A VARIATIONAL DIGITIZED GREEN FUNCTION APPROACH TO LIGHT SCATTERING BY SMALL PARTICLES

BY

RAYMOND TODD LINES, B.S., M.S.

A Dissertation submitted to the Graduate School

in partial fulfillment of the requirements

for the Degree

Doctor of Philosophy

Major Subject: Physics

New Mexico State University

Las Cruces, New Mexico

May 1998

"A Variational Digitized Green Function Approach to Light Scattering by Small Particles," a dissertation prepared by Raymond Todd Lines in partial fulfillment of the requirements for the degree, Doctor of Philosophy, has been approved and accepted by the following:

Timothy J. Pettibone
Dean of Graduate School

George H. Goedecke
Chair of the Examining Committee

Date

Committee in charge:

Dr. George H. Goedecke, Chair

Dr. Robert L. Armstrong

Dr. Tuan W. Chen

Dr. William R. Gibbs

Dr. Michael K. Giles

DEDICATION

To Christine, Britta, Nicholas, and Rachel

ACKNOWLEDGMENTS

First and foremost, I wish to thank my advisor, Dr. George Goedecke, for many hours of his time, for his patience, and for his enthusiasm. Thanks are also due to the rest of the physics department faculty who taught me the principles upon which this work was founded. I would also like to thank Young Yee, Radon Loveland, and Dr. Robert Sutherland of the U. S. Army Research Laboratory for their assistance and financial support. I owe a great debt to Dr. Richard Savage of Raytheon Systems Company for his encouragement and kind assistance. I would also like to thank Dr. Marian Hollingshead and Dr. Vance McCollough of Raytheon Systems Company for the use of computer time and for their patience. I also owe great thanks to Jan Bailey for all her help. I would like to thank my parents, for without their initial encouragement none of this would have been possible. Most of all I would like to thank my wife and children for their faith, patience, and willingness to sacrifice on my behalf.

VITA

1965	Born February 7, Ogden, Utah
1983	Graduated from Weber High School, North Ogden, Utah
1989	Undergraduate Assistant, Nuclear Accelerator Laboratory, Brigham Young University, Provo, Utah
1990	BS in Physics, Brigham Young University, Provo, Utah
1993	MS in Physics, New Mexico State University, Las Cruces, New Mexico
1990-1996	Graduate Physicist, Army Research Laboratory, White Sands, New Mexico
1996-1998	Scientist, Raytheon Systems Company, Denver Colorado

PUBLICATIONS

1 Lines, Todd, Yee, Young. P., "Temperature Profile of the Nocturnal Stable Boundary Layer Over Homogeneous Desert Using LA-TEAMS", *Proceedings of the 1994 Battlefield Atmospherics Conference, US Army Research Laboratory*, p.529-534. (M. E. Creegan, J. R. Elrick Ed.)

2 Goedecke, G., R. Loveland, R. Lines, R. Gonzalez, J. Thompson, Y. Yee, "Lidar inversion technique using total attenuation and the lidar system constant", *Proceedings, SPIE Atmospheric Propagation and Remote Sensing*, 21-23 April 1992, Orlando FL.

3 Lines, R. Todd, “Evaluation of the Stokes Parameters With Respect to Wind Vector Phenomenology,” Raytheon Systems Company Conical Scanning Microwave Imager/Sounder System Engineering Report, 10 September 1997.

4 Lines, R. Todd, “Recommendations for Wind Dependent Emission Modeling for RADTRAN,” Raytheon Systems Company Conical Scanning Microwave Imager/Sounder System Engineering Report, 7 October 1997.

5 Lines, R. Todd, “Possible One-Look Solution to the Sea Surface Wind Direction Measurement,” Raytheon Systems Company Conical Scanning Microwave Imager/Sounder System Engineering Report, 8 October 1997.

6 Lines,R. Todd, “Preliminary Wind Vector Algorithm,” Raytheon Systems Company Conical Scanning Microwave Imager/Sounder System Engineering Report, 10 November 1997.

7 Lines, R. Todd and George H. Goedecke, “Scattering by Irregular Homogeneous Particles using a Variational Green Function algorithm,” Submitted to *Applied Optics*, February 1998. *Proceedings, SPIE Atmospheric Propagation and Remote Sensing*, 21-23 April 1992 Orlando FL., pp. 171-77 (Anton Kohnle, Walter B. Miller Ed.)

PRESENTATIONS

1 Lines, R. Todd, “Detectability of Cloud Properties Using the Conical Scanning Microwave Imager/Sounder: Trade Study Results,” Presentation to the National Polar Orbiting Operational Satellite System Integrated Program

Office, November 6, 1996

2 Lines, R. Todd and Richard C. Savage, "Detectability of Icing Conditions Using Passive Microwave Devices," 1997 In-flight Remote Sensing Icing Avoidance Workshop, April 1-2, 1997, NASA Lewis Research Center, Cleveland, OH.

3 Lines, R. Todd , "The Detection of Ocean Wind Vectors Using the Conical Scanning Microwave Imager/Sounder," Presentation to the National Polar Orbiting Operational Satellite System Integrated Program Office, System Integration Team Meeting October 1, 1997

4 Lines, R. Todd, "Preliminary Algorithm for the Detection of Ocean Wind Vectors using Passive Microwave Devices," Presentation to the National Polar Orbiting Operational Satellite System Integrated Program Office, System Integration Team Meeting, November 7, 1997

5 Lines, R. Todd , "Detection of Cloud Ice Water Path Using Passive Microwave Devices," Presentation to the National Polar Orbiting Operational Satellite System Integrated Program Office, System Integration Team Meeting, December 18, 1997

ABSTRACT

A VARIATIONAL DIGITIZED GREEN FUNCTION APPROACH TO LIGHT SCATTERING BY SMALL PARTICLES

BY

RAYMOND TODD LINES, B.S., M.S.

Doctor of Philosophy in Physics

New Mexico State University

Las Cruces, New Mexico, 1998

Dr. George H. Goedecke, Chair

The variational Green function (VGF) algorithm and the underlying theory are presented. This coupled dipole algorithm models homogeneous dielectric particles of arbitrary shape by subdividing the particle into small cells. Computation is achieved without the huge matrix inversion and storage required by standard coupled dipole methods. Instead, a variational technique is used to find the coefficients of the internal field in a plane wave expansion. Predictions of differential cross sections are compared with predictions from Mie, T-matrix, and DGF calculations for homogeneous spheres, spheroids, and cubes. Convergence tests are presented for the VGF calculations as the number of plane wave terms is increased. Storage requirements and computation time are discussed. Applications and future studies are suggested.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

There is a continued interest in measuring meteorological parameters using visible, infrared and microwave sensors. In these measurements, the effects due to transfer of radiation from source to detector must be taken into account. Absorption is relatively easily taken into account through the radiative transfer equation for nonscattering atmospheres[1], but scattering is a more difficult problem.

Scattering from homogeneous or layered spheres has been calculated using the Mie theory[2]. Atmospheric hydrometeors, such as rain drops, ice crystals, and snow flakes, are not spherical. Dust, ash, and military obscurants are, in fact, highly non-spherical. Considerable work has been done on calculation of scattering from arbitrary particles[3]- [10]. The principal limitation of such work has been the immense number of calculations and the extremely large storage space required for even moderately sized particles. A numerical solution that is not so limited would allow larger particles to be modeled for propagation studies, numerical forecasting, and remote sensing algorithm development.

Program of Investigation

This develops a new variational approach to the small particle scattering problem. Though it builds on prior work in the field of small particle scattering, namely the Digitized Green Function Method, this variational approach is shown to be more

computationally efficient than other techniques in current use. The new method provides researchers with moderate or even modest computer resources a method of studying the scattering from large particles.

The Development of Scattering Models for Non-Spherical Particles

In 1908 Mie presented a solution to the scattering problem for a homogeneous sphere immersed in an electromagnetic plane wave[2]. This technique utilized an expansion of the internal and external fields in vector spherical harmonics and spherical Bessel functions, and application of the general electromagnetic boundary conditions at the sphere surface. Many variations on this original technique have allowed for solutions for other symmetric objects. The solution for normal incidence on an infinite cylinder was found by Rayleigh and generalized to oblique incidence by Wait[11]. These results were expanded upon by van de Hulst[12] and Chandrasekhar[13] (who developed the framework for completing the radiative transfer calculation) and others[3],[14]- [19]. Yeh[16],[17] extended the solution for infinite cylinders of elliptical cross section. Asano and Yamamoto[3] presented a solution for the scattering problem by prolate and oblate spheroids.

Waterman[18], and then Barber and Yeh[19] took a slightly different approach. They replaced the scattering object with an equivalent surface current distribution. The solution was formed by a vector spheroidal expansion and using boundary conditions. This method has become known as the Extended Boundary Condition Method (EBCM) and works quite well for spheroids with moderate aspect ratios. The method

does not do well with particles that have sharp corners or protrusions. Iskander et. al. [4],[20]-[21] partially eliminated this shortcoming using an iterative technique.

Fuller [22] has used the same basic concepts as Mie and Waterman in finding solutions to two-sphere systems. Examples of implementations of these techniques can be found in the book by Barber and Hill[23].

A totally different approach to the problem of small particle scattering was pioneered by Richmond[5] and then Purcell and Pennypacker[6]. This approach subdivided the particle into many tiny scatterers, usually electric dipoles. The technique is commonly referred to as the Coupled Dipole (CD) method. This technique was very successful despite a limitation in the original formulation which did not account for self-field corrections to the dipole polarizabilities and thus did not obtain computational agreement with the optical theorem.

CD methods have been used to model viral and bacterial types[24], to describe the scattering of microwaves from snowflakes [25], to describe the optical properties of soot [26] and military obscurants[27], and to study interstellar dust[28].

A variation on the CD technique was made, first by Acuista[10], then by Singham and Bohren[29]. This technique subdivided the particle into N_c cells or dipoles and then used successive orders of multiple scattering to iteratively approach the internal field of the particle. This greatly reduced the CD storage requirements because the $3N_c \times 3N_c$ matrix was never stored. The utility of the method is limited because the order of scattering series diverges for large indices of refraction and/or large values of N_c[30].

Cammack[7], and Goedecke and O'Brien[8],[31]-[25] in their Digitized Green Function (DGF) method exemplified the CD approach. The method subdivided the particle into small cubical cells, and included the self-field correction for each cell. This DGF method is equivalent to the formulation by Purcell and Pennypacker but has self-consistent polarizabilities, and it does permit agreement with the optical theorem. Draine and Goodman[9] optimized this for an arrangement of cells in a cubic lattice. In theory, DGF type solutions would be correct to any accuracy if the cell size could be made small enough. Small cell size, however, requires a larger number of cells to represent the volume of the particle. The DGF and other CD methods require the inversion of a $3N_c \times 3N_c$ complex matrix. This imposes an effective upper limit on size and index of refraction that can be used in calculation[8]. Flatau and Stephens[32] and Draine and Flatau[33] have experimented with more powerful mathematical techniques for solving the matrix equation of the CD method. They have extended the possible values of N_c from about 125 to about 3000. Both Draine and Flatau[33] and O'Brien[31] show that the accuracy of the solution depends critically on N_c being large enough, or equally, the cell size being small enough. Calculation of backscatter is shown by Draine and Flatau to require a higher value of N_c to achieve the same accuracy as forward scattering. O'Brien's results also imply that spheroids require larger values of N_c than spheres. O'Brien also noted that his results seem to indicate that rotated spheroids may require even larger values of N_c.

In the chapters that follow, a new variational Green function (VGF) coupled dipole solution is presented that avoids the $3N_c \times 3N_c$ matrix inversion of the

classical CD methods. This new method provides two important benefits. First, the $3N_c \times 3N_c$ matrix is replaced by a $3N_k \times 3N_k$ matrix where $N_k \ll N_c$, greatly reducing the large storage requirements of most coupled dipole methods. Second, a much smaller $3N_k \times 3N_k$ matrix inversion is employed to solve the scattering problem, thus avoiding the computational problems (round off and instability) involved in the inversion of large matrices. Both advantages give the researcher the ability to explore homogeneous, arbitrary shaped particles with large sizes and indices of refraction. The method is described in the pages that follow.

Chapter 2

SCATTERING THEORY

The general electromagnetic theory of light scattering from small particles is presented in this chapter. The standard equations for scattering are presented and modified to the form used in this study.

Scattering Geometry

The geometry of the scattering problem is given in Figure 1. The traditional description was given by Chandrasekhar[13] and is used by van de Hulst[12] and Bohren and Huffman[34]. The incident and scattered fields are described in terms of two components, parallel to or perpendicular to the scattering plane. The scattering plane is defined by the incident or external plane wave propagation vector $\hat{\mathbf{k}}$, and the scattered wave direction $\hat{\mathbf{k}}_s$. In the field of remote sensing the field components have taken the names *vertical* for Chandrasekhar's perpendicular and *horizontal* for Chandrasekhar's parallel. This notation will be used here.

The incident field will be assumed to travel such that $\hat{\mathbf{k}}$ is in the $+\hat{\mathbf{z}}$ direction. It is desirable, however, to place the particle in different orientations and observe the difference in the scattered field. To accomplish this we will define three rotations.

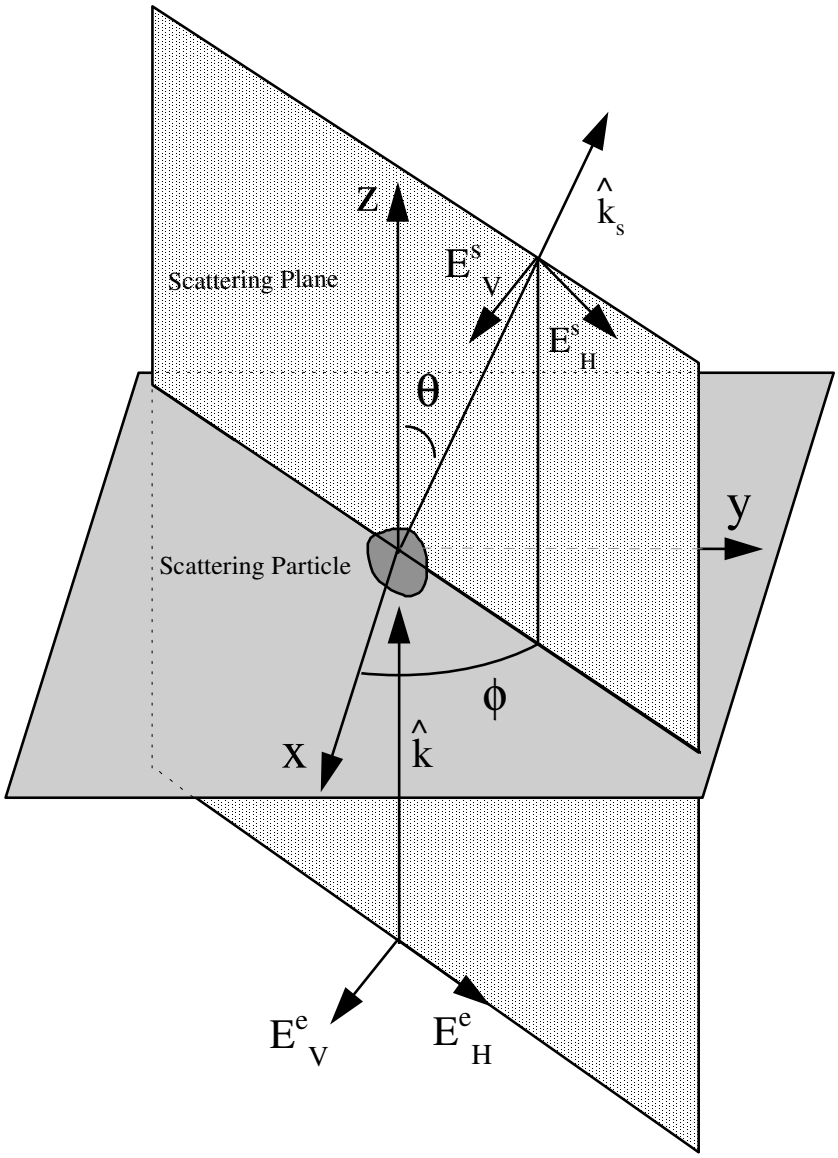


Figure 1. Geometry of the scattering problem: The incident plane wave travels along the positive z-axis. The incident and scattered electric-fields are resolved into horizontal (parallel to the scattering plane) and vertical (perpendicular to the scattering plane) components. The scattering plane is indicated by the shaded region between the incident and scattered directions

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix} \quad (2)$$

$$R_z(\gamma) = \begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

where α, β, and γ are the usual Euler angles shown in Figure 2. Combining these three rotations yields the transformation matrix

$$R(\alpha,\beta,\gamma) = R_z(\gamma)\,R_y(\beta)\,R_z(\alpha) \quad (4)$$

$$= \begin{pmatrix} \cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma & \sin\alpha\cos\beta\cos\gamma + \cos\alpha\sin\gamma & -\sin\beta\cos\gamma \\ -\cos\alpha\cos\beta\sin\gamma - \sin\alpha\cos\gamma & -\sin\alpha\cos\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\beta\sin\gamma \\ \cos\alpha\sin\beta & \sin\alpha\sin\beta & \cos\beta \end{pmatrix}$$

However, it is easier to define the shape of a particle with respect to a set of unrotated particle body frame coordinates. Although the incident field direction has been defined as $\hat{\mathbf{k}} = +\hat{\mathbf{z}}$, we may use the inverse transformation $R^{-1}(\alpha,\beta,\gamma) = R^T(\alpha,\beta,\gamma)$ to transform $\hat{\mathbf{k}}$ and the fields into the particle body frame, e.g.

$$\hat{\mathbf{k}}' = R^T(\alpha,\beta,\gamma)\hat{\mathbf{k}} \quad (5)$$

The computations described later are actually performed in the particle body frame with the rotated field. The resultant field is then transformed back into the global coordinate frame.

