4 Direct Experiment Simulation

The third path from data to science is shown on the left of Fig. 9.1. This approach is a direct simulation of the data measured at the detectors. It is based on Monte Carlo codes that are used in the neutron community for simulation of instrument performance. These codes have been tested and validated against the performance of real instruments. We will put these Monte Carlo simulations together with molecular dynamics simulations to perform direct simulations of experimental data. A number of molecular dynamics simulations are available to the theory community. These simulations are complementary to the analytical models of Sect. 9.1.3, and are sometimes advantageous. For example, no approximations are needed when implementing structural models of disordered solids, which are not handled well by the methods of Sect. 9.1.3 (except when the Q information is ignored as in methods 3 and 4).

One possibility is to include the sample into the Monte Carlo computations as a component of the instrument, transforming an individual incident neutron into a neutron scattered into the detector array. Alternatively, simulation results for the primary flight path could be stored and used for several simulations. For lattice dynamics simulations, we are tentatively planning on treating the scattering as individual events in the first Born approximation. This is equivalent to sampling the velocity-velocity correlation function of the atoms in the sample. These motions will be obtained from a molecular dynamics simulation, embedded as a core in a Python component. This same path is followed for classical spin dynamics simulations.

9.2 An Architecture for Distributed Data Analysis

9.2.1 Overview

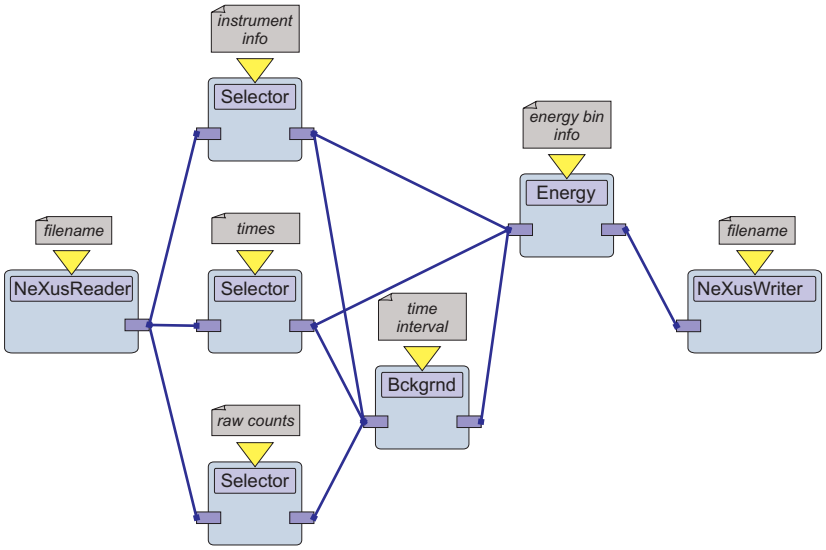


Fig. 9.2. Schematic of the conversion of raw time of flight data into an energy histogram.

Most common data analysis tasks can be cast as the results of *components* acting on *data streams*, very much like the electrical engineer's concept of a signal processing system. As an example, consider Figure 9.2. The module `NeXusReader` is responsible for reading a file that contains raw data and instrument information in some standard format and converting it into a data stream. This stream is fed to three filters, each of which selects a particular

subset of the information in the stream, such as instrument information, arrival times and raw detector counts. The outputs of the three filters are data streams that are in turn fed to `Bckgrnd`, which corrects for the instrument background. The conversion to a histogram of intensity as a function of energy is carried out by `Energy`, which requires details about the instrument, the times of flight and the background-corrected counts from `Bckgrnd`. In this example, the resulting stream is fed to `NeXusWriter` for storing in a file, but one can easily imagine that other components, perhaps those of a visualization system, might be involved in the further manipulation of the data stream. Similarly, the input stream in this example was generated by reading a file. However, one can easily envision an interface to the instrument that makes the data available as it is collected by an experiment in progress.

The diagram in Figure 9.2 is sufficiently intuitive that certain conventions are easily inferred. For example, most people deduce correctly that data streams flow from left to right, hence the protrusions to the left of a component represent its inputs and those to the right its outputs. Further, components may require information such as file names, or energy bin layout, that is best provided by the user rather than another component. This information is represented by the gray boxes above the yellow triangles. Most readers also infer that components *nest*, and complex modules such as `Bckgrnd` and `Energy` are themselves implemented in terms of lower-level components, details of which are not shown for the sake of clarity.

The *DANSE* data analysis software is organized using this *data flow* paradigm as the basic abstraction. The architecture is a set of services that enable the encapsulation of the computational engines, establish the transport of data between these engines, and provide uniform access to user input. The framework shields the computational engines from the user interface, allowing the construction of interfaces that are suitable for a variety of end-user environments. Advanced user interfaces allow the direct manipulation of the analysis network and provide visual ways to interact with the components and display the results of the analysis.

The remainder of this section describes in detail some of the framework services and their interactions.

9.2.2 Components

The bulk of the intellectual capital in data analysis software lies in the routines developed by instrument scientists and expert users that enable the physical interpretation of the raw detector signals, or facilitate the comparison with theoretical models. Certain clusters of these routines carry out identifiable higher-level tasks and have enough in common to constitute reusable modules. A major goal of the proposed architecture is to enable scientists to make contributions to a sophisticated software package with as little exposure to the complexity of the system as possible. Conversely, the rest of the

software should be as shielded as possible from the requirements of a particular computational engine. A *component* is the architectural element that acts as the mediator between a low-level module and its environment. We will refer to these low-level routines as the component's *core*, as illustrated in Figure 9.3.

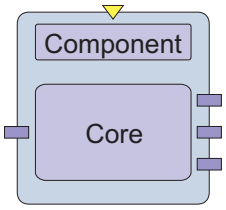


Fig. 9.3. Schematic representation of a software component.

A component is responsible for the initialization of its core, which may involve delivering user-supplied information. We will refer to such information as the component's *properties*. When a component is instantiated, it provides the framework with a table of pairs of strings with property names and default values. The framework makes this information available to the user interface, which is responsible for soliciting the user's input. The property table is read at component execution time. It is used to initialize the core by converting each property string to its native representation. This mechanism bypasses much of the complexity of component initialization and allows for the complete decoupling of the core from the user interface.

Components interact with each other by exchanging information in data streams. Streams are a useful abstraction because they promote a weak, standardized coupling between components. For each application domain, there is typically only a handful of different data types that are exchanged between components. For the analysis of neutron scattering data, the majority of data exchanges involve *tables* and generalized *histograms*, described in more detail in Section 9.2.3. Components negotiate the actual details of the data exchange when they are first connected to each other.

When a stream delivers data at the *input port* of a component, the component must make the data available to its core. Conversely, the component must place results in the data stream connected to its *output ports*. The details of this process depend rather strongly on the choices of programming languages, operating systems and compilers. Fortunately, these platform dependencies are rather generic and can be handled as part of the services offered by a component[1].

[1] This implies delegating part of the task of resolving the platform dependencies to the configuration management system.

Typical component cores manipulate large data sets and can be computationally intensive. They are appropriately implemented in low-level languages, such as `FORTRAN` and C. To the extent possible, the *DANSE* software is language neutral so it can provide a forward migration path for legacy codes. The software provides explicit support for parallel programming.

The work required to transform a data analsis program into a component is discussed in Sect. 9.3 below. Common tasks, such as accessing the contents of data streams, are done with a standardized interface. For certain types of components, such as those that manipulate a single data stream, upgrading may involve little beyond using the services provided by the framework. However, component cores have somewhat more stringent quality requirements than the typical routines in monolithic codes that run on a single workstation. Software defects, such as those that lead to core dumps or, even worse, memory leaks, are much more destructive and annoying to the user when they disrupt a large scale distributed computation that involves the allocation of scarce resources in remote facilities. Clearly, contributed components that are targeted for wide usage by typical users must go through extensive testing and a quality approval process.

For similar reasons, component cores cannot simply abort a computation that failed to converge or appears to be going astray by calling system routines such as `exit` that cause the unconditional termination of execution. Further, practices such as printing to the user console become much more complicated for components that may not have a user console or may not even have a controlling terminal. Components have access to a centralized mechanism for structured logging of status and error messages. Legacy cores may require some minor re-engineering to comply with this requirement.

9.2.3 Data Streams

Data streams are the conceptual encapsulation of the mechanism for data exchange among components. Streams can be thought of as single-port components that merely copy data present in their input port to their output port. They are an essential architectural element because they enable the decoupling of components from one another and hide the details of the data transport mechanism.

The *conceptual* decoupling of components from each other through the use of streams enables their *physical* decoupling. This makes it possible to distribute the computation among multiple process spaces. The user can choose whether components reside in the same process space, as separate threads, as separate processes on the same machine or are deployed across the network. Connections that cross process boundaries can discover where the components they connect are physically deployed and then determine an appropriate mechanism for data transfer. The choice is completely transparent to the components themselves and therefore the control and data transfer mechanisms can be implemented independently by an expert. Further, one can

take advantage of emerging protocols and services for distributed computing without disturbing any of the existing components.

Connections between components are established at run time when one cannot rely on compilers to ensure that an input port receives data compatible with its expectations. A frequently-used (but extreme) solution is to bypass the problem by allowing streams to carry only one data type, such as three-dimensional arrays of doubles. At the other extreme is a solution commonly employed by industrial-strength component systems that allows components to exchange arbitrary data types. A specialized language allows components to describe the types of data streams they consume or generate, providing in essence a run time typing system. The first extreme is rather restrictive, while the latter is fairly complex to implement and maintain. Instead, we propose a compromise that takes the requirements of our specific application domain into proper account and restricts the data stream types without sacrificing generality.