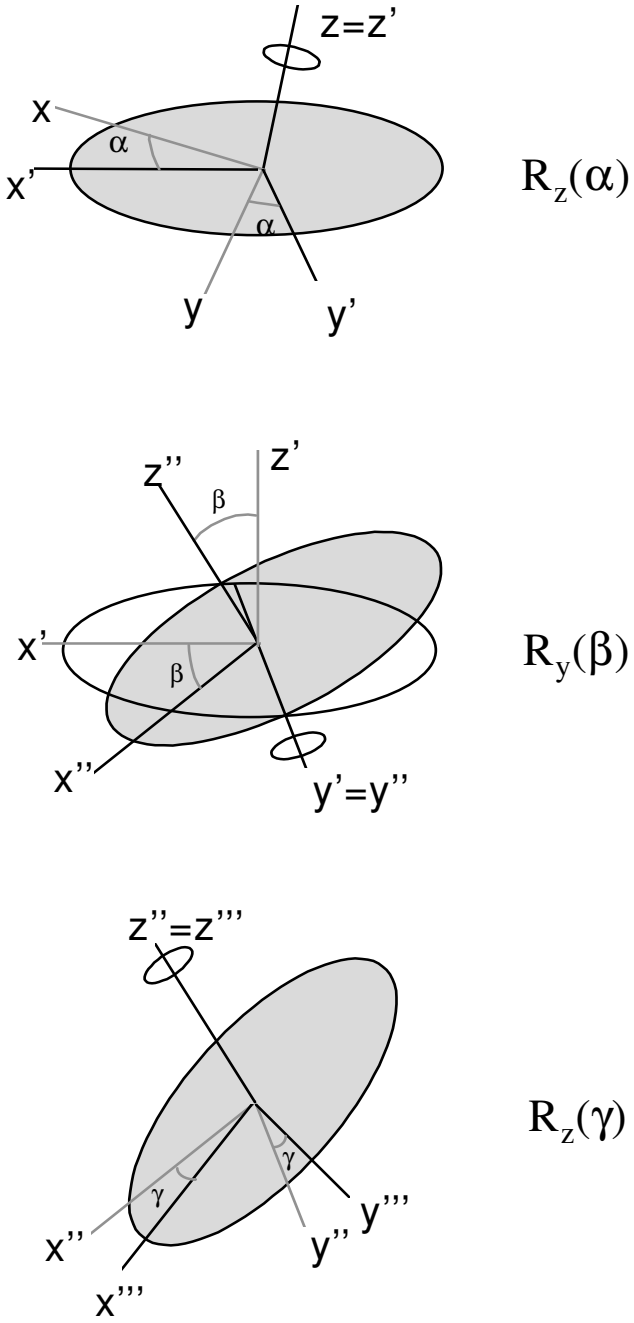


Figure 2. Euler Rotations: Geometry of the coordinate transformation between global and body frame coordinates.

General Scattering Equations

To describe the scattering problem, we begin with Maxwell's equations in Gaussian units,

$$\boldsymbol{\nabla} \cdot \mathbf{E} = 4\pi\rho \tag{6}$$

$$\boldsymbol{\nabla} \cdot \mathbf{B} = 0 \tag{7}$$

$$\boldsymbol{\nabla} \times \mathbf{E} = -\frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} \tag{8}$$

$$\boldsymbol{\nabla} \times \mathbf{B} = \frac{1}{c}\frac{\partial \mathbf{E}}{\partial t} + \frac{4\pi}{c}\mathbf{j} \tag{9}$$

where $\mathbf{E}$ and $\mathbf{B}$ are the electric and magnetic fields, ρ is the charge density, $\mathbf{j}$ is the current density, c is the speed of light and t is time. The operator $\boldsymbol{\nabla}$ is the usual gradient operator.

For material media, we may always write

$$\rho = -\boldsymbol{\nabla} \cdot \mathbf{P}(\mathbf{r}, t) \tag{10}$$

$$\mathbf{j} = \frac{\partial \mathbf{P}(\mathbf{r}, t)}{\partial t} + c\boldsymbol{\nabla} \times \mathbf{M}(\mathbf{r}, t) \tag{11}$$

where $\mathbf{P}(\mathbf{r}, t)$ is the polarization and $\mathbf{M}(\mathbf{r}, t)$ is the magnetization. We will take the usual time dependence for all electromagnetic fields. For example

$$\mathbf{E}(\mathbf{r}, t) = \mathbf{E}(\mathbf{r}) \exp(-i\omega t) + c.c. \tag{12}$$

where $\mathbf{E}(\mathbf{r})$ stands for the complex valued field amplitude vector, a function of position $\mathbf{r}$, and where ω is the angular frequency. The notation $c.c.$ refers to the complex

conjugate of the preceding term. The wave number k is given by

$$k = \frac{\omega}{c} = \frac{2\pi}{\lambda} \tag{13}$$

where λ is the wavelength. Only linear, static, nonmagnetic media will be considered, so that the complex amplitudes $[\mathbf{P}(\mathbf{r}), \mathbf{M}(\mathbf{r})]$ may be written as

$$\mathbf{P} = \chi \mathbf{E} \tag{14}$$

$$\mathbf{M} = \mathbf{0} \tag{15}$$

where the explicit dependence on position has not been written, and χ is the complex electric susceptibility. Using equations (12-15), the Maxwell equations (6-9) may be written as

$$\boldsymbol{\nabla} \cdot \mathbf{E} = -4\pi \boldsymbol{\nabla} \cdot \mathbf{P} \tag{16}$$

$$\boldsymbol{\nabla} \cdot \mathbf{B} = 0 \tag{17}$$

$$\boldsymbol{\nabla} \times \mathbf{E} = ik\mathbf{B} \tag{18}$$

$$\boldsymbol{\nabla} \times \mathbf{B} = -ik(\mathbf{E} + 4\pi \mathbf{P}) \tag{19}$$

The complex permittivity is given in terms of the complex susceptibility by

$$\epsilon(\mathbf{r}) = 1 + 4\pi\chi(\mathbf{r}) \equiv m^2(\mathbf{r}) \tag{20}$$

where the last equality defines the complex refractive index $m(\mathbf{r})$. Using this last equation, the displacement field $\mathbf{D}$ may be written as

$$\mathbf{D} = \mathbf{E} + 4\pi \mathbf{P} = \boldsymbol{\epsilon} \mathbf{E} \tag{21}$$

From this form (16-19) of the Maxwell equations, the field resulting from a plane wave impinging on the particle may be found by inserting the curl of (18) into the left

hand side of (19) which gives

$$\left(\boldsymbol{\nabla}^2 + k^2\right)\mathbf{E} = -4\pi\left(k^2\boldsymbol{\delta} + \boldsymbol{\nabla}\boldsymbol{\nabla}\right)\cdot\mathbf{P} \tag{22}$$

where $\boldsymbol{\delta}$ is the identity dyad. The *causal Green function*[35] for the Helmholtz operator $\boldsymbol{\nabla}^2 + k^2$ is found by requiring

$$(\boldsymbol{\nabla}^2 + k^2)G(\mathbf{r},\mathbf{r}') = -4\pi\delta(\mathbf{r}-\mathbf{r}') \tag{23}$$

with outgoing (causal) solution

$$G(\mathbf{r},\mathbf{r}') = \frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|} \tag{24}$$

Using the Green function the electric field as may be written as

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}^e(\mathbf{r}) + \int d^3\mathbf{r}'\,\frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|}(k^2\boldsymbol{\delta} + \boldsymbol{\nabla}_{\mathbf{r}'}\boldsymbol{\nabla}_{\mathbf{r}'})\mathbf{P}(\mathbf{r}') \tag{25}$$

Here $\mathbf{E}(\mathbf{r})$ is the total electric field, $\mathbf{E}^e(\mathbf{r})$ is the incident plane wave, and $\boldsymbol{\nabla}_{\mathbf{r}'}$ represents the gradient with respect to $\mathbf{r}'$. Equation (25) is valid at all $\mathbf{r}$, both inside and outside the particle.

The last term in (25) can be evaluated using integration by parts twice to give

$$\int d^3\mathbf{r}'\frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|}\boldsymbol{\nabla}_{\mathbf{r}'}\boldsymbol{\nabla}_{\mathbf{r}'}\mathbf{P}(\mathbf{r}') = \int d^3\mathbf{r}'\left(\boldsymbol{\nabla}_{\mathbf{r}'}\boldsymbol{\nabla}_{\mathbf{r}'}\frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|}\right)\mathbf{P}(\mathbf{r}') \tag{26}$$

The fact that $\mathbf{P}(\mathbf{r}')$ is nonzero only inside a bounded volume has been used to discard the surface integrals at spatial infinity. Then

$$\begin{aligned}\mathbf{E}(\mathbf{r}) &= \mathbf{E}^e(\mathbf{r}) + \int d^3\mathbf{r}'\,\frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|}k^2G\boldsymbol{\delta}\cdot\mathbf{P}(\mathbf{r}') \\ &\quad + \int d^3\mathbf{r}'\left(\boldsymbol{\nabla}_{\mathbf{r}'}\boldsymbol{\nabla}_{\mathbf{r}'}\frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|}\right)\mathbf{P}(\mathbf{r}')\end{aligned} \tag{27}$$

Noting

$$\boldsymbol{\nabla}_{\mathbf{r}'}\frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|} = -\boldsymbol{\nabla}_{\mathbf{r}}\frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|} \tag{28}$$

it follows that

$$\begin{aligned}\mathbf{E}(\mathbf{r}) &= \mathbf{E}^e(\mathbf{r}) + \int d^3\mathbf{r}' \, \frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|} k^2 \boldsymbol{\delta} \cdot \mathbf{P}(\mathbf{r}') \\ &\quad + \int d^3\mathbf{r}' \left(\boldsymbol{\nabla}_\mathbf{r} \boldsymbol{\nabla}_\mathbf{r} \frac{\exp(ik\,|\mathbf{r}-\mathbf{r}'|)}{|\mathbf{r}-\mathbf{r}'|} \right) \mathbf{P}(\mathbf{r}')\end{aligned} \tag{29}$$

The derivatives in (29) can be evaluated. Writing $\mathbf{R} = \mathbf{r} - \mathbf{r}'$,

$$\begin{aligned}\boldsymbol{\nabla}_\mathbf{r} \boldsymbol{\nabla}_\mathbf{r} \frac{\exp(ikR)}{R} &= \boldsymbol{\nabla}_\mathbf{r} \left\{ \left[\mathbf{R} \exp(ikR) \left(\frac{ik}{R^2} - \frac{1}{R^3} \right) \right] \right\} \\ &= \exp(ikR) \left\{ \frac{-k^2 \widehat{\mathbf{R}}\widehat{\mathbf{R}}}{R} + \left(\boldsymbol{\delta} - 3\widehat{\mathbf{R}}\widehat{\mathbf{R}} \right) \left(\frac{ik}{R^2} - \frac{1}{R^3} \right) \right\} - \frac{4\pi}{3} \boldsymbol{\delta} \delta(\mathbf{R})\end{aligned} \tag{30}$$

The last term in equation (30) ($-4\pi\boldsymbol{\delta}\delta(\mathbf{R})/3$) is seen to be correct by integrating the expression[35]

$$\nabla_\mathbf{R} \left(\frac{\mathbf{R}}{R^3} \right) \tag{31}$$

over a sphere of any radius centered at the origin. Then

$$\int d^3R \nabla_\mathbf{R} \left(\frac{\mathbf{R}}{R^3} \right) = \frac{1}{3} \boldsymbol{\delta} \cdot 4\pi \tag{32}$$

results from the generalized divergence theorem.

Substituting (30) into (29) gives the equation for the electric field for a plane wave scattered by an arbitrary particle.

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}^e(\mathbf{r}) - \frac{4\pi}{3} \mathbf{P}(\mathbf{r}) + \int d^3\mathbf{r}' \, \mathbf{G}(\mathbf{R}) \cdot \mathbf{P}(\mathbf{r}') \tag{33}$$

where $\mathbf{G}(\mathbf{R})$ is the *dyadic Green function* given by

$$\mathbf{G}(\mathbf{R}) = \exp(ikR) \left\{ \frac{k^2 \left(\boldsymbol{\delta} - \widehat{\mathbf{R}}\widehat{\mathbf{R}} \right)}{R} + \left(\boldsymbol{\delta} - 3\widehat{\mathbf{R}}\widehat{\mathbf{R}} \right) \left(\frac{ik}{R^2} - \frac{1}{R^3} \right) \right\} \tag{34}$$

This equation forms the basis of the DGF and other CD solutions. Equation (33) gives the total electric field in terms of the incident field and a term dependent on the morphology and optical properties of the particle. We may refer to this particle

dependent term as the scattered electric field

$$\mathbf{E}^s(\mathbf{r}) = -\frac{4\pi}{3}\mathbf{P}(\mathbf{r}) + \int d^3\mathbf{r}'\, \mathbf{G}(\mathbf{R}) \cdot \mathbf{P}(\mathbf{r}') \tag{35}$$

giving the final form for the electric field

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}^e(\mathbf{r}) + \mathbf{E}^s(\mathbf{r}) \tag{36}$$

The F-Field

The general electric field equation (33) may be rewritten in a form that will be more convenient for numerical calculation. Using equation (14) yields

$$\mathbf{E}(\mathbf{r})(1 + \frac{4\pi}{3}\chi) = \mathbf{E}^e(\mathbf{r}) + \int d^3\mathbf{r}'\mathbf{G}\left(\mathbf{r}, \mathbf{r}'\right) \cdot \chi(\mathbf{r}')\mathbf{E}(\mathbf{r}') \tag{37}$$

Note that the scattered field depends upon the incident field. Also, $\chi(\mathbf{r})$ is nonzero only within a bounded volume, V, around the origin.

A new field, $\mathbf{F}(\mathbf{r})$, may be defined such that

$$\mathbf{F}(\mathbf{r}) \equiv \mathbf{E}(\mathbf{r})(1 + \frac{4\pi}{3}\chi(\mathbf{r})) \tag{38}$$

and, for convenience, the quantity $W\left(\mathbf{r}\right)$ is also defined such that

$$W\left(\mathbf{r}\right) = \frac{\chi(\mathbf{r})}{(1 + \frac{4\pi}{3}\chi(\mathbf{r}))} \tag{39}$$

Then, using equation (38) and equation (39), equation (37) may be written as

$$\mathbf{F}(\mathbf{r}) = \mathbf{E}^e(\mathbf{r}) + \int d^3\mathbf{r}'\mathbf{G}\left(\mathbf{r}, \mathbf{r}'\right) \cdot W\left(\mathbf{r}'\right)\mathbf{F}(\mathbf{r}') \tag{40}$$

This places all the explicit dependence on $\chi(\mathbf{r})$ within the factor $W\left(\mathbf{r}\right)$ inside the integral. This field, $\mathbf{F}$, was used by Singham and Bohren [29] where it was called the "effective electric field."

Scattering Amplitude

The scattering amplitude is calculated from the electric field. It is traditional to define the vector scattering amplitude $\mathbf{f}$ such that, in the far-field or radiation zone limit,

$$\mathbf{E}^s(\mathbf{r}) \approx \frac{e^{ikr}}{ikr}\mathbf{f}\left(\mathbf{r}\right) \tag{41}$$

where $\mathbf{E}^s(\mathbf{r})$ is the scattered field defined in equation (35). Recall that $\mathbf{R} = \mathbf{r} - \mathbf{r}'$. In the far field limit the terms in $1/R^2$ and $1/R^3$ in equation (34) are small and can be dropped. For the exponent, $\mathbf{R} \approx \mathbf{r} - \hat{\mathbf{r}} \cdot \mathbf{r}'$ may be used, and $\mathbf{R} \approx \mathbf{r}$ elsewhere. Thus the conventional far-field scattering amplitude is

$$\mathbf{f}\left(\hat{\mathbf{r}}\right) = ik^3\left(\boldsymbol{\delta} - \hat{\mathbf{r}}\hat{\mathbf{r}}\right) \cdot \int d^3r' \; e^{-ik\hat{\mathbf{r}}\cdot\mathbf{r}'}\chi\left(\mathbf{r}'\right)\mathbf{E}\left(\mathbf{r}'\right) \tag{42}$$

Examining equation (42) it is clear that the factor $(\delta - \hat{\mathbf{r}}\hat{\mathbf{r}})$ guarantees that $\hat{\mathbf{r}}\cdot\mathbf{f} = 0$. Thus, $\mathbf{f}$ has only two independent components. These are expressed as being horizontal (parallel) and vertical (perpendicular) to the scattering plane defined by $\left(\hat{k}, \hat{\mathbf{r}}\right)$.

$$\mathbf{f}\left(\hat{\mathbf{r}}\right) = \hat{\boldsymbol{\theta}} f_H\left(\hat{\mathbf{r}}\right) + \hat{\boldsymbol{\phi}} f_V\left(\hat{\mathbf{r}}\right) \tag{43}$$

where we define

$$\hat{\boldsymbol{\theta}} = \hat{\mathbf{e}}_1 \cos\theta\cos\phi + \hat{\mathbf{e}}_2 \cos\theta \sin\phi - \hat{\mathbf{e}}_3 \sin\theta \tag{44}$$

$$\hat{\boldsymbol{\phi}} = -\hat{\mathbf{e}}_1 \sin\phi + \hat{\mathbf{e}}_2 \cos\phi \tag{45}$$

and where the $\hat{\mathbf{e}}_i$ are the Cartesian unit vectors.