We have identified two abstract types, *tables* and *histograms*, that appear to be sufficiently general to satisfy the data exchange needs of our components for the analysis of neutron scattering data. Tables are inspired by the database concept of the same name. Potential uses include the storage of raw events in a detector. The data are conceptually organized in rows, each of which has the same number of named columns. A description of the table stores the name and data type of each column, along with the number of rows, if known. This information constitutes the *table meta-data*. Histograms are a generalization of multi-dimensional dense arrays of floating point numbers and are the typical structure for exchanges between components that manipulate processed data. The *histogram meta-data* consists of the number of dimensions plus an array for each dimension that describes the histogram bins. Other data types may be added in the future if enough components exchange information not expressible as histograms or tables.

There is a clear separation of large data sets from the lightweight information that describes them. Data storage and data transport are entirely opaque to the data producers and consumers, and these data handling functions are considered implementation details that differ for each platform. Transport of the actual data is considered an expensive operation, since it scales with the amount of data in the stream, whereas the meta-data can be exchanged freely between components. We will construct a simple stream description language based on XML to allow components to explain to each other the content flowing in the data streams connected to their ports. The meta-data will be part of the negotiation protocol for component connections and will be used to issue proper diagnostics when irreparable incompatibilities are detected.

9.2.4 Implementation

The software is written as components in Python, a modern very-high-level computer language, and as C++ cores with Python wrappers. The C++ cores are then accessible as Python functions, but they perform as compiled code. The Python interpreter makes it easy to arrange components into custom scripts without recompilation. Interpreted Python scripts provide users with great flexibility in constructing custom analysis procedures from a well-stocked software toolkit. Scripts for common types of data analysis will be nested into "one-click" analysis packages, but their customization should be readily possible.

Data analysis as a service accessed through an Internet web site is illustrated in Figure 9.4. Components can be maintained and run centrally on well-tested platforms. A compute server, not the user, arranges for computation on the appropriate hardware. A user could elect for little code and no raw data to reside on his or her local computer, while still directing the reuse and reconfiguring of the needed Python components, including those that run on specialized hardware such as Beowulf clusters. Our intent is to ensure that any user can utilize the highest-performance hardware without buying and maintaining it.

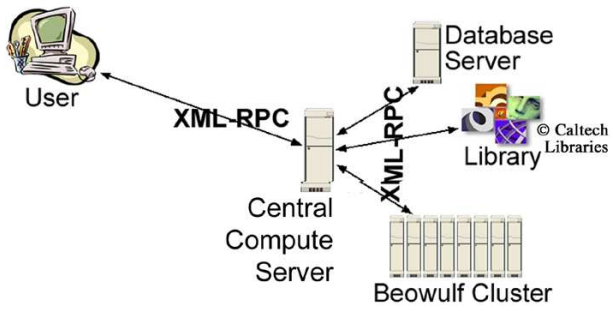


Fig. 9.4. Architecture for distributed computing.

Today the computing resources are located at Caltech, where we are relatively free to experiment with them. The essence of a working web portal and two user interfaces have been demonstrated. In one interface, the user logs on to our web portal through a standard browser such as Internet Explorer, and receives a Java applet that runs under the browser. The applet provides a "Labview-like" graphical user interface (GUI) where the user "wires together" a set of data analysis components. Changing the present GUI is not difficult because of a clean separation between the user's depiction of the data analysis procedure, and the server's execution of the data analysis procedure. With this separation, a user could select among several GUIs, depending on needs or preferences. We have already tested two of them.

After the user arranges an analysis procedure in the GUI, the Java applet transmits instructions to the web server using XML-RPC protocols. The software on the server arranges an appropriate Python script and executes it. Execution may occur locally or on other computing resources, today accessed by XML-RPC protocols. The response to the user is through his or her Internet browser.

Many of the data analysis components could be selected to run locally on the user's computer. Nevertheless, software released for running on a user's own computer will include automatic and transparent Internet access to the *DANSE* compute server. Such a remote access to the *DANSE* system will overcome a traditional problem with software releases for a user's local computer, where software developers must design for the least-capable hardware. The user could specify what parts of the analysis would be performed locally or remotely. With such access to the central service, capabilities for large simulations on the *DANSE* system, for example, would be available to the user through a familiar software package.

In principle, the *DANSE* architecture can serve all instruments at all neutron sources. Moving this principle into practice is underway as of this writing in 2003.

9.2.5 Advantages of a User-Directed, Distributed Architecture

- The architecture offers access to the best combination of hardware and software.
- The architecture allows the user to select which processes are executed locally, and which ones are distributed automatically to the central *DANSE* hardware. The user can analyze data with any platform, although it may be advantageous to have a local capability for some frequently-used computations.
- When experimental data and analysis codes reside on central servers, interaction with the data across the Internet requires minimal data bandwidth.
- A clean separation of the user interface from the analysis code allows interchanging the user interface without affecting the core service.
- Centralization of the main codes simplifies their maintenance, and computing resources can be changed without affecting the user.
- One web portal will serve all neutron instruments. This consistency is possible for other neutron facilities too.

9.2.6 Extensibility by Scientists

Python scripts can be prepared for routine analysis, and used easily. Developing code for the proposed architecture can be done at three levels of increasing complexity. New users may be satisfied by the first level. Arbitrary data analysis is possible at the second level, but optimal performance is offered by the third level.

1. A user could select a complete script from a menu of tested procedures. A set of standard data reduction and visualization could be performed as a "one-click" operation.
2. For altering existing Python scripts, the entry barrier is very low. This low barrier is possible with a "Labview"-style interface that allows users to "wire-up" modules of data and code through a graphical user interface.
3. For writing new Python code, the entry barrier is as low as possible with any computer language. Python programs are interpreted and easy to debug. Their performance is comparable to what is expected of proprietary scientific data analysis packages such as IDL or Matlab, which may run some two orders of magnitude slower than compiled code. If this is a difference between 10 microseconds and 1 millisecond, compilation may be unnecessary. The user can decide when it is appropriate to make the transition from Python to compiled code.
4. For optimal computing performance, a transition path to a lower-level language such as C++ is available. Code written in C++ needs Python bindings so the entry points to the compiled C++ code and data are known to the Python interpreter. Writing these bindings can be done by scientists, although this involves understanding some details of how Python functions are implemented. Automated tools exist today to do this reliably, and they could do the bindings elegantly if they were developed further in the proposed effort on central resources. Another important product of the central resource effort will be a standardized object model to allow low-level codes written in `FORTRAN`, C, or C++ to be incorporated as cores of Python components. Writing these C++ extensions for Python is the subject of the next section.

9.3 Extending *DANSE*: Writing C++ Extensions to Python

Important sources of information on this topic are available elsewhere: the Python extension and embedding manual, and the Python-C API reference manual. These documents are available at the Python website: http://python.org, but better yet, just look in the 'ext' and 'api' subdirectories of your Python distribution's Doc directory for HTML versions.

This section was written to supplement those documents, first by giving an example of dynamically allocating C++ objects and keeping track of them, and second by introducing some of the API functions for working with aggregate types like tuples and lists, for which the easiest Python-C conversion tools don't work. Also, I've concocted some simple, try-this-at-home examples to illustrate the process. Finally, I've added a more real-life example.

Note: In the following, C++ is used to mean both C and C++. The Python-C API is written in C, but for the most part, but that can be called

seamlessly from C++. There's only one item that must have C linkage, the init function, described below.

9.3.1 Overview

Why write C++ extensions to Python? To reuse existing code, and to gain better performance. A great deal of software has already been written in C and C++ (not to mention FORTRAN), and, at least at the present, nothing beats compiled languages for performance in the numerically intensive codes that *DANSE* supports. Extending Python allows us to turn all that code into building blocks for solutions. Libraries of extensions package those building blocks into kits that users can adapt to solve their problems.

Of course you can do all that without Python, so why use it at all? Python has an easier learning curve than C++, and it's more flexible and immediate, making it available to a wider set of users. Once a library of extensions is available on a platform, users can call out to that library, mixing and matching components and testing combinations, without the overhead of compiling and linking. (This immediacy should not be underestimated.) And with all its standard library packages, Python can be manipulated to do some pretty amazing things that would be more difficult to realize in C++, like parsing XML documents, setting up computations, etc.

Writing C++ extensions for Python can be learned in an afternoon, especially if there are some examples to follow.[2] The idea is this: you have a C++ class or function, and you'd like to make it callable from the Python interpreter. You'll need to (I) write some wrapper code in C++, and (II) compile it into a C++ library that the Python interpreter can dynamically load and use. You then typically (III.) write a Python module that mediates between the Python user and this library. This makes life plush for those users who don't want to be bothered with the details of finding out what's in the library and how to call it. More importantly, it gives us the chance to check inputs as soon as possible for errors. When a bad pointer is sent to the C++ level, the results are disastrous (a core dump on a good day), so our software can be made more robust if we handle the pointers ourselves.

It takes more care to create a library that is complete: catches exceptions, checks that preconditions and postconditions have been met, and so on. These

[2] If it's so easy, why hasn't someone written a program to write the wrappers automatically? They have! Packages like the Simple Wrapper Interface Generator (SWIG, `http://www.swig.org`) will do nearly all of the work for you. There are arguments for and against automatic code-generators like SWIG, and I've worked both ways. At the moment, writing the bindings is such an easy task, and I do so relatively little of it, that I prefer to do it myself. Others no doubt feel differently, and I have no interest in changing their minds. But even if you're going to use SWIG, or a similar library, there's merit in putting in some time writing your own wrappers to learn how and why things get done. Then you can judge well for yourself which approach suits your situation.

are more advanced topics, covered in other parts of the book (*SOMEDAY NAME A FEW??*). Low-level *DANSE* programmers will be expected to incorporate these techniques into their code, but first things first.

9.3.2 A Little More Detail

Here are those three easy steps again, in slightly more detail:

- Write the bindings. There are three essential components:
  1. Wrapper function(s). (One for each function you want callable from Python). The wrapper typically calls a function or a class method, or it creates a heap object. Once you learn how to write one wrapper, you know how to do it, because all wrappers do the same three things.
  2. Method table. (One entry in the table for each function you want callable from Python). The method table tells the Python interpreter which C++ functions it can call from the library.
  3. init function. (One per module) This is the interpreter's entry point into the C++ library.