Considering only the dyadic from equation (42), dot products can be formed with $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\phi}}$.

$$\hat{\boldsymbol{\theta}} \cdot (\boldsymbol{\delta} - \hat{\mathbf{r}}\hat{\mathbf{r}}) = \hat{\boldsymbol{\theta}} \cdot \boldsymbol{\delta} - \hat{\boldsymbol{\theta}} \cdot \hat{\mathbf{r}}\hat{\mathbf{r}} \tag{46}$$

$$= \hat{\boldsymbol{\theta}} \cdot \boldsymbol{\delta} = \hat{\boldsymbol{\theta}}$$

$$\hat{\boldsymbol{\phi}} \cdot (\boldsymbol{\delta} - \hat{\mathbf{r}}\hat{\mathbf{r}}) = \hat{\boldsymbol{\phi}} \tag{47}$$

because $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\phi}}$ are by definition orthogonal to $\hat{\mathbf{r}}$. Thus we have

$$f_H(\hat{\mathbf{r}}) = \hat{\boldsymbol{\theta}} \cdot \mathbf{f}(\hat{\mathbf{r}}) \tag{48}$$

$$f_V(\hat{\mathbf{r}}) = \hat{\boldsymbol{\phi}} \cdot \mathbf{f}(\hat{\mathbf{r}}) \tag{49}$$

The Scattering Amplitude Matrix

If the external field $\mathbf{E}^e(\mathbf{r})$ is taken to be a plane wave $\mathbf{E}^e(\mathbf{r}) = \mathbf{E}^e e^{i\mathbf{k}\cdot\mathbf{r}}$, then the scattering amplitude dyadic is defined by.

$$\mathbf{E}^s(\mathbf{r}) \approx \frac{e^{ikr}}{r}\mathbf{S}(\hat{\mathbf{r}}, \mathbf{k}) \cdot \mathbf{E}^e \tag{50}$$

Note that $\mathbf{E}^e$ is just the complex amplitude of the external plane wave. From equation (41), we see that the scattering amplitude dyadic is related to the vector scattering amplitude by

$$\mathbf{S}(\hat{\mathbf{r}}, \mathbf{k}) \cdot \mathbf{E}^e = \frac{\mathbf{f}}{ik} \tag{51}$$

The scattering amplitude matrix elements are derived by first noting that an incident plane wave may be resolved into two components, one horizontal (parallel) and one vertical (perpendicular) to the scattering plane. If the x-z plane is chosen for the scattering plane, then

$$\mathbf{E}^e = \hat{\mathbf{e}}_1 \mathbf{E}_1^e + \hat{\mathbf{e}}_2 \mathbf{E}_2^e = \hat{\mathbf{e}}_H \mathbf{E}_H^e + \hat{\mathbf{e}}_V \mathbf{E}_V^e \tag{52}$$

where

$$\begin{pmatrix} \hat{\mathbf{e}}_H \\ \hat{\mathbf{e}}_V \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{e}}_1 \\ \hat{\mathbf{e}}_2 \end{pmatrix} \tag{53}$$

From equations (44) and (45) the unit vectors are defined as

$$\begin{pmatrix} \hat{\mathbf{e}}'_H \\ \hat{\mathbf{e}}'_V \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\theta}} \\ \hat{\boldsymbol{\phi}} \end{pmatrix} \tag{54}$$

Then

$$\begin{pmatrix} \mathbf{E}^s_H \\ \mathbf{E}^s_V \end{pmatrix} = \frac{e^{ikr}}{r} \begin{pmatrix} S_{HH} & S_{HV} \\ S_{VH} & S_{VV} \end{pmatrix} \begin{pmatrix} \mathbf{E}^e_H \\ \mathbf{E}^e_V \end{pmatrix} \tag{55}$$

defines the scattering amplitude matrix, where

$$S_{HH} = \hat{\mathbf{e}}'_H \cdot \mathbf{S}(\mathbf{r}, \mathbf{k}) \cdot \hat{\mathbf{e}}_H \tag{56}$$

$$S_{HV} = \hat{\mathbf{e}}'_H \cdot \mathbf{S}(\mathbf{r}, \mathbf{k}) \cdot \hat{\mathbf{e}}_V \tag{57}$$

$$S_{VH} = \hat{\mathbf{e}}'_V \cdot \mathbf{S}(\mathbf{r}, \mathbf{k}) \cdot \hat{\mathbf{e}}_H \tag{58}$$

$$S_{VV} = \hat{\mathbf{e}}'_V \cdot \mathbf{S}(\mathbf{r}, \mathbf{k}) \cdot \hat{\mathbf{e}}_V \tag{59}$$

Conservation Laws

The extinction cross section gives the ratio of the total power taken from the incident wave to the incident power per unit area and is the sum of the total scattering and absorption cross sections. This relationship among the cross sections is part of the optical theorem. The equations describing the optical theorem and each cross section individually are developed below.

Optical Theorem

The relationship between the extinction cross section and the scattering cross section, absorption cross section, and the scattering amplitude can be found using conservation laws following the treatment by Goedecke[30]. First the conservation relations will be derived for energy, momentum, and power, then the expressions for the cross sections will be found. Finally, the correspondence to the optical theorem will be shown.

Conservation of Energy

The rate per unit volume at which work is done on the charge-current distribution by an electromagnetic field is

$$\mathbf{j} \cdot \mathbf{E} \tag{60}$$

The Maxwell equations with sources (6-9) may be used to write.

$$\mathbf{j} \cdot \mathbf{E} = \frac{1}{4\pi} \left((c\boldsymbol{\nabla} \times \mathbf{B}) \cdot \mathbf{E} - \frac{1}{c} \frac{\partial \mathbf{E}}{\partial t} \cdot \mathbf{E} \right) \tag{61}$$

Next the vector identity

$$\boldsymbol{\nabla} \cdot (\mathbf{E} \times \mathbf{B}) = \mathbf{B} \cdot (\boldsymbol{\nabla} \times \mathbf{E}) - \mathbf{E} \cdot (\boldsymbol{\nabla} \times \mathbf{B}) \tag{62}$$

is employed, thus yielding

$$\mathbf{j} \cdot \mathbf{E} = \frac{1}{4\pi} \left(c \left[-\boldsymbol{\nabla} \cdot (\mathbf{E} \times \mathbf{B}) + \mathbf{B} \cdot (\boldsymbol{\nabla} \times \mathbf{E}) \right] - \frac{1}{c} \frac{\partial \mathbf{E}}{\partial t} \cdot \mathbf{E} \right) \tag{63}$$

From Faraday's law, equation (8), it is seen that

$$\mathbf{j} \cdot \mathbf{E} = \frac{-1}{4\pi} \left(c\boldsymbol{\nabla} \cdot (\mathbf{E} \times \mathbf{B}) + \mathbf{B} \cdot \frac{\partial \mathbf{B}}{\partial t} + \frac{1}{c} \frac{\partial \mathbf{E}}{\partial t} \cdot \mathbf{E} \right) \tag{64}$$

The total energy density in the fields is defined as

$$u \equiv \frac{(E^2 + B^2)}{8\pi} \tag{65}$$

and the energy flux density is defined as

$$\mathbf{S} = \frac{c}{4\pi} (\mathbf{E} \times \mathbf{B}) \tag{66}$$

which is called the Poynting vector.

Then equation (64) may be written as

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{S} = -\mathbf{j} \cdot \mathbf{E} \tag{67}$$

Equation (67) is a statement of conservation of energy.

Power Conservation Laws

To continue toward power conservation laws, another identity is needed.

$$\frac{d}{dt} \int_V d^3 r \mathfrak{f} (\mathbf{r}, t) = \int_V d^3 r \frac{\partial}{\partial t} \mathfrak{f} + \int_{\Sigma(V)} d\Sigma \cdot \mathbf{v} \mathfrak{f} \tag{68}$$

where $\mathfrak{f}$ is a function of position and time, and $\Sigma(V)$ is the surface bounding any volume V, which is allowed to move and changes shape and size. Here, $\mathbf{v}$ is the instantaneous velocity of the surface element $d\Sigma$, and $d\Sigma$ points in the direction outward from V. Equation (67) is now integrated over such a moving volume, to get

$$\int_V d^3 r \frac{\partial u}{\partial t} + \int_V d^3 r \boldsymbol{\nabla} \cdot \mathbf{S} = -\int_V d^3 r \mathbf{j} \cdot \mathbf{E} \tag{69}$$

The first term is of the same form as the second term in equation (68), thus

$$\dot{\mathcal{E}} (V) \equiv \frac{\partial}{\partial t} \int_V d^3 r u = -\int_V d^3 r \mathbf{j} \cdot \mathbf{E} - \oint_\Sigma d\Sigma \cdot (\mathbf{S} - \mathbf{v} u) \tag{70}$$

where $\dot{\mathcal{E}} (V)$ is the time derivative of the energy in the field in the volume V. The divergence theorem has been used to rewrite the term in $\mathbf{S}$.

A distinction can be made between the fields due only to sources inside V (self-fields) and those due to external sources (external fields). Because of the bilinearity of the energy, momentum, and stress equations in the fields $\mathbf{E}$ and $\mathbf{B}$, there will be cross terms. For example, the energy density u is given by

$$u = u^s + u^e + u^{se} \tag{71}$$

where the superscripts s, e, and se represent the self, external (incident), and self-external cross terms, respectively. The forms of u^{se}, and $\mathbf{S}^{se}$ are

$$u^{se} = \frac{1}{4\pi}\left(\mathbf{E}^s \cdot \mathbf{E}^e + \mathbf{B}^s \cdot \mathbf{B}^e\right) \tag{72}$$

$$\mathbf{S}^{se} = \frac{c}{4\pi}\left(\mathbf{E}^s \times \mathbf{B}^e + \mathbf{E}^e \times \mathbf{B}^s\right) \tag{73}$$

while $(u^{ss}, \mathbf{S}^{ss})$ contain self-fields only, $(u^{ee}, \mathbf{S}^{ee})$ contain external fields only. Then it follows that

$$\frac{\partial u^s}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{S}^s = -\mathbf{j}^s \cdot \mathbf{E}^s \tag{74}$$

$$\frac{\partial u^e}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{S}^e = -\mathbf{j}^e \cdot \mathbf{E}^e \tag{75}$$

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{S} = -\mathbf{j} \cdot \mathbf{E} = -\left(\mathbf{j}^s + \mathbf{j}^e\right) \cdot \left(\mathbf{E}^s + \mathbf{E}^e\right) \tag{76}$$

$$\frac{\partial u^{se}}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{S}^{se} = -\left(\mathbf{j}^s \cdot \mathbf{E}^e + \mathbf{j}^e \cdot \mathbf{E}^s\right) \tag{77}$$

Three of these are independent. Each may be integrated over a moving volume V. Beginning with the self-field terms,

$$\dot{\mathcal{E}}^s\left(V\right) = -\int_V d^3 r \mathbf{j}^s \cdot \mathbf{E}^s - \oint_\Sigma d\Sigma \cdot \left(\mathbf{S}^s - \mathbf{v}u^s\right) \tag{78}$$

The expression

$$\mathcal{P}^{As}\left(V\right) = \int_V d^3 r \mathbf{j}^s \cdot \mathbf{E}^s \tag{79}$$

is clearly the instantaneous power absorbed by the charges in V from the self-fields, and

$$\mathcal{P}^{Rs}(V) = \oint_{\Sigma} d\Sigma \cdot (\mathbf{S}^{s} - \mathbf{v}u^{s}) \tag{80}$$

is the instantaneous power carried out of V due to the self-fields. Then equation (70) may be written as

$$\dot{\mathcal{E}}^{s}(V) = -\mathcal{P}^{As}(V) - \mathcal{P}^{Rs}(V) \tag{81}$$

For the external fields

$$\dot{\mathcal{E}}^{e}(V) = -\int_{V} d^{3}r \mathbf{j}^{e} \cdot \mathbf{E}^{e} - \int_{\Sigma} d\Sigma \cdot (\mathbf{S}^{e} - \mathbf{v}u^{e}) \tag{82}$$

The first term on the right hand side is clearly zero (because $\mathbf{j}^{e} \equiv 0$ inside V). Defining

$$\mathcal{P}^{Re}(V) = \int_{\Sigma} d\Sigma \cdot (\mathbf{S}^{e} - \mathbf{v}u^{e}) \tag{83}$$

as the instantaneous power carried out of V due to the external fields, equation (82) is

$$\dot{\mathcal{E}}^{e}(V) = -\mathcal{P}^{Re}(V) \tag{84}$$

The cross term and the total energy density will both include the sources inside V within the integrations and will thus look like the self-field term

$$\dot{\mathcal{E}}^{se}(V) = -\mathcal{P}^{Ase}(V) - \mathcal{P}^{Rse}(V) \tag{85}$$

$$\dot{\mathcal{E}}(V) = -\mathcal{P}^{A}(V) - \mathcal{P}^{R}(V) \tag{86}$$

where $\mathcal{P}^{A}$ is the total instantaneous power absorbed,

$$\mathcal{P}^{A} = \mathcal{P}^{As} + \mathcal{P}^{Ase} \tag{87}$$

and $\mathcal{P}^{R}(V)$ is the total instantaneous electromagnetic power crossing $\Sigma(V)$,

$$\mathcal{P}^{R}(V) = \mathcal{P}^{Rs}(V) + \mathcal{P}^{Rse}(V) + \mathcal{P}^{Re}(V) \tag{88}$$

If equations (81), (84), (85), and (86) are applied to steady state but fluctuating systems, and averaged over some suitable time, long enough so that the time average of a time derivative is zero, we find

$$\left\langle \mathcal{P}^{Rs}\left(V\right)\right\rangle = -\left\langle \mathcal{P}^{As}\left(V\right)\right\rangle \tag{89}$$

$$\left\langle \mathcal{P}^{Re}\left(V\right)\right\rangle = 0 \tag{90}$$

$$\left\langle \mathcal{P}^{Rse}\left(V\right)\right\rangle = -\left\langle \mathcal{P}^{Ase}\left(V\right)\right\rangle \tag{91}$$

$$\left\langle \mathcal{P}^{R}\left(V\right)\right\rangle = -\left\langle \mathcal{P}^{A}\left(V\right)\right\rangle \tag{92}$$

For the monochromatic situation considered here, the time average of a time derivative over one period is identically zero.

Now suppose $\Sigma\left(V\right)$ is taken to be the surface $\Sigma\left(r'\right)$, the surface of a very large fixed sphere. This can be done without loss of generality if the external field sources are outside the sphere. Then from equation (78)

$$0 = -\left\langle \int_V d^3r \mathbf{j}^s \cdot \mathbf{E}^s \right\rangle - \left\langle \oint_{\Sigma(r')} d\Sigma \cdot \mathbf{S}^s \right\rangle \tag{93}$$

since there are no sources in the volume between $\Sigma\left(V\right)$ and $\Sigma\left(r'\right)$. This yields

$$\left\langle \mathcal{P}^{Rs}\left(V\right)\right\rangle = \left\langle \int_{\Sigma(r')} d\Sigma \cdot \left(\mathbf{S}^{Rs}\right) \right\rangle = \left\langle \mathcal{P}^{Rs}\left(r'\right)\right\rangle = \left\langle \mathcal{P}^{As}\left(r'\right)\right\rangle \tag{94}$$

Likewise, if the external fields have no sources inside $\Sigma\left(r'\right)$, then equation (91) can be written as

$$-\left\langle \mathcal{P}^{Ase}\left(V\right)\right\rangle = \left\langle \mathcal{P}^{Rse}\left(r'\right)\right\rangle = \left\langle \int_{\Sigma(r')} d\Sigma \cdot \left(\mathbf{S}^{Rse}\right) \right\rangle \tag{95}$$

The total average power passing outward from r' (or V) is found as the sum of the partial power terms,

$$\left\langle \mathcal{P}^{R}\left(V\right)\right\rangle = \left\langle \mathcal{P}^{Rs}\left(r'\right)\right\rangle + \left\langle \mathcal{P}^{Rse}\left(r'\right)\right\rangle = -\left\langle \mathcal{P}^{A}\left(V\right)\right\rangle \tag{96}$$

Rearranging terms yields

$$\left\langle \mathcal{P}^{A}\left(V\right)\right\rangle +\left\langle \mathcal{P}\right\rangle ^{Rs}\left(r'\right)=-\left\langle \mathcal{P}^{Rse}\left(r'\right)\right\rangle \tag{97}$$

Since the left hand side is the sum of the power absorbed by the sources in V and the power radiated by them, this equation implies the definition

$$\left\langle \mathcal{P}^{in}\right\rangle \equiv-\left\langle \mathcal{P}^{Rse}\right\rangle =-\left\langle \int_{\Sigma(r')}d\Sigma\cdot\left(\mathbf{S}^{Rse}\right)\right\rangle =\left\langle \mathcal{P}^{Ase}\left(V\right)\right\rangle \tag{98}$$

as the time average incident power, the average net total electromagnetic field power provided to the system. The interpretation here is clear because $\left\langle \mathcal{P}^{Ase}\left(V\right)\right\rangle$ is the rate at which work is done on the charges in V by the external fields.

The conservation laws from equations (89)-(98) provide the framework needed to derive the optical theorem.

Extinction Cross Section and Optical Theorem

If the self-field time averaged scattered power is evaluated for r' in the far field of the scatterer, equations (66) and (94) yield

$$\begin{aligned}\left\langle \mathcal{P}^{As}\right\rangle &= \left\langle \mathcal{P}^{Rs}\right\rangle =\frac{c}{4\pi}\int d\Omega r^{2}\left\langle \left(\mathbf{E}^{s}\times\mathbf{B}^{s}\right)\cdot\hat{\mathbf{r}}\right\rangle \\ &= \frac{c}{4\pi}\int d\Omega r^{2}\left\langle \left|\mathbf{E}^{s}\right|^{2}\right\rangle\end{aligned} \tag{99}$$

where the last equality follows because $\mathbf{B}^{s}=\hat{\mathbf{r}}\times\mathbf{E}^{s}$ and $\hat{\mathbf{r}}\cdot\mathbf{E}^{s}=0$ in the radiation zone. Likewise, the current may be written in terms of the polarization vector, so the self absorbed average power is

$$\left\langle \mathcal{P}^{As}\right\rangle =\int_{V}d^{3}r\left\langle \mathbf{j}^{s}\cdot\mathbf{E}^{s}\right\rangle =\int_{V}d^{3}r\left\langle \frac{\partial\mathbf{P}}{\partial t}\cdot\mathbf{E}^{s}\right\rangle \tag{100}$$

From equations (98) and (73) the net incident scattered power can be written as

$$\begin{aligned} \left\langle \mathcal{P}^{in} \right\rangle &= -\left\langle \int_{\Sigma(r')} d\Sigma \cdot \left(\mathbf{S}^{Rse} \right) \right\rangle \\ &= -\int d\Omega r^2 \hat{\mathbf{r}} \cdot \frac{c}{4\pi} \left\langle \mathbf{E}^s \times \mathbf{B}^e + \mathbf{E}^e \times \mathbf{B}^s \right\rangle \end{aligned} \tag{101}$$

Likewise, the cross term absorbed power may be written as

$$\left\langle \mathcal{P}^{Ase} \right\rangle = \int d^3 r \left\langle \mathbf{j} \cdot \mathbf{E}^e \right\rangle = \int_V d^3 r \left\langle \frac{\partial \mathbf{P}}{\partial t} \cdot \mathbf{E}^e \right\rangle \tag{102}$$

in terms of the susceptibility $\chi(\mathbf{r})$, this last expression may be written as

$$\begin{aligned} \left\langle \mathcal{P}^{Ase} \right\rangle &= \int_V d^3 r \left\langle \left[-i\omega\chi(\mathbf{r}) E(\mathbf{r}) e^{-i\omega t} + c.c. \right] \cdot \left[E^e(\mathbf{r}) e^{i\mathbf{k}\cdot\mathbf{r} - i\omega t} + c.c. \right] \right\rangle \\ &= -i\omega \mathbf{E}^{e*} \cdot \int d^3 r e^{-i\mathbf{k}\cdot\mathbf{r}} \chi(\mathbf{r}) \mathbf{E}(\mathbf{r}) + c.c \end{aligned} \tag{103}$$

But using the definition of the vector scattering amplitude in equation (42), this is

$$\left\langle \mathcal{P}^{in} \right\rangle = \left\langle \mathcal{P}^{Ase} \right\rangle = \frac{-c}{k^2} \mathbf{E}^{e*} \cdot \mathbf{f}\left(\hat{\mathbf{k}} \right) + c.c \tag{104}$$

where $\mathbf{f}\left(\hat{k}\right)$ is the forward scattering amplitude, i.e., for $\hat{\mathbf{r}} = \hat{\mathbf{k}}$. The incoming energy flux is

$$\Phi = \frac{c}{4\pi} \left\langle \mathbf{E}^e(\mathbf{r}, t) \times \mathbf{B}^e(\mathbf{r}, t) \right\rangle = \frac{c}{2\pi} \left| \mathbf{E}^e \right|^2 \tag{105}$$

where $\mathbf{E}^e$ is the complex amplitude of the incident plane wave. The so-called extinction cross section of the scatterer is defined by

$$\sigma^{ext} \equiv \frac{\left\langle \mathcal{P}^{in} \right\rangle}{\Phi} \tag{106}$$

Using equations (104), (105) and (51) yields

$$\begin{aligned} \sigma^{ext} &= -\frac{4\pi}{k^2 \left| \mathbf{E}^e \right|^2} \mathrm{Re}\left(\mathbf{E}^{e*} \cdot \mathbf{f}\left(\hat{k} \right) \right) \\ &= \frac{4\pi}{k \left| \mathbf{E}^e \right|^2} \mathbf{E}^{e*} \cdot \mathrm{Im}\left(\mathbf{S}\left(\hat{\mathbf{k}} \right) \right) \cdot \mathbf{E}^e \end{aligned} \tag{107}$$

Now from the conservation laws,

$$0 = \left[\mathcal{P}^{Rse} + \mathcal{P}^{Ase}\right] + \left[\mathcal{P}^{Rs} + \mathcal{P}^{As}\right] \tag{108}$$

rearranging these yields

$$-\mathcal{P}^{Rse} = \mathcal{P}^{Ase} + \mathcal{P}^{As} + \mathcal{P}^{Rs} \tag{109}$$

Now $\mathcal{P}^{Rse} = -\mathcal{P}^{Ase} = \mathcal{P}^{in}$ thus the total power absorbed is $\mathcal{P}^{A} = \mathcal{P}^{Ase} + \mathcal{P}^{As}$. Then

$$\mathcal{P}^{in} = \mathcal{P}^{A} + \mathcal{P}^{R} \tag{110}$$

Multiplying each term by $c/\left(\Phi k^2\right)$ yields

$$\sigma^{ext} = \sigma^{abs} + \sigma^{scat} \tag{111}$$

where σ^{abs} and σ^{scat} are the absorption and scattering cross sections. Equations (107) and (111) together comprise the optical theorem.