  These three steps are accomplished with generous aid from the Python-C extension API.
- Compile it. This is slightly platform dependent (Michael Aivazis's system for processing source code removes this platform dependence for UNIX flavors, including cygwin; Windows is in the works). The goal is to compile into a shared object library (unix) or a dynamically linked library (the beloved Windows dll).
- Call it from Python. One typically writes a Python module that acts as a layer between the user and the C++ library. By doing things like providing Python classes that mirror the C++ classes, one can make the experience quite similar. Or dissimilar. The choice is yours.

9.3.3 A Lot More Detail: Wrappers

Every wrapper function does three things:

a) Converts a Python object with the arguments to the C++ function into C++ objects,
b) calls the C++ function,
c) converts the output to a Python object with the result and return it.

Let's first run through these steps with numbers and strings, for which there's an immediate connection between Python and C++ types; we can use a function, `PyArg_ParseTuple()`, which is built in to the API. Later examples look at tasks like working with C++ class instances and using aggregate Python types, such as dictionaries and lists.

Simple Example: PyArg_ParseTuple, Py_BuildValue
a) Convert arguments from Python to C++

To convert the Python arguments into C++ objects, first define variables of the appropriate type, one for each C++ argument. Then pass the addresses of these variables into `PyArg_ParseTuple()`. This function takes the args tuple, pulls PyObject's out of it, and converts them to C++ types according to a format string.

Here are some examples:

```
//One integer:
int a;
int ok = PyArg_ParseTuple(args, "i",&a);
if(!ok) return o;
//Two integers
int a, b;
int ok = PyArg_ParseTuple(args, "i",&a, &b);
if(!ok) return 0;
//One integer, a string, two doubles
int anint;
char * astring;
double dub1, dub2;
int ok = PyArg_ParseTuple(args, "isdd", &anint, &astring,
                          &dub1, &dub2);
if(!ok) return 0;
```

A complete list of what can go into the format string is given in the extension documentation (look in the ext subdirectory of the doc directory in your Python distribution, or look online at http://python.org/doc/current/ext/ext.html). In the current (Oct. '02) documentation, you want section 1.7, "Extracting Parameters in Extension Functions".

`PyArg_ParseTuple()` checks the types of the objects in the tuple args against the types given in the format string. If there's a discrepancy, it sets the exception context and returns 0 to our wrapper. If our wrapper detects that, it returns 0 to the Python interpreter, which understands that to mean failure, and raises an exception. So if you're using `PyArg_ParseTuple()`, most of the error checking is done for you! It is not idiot-proof, but it is smart-friendly.

Once `PyArg_ParseTuple` has successfully returned, do any additional processing or checking of the input data that's appropriate. For instance, Python does not have an unsigned integer type. With a C++ function that takes an unsigned int, you'll need to pass an int to `PyArg_ParseTuple`, check that the int is greater than -1, and then convert it to an unsigned int.

b) Call your code

It's your function, call it.

c) Convert output to a Python object, and return it

The API gives a function called `Py_BuildValue`. It returns a pointer to a PyObject; it takes a format string and variables. The format strings are the same as those used in

```
PyObject *py_result = Py_BuildValue("i", result);
```

or—

```
PyObject *py_result = Py_BuildValue("s", astring);
```

etc. What's returned by `Py_BuildValue` is what the wrapper will return. One note: don't return 0; the interpreter will take this as sign of failure. You could, of course, return a Python integer with value zero: `Py_BuildValue("i",0)`. You can specify more than one item to return, in which case `Py_BuildValue` will place the items in a tuple.

Here's a complete example of wrapping a function, "bogus", that takes a double, a string, and an int (in that order) and returns an int. We expect that the arguments will come from Python in the order string, int, double.

```
static PyObject *wrap_bogus(PyObject *, PyObject * args){
 //First, get the arguments from Python
 int anint = 0;
 double adub = 0;
 char * astring = 0;
 int ok = PyArg_ParseTuple(args,"sid",&astring, &anint, &adub);
 if(!ok) return 0;
 //do any checking of arguments here
 //Second, make the function call
 int result = bogus(adub, astring, anint);
 //do any extra stuff you want with the return result here
 //Third, build a Python object to return
 return PyBuildValue("i",result);
}
```

Wrapping classes: PyCObject_FromVoidPtr, PyCObject_AsVoidPtr.

a) Creating C++ objects

Wrapping functions is well and good, but what about C++ classes? Python can work with a C++ object by dynamically allocating it and holding onto a pointer. That pointer can be passed back to subsequent wrappers that can then invoke class methods on the object. The pointer is handled in Python by a type PyCObject. To convert a pointer to a PyCObject, use `PyCObject_FromVoidPtr()`. This API function takes two arguments: the void pointer, and a pointer to a function that takes a void pointer and no return. The purpose of the function is to delete the C++ object when nothing in the Python session is paying attention to it any more.

We get the arguments to the constructor from the Python API, create the object using new, and return a pointer to that object to the interpreter. Use the API function `PyCObject_FromVoidPtr()` to create the Python object to return to Python. Here's an example with a real (if dull) class called Numbers:

```
class Numbers
```

```
{
public:
 Numbers(int first, double second) : m_first(first), m_second(second){;}
 ~Numbers(void){;}
 double NumMemberMult(void){return (double)m_first*m_second;}
private:
 int m_first;
 double m_second;
};
```

Here's a wrapper that creates a new instance of Numbers:

```
PyObject *wrap_new_Numbers(PyObject *, PyObject* args){
 //First, extract the arguments from a Python tuple
 int arg1;
 double arg2;
 int ok = PyArg_ParseTuple(args,"id",&arg1,&arg2);
 if(!ok) return 0;
 //Second, dynamically allocate a new object
 Numbers *newnum = new Numbers(arg1, arg2);
 //Third, build a Python object to return
 PyObject * py_newnum = PyCObject_FromVoidPtr( static_cast<void *>(newnum),
 &DelNumbers);
 return py_newnum;
}
```

Look familiar? This wrapper has essentially the same form as `wrap_bogus()`. That's because ALL wrappers have essentially this form.

The pointer to the dynamically allocated object, newnum, goes out of scope as soon as `wrap_new_Numbers()` returns. The only thing keeping this from being the mega-classic memory leak is that the Python interpreter has an object that will keep track of the address of the new object. The interpreter keeps track of that object for us, and when we lose interest in it (when its reference count goes to zero), the interpreter will call a C++ function to delete the C++ object. So, the second slot in `PyCObject_AsVoidPtr()` is a pointer to a function with return type void and one void argument. The signature of `PyCObject_AsVoidPtr()` is:

```
PyObject * PyCObject_FromVoidPtr( void *, void (*DeleteFunction)(void*));
```

You must supply the function pointed to (in this example called DelNumbers). It has the delete corresponding to the new above. Here's it is:

```
static void DelNumbers(void *ptr)
{
    Numbers * oldnum = static_cast<Numbers *>(ptr);
    delete oldnum;
    return;
}
```

This strategy can be used for any dynamically allocated resource, such as file handles or arrays.

Using the object

The user can't actually do anything with the pointer in the Python layer, except send it back to the C++ layer to do something else: call a C++ class method on it, or give it as an argument to another function. Here's an example of wrapping a class method.

```
#include <Python.h>
...
PyObject *wrap_Numbers_MemberMult(PyObject *, PyObject* args)
{
 // First, extract the PyCObject that has the
 // Python version of the address
 // from the args tuple
 PyObject *pynum = 0;
 int ok = PyArg_ParseTuple(args, "O", &pynum);
 //"O" is for Object
 if(!ok) return NULL;
 //Convert the PyCObject to a void *
 void * temp = PyCObject_AsVoidPtr(pynum);
 //cast void pointer to Numbers pointer
 Numbers * thisnum = static_cast<Numbers *>(temp);
 //Can combine the two lines into one:
 //Numbers *thisNum = static_cast<Numbers *>(
 //                        PyCObject_AsVoidPtr(pynum));
 //Second, make the function call
 double result = thisnum->NumMemberMult();
 //Third, build a Python object with the return value
 return Py_BuildValue("d",result);
}
```

All you have to do is fish the PyCObject out of the tuple, extract the void pointer to a C++ variable, cast it to the appropriate type, and use it; then bundle up the result and send it back to the interpreter.

Working with composite types. What if you want to pass a Python list of numbers to a C++ function? There's no format code to pass to `PyArg_ParseTuple` for lists. The solution is to extract the list from the `args` tuple as a PyObject (format code: "O"). Then use the Python-C API functions for working with lists to load the Python list, item-by-item, into a C++ array. Suppose our target function has the signature

```
double sum_some_numbers(double *numbers, int array_length)
```

Here's some code that could wrap this function. Note that we have to take responsibility for error checking. We can set the exception using `PyErr_SetString()`. We can verify that Python objects are the *type* what we think they are by Py*type*_Check().

```
#include <Python.h>
#include <valarray> // This example uses the std::valarray class.
...
PyObject *py_sum_some_numbers(PyObject *, PyObject* args)
{
    PyObject *pyList;
    int ok = PyArg_ParseTuple(args, "O", &pyList);
    if(!ok) return 0;
```

```
    //Did the user send a Python list?
    int isList = PyList_Check(pyList);
    if(!isList)
    {
        //If not, complain to the Python user and raise an exception:
        PyErr_SetString(PyExc_TypeError, "You fool! That's not a list!");
        return 0;
    }
    //How many items are in the list?
    int numNums = PyList_Size(pyList);
    //Maybe you want to do something here if the size of the list is 0.
    // Now transfer the contents of the list to an array
    // valarray: this C++ standard library class is a great
    // way to avoid memory leaks, and much more
    std::valarray<double> nums(0.0, PyList_Size);
    for(int i=0; i<numNums; i++)
    {
        //Extract the next object in the list:
        PyObject *temp = PyList_GetItem(pyList, i);
        // Was the list item a Python float? If not, quit.
        // Note that we don't need to worry about cleaning
        // up the memory: nums will be automatically destroyed
        // when execution exits the scope of nums.
        if( !PyFloat_Check(temp))
        {
            //Just what was in that list?
            PyErr_SetString(PyExc_TypeError, "You fool! That's not a float";
            return 0;
        }
        // Now convert the Python float to a double, and load
        nums[i] = PyFloat_AsDouble(temp);
    }
    //Step 2: Call the function
    double sum = sum_some_numbers(&nums[0], nums.size());
    //Step 3: return result.
    return Py_BuildValue("d", sum);
}
```

Sometimes you'll want to return a Python object, such as a list. In this case, you'll have created a PyObject pointer at some point. You can return that pointer directly. If you need to return several Python objects, you can use `Py_BuildValue()` with the "O" format code.