Equation (111) is the desired relationship between the extinction, absorption, and scattering cross sections. Given expressions for the scattering and absorption cross sections, equation (111) provides a method of obtaining the extinction cross section which is computationally more accurate than equation (107). But the latter equation is useful as a check on whether energy is properly conserved computationally Expressions for the scattering and absorption cross sections are developed below.

Differential Scattering Cross Section

The differential cross section is defined as the ratio of the time-averaged power per unit solid angle radiated in a particular direction $\hat{\mathbf{r}}$, to the incident flux

$$\frac{d\sigma}{d\Omega} = \frac{\left\langle \frac{d\mathcal{P}^s}{d\Omega} \right\rangle}{\Phi} \tag{112}$$

The time average scattered power is given by equation (99)

$$\left\langle \frac{d\mathcal{P}^s}{d\Omega} \right\rangle = \frac{c}{4\pi} r^2 \left\langle \left| \mathbf{E}^s \right|^2 \right\rangle \tag{113}$$

It follows from equation (41) that the scattered power is given by

$$\left\langle \frac{d\mathcal{P}^s}{d\Omega} \right\rangle = \frac{c}{2\pi k^2} \mathbf{f} \cdot \mathbf{f}^* \tag{114}$$

The average power per unit solid angle scattered into a final polarization a given an incident plane wave with polarization b is given by

$$\left\langle \frac{d\mathcal{P}^s_{ab}}{d\Omega} \right\rangle = \frac{c}{2\pi k^2} \mathbf{f}_a \cdot \mathbf{f}_b^* \tag{115}$$

The incident flux with polarization b is given by

$$\Phi_b = \frac{c}{4\pi} \left\langle \mathbf{E}^e_b \left(\mathbf{r}, t \right) \times \mathbf{B}^e_b \left(\mathbf{r}, t \right) \right\rangle = \frac{c}{2\pi} \left| \mathbf{E}^e_b \right|^2 = \frac{c}{2\pi} \tag{116}$$

where for convenience here and in what follows, the complex amplitude of the incident plane wave is taken to have unit magnitude.

The differential cross section is thus

$$\frac{d\sigma^{scat}_{ab}}{d\Omega} = \frac{1}{\Phi_b} \left\langle \frac{d\mathcal{P}^s_{ab}}{d\Omega} \right\rangle = k^{-2} \mathbf{f}_a \cdot \mathbf{f}_b^* \tag{117}$$

Taking the horizontal and vertical components yields

$$\frac{d\sigma^{scat}_{HH}}{d\Omega} = k^{-2} \left| \mathbf{f}_H \right|^2 \tag{118}$$

$$\frac{d\sigma^{scat}_{VV}}{d\Omega} = k^{-2} \left| \mathbf{f}_V \right|^2 \tag{119}$$

where the magnitudes f_H and f_V are given in equations (48) and (49).

The total scattering cross section is the integral of the differential cross section over all solid angles including both components.

$$\sigma_s = \int \left(\frac{d\sigma}{d\Omega} \right) d\Omega = \int d\Omega k^{-2} \mathbf{f} \cdot \mathbf{f}^* \tag{120}$$

Absorption Cross Section

The absorption cross section is calculated from equation (70). The total power absorbed analogously to equation (79) is given by

$$\left\langle \mathcal{P}^{A}\left(V\right)\right\rangle =\int_{V}d^{3}r\mathbf{j}\cdot\mathbf{E}=\int dr^{3}\left\langle \mathbf{E}\cdot\frac{\partial\mathbf{P}}{\partial t}\right\rangle \tag{121}$$

Writing the time dependence explicitly, the total field is

$$\mathbf{E}\left(\mathbf{r},t\right)=\mathbf{E}\left(\mathbf{r}\right)e^{-i\omega t}+\mathbf{E}^{*}\left(\mathbf{r}\right)e^{i\omega t} \tag{122}$$

and the polarization is

$$\mathbf{P}\left(\mathbf{r},t\right)=\chi\mathbf{E}\left(\mathbf{r}\right)e^{-i\omega t}+\chi^{*}\mathbf{E}^{*}\left(\mathbf{r}\right)e^{i\omega t} \tag{123}$$

Thus the time derivative of $\mathbf{P}$ is given as

$$\frac{\partial\mathbf{P}\left(\mathbf{r},t\right)}{\partial t}=-i\omega\chi\mathbf{E}\left(\mathbf{r}\right)e^{-i\omega t}+i\omega\chi^{*}\mathbf{E}^{*}\left(\mathbf{r}\right)e^{i\omega t} \tag{124}$$

Substitution yields

$$\left\langle E\left(\mathbf{r},t\right)\cdot\frac{\partial\mathbf{P}}{\partial t}\right\rangle =2\omega\left|\mathbf{E}\left(\mathbf{r}\right)\right|^{2}\mathrm{Im}\left(\chi\right) \tag{125}$$

then

$$\left\langle \mathcal{P}^{A}\left(V\right)\right\rangle =\int dr^{3}2\omega\left|\mathbf{E}\left(\mathbf{r}\right)\right|^{2}\mathrm{Im}\left(\chi\right) \tag{126}$$

and dividing by the incident flux $\frac{c}{2\pi}\left|\mathbf{E}_{0}\right|^{2}$, with $\left|\mathbf{E}_{0}\right|^{2}=1$ yields the absorption cross section, σ^{abs}.

$$\sigma^{abs}=4\pi k\int dr^{3}\left|\mathbf{E}\left(\mathbf{r}\right)\right|^{2}\mathrm{Im}\left(\chi\right) \tag{127}$$

Chapter 3

DIGITIZATION

In this chapter the equations developed in Chapter 2 are discretized or digitized for use in numerical calculations. A homogeneous particle is assumed for this study. This particle is subdivided into a three dimensional lattice of cubical cells. Each cell is assumed to act like a dipole[6][8][33]. It is important to note that not all cells will be completely filled. The volume of each cell is weighted by the fraction of the cell that is filled. An effective cell volume is calculated, and from this a new weighted cell dimension, d_ν, can be found where the ν indicates the cell number. This new weighted d_ν is the form used in the calculations which follow.

Green's Function Digitization

Equation (40) can be written as a sum where the infinitesimal volume d^3r' is now a finite volume d_ν^3:

$$\begin{aligned} F_i(\mathbf{r}_\mu) &= E_i^{in}(\mathbf{r}_\mu) + \sum_{\nu\neq\mu} d_\nu^3 G_{ij}(\mathbf{r}_{\mu\nu}) W F_j(\mathbf{r}'_\nu) + I_i(\mathbf{r}_\mu) \\ &= E_{\mu,i}^{in} + W \sum_{\nu\neq\mu} d_\nu^3 G_{\mu i\nu j} F_{\nu j} + I_{\mu i} \end{aligned} \tag{128}$$

Here $I_{\mu i}$ is the self term, and Greek indices run and sum over dipole cell number and the Latin indices run and sum over vector components. Also, $\mathbf{r}_\mu$ is at the center of cell number μ. Equation (128) only approximates equation (40) well if d_μ^3 is small.

Experience shows that $|m|\, kd < 1$ should be used, where m is the complex refractive index of the homogenous particle.

The Green's function is discretized as

$$G_{\mu i \nu j} = \exp(ikR_{\mu\nu}) \left[\frac{k^2}{R_{\mu\nu}} \left(\delta_{ij} - \widehat{R_{\mu\nu i}} \widehat{R_{\mu\nu j}} \right) + \left(\frac{ik}{R_{\mu\nu}^2} - \frac{1}{R_{\mu\nu}^3} \right) \left(\delta_{ij} - 3\widehat{R_{\mu\nu i}} \widehat{R_{\mu\nu j}} \right) \right] \tag{129}$$

where $\mathbf{R}_{\mu\nu} = \mathbf{r}_\mu - \mathbf{r}_\nu$. This equation is very similar to that used by Goedecke and O'Brien.[8]

Self-Term

A problem occurs when the summation index ν is equal to μ : Terms with $R_{\mu\mu}$ in the denominator would diverge. Thus in equation (128) these terms have been separated. Because such a term results from the action of the cell field on itself, it is known as the self-term. The self-term results from letting $\mathbf{r} = \mathbf{r}_\mu$, and $\mathbf{r}' = \mathbf{r}_\mu + \boldsymbol{\xi}$, in equation (40) , and taking $\chi(\mathbf{r}')\, \mathbf{F}_j(\mathbf{r}') = \chi(\mathbf{r}_\mu)\, \mathbf{F}_j(\mathbf{r}_\mu)$. One gets

$$I_{\mu,i} = \left[\int_\square d^3\xi G_{ij}(\boldsymbol{\xi}) \right] W F_j(\mathbf{r}_\mu) \tag{130}$$

The $\square$ indicates that the integral is over the small dipole cell, of volume d_μ^3. The integral in brackets in equation (130) can be written as

$$\int_\square d^3\xi G_{ij}(\boldsymbol{\xi}) = \int_\square d^3\xi e^{ik\xi} \left[\frac{k^2}{\xi} \left(\delta_{ij} - \hat{\xi}_i \hat{\xi}_j \right) + \left(\frac{ik}{\xi^2} - \frac{1}{\xi^3} \right) \left(\delta_{ij} - 3\hat{\xi}_i \hat{\xi}_j \right) \right] \tag{131}$$

Employing the identity

$$\int_\square d\Omega_\xi \hat{\xi}_i \hat{\xi}_j = \delta_{ij} \frac{1}{3} \int_\square d\Omega_\xi \tag{132}$$

the last term vanishes due to symmetry of cubical cells. Also, the remaining terms can be combined to yield

$$\int_{\square} d^3\xi G_{ij}\left(\boldsymbol{\xi}\right) = \delta_{ij}\frac{2k^2}{3}\int_{\square} d^3\xi \frac{k^2 e^{ik\xi}}{\xi} \tag{133}$$

A series expansion of the integrand on the right hand side yields

$$\frac{\exp(ikx)}{x} = \left[\frac{1}{x} + ik - \frac{1}{2}k^2 x - \frac{1}{6}ik^3x^2 + \frac{1}{24}k^4x^3 + Ox^4\right] \tag{134}$$

which can be substituted into the previous expression to yield

$$\begin{aligned}\int_{\square} d^3\xi G_{ij}\left(\boldsymbol{\xi}\right) &= \delta_{ij}\frac{2k^2}{3}\int_{\square} d^3\xi \left(\frac{1}{\xi} + ik - \frac{1}{2}k^2\xi - \frac{1}{6}ik^3\xi^2 + \frac{1}{24}k^4\xi^3 + O\left(\xi^4\right)\right) \\ &\equiv \delta_{ij}\frac{4\pi}{3}\Gamma \end{aligned} \tag{135}$$

where this equation defines Γ as

$$\Gamma = \left((ka)^2 + \frac{2}{3}i\left(ka\right)^3 - \frac{1}{4}\left(ka\right)^4 - \frac{1}{15}i\left(ka\right)^5 + O\left(ka\right)^6\right) \tag{136}$$

and the integral has been done by using an equivalent volume sphere with radius a_μ

$$a_\mu = d_\mu\left(\frac{3}{4\pi}\right)^{\frac{1}{3}} \tag{137}$$

thus

$$\begin{aligned}\Gamma_\mu &= \left(\frac{3}{4\pi}\right)^{\frac{2}{3}}\left(kd_\mu\right)^2 + \left(\frac{i}{2\pi}\right)\left(kd_\mu\right)^3 - \frac{1}{4}\left(\frac{3}{4\pi}\right)^{\frac{4}{3}}\left(kd_\mu\right)^4 \\ &\quad -\frac{1}{15}i\left(\frac{3}{4\pi}\right)^{\frac{5}{3}}\left(kd_\mu\right)^5 + O\left(\left(kd_\mu\right)^6\right)\end{aligned} \tag{138}$$

The expression for $I_{\mu i}$ can then be written as

$$I_{\mu i} = \delta_{ij}\frac{4\pi}{3}W\Gamma_\mu F_{\mu j} = \frac{4\pi}{3}W\Gamma_\mu F_{\mu i} \tag{139}$$

The first two terms in Γ_μ were used by Goedecke and O'Brien in their formulation of the DGF scattering code [8]. Draine and Goodman obtained a somewhat different

expression for Γ_μ for placement of the dipole cells on a cubic lattice[9], i.e. the coefficients of the various powers of kd_μ are different than in equation(138). However, for ease in comparison with the DGF method, the form given by Goedecke and O'Brien is used here. The corrections of order $(kd_\mu)^4$and higher are insignificant; and even the term in $(kd_\mu)^2$ is not important if $kd_\mu \ll 1$. But, as shown by Goedecke and O'Brien, the lowest order imaginary term proportional to $(kd_\mu)^3$ is essential for agreement with the optical theorem, and this term is the same in the Draine and Goodman expression.

Digitized F-Field

Using equation (139), equation (128) can be written as

$$F_{\mu i} = E_i^{in}(\mathbf{r}_\mu) + \sum_{\nu \neq \mu} d_\nu^3 G_{\mu i \nu j} W F_{\nu j} + \frac{4\pi}{3}\Gamma_\nu W F_{\nu i} \tag{140}$$

where $W(\mathbf{r}) = W$ is constant because a homogeneous particle has been assumed, namely,

$$W = \frac{\chi}{(1 + \frac{4\pi}{3}\chi)} \tag{141}$$

The diagonal terms in the dyadic Green function, $G_{\mu i \nu j}$, are filled in by defining

$$G_{\mu i \mu j} = \frac{4\pi}{3 d_\mu^3}\Gamma_\mu \delta_{ij} \tag{142}$$

Then the F-field is given by

$$F_{\mu i} = E_i^{in}(\mathbf{r}_\mu) + d_\nu^3 G_{\mu i \nu j} W F_{\nu j} \tag{143}$$

where summation notation is employed. This is the form of the scattering equation that is used in the calculations that follow. The reconstruction of the internal electric

field from the F-field is obtained from (38).

$$E_{\mu i} = \frac{F_{\mu i}}{\left(1 + \frac{4\pi}{3}\chi\right)} \tag{144}$$

Digitized Scattering Amplitude

Digitization of the scattering amplitude follows immediately from equations (48) and (49).

$$f_H\left(\hat{\mathbf{r}}\right) = \hat{\theta} \cdot \mathbf{f}\left(\hat{\mathbf{r}}\right) = ik^3 \chi_\mu d_\mu^3 e^{-ik\hat{r}\cdot\mathbf{r}_\mu} \hat{\theta}_j E_{\mu,j} \tag{145}$$

$$f_V\left(\hat{\mathbf{r}}\right) = \hat{\phi} \cdot \mathbf{f}\left(\hat{\mathbf{r}}\right) = ik^3 \chi_\mu d_\mu^3 e^{-ik\hat{r}\cdot\mathbf{r}_\mu} \hat{\phi}_j E_{\mu,j} \tag{146}$$

The calculation of these last two equations is easier to do numerically in the body frame coordinates. Where the body frame values for $\mathbf{r}'$, $\mathbf{E}'_\mu$ and $\hat{\mathbf{k}}'$ defined as follows:

$$\hat{r} \cdot \mathbf{r}_\mu = \hat{r}_i r_{\mu,i} = R_{ij} \hat{r}_j r'_{\mu,i} \tag{147}$$

$$\hat{\theta} \cdot \mathbf{E}_\mu = R_{ij} \hat{\theta}_j \mathbf{E}'_{\mu,i} = \hat{\theta}'_i \mathbf{E}'_{\mu,i} \tag{148}$$

$$\hat{\phi} \cdot \mathbf{E}_\mu = R_{ij} \hat{\phi}_j \mathbf{E}'_{\mu,i} = \hat{\phi}'_i \mathbf{E}'_{\mu,i} \tag{149}$$

Thus, the scattering amplitude components are given by

$$f_H\left(\hat{\mathbf{r}}\right) = \hat{\theta} \cdot \mathbf{f}\left(\hat{\mathbf{r}}\right) = ik^3 \chi_\mu d_\mu^3 e^{-ik\hat{r}_\kappa r'_{\mu\kappa}} \hat{\theta}'_i \mathbf{E}'_{\mu,i} \tag{150}$$

$$f_V\left(\hat{\mathbf{r}}\right) = \hat{\phi} \cdot \mathbf{f}\left(\hat{\mathbf{r}}\right) = ik^3 \chi_\mu d_\mu^3 e^{-ik\hat{r}_\kappa r'_{\mu\kappa}} \hat{\phi}'_i \mathbf{E}'_{\mu,i} \tag{151}$$

where the dependence of R on the Euler angles α, β, and γ has been dropped for brevity.

Digitized Scattering Cross Section

The differential scattering cross section is calculated from the digitized form of the scattering amplitude given in equations (150) and (151).

$$\frac{d\sigma_{HH}^{scat}}{d\Omega} = k^{-2} f_H f_H^* \tag{152}$$

$$\frac{d\sigma_{VV}^{scat}}{d\Omega} = k^{-2} f_V f_V^* \tag{153}$$

The total scattering cross section is calculated as

$$\sigma_s = \int d\Omega k^{-2} \left|\mathbf{f}\right|^2 = k^{-2} \int \left(f_H f_H^* + f_V f_V^*\right) \cos\theta d\theta d\phi \tag{154}$$

This last equation is not easily digitized in closed form. For the present work, a two dimensional Simpson's rule numerical integration is employed[36].