The Python-C api is very complete and well-documented. Similar functions exist for inserting objects into lists, and working with dictionaries, tuples, modules, and so on. Consult Chapter 7 of the api reference. Hopefully, these examples have given you the flavor for wrapping C++ functions.

9.3.4 A Lot More Detail: Method Table

The method table is sort of like a table of contents for the Python interpreter. When it loads the library, it reads the method table to find pointers to the functions in the library.

```
static PyMethodDef numbersMethods[] = {
    {"PyNumbers", wrap_new_Numbers, METH_VARARGS,
         "Create new Numbers object"},
    {"PyNumbers_MembMult", wrap_Numbers_MemberMult, METH_VARARGS,
         "Multiply Numbers object's members"},
    {NULL,NULL}
};
```

The name of the table must match the second argument in the init function. Each function in the library gets an entry in the table, and each entry has four components.

1. The string ("PyNumbers" or "PyNumbers_MemberMult") is what you'll call from the interpreter.
2. The name `wrap_whatever` is the name of the corresponding C++ function.
3. `METH_VARARGS` indicates that one is using the "tuple named args" approach.
4. The final string will appear as the docstring for this function in the Python layer.

The `{NULL,NULL}` marks the end of the table for the interpreter.

9.3.5 A Lot More Detail: Init Function

The final component in the bindings is the init function. This function has to be named init*name_of_library*(). If the filename of the library is numbers.dll or numbers.so, this function must be named `initnumbers`; if it's _numbers.dll, then this function is named `init_numbers`. For this example, let's call the extension library _numbers. The init function's return type is void, and it takes no arguments. It calls `Py_InitModule()`, which takes two arguments: a string literal with the name of the library, and the name of the Methods table. It *must* match the name of the Method table. Also, the function must have C linkage, not C++, meaning the function must be declared extern "C" if you're using a C++ compiler. Also, on Windows, the function must be exported by the dll, hence the `__declspec(dllexport)`. To keep some platform independence, wrap this in a pre-processor conditional. This function is executed when the library is loaded, so if there's other initialization steps you need to take, this is the place.

```
extern "C"
#ifdef WIN32 || _WIN32
__declspec(dllexport)
```

```
#endif
void init_numbers(void)
{
    (void) Py_InitModule("_numbers", numbersMethods);
}
```

9.3.6 More Detail: Compile

Compiling the Numbers example under Linux. Here's a way to compile the Numbers example under Linux. You'll need files with the Numbers source code (Numbers.cpp and Numbers.h), and the wrapper, Numbers_bindings.cpp. I assume you're using gcc; if not, you'll need to modify the compiler flags appropriately.

To compile, compile each source file:

```
gcc -I/usr/include/python2.2 -I. -c -fpic  Numbers.cpp
gcc -I/usr/include/python2.2 -I. -c -fpic Numbers_bindings.cpp
```

and link them into a shared library:

```
gcc Numbers.o Numbers_bindgins.o -lm -lc -fpic -shared -o _numbers.so
```

In the compile lines, you'll of course need to make sure that you've pointed to the directory where your Python.h file lives. The "-fpic" specifies position independent code, "-shared" a shared library that can be dynamically linked.

Once you've compiled _numbers.so, move it into a place on your system's PYTHON_PATH and go to town.

Compiling the Numbers example under Windows. Well, of course we want all our ARCS modules to run under Linux/Unix, but for all those times Windows needs a helping hand, here's how to do it:

For working in Windows, it may be best to use MS Visual C++. Here's what you'd do to create a project and so on in VC7.

1. From the Start page select New Project.
2. From the Project Types, pick Visual C++ Projects, from the Templates, choose Win32 project. Fill in the name for your library.
3. On the next window, pick Application Settings, and set Application Type to DLL.
4. In the Solution Explorer,
   - get rid of stdafx.cpp
   - right-click on the name of the project, and choose Properties
   - Under the C/C++/General folder, add the additional include directory in which your distribution's Python.h file resides.
   - Under the Linker/General tab, set the output file to _your_project_name.dll. Note that everything up the ".dll" must be the same as what follows "init" in the `init_your_project_name` function in the bindings. In the Numbers example, we called that function init_numbers (it's the last function in the file). So the dll has to be called _numbers.dll. If the

name of the file and the init function don't agree, the interpreter will get lost.

5. In Solution Explorer, look in the file your_project_name.cpp– you'll need to get rid of that crap about APIENTRY DllMain. Better yet, just get rid of the automatically generated code.

Now fill add the source files Numbers.cpp, Numbers_bindings.cpp, etc. Build, and when the _your_project_name.dll appears in the output directory, move it into your PYTHON_PATH, and enjoy.

9.3.7 More Detail: Call it from Python

In keeping with the Numbers example used above, here's a Python class called Numbers. It's a "shadow class" for the C++ Numbers class. Pretty much everything you can do with the C++ class can also be done with the Python.

```
import _numbers
class Numbers:
    def __init__(self, an_int, a_float):
        #Check an_int
        if type(an_int) != type(1):
            raise TypeError, "Fool! an_int must be an integer"
        #Check a_float
        if type(a_float) != type(3.14159)
            raise TypeError, "Fool! a_float must be a float"
        self.this = numbers.PyNumbers(an_int, a_float)
    def MemberMult(self):
        return numbers.PyNumbers_MembMult(self.this)
```

If this were saved in a module named numbers.py (stored somewhere on your PYTHON_PATH), then an interpreter command line session might look like

```
>>> import numbers
>>> n = numbers.Numbers(2,3.14)
New Numbers object created
>>> n.MemberMult()
6.280000000002
>>> n = 1
```

The variable n is the only thing keeping track of the Numbers object. When we reassign n, the interpreter calls our bit of code from the C++ library that deletes the object, preventing the resource leak.

So now the Python user has something like "interpreted C++". Pretty cool, eh?

9.3.8 More realistic example

Let's wrap a function from the NeXus API, NXopen, with all the bells and whistles. This wrapper is part of a larger library that wraps the entire NeXus

C API. The latter, of course, is a simplified interface to the HDF libraries that the NeXus standard currently uses.

Since this is one of several dozen libraries, we split the bindings up into several files. The wrappers live in pairs of files, one pair for each major functional group: file.h/file.cc for file level operations, group.h/group.cc for group level, etc. Only two file level operations from the original NeXus C API are in the Python NeXus API: NXopen and NXflush. Therefore, file.h looks like the following:

```
#ifndef NeXus_file_h
#define NeXus_file_h
// Python bindings for file level operations:
// NXopen
extern char pyNeXus_NXopen__name__[];
extern char pyNeXus_NXopen__doc__[];
extern "C" PyObject * pyNeXus_NXopen(PyObject *, PyObject *args);
// NXflush
extern char pyNeXus_NXflush__name__[];
extern char pyNeXus_NXflush__doc__[];
extern "C" PyObject * pyNeXus_NXflush(PyObject *, PyObject *args);
#endif
```

This file will be included into the file that contains the methods table. Note that in addition to the actual wrapper functions, we declare two char arrays for the name and docstring. Defined in file.cc, these variables keep the methods table neat (the docstrings in particular may get lengthy).

How do we implement the wrapper for NXopen? Begin with the signature of NXopen, located in napi.h in the NeXus source distribution:

```
NX_EXTERNAL  NXstatus CALLING_STYLE NXopen(CONSTCHAR * filename,
                NXaccess access_method, NXhandle* pHandle);
```

The various types NXstatus, CONSTCHAR, NXaccess, and NXhandle are defined in the NeXus C API header files; we need to track them down so we can know what the Python user will have to give us in order to satisfy the function call.

Searching through napi.h, we learn that `NXstatus` is a `typedef` for `int`, `CONSTCHAR` is a `typedef` for char,[3] `NXaccess` is an enumeration with members like `NXACC_READ`, and `NXhandle` is a `typedef` for `void *`. We can't map the `NXaccess` enumeration directly into Python types, so we'll expect a string from the user; by comparing values of the string we'll assign the proper value to an `NXaccess` variable. As for the `CONSTCHAR * filename`, we can derive that directly from a Python string.

What about the `NXhandle *pHandle`? This is interesting. Let's open up the NeXus C API source code and find out what exactly is done with that void pointer. Reading through napi.c, we discover that `NXopen` dynamically

[3] Prefering to not mislead the readers of our code, we might have chosen CONSTCHAR as typedef const char.

allocates a structure of type `NexusFunction`, and that the `NXhandle` we pass to `NXopen` becomes a handle to that object. In other words, `pHandle` is an output of `NXopen`, not an input. We don't need to trouble the Python user with giving us a `NXhandle` object; instead, we'll give them one.

Expecting two inputs, both Python strings, we write the first few lines of the wrapper as follows:

```
PyObject * pyNeXus_NXopen(PyObject *, PyObject *args)
{
    char *filename = 0;
    char *acc_method = 0;
    int ok = PyArg_ParseTuple(args, "ss", &filename, &acc_method);
    if(!ok) return 0;
```

and we've got what we needed from the Python user. Or do we? We owe it to ourselves to check the inputs from the user. This could be done either here or in the Python layer. Checking that the inputs are Python strings has already been performed by `PyArg_ParseTuple`. There are two questions about the filename: is it the name of an actual file, and is it the name of an appropriate HDF file? The first question is easily answered in a platform independent way in Python. The second can only be answered by essentially doing what we do anyway in `NXopen`. We need to inspect the value of `acc_method` anyway to convert it to the appropriate member of the NXaccess enumeration. We can do that easily using the C++ standard library class `string`:

```
//Check access_method:
std::string methstring(acc_method);
NXaccess mode;
if     (methstring == "r" ) mode = NXACC_READ;
else if(methstring == "rw") mode = NXACC_RDWR;
else if(methstring == "c" ) mode = NXACC_CREATE;
else if(methstring == "c4") mode = NXACC_CREATE4;
else if(methstring == "c5") mode = NXACC_CREATE5;
else
{
    std::string errstr("NeXus_bindings.cc pyNeXus_NXopen(): ");
    errstr += "unrecognized access_method string.";
    PyErr_SetString( PyExc_ValueError, errstr.c_str() );
    return 0;
}
```

Here we've used `PyErr_SetString` to set the (Python) exception context if we don't recognize what the Python user wants, and then forced the interpreter to raise the exception by returning 0.

At this point, we're nearly ready to call `NXopen`—we only need to declare a variable of type NXhandle:

```
NXhandle handle;
NXstatus status = NXopen(filename, mode, &handle);
if(status != NX_OK)
{
    std::string errstr("NeXus_bindings.cc pyNeXus_NXopen(): ");
```

```
        errstr += "NXopen failed.";
        PyErr_SetString(PyExc_IOError, errstr.c_str() );
        return 0;
    }
```

If we get to this stage, we're almost ready to return. Since we are not content with any `NXstatus` but `NX_OK`, all we need to do is to return the `NXhandle` initialized by `NXopen`. But there's another issue here: `NXopen` allocates a resource, so we must release that resource when we're finished. As with the discussion of allocating C++ objects (§ 9.3.3), we can use `PyCObject_FromVoidPtr` and a helper function to release the resource when the Python user is finished:

```
        return PyCObject_FromVoidPtr(handle, pyNeXus_NXclose);
    }
```

The helper function (declared `static` to avoid) wraps `NXclose`, which in turn disposes of the resources allocated in `NXopen`:

```
    static void pyNeXus_NXclose(void *file)
    {
        NXhandle oldnxh = static_cast<NXhandle >(file);
        NXclose( &oldnxh);
        return;
    }
```

Finally, we define the name and docstring for this function. The name should be something sensible, while the docstring is an opportunity to incorporate a little documentation.

```
    char pyNeXus_NXopen__name__[] = "nxopen";
    char pyNeXus_NXopen__doc__[] = "Open a nexus file\n"
    "2 Arguments: filename, access_method\n"
    "Input: \n"
    "      filename (Python string)\n"
    "      access_method (Python string)\n"
    "           allowed values: r (read only)\n"
    "                           rw (read/write)\n"
    "                           c (create)\n"
    "                           c4 (create HDF 4)\n"
    "                           c5 (create HDF 5)\n"
    "Output: (return)\n"
    "       PyCObject holding pointer to NXhandle\n"
    "Exceptions: ValueError, IOError\n";
```

9.4 Data Stream Protocols

How much should we say about this?

Further Reading

The contents of the following are described in the Bibliography.

H. A. Abelson and G. J. Sussman: *Structure and Interpretation of Computer Programs* (MIT Press, Cambridge Mass, 2001).

Mark Lutz and David Ascher: *Learning Python* (O'Reilly & Associates, Inc. 1999).

Bernard D. Cullity: *Elements of X-Ray Diffraction*, (Addison-Wesley, Reading, MA 1978).

1. B. Fultz and J.M. Howe, *Transmission Electron Microscopy and Diffractometry of Materials* (Second Edition, Springer–Verlag, Heidelberg, 2002).
2. T. Egami and S.J.L. Billinge, *Underneath the Bragg Peaks: Structural analysis of complex materials* (Pergamon Press Elsevier, Oxford England, 2003).
3. M. Aivazis, W.A. Goddard, D. Meiron, M. Ortiz, J. Pool, and J. Shepherd, "A Virtual Test Facility for Simulating the Dynamic Response of Materials," Computing in Science and Engineering **2**, 42 (2000).
4. TeraGrid `http://www.teragrid.org/`
5. Center for Advanced Computing Research `http://www.cacr.caltech.edu/`
6. Python Programming Language `http://www.python.org/`
7. M. Lutz and D. Ascher, *Learning Python* (O'Reilly, Sebastopol, CA 1999).
8. Extensible Markup Language (XML) `http://www.w3.org/XML/`
9. E.T. Ray, *Learning XML* (O'Reilly, Sebastopol, CA 2001).
10. The MathWorks - MATLAB `http://www.mathworks.com/products/matlab/`
11. Research Systems, Inc. - IDL Software `http://www.rsinc.com/idl/index.asp`
12. WaveMetrics IGOR Pro `http://www.wavemetrics.com/Products/IGORPro/IgorPro.html`
13. Extending and Embedding the Python Interpreter `http://www.python.org/doc/current/ext/ext.html`
14. Python/C API Reference Manual `http://www.python.org/doc/current/api/api.html`
15. VASP Group, Theoretical Physics Department, Vienna `http://cms.mpi.univie.ac.at/vasp/`
16. M.C. Payne, M.P. Teter, D.C. Allan, T.A. Arias, J.D. Joannopoulos, "Iterative minimization techniques for ab initio total-energy calculations - molecular-dynamics and conjugate gradients", Rev. Mod. Phys. **64** 1045-1097 (1992); CASTEP `http://www.tcm.phy.cam.ac.uk/castep/`
17. X. Gonze, J.M. Beuken, R. Caracas, F. Detraux, M. Fuchs, G.M. Rignanese, L. Sindic, M. Verstraete, G. Zerah, F. Jollet, M. Torrent, A. Roy, M. Mikami, P. Ghosez, J.Y. Raty, D.C. Allan, "First-principles computation of material properties: the ABINIT software project", Comp. Mater. Sci. **25** 478-492 (2002); ABINIT HOME PAGE `http://www.abinit.org/`
18. WIEN 2k `http://www.wien2k.at/`
19. Gaussian.Com `http://www.gaussian.com/`
20. F.P. Brooks, Jr., *The Mythical Man-Month – Essays on Software Engineering, Anniversary Edition*, (Addison-Wesley, Reading, Mass 1995).
21. M.A.G. Aivazis and B. Fultz, "DANSE – Distributed Data Analysis for Neutron Scattering Experiments" `http://arcs.cacr.caltech.edu:8000/arcs/uploads/1/danse.pdf`
22. H. Abelson and G.J. Sussman, *Structure and Interpretation of Computer Programs* (McGraw–Hill/MIT Press, Blacklick, Ohio, Cambridge, Mass., 1996).
23. C.A. Jones and F.L. Drake, Jr., *Python & XML*, (O'Reilly, Sebastopol, CA 2002).
24. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns* (Addison–Wesley, Reading, Mass., 1995).
25. J. Lakos, *Large Scale C++ Software Design*, (Addison-Wesley, Boston, 1996).
26. ViPEr: a visual programming environment for Python `http://www.scripps.edu/ stoffler/proj/ViPEr/viper.html`
27. IPNS - Computing - Web Project(ISAW) `http://www.pns.anl.gov/computing/isaw/`
28. H.M. Rietveld, "Line profiles of neutron powder diffraction peaks for structure refinement," Acta Cryst. **22**, 151 (1967). H.M. Rietveld, "A profile refinement method for nuclear and magnetic structures." J. Appl. Cryst. **2**, 65 (1969).
29. McStas - A neutron ray-trace simulation package `http://neutron.risoe.dk/`
30. W.-T. Lee and X.-L. Wang, "IDEAS, a general-purpose software for simulating neutron scattering instruments," Neutron News, **13** 30-34 (2002). `www.sns.gov/ideas`