Digitized Absorption Cross Section

The digitized form of the total absorption cross section is given from equation (127)

$$\sigma^{abs} = 4\pi \,\mathrm{Im}\left(\chi\right) d_\mu^3 E_{\mu i}\left(\mathbf{r}\right) E_{\mu i}^*\left(\mathbf{r}\right) \tag{155}$$

where the summation is over both μ and i.

Digitized Extinction Cross Section

The extinction cross section is given by equation (111) by simply adding equation (154) and equation (155). This may be compared to the optical theorem result from

equation (107) which may be digitized using equation (42) as

$$\sigma^{ext} = \frac{-4\pi}{k^2} \operatorname{Re}\left[E_j^{e*} i k^3 \chi_\mu d_\mu^3 e^{-i\hat{k}\hat{r}\cdot\mathbf{r}_\mu} E_{\mu,j}\right] \tag{156}$$

Integration Interval d^3

Some care must be taken in digitization of the integrals in equation (40) that the integration step d^3 be small enough to accurately represent the integral. The criteria should be that the internal field remain nearly constant over d, so

$$d < \frac{\lambda}{2\pi\,|m|} \tag{157}$$

If we take V as the total volume of the particle, then the number of cells N should be more than V/d^3or

$$N_c > \frac{V}{d^3} = \left(\frac{2\pi\,|m|}{\lambda}\right)^3 V \tag{158}$$

Thus the particle is divided into N_c cells of size d. The weighted cell size, d_μ, weights d by the percentage of the cell that is actually filled by the particle.

Chapter 4

ALGORITHM

Solution of the Scattering Equation

Continuing with the assumption of a homogeneous particle, we can write the F-field

$$F_{\mu i} = E^{in}_{\mu i} + \sum_{\nu} G_{\mu i \nu j} W_{\nu} F_{\nu j} \tag{159}$$

where, for convenience, the cell dimension d^3_{μ} is included in the W term.

$$W_{\nu} = W d^3_{\nu} \tag{160}$$

Symbolically, this may be written as

$$\mathbf{F} = \mathbf{E}^{in} + \mathbf{GWF} \tag{161}$$

Proceeding directly to a numerical solution at this point would be equivalent to the DGF. Instead, a variational solution is sought. First, an error $\boldsymbol{\varepsilon}$ is defined such that

$$\boldsymbol{\varepsilon} = \mathbf{F} - \mathbf{E}^{in} - \mathbf{GWF} \tag{162}$$

Writing components yields

$$\varepsilon_{\alpha i} = \sum_{\beta} \sum_{j} \left(1_{\alpha i \beta j} - G_{\alpha i \beta j} W_{\beta}\right) F_{\beta j} - E^{in}_{\alpha i} \tag{163}$$

Here the factor

$$F^{st}_{\alpha i} = -W_{\alpha} G_{\alpha i \alpha j} F_{\alpha j} \tag{164}$$

is the specially defined self term discussed in Chapter 3 Section 2. Note that

$$1_{\alpha i \beta j} = \delta_{\alpha \beta} \delta_{ij} \tag{165}$$

Equation (139), gives a definition of the diagonal elements of the dyadic Green function taken from the self term,

$$F_{\alpha i}^{st} = \delta_{\alpha\beta}\left(\frac{4\pi}{3}\Gamma_\alpha\right) W_\alpha F_{\alpha i} \tag{166}$$

Equation (142) gives a discretized form of the dyadic Green function diagonal terms.

$$G_{\alpha i\alpha j} = \frac{4\pi}{3d_\alpha^3}\Gamma_\alpha \delta_{ij} \tag{167}$$

Now, for convenience, define the quantity

$$M_{\alpha i\beta j} = (1_{\alpha i\beta j} - W_\beta G_{\alpha i\beta j}) \tag{168}$$

where the α, i, β,and j are not summed over. Then

$$\varepsilon_{\alpha i} = M_{\alpha i\beta j} F_{\beta j} - E_{\alpha i}^o \tag{169}$$

where in equation (169) the summation convention has been employed throughout, on both Latin and Greek indices.

The function Φ to extremize is defined as the squared magnitude of the error $\boldsymbol{\varepsilon}$,

$$\Phi = \boldsymbol{\varepsilon}_{\alpha i}\boldsymbol{\varepsilon}_{\alpha i}^* = F_{\beta j} M_{\beta j\alpha i}^T M_{\alpha i\gamma l}^* F_{\gamma l}^* - E_{\alpha i}^{o*} M_{\alpha i\beta j} F_{\beta j} - E_{\alpha i}^o M_{\alpha i\beta j}^* F_{\beta j}^* + E_{\alpha i}^o E_{\alpha i}^{o*} \tag{170}$$

Again for convenience two additional quantities are defined which will shorten the notation

$$K_{\gamma l\beta j} = M_{\gamma l\alpha i}^{*T} M_{\alpha i\beta j} \tag{171}$$

$$X_{\beta j}^* = E_{\alpha i}^{o*} M_{\alpha i\beta j} \tag{172}$$

Then equation (170) can be written compactly as

$$\Phi = F_{\gamma l}^* K_{\gamma l\beta j} F_{\beta j} - \left[X_{\beta j}^* F_{\beta j} + X_{\beta j} F_{\beta j}^*\right] + E_{\alpha i}^o E_{\alpha i}^{o*} \tag{173}$$

Because the form of $\mathbf{F}$ is not known, a trial function $\tilde{\mathbf{F}}$ can be used which is an expansion in some basis function set $\Psi_{\beta N}$. We define the trial function $\tilde{\mathbf{F}}$ by

$$\tilde{F}_{\beta j}=\sum_{N=1}^{N_k} a_{Nj}\Psi_{\beta N} \tag{174}$$

where the sum has been written explicitly. The function $\tilde{\mathbf{F}}$ is substituted for $\mathbf{F}$. Then equation (170) is approximately given by

$$\Phi = \Psi^*_{\gamma M}a^*_{Ml}K_{\gamma l\beta j}a_{Nj}\Psi_{\beta N} - \left[X^*_{\beta j}\Psi_{\beta N}a_{Nj} + X_{\beta j}\Psi^*_{\beta N}a^*_{Nj}\right] + E^o_{\alpha i}E^{o*}_{\alpha i} \tag{175}$$

Now the notation can again be simplified by defining two functions

$$H_{MlNj} = \Psi^*_{\gamma M}K_{\gamma l\beta j}\Psi_{\beta N} \tag{176}$$

$$Y^*_{Nj} = \Psi^T_{N\beta}X^*_{\beta j} \tag{177}$$

Then the final form of equation (170) is

$$\Phi = a^*_{Ml}H_{MlNj}a_{Nj} - \left[Y^*_{Nj}a_{Nj} + Y_{Nj}a^*_{Nj}\right] + E^o_{\alpha i}E^{o*}_{\alpha i} \tag{178}$$

To extremize Φ, equation (178) is differentiated with respect to a^*_{Li}, yielding

$$\frac{\partial\Phi}{\partial a^*_{Li}} = H_{LiNj}a_{Nj} - Y_{Li} \tag{179}$$

The result is set equal to zero. Equation (179) can be written in matrix form as

$$(\mathbf{H})\,(\mathbf{a}) = (\mathbf{Y}) \tag{180}$$

The solution for $(\mathbf{a})$ is given by

$$(\mathbf{a}) = (\mathbf{H})^{-1}\,(\mathbf{Y}) \tag{181}$$

The matrix $(\mathbf{H})$ has dimensions $3N_k \times 3N_k$. In the calculations that follow the value of $N_k \leq$ ranges from 10 to 120 which makes the dimensions of $(\mathbf{H})$ less than 360×360. This is within the limits of common matrix inversion techniques[36]. The

matrix $(\mathbf{H})$ is significantly smaller than the $3N_c \times 3N_c$ matrix that would result from a direct solution from equation (159).

Writing out the $(\mathbf{H})$and $(\mathbf{Y})$ matrix explicitly in components we have

$$\begin{aligned} H_{MlNj} &= \Psi^*_{\gamma M}\left(1_{\alpha i\gamma l} - W^*_\beta G^*_{\alpha i\gamma l}\right)\left(1_{\alpha i\beta j} - W_\beta G_{\alpha i\beta j}\right)\Psi_{\beta N} \\ Y_{Nj} &= \left(1_{\alpha i\beta j} - W_\beta G_{\alpha i\beta j}\right)^* E^o_{\alpha i}\Psi^*_{\beta N} \end{aligned} \tag{182}$$

The functions $\Psi_{\beta N}$ could be any set of functions. A simple choice for larger particles with refractive indices that are not very different from unity is a set of plane waves:

$$\Psi_{\beta N} = e^{imk\hat{\mathbf{k}}_N \cdot \mathbf{r}_\beta} \tag{183}$$

where as before m is the complex refractive index of the particle. These functions obey the correct wave equation inside the particle. The trial functions $\tilde{F}_{\beta j}$ are given by

$$\tilde{F}_{\beta j} = \sum_{N=1}^{N_k} a_{Nj} e^{imk\hat{\mathbf{k}}_N \cdot \mathbf{r}_\beta} \tag{184}$$

Again, the summation has been written explicitly. The $\hat{\mathbf{k}}_N$ or "k-vectors" must be chosen to represent the internal wave well. A "goodness of choice" criterion for the number and direction of the $\hat{\mathbf{k}}_N$ vectors results from evaluating Φ explicitly. Observing the values of the a_{Nj} can determine which of the k-vectors are not contributing well. If the magnitude a_N is small, then the $\hat{\mathbf{k}}_N$ is not well placed or may not be needed.

For the sake of computation, a temporary variable is employed in the calculations that follow.

$$T_{\alpha iNj} = \sum_\beta \left(1_{\alpha i\beta j} - G_{\alpha i\beta j} W_\beta\right)\Psi_{\beta N} \tag{185}$$

Using $T_{\alpha iNj}$, H_{MlNj} and Y_{Nj} can be expressed as

$$H_{MlNj} = \sum_{\alpha}\sum_{i} T^{*}_{\alpha iMl} T_{\alpha iNj} \tag{186}$$
$$Y_{Nj} = \sum_{\alpha}\sum_{i} T^{*}_{\alpha iNj} E^{o}_{\alpha i}$$

This is the form of the quantities used in the actual algorithms.

Because of the static nature of FORTRAN77, the memory allocated to $T_{\alpha iNj}$ cannot be freed for other use. The storage requirement for $T_{\alpha iNj}$ is $(3N_c \times 3N_k) * N_{bytes}$ where $N_k \ll N_c$. Thus we may compare the storage requirements of $T_{\alpha iNj}$ with the $(3N_c \times 3N_c)$ requirements of a standard CD implementation like the DGF. It should be noted that the use of $T_{\alpha iNj}$ is a compromise between storage and speed. For a faster computer system speed may not be an issue, then only the $(\mathbf{H})$ matrix need be stored. Of course there are other storage requirements, constants, indices, the vector $\mathbf{Y}$, the cell locations, etc., but these are nearly negligible compared to the large matrix in the DGF case, and are the same for both algorithms. In Table 1, an example of the storage requirements (see Figure 5) is compared to a typical CD implementation like the DGF. In the calculations, $N_k = 32$ and $N_{bytes} = 8$. For $N_c = 8360$, the ideal VGF storage requirement is dominated by the cell location array.

Table 1: VGF and CD Memory Usage

N_c	Typical CD (bytes)	VGF (bytes)	Ideal VGF (bytes)
408	11,985,408	940,032	9216
1184	100,933,632	2,727,936	9216
2632	498,774,528	6,064,128	9216
8360	5,032,051,200	19,261,440	25,080

Particle Sub-Division

As previously discussed, the particle must be divided into cells small enough that each acts like an electric dipole. To do this a cubical region of cells is defined. The region is made larger than the biggest dimension of the particle. Then, the particle is placed in the region, and the subcells that contain a portion of the particle within their volume are included in the calculation. In this way, the cells to be used as part of the particle are "carved out" of the larger region of cells. Cells along the particle boundary may be only partially filled. To include these cells without modification would overestimate the volume of the particle. To remove these cells would underestimate the particle volume. In order to overcome this difficulty, the cells are further divided into a set of subcells. If the subcell contains part of the particle within its much smaller volume the subcell is counted. The volume of the cell itself is weighted by the number of subcells within its volume that are counted. This weighting is accomplished by adjusting the value of d_μ which is the cell dimension such that d_μ^3 is equal to the volume of the fraction of subcells counted.

Chapter 5

RESULTS

This chapter presents results from the variational coupled dipole solution. The results are divided into three sections. The first section will compare the VGF solution to the results of Mie theory for homogeneous spheres. A Mie code from Barber and Hill[23] is used. Several cases are presented. The second section presents the results from the VGF for prolate and oblate spheroids. These results are compared to the T-matrix method for axisymmetric objects, again using a code from Barber and Hill[23]. The final section presents the VGF results for a cubical particle. Results are compared to the DGF code from Goedecke and O'Brien[8]. The question of how many k-vectors are necessary and their placement is addressed.

In each case the incident plane wave is assumed to travel in the $+z$-direction. The x-z plane is taken as the scattering plane.

Spheres

It should be pointed out that a discrete dipole approach cannot produce the scattering from a true sphere. This is because the subdivision of the particle into cells cannot accurately represent the sphere's surface. Draine and Flatau use the word "pseudosphere" to remind the reader of this source of error[33]. Here, the word "sphere" will be used for brevity, where the inaccuracy in the discretization process will be understood.

The first example for a sphere is presented in Figure 3. The sphere was relatively small with a radius of $1.125\ \mu m$. The sphere was illuminated by a plane wave travelling along the z-axis with a wavelength, λ, of $10\ \mu m$. This gave a size parameter, $X = 2\pi a/\lambda$, of approximately 0.707. The index of refraction was $1.33 + i0.1$. The sphere was subdivided into 117 subcells. Both polarizations are shown. The VGF solution utilized 10 k-vectors distributed in two rings, one with $\theta_k = 60°$ and one at $\theta_k = 120°$. In each ring there were four vectors at $\phi_k = 90°$, $180°$, $270°$, and $360°$. There was an additional vector in the forward $(\theta_k, \phi_k) = (0.0, 0.0)$ direction and backward $(\theta_k, \phi_k) = (\pi, 0.0)$ direction. The angles θ_k and ϕ_k are defined in Figure 4. In Figure 3, the VGF result is compared to the exact result using the Mie code. For comparison, the DGF result for same parameters is also given. This case was first examined by Cammack[7] and then by Goedecke and O'Brien [8]. The agreement is good with a difference in the forward peak less than 0.2%.

The second example, Figure 5, is a larger sphere with a radius of $4.77\ \mu m$. Again the sphere was illuminated with $10\ \mu m$ light giving a size parameter $X = 3$. The index of refraction was $1.33 + i0.01$. The particle was subdivided into cells with the different lines on the graph representing different numbers of cells. The number of cells in each case result from carving out a sphere from a cube of 24^3, 16^3, 12^3, or 8^3 cells. This process yielded 8360, 2632, 1184, and 408 occupied cells respectively. The values for the cell size, d, ranged from $0.398\ \mu m$ to $1.193\ \mu m$. For comparison, taking the equal sign in equation (157) would give $d = 1.1966\ \mu m$. Clearly the greater than symbol should be taken seriously for larger particles. The k-vectors were placed in five rings

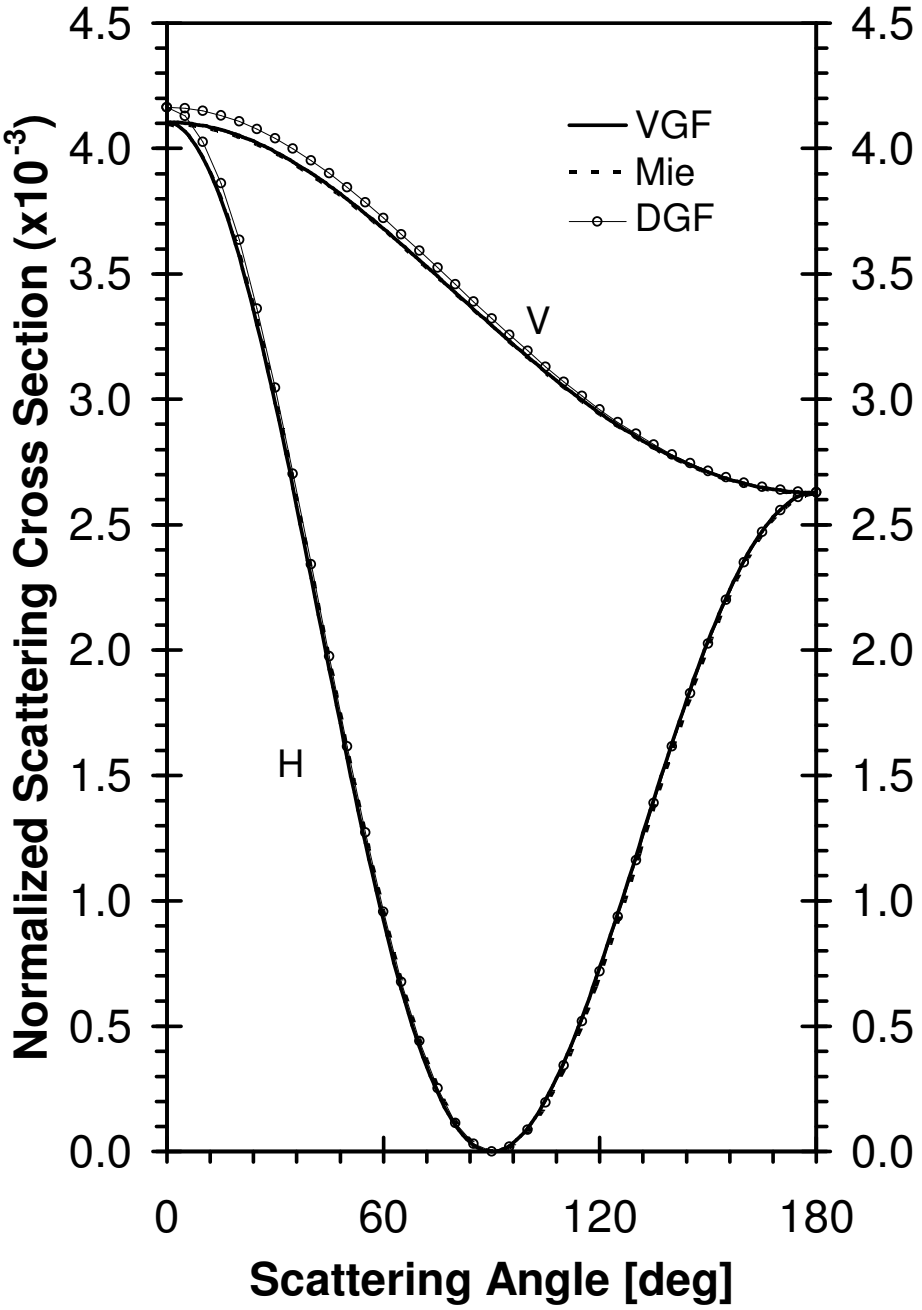


Figure 3. Scattering From a Small Sphere: The sphere has a radius $a = 1.55\ \mu m$ and refractive index $m = 1.2 + i0.05$. The unweighted cell length is $d = 0.45\ \mu m$, $N = 117$, and $\lambda = 10\ \mu m$. Both polarizations are shown. The rotation angles are $\alpha = 0^\circ$ $\beta = 0^\circ$, and $\gamma = 0^\circ$. The heavy solid curve is the VGF result, the dotted curve is the Mie result for comparison. There were 10 k-vectors evenly distributed in θ and ϕ.