31. These classic books begin with structure: J.W. Christian *Theory of Transformations in Metals and Alloys* (Pergamon, Oxford 1975). C. Kittel, *Introduction to Solid State Physics* (J. Wiley & Sons, New York 1971).
32. A. Le Bail, H. Duroy, J. L. Fourquet, "Ab initio Structure Determination of $LiSbWO_6$ by X-ray Powder Diffraction." mater. res. Bull. **23**, 447-452 (1988).
33. G.S. Pawley "Unit Cell Refinement from Powder Diffraction Scans," J. Appl. Cryst. **14**, 357-361 (1981).
34. V. Gerold and J. Kern, "The Determination of Atomic Interaction Energies in Solid-Solutions from Short-Range Order Coefficients – An Inverse Monte-Carlo Method," Acta Metall **35**, 393-399 (1987).
35. A.K. Soper, "Empirical potential Monte Carlo simulation of fluid structure." Chemical Phys. **202** 295-306 (1996).
36. S.J.L. Billinge and M.G. Kanatzidis, "Beyond Crystallography: the study of disorder, nanocrystallinity and crystallographically challenged materials," Chem. Commun., in press.
37. R.B. Von Dreele, P.W. Stephens, G.D. Smith and R.H. Blessing, "The first protein crystal structure determined from high-resolution X-ray powder diffraction data: a variant of T3R3 human insulin-zinc complex produced by grinding," Acta Crystallogr. D **56** 1549-1553 (2000).
38. D. Dragoi, E. Üstündag, B. Clausen and M.A.M. Bourke, "Investigation of thermal residual stresses in tungsten-fiber/bulk metallic glass matrix composites," Scripta Mater. **45**, 245-252 (2001).
39. B. Clausen, S.Y. Lee, E. Üstündag, C.C. Aydiner, R.D. Conner and M.A.M. Bourke, "Compressive yielding of tungsten fiber reinforced bulk metallic glass composites," Scripta Mater. **49**, 123-128 (2003).
40. B. Clausen, T. Lorentzen and T. Leffers, "Self-consistent modeling of the plastic deformation of FCC polycrystals and its implications for diffraction measurements of internal stresses," Acta Mater. **46**, 3087-3098 (1998).
41. R.C. Rogan, E. Üstündag, B. Clausen and M.R. Daymond, "Texture and strain analysis of the ferroelastic behavior of $Pb(Zr,Ti)O_3$ by in situ neutron diffraction," J. Appl. Phys. **93**, 4104-4111 (2003).
42. Y.D. Wang, H. Tian, A.D. Stoica, X.-L.Wang, P.K. Liaw, and J.W. Richardson, "Evidence on the Development of Large Grain-Orientation-Dependent Residual Stresses in a Cyclically-Deformed Alloy," Nature Materials, **2**, 103-106 (2003).
43. E. Üstündag, B. Clausen and M.A.M. Bourke, "Neutron diffraction study of the reduction of $NiAl_2O_4$," Appl. Phys. Lett. **76**, 694-696 (2000).
44. M.A.M. Bourke, D.C. Dunand and E. Üstündag, "SMARTS – a spectrometer for strain measurement in engineering materials," Appl. Phys. A **74**, S1707-S1709 (2002).
45. I.C. Noyan and J.B. Cohen, Residual Stress: Measurement by Diffraction and Interpretation, Springer–Verlag, New York (1987).
46. D. Chidambarrao, Y.C. Song and I.C. Noyan, "Numerical simulation of the X-ray stress analysis technique in polycrystalline materials under elastic loading," Metall. and Mater. Trans. A **28**, 2515-2525 (1997).
47. X.-L. Wang, Y.D. Wang, and J.W. Richardson, "Experimental Error due to Displacement of Sample in Time-of-flight Diffractometry," J. Appl. Cryst. **35**, 533-537 (2002).
48. D.C. Montgomery, Design and Analysis of Experiments, 5th ed., John Wiley & Sons, New York (2001).
49. P.J. Withers, M.R. Daymond and M.W. Johnson, "The precision of diffraction peak location," J. Appl. Cryst. **34**, 737-743 (2001).
50. *ABAQUS User Manual*, version 6.3, Hibbitt, Karlsson and Sorensen, Inc., 2002.
51. P.R. Dawson, "Computational crystal plasticity," Int. J. Solids and Structures **37**, 115-130 (2000).
52. A. Guinier, *X-ray Diffraction in Crystals, imperfect Crystals, and Amorphous Bodies* (Dover, Mineola, NY 1994).

53. P.-G. de Gennes *Scaling Concepts in Polymer Physics* (Cornell University Press, Ithaca, NY, 1979).
54. L. Liebler, "Theory of Microphase Separation in Block Co-Polymers," Macromolecules **13**, 1602-1617 (1980).
55. E. W. Cochran, D.C. Morse, and F.S. Bates "Design of ABC triblock copolymers near the ODT with the random phase approximation," Macromolecules **36**, 782-792 (2003).
56. D.I. Svergun and M.H.J Koch "Small-angle scattering studies of biological macromolecules in solution," Reports on Progress in Physics **66** 1735-1782 (2003).
57. M.E. Wall, S.C. gallagher, C.S. tung, and J. Trewhella, "A molecular model of the troponin C/troponin I interaction using constraints from X-ray crystallography, NMR neutron scattering, and cross-linking," Biophysical Journal **78**(1) 366A (2000).
58. P. Chacon, F. Moran, J.F. Diaz, E. Pantos, and J.M. Andreu, "Low-resolution structures of proteins in solution retrieved from X-ray scattering with a genetic algorithm," Biophysical Journal **74**(6) 2760-2775 (1998).
59. G.W. Lynn, M.V. Buchanan, P.D. Butler, L.J. Magid, and G.D. Wignall, "New high-flux small-angle neutron scattering instrumentation and the center for structural and molecular biology at Oak Ridge National Laboratory," J. Appl. Cryst. **36** 829-831 (2003).
60. S. Krueger, "Neutron reflection from interfaces with biological and biomimetic materials," Current Opinion in Colloid & Interface Science **6**, 111 (2001).
61. M.R. Fitzsimmons, S.D. Bader, J.A. Borchers, G.P. Felcher, J.K. Furdyna, A. Hoffmann, J.B. Kortright, Ivan K. Schuller, T.C. Schulthess, S.K. Sinha, M.F. Toney, D. Weller, and S. Wolf, "Neutron scattering studies of nanomagnetism and artificially structured materials," Journal of Magnetism and Magnetic Materials, *in press*. Preprint at `http://www.ncnr.nist.gov/programs/reflect/rp/magnetism/nanomag_21july.pdf`.
62. S.K. Sinha, E.B. Sirota, S. Garoff, and H.B. Stanley, "X-ray and neutron scattering from rough surfaces," Phys. Rev. B **38**, 2297 (1988); R. Pynn, Neutron scattering by rough surfaces at grazing incidence. Phys. Rev. B **45**, 602 (1992).
63. B.P. Toperverg, "Specular reflection and off-specular scattering of polarized neutrons," Physica B **297**, 160 (2001). B. Toperverg, O. Nikonov, V. Lauter-Pasyuk, and H.J. Lauter, "Towards 3D polarization analysis in neutron reflectometry," Physica B **297**, 169 (2001). A. Rühm, B. P. Toperverg, and H. Dosch, "Supermatrix approach to polarized neutron reflectivity from arbitrary spin structures," Phys. Rev. B **60**, 16073 (1999). V. Lauter-Pasyuk, H.J. Lauter, B.P. Toperverg, L. Romashev and V. Ustinov, "Transverse and lateral structure of the spin-flop phase in Fe/Cr antiferromagnetic superlattices," Phys. Rev. Lett. **89**, 167203 (2002).
64. For example, Bede REFS 4.0 `http://www.bede.co.uk/overview.php?overviewID=1015939212.63555`
65. J.F. Ankner and C.F. Majkrzak, "Subsurface profile refinement for neutron specular reflectivity," Neutron Optical Devices and Appl. **1738**, 260 (1992).
66. N.F. Berk and C.F. Majkrzak, "Using parametric B splines to fit specular reflectvities," Phys. Rev. B **51**, 11296 (1995).
67. S.W. Lovesey, *Theory of neutron scattering from condensed matter* Vol. 1 (Clarendon Press, Oxford, 1984).
68. G. Shirane, S.M. Shapiro and J.M. Tranquada, *Neutron scattering with a triple axis spectrometer* (Cambridge Univ. Press, Cambridge, 2002).
69. G.L. Squires, *Introduction to the theory of thermal neutron scattering* (Cambridge Univ. Press, Cambridge, 1978), reprinted by Dover, Mineola, NY, 1996.
70. K. Sköld, D.L. Price, "Neutron scattering," in *Methods of Experimental Physics* Vol. 23, R. Celotta and J. Levine, Eds. (Academic Press, Orlando, 1986).

71. A.F. Yue, I. Halevy, A. Papandrew, P.D. Bogdanoff, B. Fultz, W. Sturhahn, E.E. Alp, and T.S. Toellner, "Mass Effects on Optical Phonons in $L1_2$-Ordered $Pt_3^{57}Fe$ and $Pd_3^{57}Fe$," Hyperfine Interact. **141**, 249 (2002).
72. P. D. Bogdanoff, T. Swan–Wood, and B. Fultz, "The phonon entropy of alloying and ordering of Cu-Au," Phys. Rev. B **68** (1): art. no. 014301 July 1, 2003. Peter D. Bogdanoff, "The Phonon Entropy of Metals and Alloys: The effects of thermal and chemical disorder" Ph.D. Thesis California Institute of Technology, Nov. 20, 2001.
73. G.J. Kearley, "A Profile-Refinement Approach for Normal-Coordinate Analyses of Inelastic Neutron-Scattering Spectra," J. Chem. Soc.-Faraday Trans. II **82** 41, (1986).
74. S. Kasuriya, S. Namuangruk, P. Treesukol, M. Tirtowidjojo, and J. Limtrakul, "Adsorption of Ethylene, Benzene, and Ethylbenzene over Faujasite Zeolites Investigated by the ONIOM Method," J. Catalysis **219** 320 (2003).
75. H.V. Brand, L.A. Curtiss, L.E. Iton, F. R. Trouw and T.O. Brun, "Theoretical and Inelastic Neutron-Scattering Studies of Tetraethylammonium Cation as a Molecular Sieve Template," J. Phys. Chem. **98** 1293 (1994).
76. B.S. Hudson, J.S. Tse, M.Z. Zgierski, S.F. Parker, D.A. Braden, and C. Middleton, "The Inelastic Incoherent Neutron Spectrum of Crystalline Oxamide: Experiment and Simulation of a Solid," Chem. Phys. **261** 249 (2000).
77. B. Fultz, T. Kelley, J.-D. Lee, O. Delaire, T. Swan–Wood and M. Aivazis, *Experimental Inelastic Neutron Scattering* `http://arcs.caltech.edu:8000/arcs/1`.
78. T.H. Dunning, R.J. Harrison, D. Feller, and S.S. Xantheas, "Promise and challenge of high-performance computing, with examples from molecular modelling", Phil. Trans. R. Soc. London A **360**, 1079 (2002).
79. Scientific Computing and Imaging Institute `http://software.sci.utah.edu/scirun.html`
80. NeXus Data Format Home Page `http://www.neutron.anl.gov/nexus/`
81. P. R. Bevington and D. Keith Robinson, *Data Reduction and Error Analysis for the Physical Sciences*

A. Appendix

A.1 Convolutions and Correlations

A.1.1 Convolution Theorem

It is easiest to explain convolutions is in terms of a broadening of a sharp peak caused by making a measurement with a blurry instrument. The instrumental broadening function is $f(k)$.[1] We seek the true specimen diffraction profile $g(k)$. What we actually measure with our diffractometer is the convolution of $f(k)$ and $g(k)$, denoted $h(K)$ (where K is the shift of the detector across the diffraction intensity). Deconvolution will require the Fourier transforms of $f(k)$, $g(k)$, $h(K)$:

$$f(k) = \sum_n F(n)\, \mathrm{e}^{\mathrm{i}2\pi nk/l} \qquad \text{equipment,} \tag{A.1}$$

$$g(k) = \sum_{n'} G(n')\, \mathrm{e}^{\mathrm{i}2\pi n'k/l} \qquad \text{specimen,} \tag{A.2}$$

$$h(K) = \sum_{n''} H(n'')\, \mathrm{e}^{\mathrm{i}2\pi n''K/l} \qquad \text{measurement.} \tag{A.3}$$

Note that l has units of inverse distance, so n/l is a real space variable. The range in k of the Fourier series is the interval $-l/2$ to $+l/2$, which includes all features of a diffraction peak.[2] The convolution of f and g is defined as:

$$h(K) = \int_{-\infty}^{\infty} f(K-k)\, g(k)\, \mathrm{d}k \; . \tag{A.4}$$

We must choose an interval so that that f and g vanish outside the range $\pm\ l/2$, so we can change the limits of integration from $\pm\ \infty$ to $\pm\ l/2$. Substitute (A.1) and (A.2) into (A.4):

$$h(K) = \int_{-l/2}^{l/2} \sum_n F(n)\, \mathrm{e}^{\mathrm{i}2\pi n(K-k)/l} \sum_{n'} G(n')\, \mathrm{e}^{\mathrm{i}2\pi n'k/l}\, \mathrm{d}k \; . \tag{A.5}$$

[1] Measurements are typically in scattering angle, which is interpretable as a k-space variable.