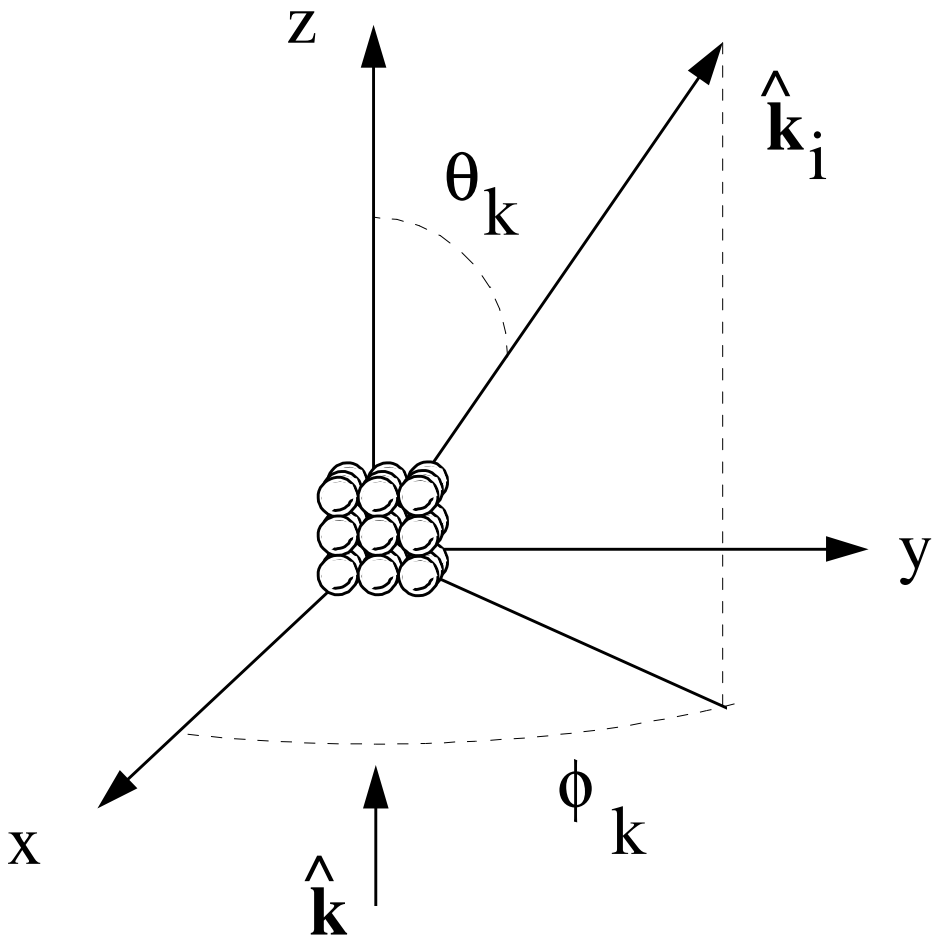


Figure 4. Geometry of the k-Vector Placement: The incident plane wave travels in the $+z$-direction.

at $\theta_k = 30°$, $60°$, $90°$, $120°$, and $150°$ with six vectors in each ring at $\phi_k = 60°$, $120°$, $180°$, $240°$, $300°$, and $360°$ with an additional vector in the forward and backward direction for a total of 32 k-vectors. The forward peak and backscatter are quite well defined for most of the curves. It is clear that as the number of cells increased, so that cell size decreased, the result approached the Mie curve.

There is a difference between the results given here and those published by Draine and Flatau[33]. This difference is due in part to the inaccuracy of the choice for the self term for the VGF code which introduces an error on the order of $(ka)^2$ where a is the radius of the effective spherical cell. Draine and Goodman have undertaken a study of optimal self term expressions[9]. Such a study is beyond the scope of the present discussion, thus the slight inaccuracy of the self term will be tolerated with the understanding that an optimal self term would provide somewhat better results. Given the above approximations, the agreement of the VGF solution with those of Mie theory is quite good.

Ellipsoids

As with spheres, a discrete dipole method cannot produce a true ellipsoid, due to the division into rectangular cells. The problem is even more severe for ellipsoids than for spheres, because along points or edges the phase of the internal field varies more rapidly. This means that smaller cells are required, yielding a larger value for N_c. In the examples that follow, the results from the VGF will be presented and compared to T-matrix code results.

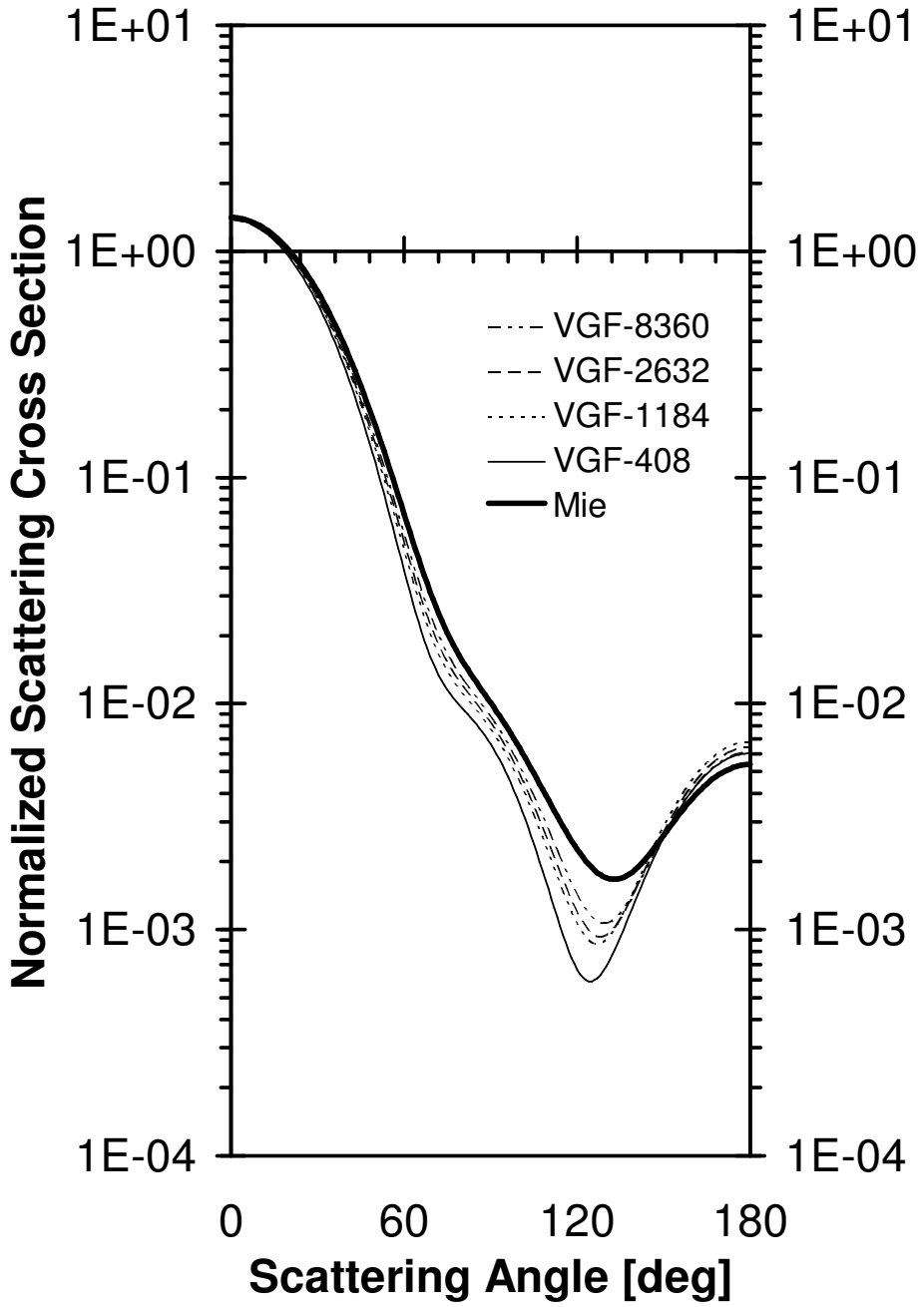


Figure 5. Scattering From a Larger Sphere: The sphere has $a = 4.77\ \mu m$, and $\lambda = 10\ \mu m$. Only the horizontal polarization is shown. The heavy solid curve is the Mie result. The remaining curves give VGF results for 408, 1184, 2632, and 8360 cells. The unweighted cell size, d, ranges from $1.193\ \mu m$ to $0.398\ \mu m$. There were 32 k-vectors evenly distributed in θ and ϕ.

Prolate Spheroids

The first example is a prolate spheroid. A prolate spheroid is an ellipsoid of rotation in which the symmetry axis is the major axis, i.e., the ratio of its symmetry axis, a, to its semiminor axis, b, is greater than one. For a first example $a = 1.55$ μm, $\lambda = 10$ μm. The index of refraction was $1.2 + i0.05$. The size parameter, X, is defined using the semi-major axis. Thus $X = 0.974$. The ratio of semi-major axis to semi-minor axis was $a/b = 1.2$. The spheroid was divided into 89 cells. The k-vectors were placed in five rings at $\theta_k = 30°$, $60°$, $90°$, $120°$, $150°$ with six vectors in each ring at $\phi_k = 60°$, $120°$, $180°$, $240°$, $300°$, $360°$ and with a vector in the forward and backward direction. The VGF results are graphed in Figure 6 where they are compared with the T-matrix solution. Barber and Hill chose $\gamma = 180°$ for axisymmetric particles, like spheroids, to simplify the T-matrix coding. This is possible because the rotation by γ about the particle symmetry axis does not effect the form of the scattering cross section for axisymmetric particles. We will follow this choice instead of the usual $\gamma = 0°$ in order to compare the VGF with their results. Thus the particle in Figure 6 has the rotation angles $\alpha = 0°$, $\beta = 0°$, and $\gamma = 180°$. The agreement is good with only a slight difference (less than 2%) in the forward peak.

For spheres, the orientation of the particle did not matter, due to the sphere's symmetry. Now, however, the spheroid orientation will make a difference. In the global frame, defined in Chapter 2, the spheroid orientation is given in terms of the three Euler angels, α, β, and γ. In the next two figures, the prolate spheroid from Figure 6 has been rotated by $\alpha = 0°$, $\beta = 45°$, $\gamma = 180°$ (Figure 7), and $\alpha = 45°$, $\beta = 45°$,

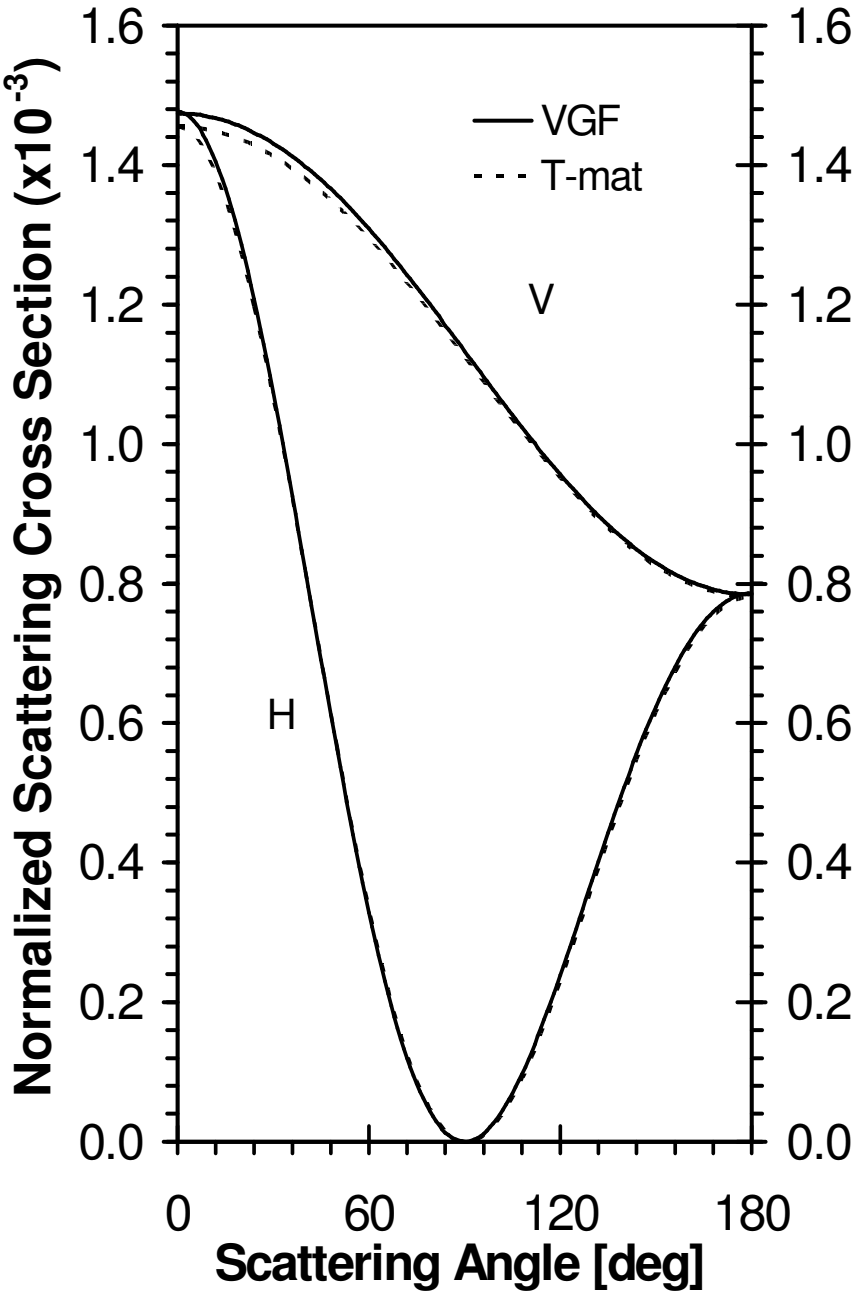


Figure 6. Scattering From a Prolate Spheroid: The spheroid has a semi-symmetry axis $a = 1.55\ \mu m$ with $a/b = 1.2$, and refractive index $m = 1.2 + i0.05$. The unweighted cell length is $d = 1.33\ \mu m$, $N = 144$, and $\lambda = 10\ \mu m$. Both polarizations are shown. The heavy solid curve is the VGF result, the dotted curve is the T-matrix result for comparison. There were 32 k-vectors evenly distributed in θ and ϕ.

$\gamma = 180^\circ$ (Figure 8). Again, the k-vectors were placed in five rings at $\theta_k = 30^\circ$, 60°, 90°, 120°, 150° with six vectors in each ring at $\phi_k = 60^\circ$, 120°, 180°, 240°, 300°, 360° and two additional vectors, one in the forward and one backward directions. For both cases the results agree well with errors less than 1.2% in the forward peak.

Oblate Spheroids

The scattering solution for oblate spheroids (symmetry axis a is the minor axis, so $a/b < 1$) is traditionally harder to find using discrete dipole methods. The first oblate spheroid example, plotted in Figure 9, had $a = 1.55\ \mu m$, $\lambda = 10\ \mu m$. The index of refraction was $1.2 + i0.05$. The size parameter, X, was defined using the semi-major axis, thus $X = 0.97389$. The ratio $a/b = 0.8$. The spheroid was divided into 296 cells. The k-vectors were placed in five rings at $\theta_k = 30^\circ$, 60°, 90°, 120°, 150° with six vectors in each ring at $\phi_k = 60^\circ$, 120°, 180°, 240°, 300°, 360° and one forward and one backward vector. The agreement with the T-matrix method is quite good with a difference in the forward peak of less than 1%.

Again, the orientation of the particle is important for oblate spheroids. In the figures below, the solution for the particle from Figure 9 is plotted, with the particle rotated by $\alpha = 0^\circ$, $\beta = 45^\circ$, $\gamma = 180^\circ$ (Figure 10) and $\alpha = 45^\circ$, $\beta = 45^\circ$, $\gamma = 180^\circ$ (Figure 11). The k-vectors were placed in five rings at $\theta_k = 30^\circ$, 60°, 90°, 120°, 150° with six vectors in each ring at $\phi_k = 60^\circ$, 120°, 180°, 240°, 300°, 360° with one vector in the forward and one vector in the backward direction. Again the VGF results agree well with the T-matrix results for both cases.