[2] We don't care about $f(k)$ and $g(k)$ outside this interval, but with (A.1)–(A.3) these Fourier transforms repeat themselves with a period of l. We confine ourselves to one period, and require that f and g vanish at its ends.

We rearrange summations over the independent variables n and n', and remove from the integral all factors independent of k:

$$h(K) = \sum_{n'} \sum_{n} G(n')F(n)\, \mathrm{e}^{\mathrm{i}2\pi nK/l} \int_{-l/2}^{l/2} \mathrm{e}^{\mathrm{i}2\pi(n'-n)k/l}\, \mathrm{d}k \; . \tag{A.6}$$

Now we employ the orthogonality condition[3]:

$$\int_{-l/2}^{l/2} \mathrm{e}^{\mathrm{i}2\pi(n'-n)k/l} \mathrm{d}k = \left\{ \begin{array}{l} l \text{ if } n' = n \\ 0 \text{ if } n' \neq n \end{array} \right\} . \tag{A.7}$$

With the orthogonality condition of (A.7), the double sum in (A.6) is reduced to a single sum:

$$h(K) = l \sum_{n} G(n)F(n)\, \mathrm{e}^{\mathrm{i}2\pi nK/l} \; . \tag{A.8}$$

Compare (A.8) to the definition for $h(K)$ in (A.3). We see that that the Fourier coefficients $H(n'')$ are proportional to the product of $G(n)$ and $F(n)$:

$$l\, G(n)F(n) = H(n) \; . \tag{A.9}$$

By comparing (A.4) and (A.9), we see that a convolution in k-space is equivalent to a multiplication in real space (with variable n/l). The converse is also true; a convolution in real space is equivalent to a multiplication in k-space. This important result is the *convolution theorem*.

A.1.2 Deconvolutions

Equation (A.9) shows how to perform the deconvolution of $f(k)$ from $h(K)$; perform a division in n-space. Specifically, when we have the full sets of Fourier coefficients $\{F(n)\}$ and $\{H(n)\}$, we perform a division in n-space for each Fourier coefficient:

$$G(n) = \frac{1}{l} \frac{H(n)}{F(n)} \; . \tag{A.10}$$

We obtain each $F(n')$ by multiplying both sides of (A.1) by $\exp(-\mathrm{i}2\pi n'k/l)$ and integrating over k:

$$\int_{-l/2}^{l/2} f(k)\, \mathrm{e}^{-\mathrm{i}2\pi n'k/l} \mathrm{d}k = \sum_{n} F(n) \int_{-l/2}^{l/2} \mathrm{e}^{\mathrm{i}2\pi(n-n')k/l} \mathrm{d}k \; . \tag{A.11}$$

[3] Verified by writing the exponential as $\cos(2\pi(n'-n)k/l) + \mathrm{i}\sin(2\pi(n'-n)k/l)$. The sine integration vanishes by symmetry. The cosine integration gives $l[2\pi(n'-n)]^{-1}[\sin(\pi(n'-n)) - \sin(\pi(n'-n))]$, which $= 0$ when $n'-n \neq 0$. In the case when $n'-n=0$, the integrand in (A.7) equals 1, so the integration gives l.

The orthogonality relationship of (A.7) causes the right-hand-side of (A.11) to equal zero unless $n = n'$. Equation (A.11) therefore becomes:

$$\frac{1}{l}\int_{-l/2}^{l/2} f(k)\,\mathrm{e}^{-\mathrm{i}2\pi n'k/l}\mathrm{d}k = F(n')\ . \tag{A.12}$$

The Fourier coefficients $H(n)$ are obtained the same way. The simple division of Fourier coefficients in (A.10) then provides the set of Fourier coefficients for the true specimen profile, $\{G(n)\}$. If we then use (A.2) to take the Fourier transform of the $\{G(n)\}$ from (A.10), we obtain $g(k)$, the true specimen diffraction profile.

A.2 Fourier Transform of Screened Coulomb Potential

In this subsection we calculate the Fourier transform of a "screened Coulomb" potential, a result that is useful in calculations of form factors of atoms for example. This screened Coulomb potential, $V(r)$, is:

$$V(r) = -\frac{Ze^2}{r}\mathrm{e}^{-r/r_0}\ . \tag{A.13}$$

The exponential factor accounts for the screening of the nuclear charge by the atomic electrons, and r_0 is an effective Bohr radius for the atom. Interestingly, the exponential decay also facilitates the mathematics of working with a potential that is otherwise strong at very large distances.

We now use the first Born approximation, (2.54), to calculate the atomic scattering factor, $f\,(\boldsymbol{\Delta k})$, as the Fourier transform of $V(\boldsymbol{r})$:

$$f_{\mathrm{el}}(\boldsymbol{\Delta k}) = -\frac{m}{2\pi\hbar^2}\int_{\text{all space}} \mathrm{e}^{-\mathrm{i}\boldsymbol{\Delta k}\cdot\boldsymbol{r}}\,V(\boldsymbol{r})\,\mathrm{d}^3\boldsymbol{r}\ . \tag{A.14}$$

Substituting the potential (A.13) into (A.14):

$$f_{\mathrm{el}}(\boldsymbol{\Delta k}) = \frac{mZe^2}{2\pi\hbar^2}\int_{\text{all space}} \mathrm{e}^{-\mathrm{i}\boldsymbol{\Delta k}\cdot\boldsymbol{r}}\frac{\mathrm{e}^{-r/r_0}}{r}\mathrm{d}^3\boldsymbol{r}\ . \tag{A.15}$$

The integral, $\mathcal{I}\,(\boldsymbol{\Delta k}, r_0)$, in (A.15) occurs in other contexts, so we pause to solve it.

$$\mathcal{I}\,(\boldsymbol{\Delta k}, r_0) \equiv \int_{\text{all space}} \mathrm{e}^{-\mathrm{i}\boldsymbol{\Delta k}\cdot\boldsymbol{r}}\frac{\mathrm{e}^{-r/r_0}}{r}\mathrm{d}^3\boldsymbol{r}\ , \tag{A.16}$$

which is the 3-dimensional Fourier transform of the screened Coulomb potential (A.13). It is natural to use spherical coordinates:

$$\mathcal{I}\left(\boldsymbol{\Delta k}, r_0\right) = \int_{r=0}^{\infty} \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} \mathrm{e}^{-\mathrm{i}\boldsymbol{\Delta k}\cdot\boldsymbol{r}} \frac{\mathrm{e}^{-r/r_0}}{r} r^2 \sin\theta \,\mathrm{d}\theta \,\mathrm{d}\phi \,\mathrm{d}r \ . \tag{A.17}$$

The trick for working with the exponential in (A.17), $\mathrm{e}^{-\mathrm{i}\boldsymbol{\Delta k}\cdot\boldsymbol{r}}$, is to align the vector $\boldsymbol{\Delta k}$ along the z-axis so that $\boldsymbol{\Delta k}\cdot\boldsymbol{r} = \Delta k z$. Also, since $z = r\cos\theta$:

$$\mathrm{d}z = -r\sin\theta \,\mathrm{d}\theta \ . \tag{A.18}$$

The limits of integration are changed as:

$$\theta = 0 \Longrightarrow \ z = r \ , \tag{A.19}$$

$$\theta = \pi \Longrightarrow \ z = -r \ . \tag{A.20}$$

With the substitution of (A.18)–(A.20) into (A.17):

$$\mathcal{I}\left(\boldsymbol{\Delta k}, r_0\right) = \int_{r=0}^{\infty} \int_{z=r}^{-r} \int_{\phi=0}^{2\pi} \mathrm{e}^{-\mathrm{i}\Delta k z} \mathrm{e}^{-r/r_0} \mathrm{d}\phi(-\mathrm{d}z)\mathrm{d}r \ , \tag{A.21}$$

$$\mathcal{I}\left(\boldsymbol{\Delta k}, r_0\right) = 2\pi \int_{r=0}^{\infty} \int_{z=-r}^{r} \mathrm{e}^{-\mathrm{i}\Delta k z} \mathrm{e}^{-r/r_0} \mathrm{d}z \,\mathrm{d}r \ . \tag{A.22}$$