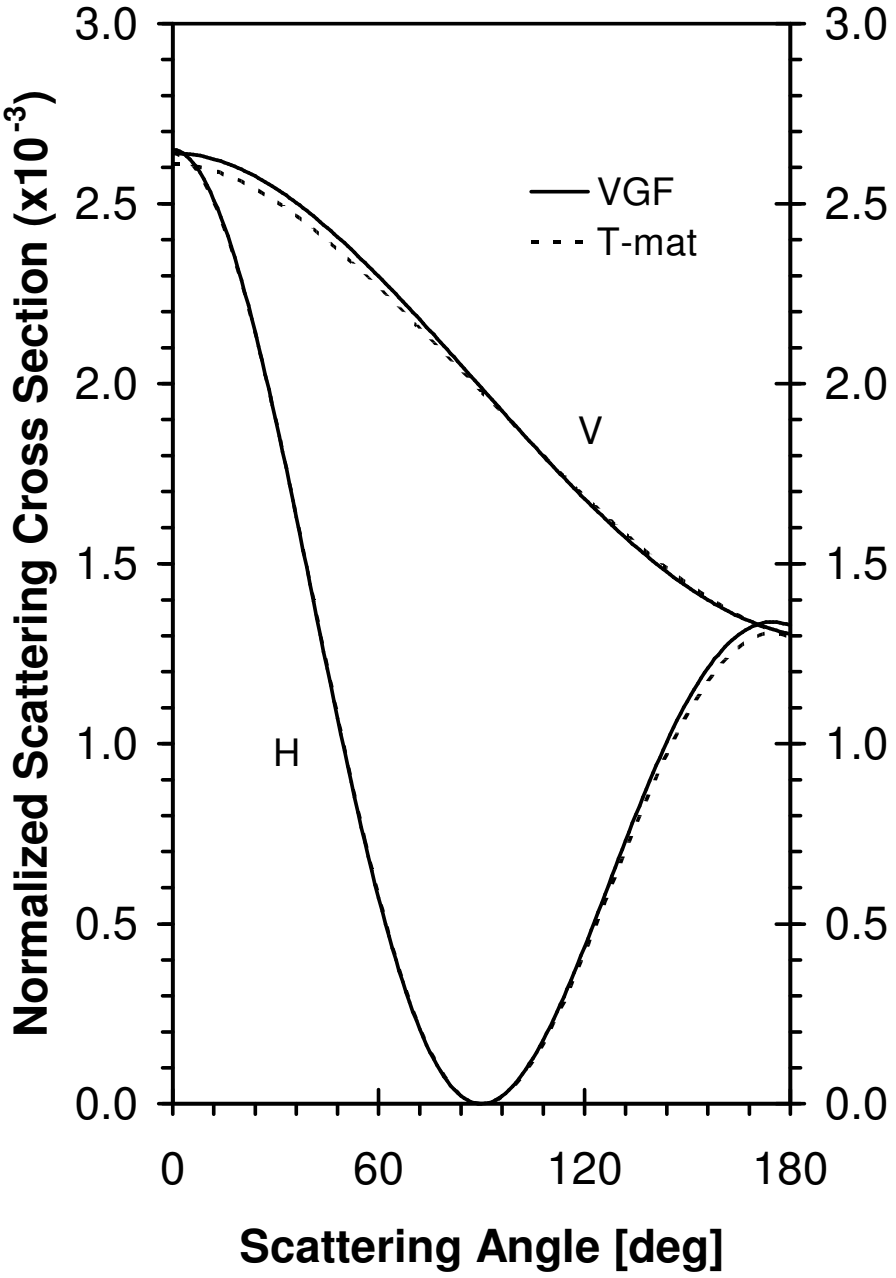


Figure 7. Scattering From a Rotated Prolate Spheroid: The spheroid has a semi-symmetry axis $a = 1.55\ \mu m$ with $a/b = 1.2$, and refractive index $m = 1.2+i0.05$. The unweighted cell length is $d = 0.31\ \mu m$, $N = 528$, and $\lambda = 10\ \mu m$. Both polarizations are shown. The rotation angles are $\alpha = 0^\circ$ $\beta = 45^\circ$, and $\gamma = 180^\circ$. The heavy solid curve is the VGF result, the dotted curve is the T-matrix result for comparison. There were 32 k-vectors evenly distributed in θ and ϕ.

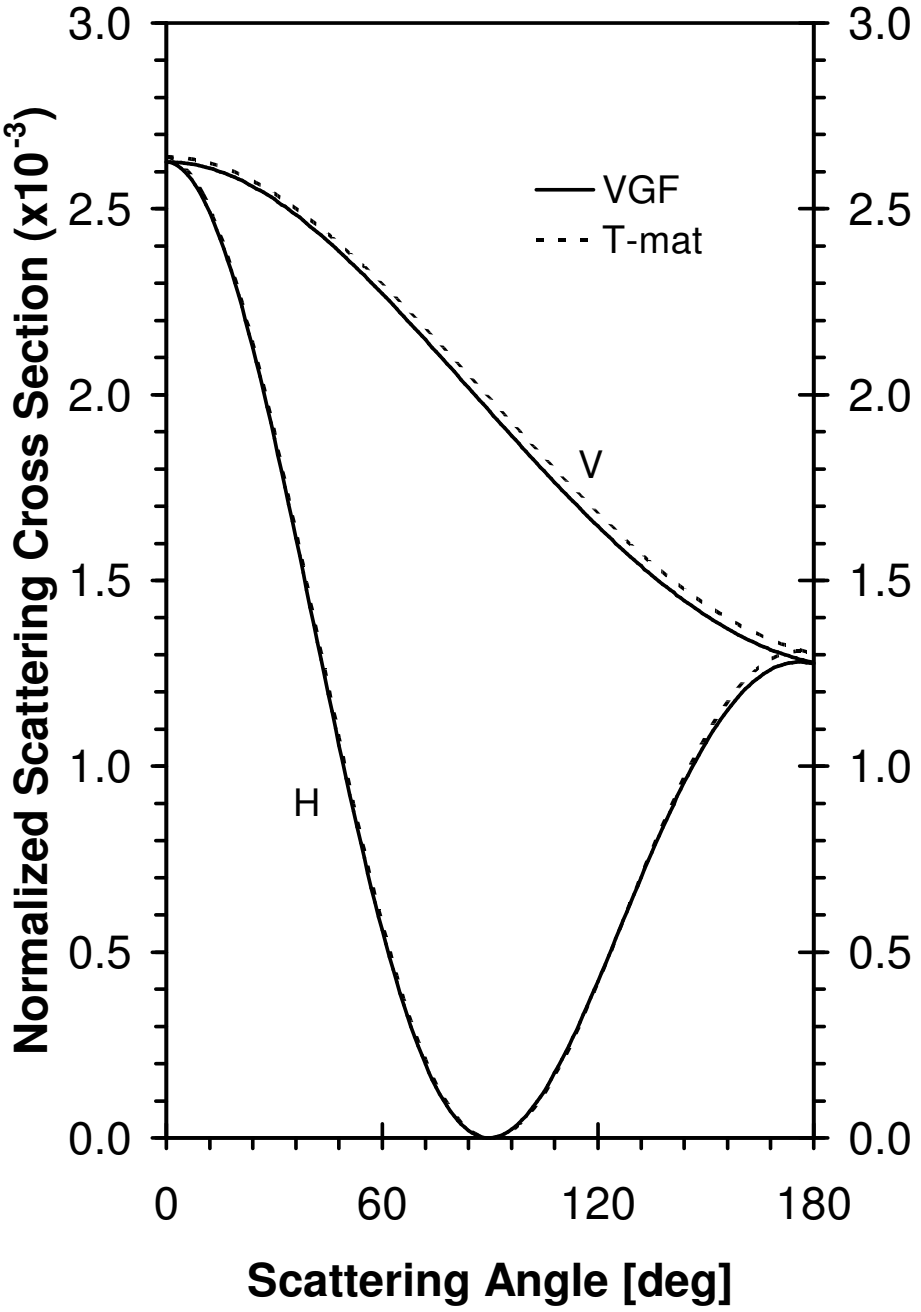


Figure 8. Scattering From a Prolate Spheroid-Rotated Twice: The spheroid has a semi-symmetry axis $a = 1.55\ \mu m$ with $a/b = 1.2$, and refractive index $m = 1.2+i0.05$. The unweighted cell length is $d = 0.31\ \mu m$, $N = 528$, and $\lambda = 10\ \mu m$. Both polarizations are shown. The rotation angles are $\alpha = 45^\circ$ $\beta = 45^\circ$, and $\gamma = 180^\circ$. The heavy solid curve is the VGF result, the dotted curve is the T-matrix result for comparison. There were 32 k-vectors evenly distributed in θ and ϕ.

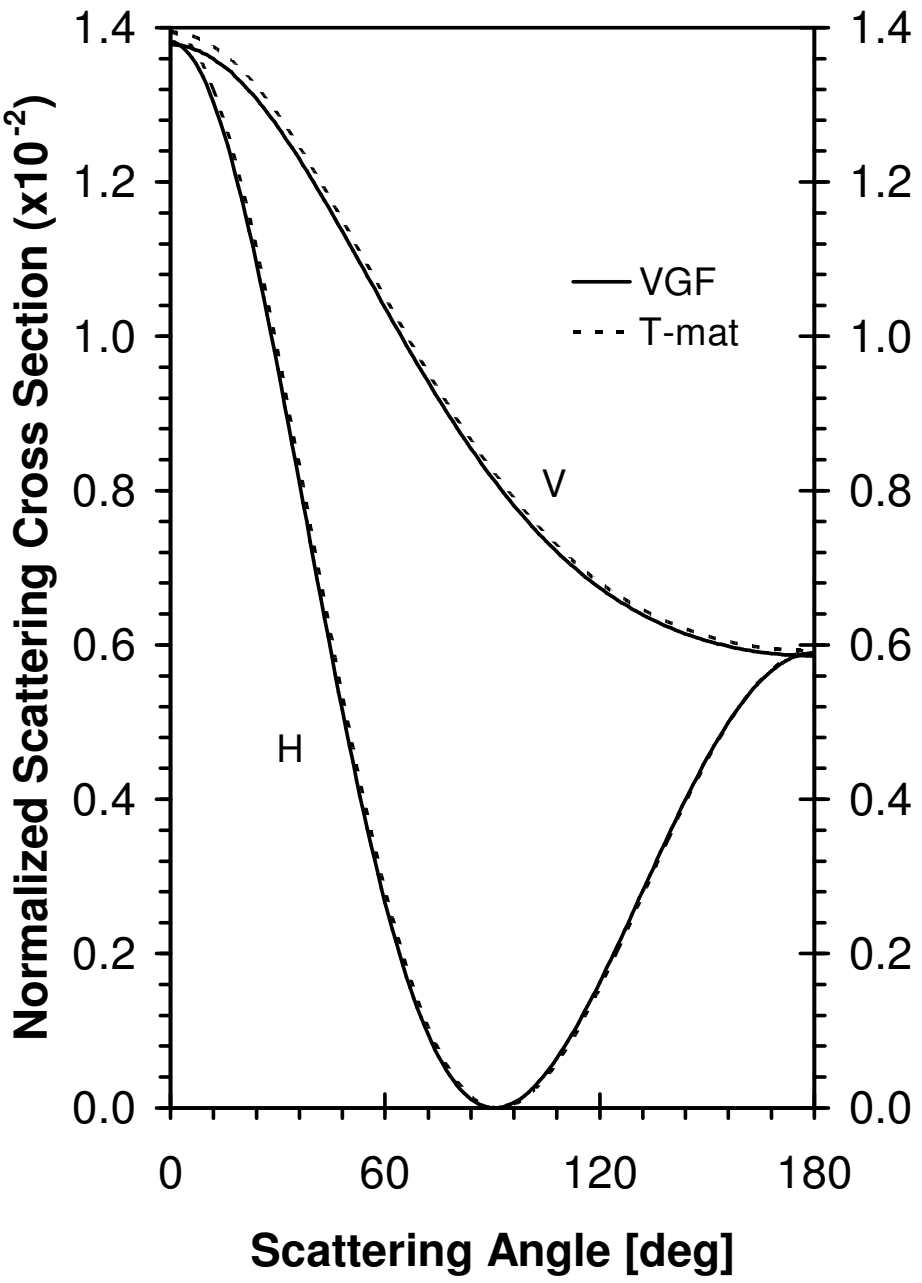


Figure 9. Scattering From an Oblate Spheroid: The spheroid has a semi-symmetry axis $a = 1.55\ \mu m$ with $a/b = 0.8$, and refractive index $m = 1.2 + i0.05$. The unweighted cell length is $d = 0.31\ \mu m$, $N = 289$, and $\lambda = 10\ \mu m$. Both polarizations are shown. The rotation angles are $\alpha = 0^\circ\ \beta = 0^\circ$, and $\gamma = 180^\circ$. The heavy solid curve is the VGF result, the dotted curve is the T-matrix result for comparison. There were 32 k-vectors evenly distributed in θ and ϕ.

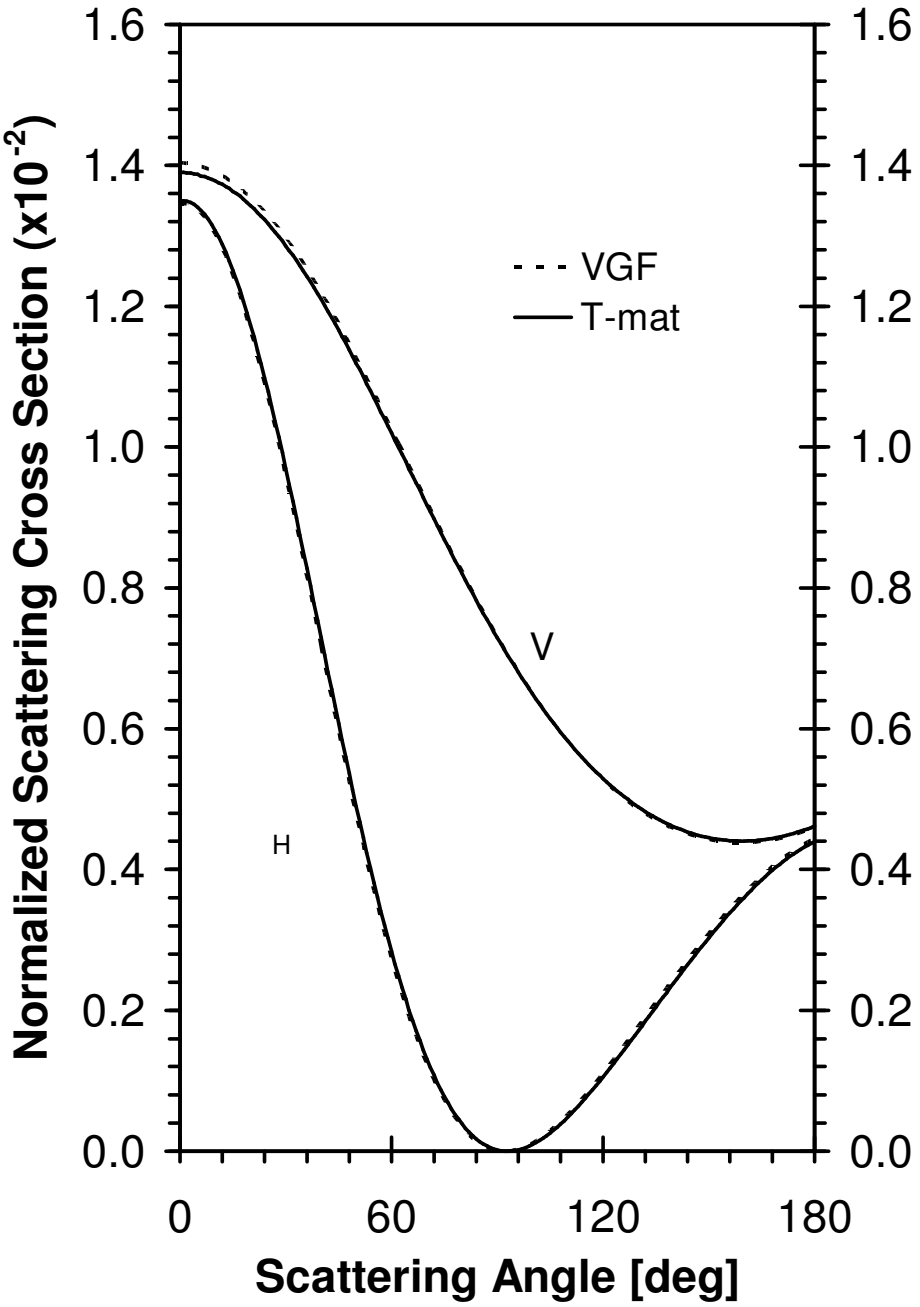


Figure 10. Scattering From a Oblate Spheroid Rotated Once: The spheroid has a semi-symmetry axis $a = 1.55\ \mu m$ with $a/b = 0.8$, and refractive index $m = 1.2 + i0.05$. The unweighted cell length is $d = 0.31\ \mu m$, $N = 289$, and $\lambda = 10\ \mu m$. Both polarizations are shown. The rotation angles are $\alpha = 0^\circ$ $\beta = 45^\circ$, and $\gamma = 180^\circ$. The heavy solid curve is the VGF result, the dotted curve is the T-matrix result for comparison. There were 32 k-vectors evenly distributed in θ and ϕ.

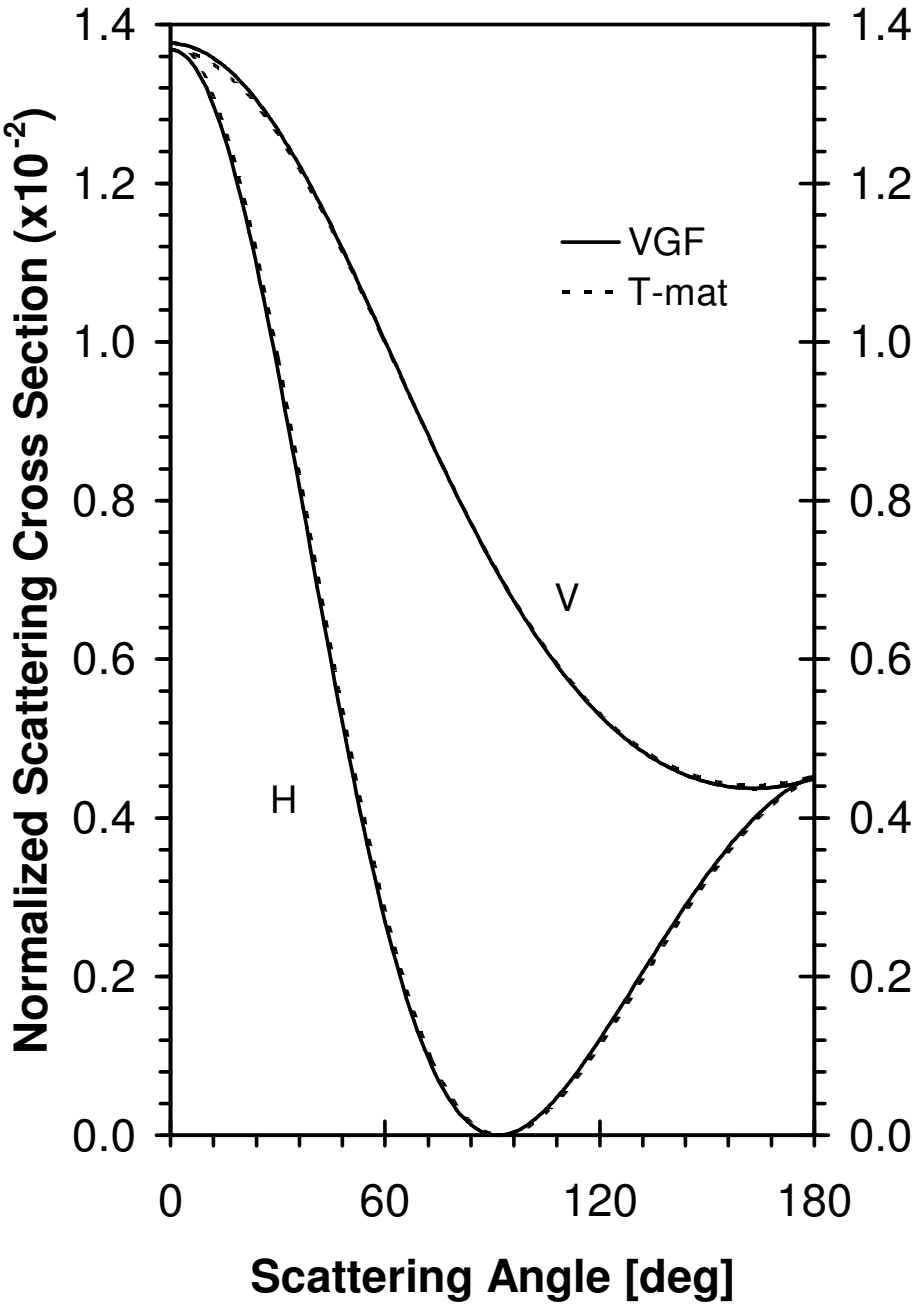


Figure 11. Scattering From a Oblate Spheroid Rotated Twice: The spheroid has a semi-symmetry axis $a = 1.55\ \mu m$ with $a/b = 0.8$, and refractive index $m = 1.2+i0.05$. The unweighted cell length is $d = 0.31\ \mu m$, $N = 289$, and $\lambda = 10\ \mu m$. Both polarizations are shown. The rotation angles are $\alpha = 0^\circ$ $\beta = 45^\circ$, and $\gamma = 180^\circ$. The heavy solid curve is the VGF result, the dotted curve is the T-matrix result for comparison. There were 32 k-vectors evenly distributed in θ and ϕ.