Writing the exponential as $\mathrm{e}^{-\mathrm{i}\Delta k z} = \cos(\Delta k z) - \mathrm{i}\sin(\Delta k z)$, the z-integration of the sine function vanishes by symmetry in the interval $-r$ to $+r$, and the cosine integral is:

$$\int_{z=-r}^{r} \cos(\Delta k z) \,\mathrm{d}z = \frac{+2}{\Delta k} \sin(\Delta k r) \ , \tag{A.23}$$

which does not depend on the direction $\widehat{\boldsymbol{\Delta k}}$. Using (A.23) for the z-integration in (A.22), we obtain:

$$\mathcal{I}\left(\Delta k, r_0\right) = \frac{4\pi}{\Delta k} \int_{r=0}^{\infty} \sin(\Delta k r) \, \mathrm{e}^{-r/r_0} \mathrm{d}r \ . \tag{A.24}$$

Equation (A.24) is the Fourier transform of a decaying exponential. This integral can be solved by twice integrating by parts.[4] The result is a Lorentzian function:

$$\int_{r=0}^{\infty} \sin\left(\Delta k r\right) \mathrm{e}^{-r/r_0} \mathrm{d}r = \frac{\Delta k}{\Delta k^2 + \frac{1}{r_0^2}} \ . \tag{A.25}$$

[4] Defining $U \equiv \mathrm{e}^{-r/r_0}$ and $\mathrm{d}V \equiv \sin(\Delta k r)\,\mathrm{d}r$, we integrate by parts: $\int U\mathrm{d}V = UV - \int V\mathrm{d}U$. The integral on the right hand side is evaluated as: $(\Delta k r_0)^{-1} \int_{r=0}^{\infty} \cos(\Delta k r)\,\mathrm{e}^{-r/r_0}\mathrm{d}r$, which we integrate by parts again to obtain: $-\left(\Delta k r_0\right)^{-2} \int_{r=0}^{\infty} \sin(\Delta k r)\,\mathrm{e}^{-r/r_0}\mathrm{d}r$. This result can be added to the $\int U\mathrm{d}V$ on the left hand side to obtain (A.25).

We substitute the result (A.25) into (A.24), completing the evaluation of (A.16):

$$\mathcal{I}\left(\Delta k, r_0\right) = \int\limits_{\text{all space}} \mathrm{e}^{-\mathrm{i}\boldsymbol{\Delta k}\cdot\boldsymbol{r}} \frac{\mathrm{e}^{-r/r_0}}{r} \mathrm{d}^3\boldsymbol{r} = \frac{4\pi}{\Delta k^2 + \frac{1}{r_0^2}} \ . \tag{A.26}$$

For later convenience, we now obtain a related result. The use of an exponential screening factor to perform a Fourier transform of the Coulomb potential is a useful mathematical trick. By letting $r_0 \to \infty$, we suppress the screening of the Coulomb potential, so $\mathrm{e}^{-r/r_0} = 1$ in (A.13). The Fourier transform of this bare Coulomb potential, with its mathematical form of $1/r$, is obtained easily from (A.26):

$$\int\limits_{\text{all space}} \mathrm{e}^{-\mathrm{i}\boldsymbol{\Delta k}\cdot\boldsymbol{r}} \frac{1}{r} \, \mathrm{d}^3\boldsymbol{r} = \frac{4\pi}{\Delta k^2} \ . \tag{A.27}$$

A.3 Fundamental and Derived Constants

Fundamental Constants

$\hbar = 1.0546 \times 10^{-27}$ erg·sec $= 6.5821 \times 10^{-16}$ eV·sec
$k_{\rm B} = 1.3807 \times 10^{-23}$ J/(atom·K) $= 8.6174 \times 10^{-5}$ eV/(atom·K)
R = 0.00198 kcal/(mole·K) = 8.3145 J/(mole·K) (gas constant)
$c = 2.998 \times 10^{10}$ cm/sec (speed of light in vacuum)
$m_{\rm e} = 0.91094 \times 10^{-27}$ g $= 0.5110$ MeV·c^{-2} (electron mass)
$m_{\rm n} = 1.6749 \times 10^{-24}$ g $= 939.55$ MeV·c^{-2} (neutron mass)
$N_{\rm A} = 6.02214 \times 10^{23}$ atoms/mole (Avogadro constant)
$e = 4.80 \times 10^{-10}$ esu $= 1.6022 \times 10^{-19}$ coulomb
$\mu_0 = 1.26 \times 10^{-6}$ henry/m
$\varepsilon_0 = 8.85 \times 10^{-12}$ farad/m

$a_0 = \hbar^2/(m_{\rm e}e^2) = 5.292 \times 10^{-9}$ cm (Bohr radius)
$e^2/(m_{\rm e}c^2) = 2.81794 \times 10^{-13}$ cm (classical electron radius)
$e^2/(2a_0) =$ R (Rydberg) $= 13.606$ eV (K-shell energy of hydrogen)
$e\hbar/(2m_{\rm e}c) = 0.9274 \times 10^{-20}$ erg/oersted (Bohr magneton)
$\hbar^2/(2m_{\rm e}) = 3.813 \times 10^{-16}$ eV s cm^{-2}

Definitions

1 becquerel (B) = 1 disintegration/second
1 Curie $= 3.7 \times 10^{10}$ disintegrations/second

radiation dose:
1 roentgen (R) = 0.000258 coulomb/kilogram
Gray (Gy) = 1 J/kG

Sievert (Sv) is a unit of "radiation dose equivalent" (meaning that doses of radiation with equal numbers of Sieverts have similar biological effects, even when the types of radiation are different). It includes a dimensionless quality factor, Q (Q∼1 for x-rays, 10 for neutrons, and 20 for α-particles), and energy distribution factor, N. The dose in Sv for an energy deposition of D in Grays [J/kG] is:
Sv = Q×N×D [J/kG]
Rad equivalent man (rem) is a unit of radiation dose equivalent approximately equal to 0.01 Sv for hard x-rays.

1 joule = 1 J = 1 W·s = 1 N·m = 1 kg·m^2·s^{-2}
1 joule $= 10^7$ erg
1 newton = 1 N = 1 kg·m·s^{-2}
1 dyne = 1 g·cm·s^{-2} $= 10^{-5}$ N
1 erg = 1 dyne·cm = 1 g·cm^2·s^{-2}
1 Pascal = 1 Pa = 1 N·m^{-2}
1 coulomb = 1 C = 1 A·s
1 ampere = 1 A = 1 C/s

1 volt = 1 V = 1 $\mathrm{W{\cdot}A^{-1}} = 1\ \mathrm{m^2{\cdot}kg{\cdot}A^{-1}{\cdot}s^{-3}}$
1 ohm = 1 Ω = 1 $\mathrm{V{\cdot}A^{-1}} = 1\ \mathrm{m^2{\cdot}kg{\cdot}A^{-2}{\cdot}s^{-3}}$
1 farad = 1 F = 1 $\mathrm{C{\cdot}V^{-1}} = 1\ \mathrm{m^{-2}{\cdot}kg^{-1}{\cdot}A^2{\cdot}s^4}$
1 henry = 1 H = 1 $\mathrm{Wb{\cdot}A^{-1}} = 1\ \mathrm{m^2{\cdot}kg{\cdot}A^{-2}{\cdot}s^{-2}}$
1 tesla = 1 T = 10,000 gauss = 1 $\mathrm{Wb{\cdot}m^{-2}} = 1\ \mathrm{V{\cdot}s{\cdot}m^{-2}} = 1\ \mathrm{kg{\cdot}s^{-2}{\cdot}A^{-1}}$

Conversion Factors

1 Å = 0.1 nm = $10^{-4}\,\mu$m = 10^{-10} m
1 b (barn) = 10^{-24} $\mathrm{cm^2}$
1 eV = 1.6045×10^{-12} erg
1 eV/atom = 23.0605 kcal/mole = 96.4853 kJ/mole
1 cal = 4.1840 J
1 bar= 10^5 Pa
1 torr = 1 T = 133 Pa
1 kG = 5.6096×10^{29} MeV$\cdot c^{-2}$

Useful Facts

energy of 1 Å photon = 12.3984 keV
$h\nu$ for 10^{12} Hz = 4.13567 meV
1 meV = 8.0655 $\mathrm{cm^{-1}}$
temperature associated with 1 eV = 11,600 K
lattice parameter of Si (in vacuum at 22.5°C) = 5.431021 Å

Neutron Wavelengths, Energies, Velocities

$E_\mathrm{n} = 81.81\,\lambda^{-2}$ (energy-wavelength relation for neutrons [meV, Å])
$\lambda_n = 3955.4/v_n$ (wavelength-velocity relation for neutrons [Å, m/s])
$E_n = 5.2276 \times 10^{-6}\ v_n^2$ (energy-velocity relation for neutrons [meV, m/s])

Some X-Ray Wavelengths [Å]

Element	$K\overline{\alpha}$	$K\alpha_1$	$K\alpha_2$	$K\beta_1$
Cr	2.29092	2.28962	2.29351	2.08480
Co	1.79021	1.78896	1.79278	1.62075
Cu	1.54178	1.54052	1.54433	1.39217
Mo	0.71069	0.70926	0.71354	0.632253
Ag	0.56083	0.55936	0.56377	0.49701

Relativistic Electron Wavelengths

For an electron of energy E [keV] and wavelength λ [Å]:

$$\lambda = h\Big[2m_\mathrm{e}E\Big(1+\frac{E}{2m_\mathrm{e}c^2}\Big)\Big]^{-1/2} = \frac{0.3877}{E^{1/2}\,(1+0.9788\times10^{-3}E)^{1/2}}$$

kinetic energy $\equiv T = \frac{1}{2}m_\mathrm{e}v^2 = \frac{1}{2}E\frac{1+\gamma}{\gamma^2}$

Table A.1. Parameters of high-energy electrons

E [keV]	λ [Å]	γ	v [c]	T [keV]
100	0.03700	1.1957	0.5482	76.79
120	0.03348	1.2348	0.5867	87.94
150	0.02956	1.2935	0.6343	102.8
200	0.02507	1.3914	0.6953	123.6
300	0.01968	1.587	0.7765	154.1
400	0.01643	1.7827	0.8279	175.1
500	0.01421	1.9785	0.8628	190.2
1000	0.008715	2.957	0.9411	226.3

Index