Cubical Particles

The calculation of scattering from a cube is much harder than for a sphere or an axisymmetric particle. EBCM or T-matrix type solutions which involve expansions in orthogonal functions do not do well with corners or straight edges. Therefore, the calculations that follow are not compared to these methods. Instead, comparisons are made with the DGF code.

The first example is for a small cube having the same volume as the small sphere of Figure 3, giving the cube a side length of $3.65\ \mu m$. The wavelength $\lambda = 10\ \mu m$. The index of refraction was $1.33 + i0.1$ and the cube was oriented with its sides parallel to the axes. The cube was divided into 216 cells ($d = 0.608$). The k-vectors were placed in five rings at $\theta_k = 30°$, $60°$, $90°$, $120°$, and $150°$ with six vectors in each ring at $\phi_k = 60°$, $120°$, $180°$, $240°$, $300°$, and $360°$ with an additional vector in the forward and backward direction for a total of 32 k-vectors. The result is given in Figure 12. The DGF solution for a cube divided into 216 cells is plotted for comparison. The VGF and DGF solutions agree very well with a difference in the forward peak of less than 0.5%.

The general form of the normalized differential scattering cross section for the small cube is similar to the spherical case. To compare the two cases, the cube solution is plotted in Figure 13 along with the solutions for a sphere of equal volume, a sphere of equal projected area, and a sphere with radius equal to half the cube side length. The DGF result is also given.

The second cube example was much larger. The cube side length was $17.188\ \mu m$ with index of refraction $m = 1.089 + i0.18$. The wavelength $\lambda = 10.8\ \mu m$. The

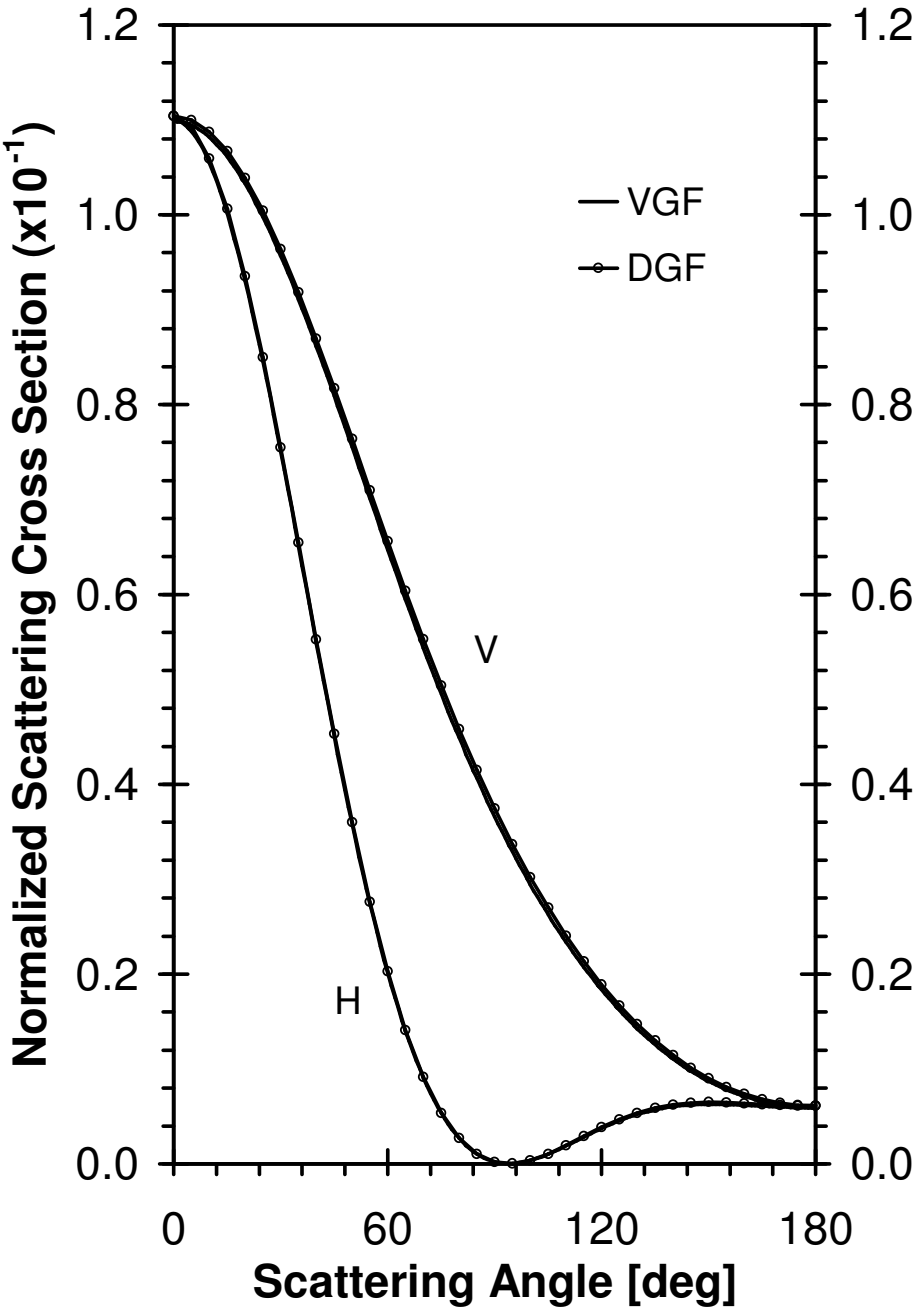


Figure 12. Scattering from a Small Cube: The cube has sides of $3.65\ \mu m$ with an index of refraction of $1.33 + i0.1$. It is illuminated by a plane wave of wavelength $\lambda = 10\ \mu m$ traveling in the $+\hat{z}$-direction. The cube is oriented with its sides parallel to the x, y, and z-axis. The particle was divided into 216 cells ($d = 0.609\ \mu m$). The k-vectors were placed in 5 rings at $\theta_k = 30°, 60°, 90°, 120°$, and $150°$. In each ring there were 10 vectors placed at $\phi_k = 60°, 120°, 180°, 240°, 300°$, and $360°$. There were two additional vectors at $(\theta_k, \phi_k) = (0°, 0°)$ and $(\theta_k, \phi_k) = (180°, 0°)$.

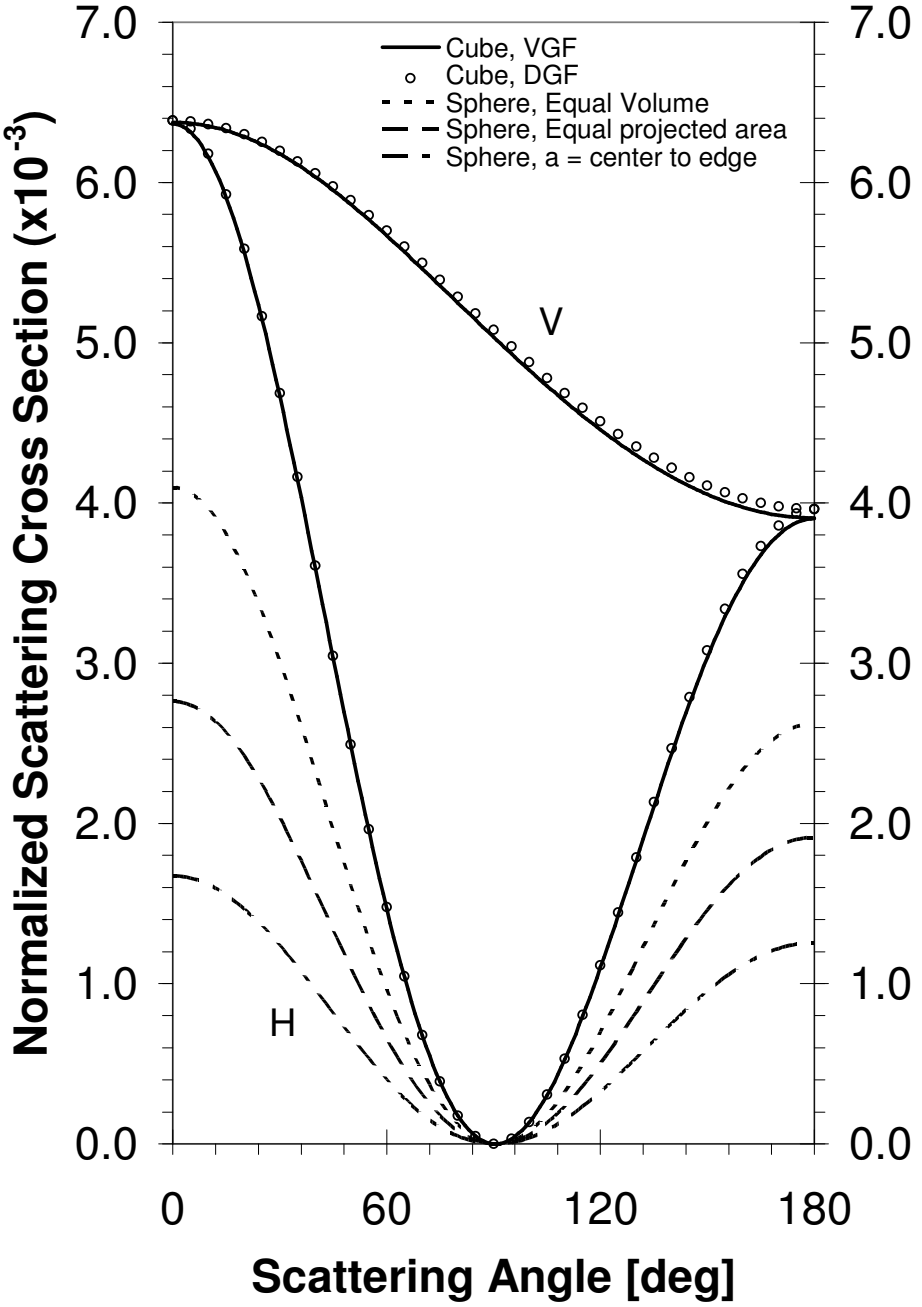


Figure 13. Scattering from a Small Cube Compared to a Sphere: The cube has sides of $1.813\ \mu m$ and an index of refraction of $1.33 + i0.1$. It is illuminated by a plane wave of wavelength $\lambda = 10\ \mu m$. The cube is oriented with its sides parallel to the x, y, and z-axes. The cube is divided into 1000 cells $(d = 0.18)$. The VGF scattering solution is the solid line. The solutions for a sphere of equal volume and equal projected area are given as the dotted and dashed lines respectively. The solution for a sphere of radius equal to half the cube side length is the dash-dotted curve. The small circles give the DGF solution.

cube was oriented such that its sides were parallel to the axes. The cube was divided into 1728 cells with $d = 1.43\ \mu m$. The k-vectors were placed in 8 rings at $\theta_k = 20°$, $40°$, $60°$, $80°$, $100°$, $120°$, $140°$, and $160°$. In each ring there were 10 vectors placed at $\phi_k = 36°$, $72°$, $108°$, $144°$, $180°$, $216°$, $252°$, $288°$, $324°$, and $360°$. In addition to the rings there was one k-vector in the forward direction and one in the backward direction making a total of 82 k-vectors. The VGF and DGF solution for this cube are plotted in Figure 14 and again show good agreement with a difference in the forward peak less than 0.2%.

Number and Position of k-Vectors

The derivation of the scattering solution did not give any information as to the number and position of the k-vectors in equation (184). Clearly there must be an optimal number of expansion vectors. A very small number is probably not accurate enough, and a very large number of vectors would defeat the purpose of this technique. The placing of these vectors must also be important.

To answer the question of how many expansion vectors are required, the scattering solution presented in Figure 14 was again computed, but with different numbers of k-vectors. In each case the incident radiation had $\lambda = 10\ \mu m$. The results are presented in Figure 15 for four cases: with two rings of five vectors, five rings of six vectors, six rings of eight vectors, and ten rings of twelve vectors. In each case there was a vector in the forward and backward directions. The value of θ_k for each ring was obtained by evenly spacing the rings between $0\,°$ and $180\,°$ exclusively. The vectors in a given ring

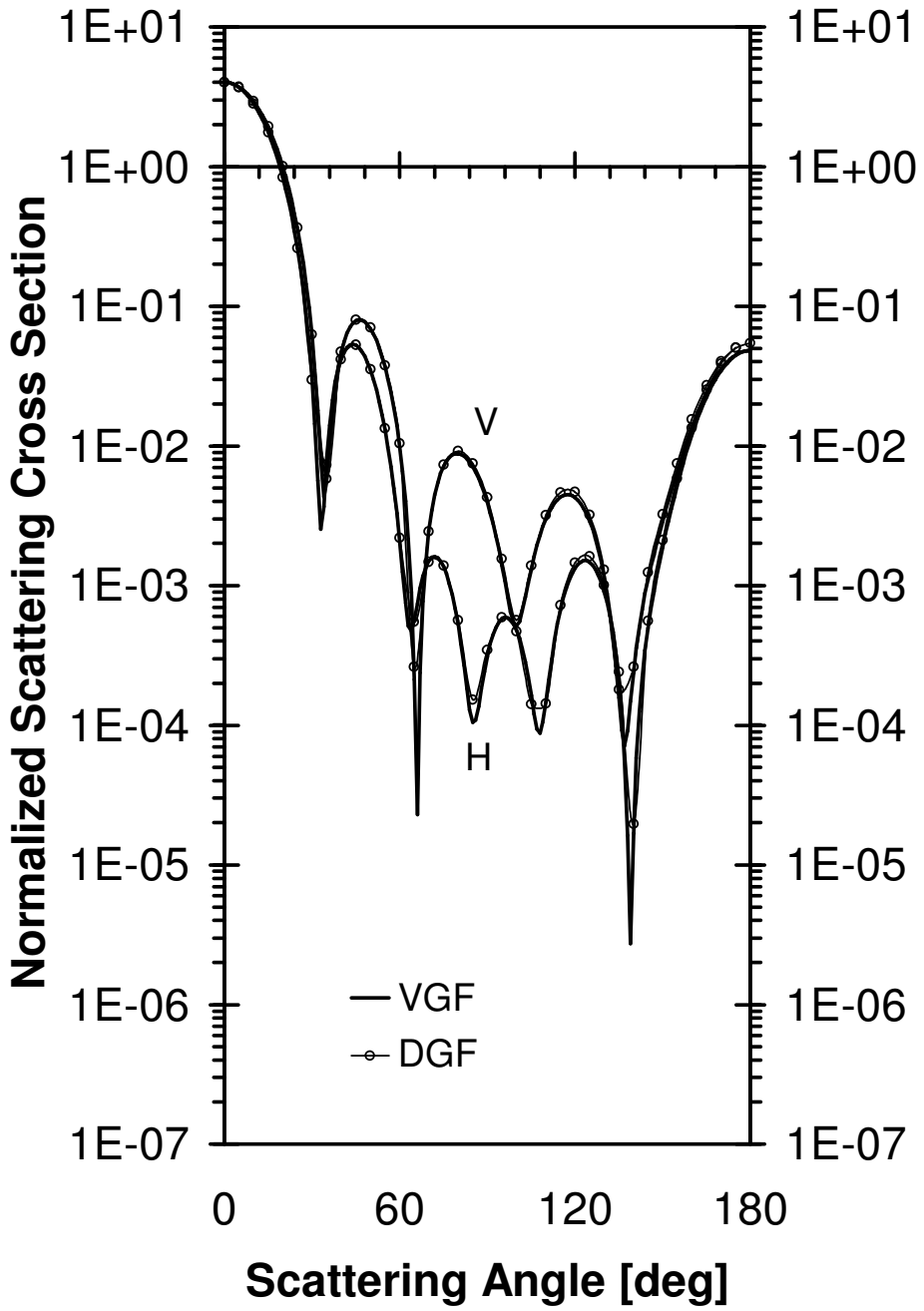


Figure 14. Scattering from a Large Cube: The cube has sides of $17.19\ \mu m$ with an index of refraction of $1.089 + i0.18$. It is illuminated by a plane wave of wavelength $\lambda = 10.8\ \mu m$ traveling in the $+\hat{z}$-direction. The cube is oriented with its sides parallel to the x, y, and z-axis. The particle was divided into 1728 cells ($d = 1.43\ \mu m$). The k-vectors were placed in 8 rings at $\theta_k = 20°$, $40°$, $60°$, $80°$, $100°$, $120°$, $140°$, and $160°$. In each ring there were 10 vectors placed at $\phi_k = 36°$, $72°$, $108°$, $144°$, $180°$, $216°$, $252°$, $288°$, $324°$, and $360°$. There were two additional vectors at $(\theta_k, \phi_k) = (0°, 0°)$ and $(\theta_k, \phi_k) = (180°, 0°)$.

were evenly spaced in azimuth between $0\,^\circ$ and $360\,^\circ$ with the final vector in each ring at $360\,^\circ$. As the number of expansion vectors increased, the solution improved, though there was less improvement between the 82 vector case and the 122 vector case than there was for the other changes in number of k-vectors. This seems to imply that a limit has been reached, beyond which increasing the number of k-vectors does not yield significantly improved accuracy. For particles with complex shapes, more k-vectors were required to adequately represent the internal electric field. The cube in Figure 14 required 82 k-vectors while 32 were used for the sphere in Figure 5. The value of the error Φ acts as a guide in choosing the number of k-vectors. The value of Φ is plotted in Figure 16 for the particle from Figure 15 with 12, 32, 50, 82, and 120 k-vectors.

To help answer the question of how best to choose the directions of the k-vectors, a spherical particle was studied with k-vectors in the forward and backward directions, and a ring of 18 k-vectors evenly spaced in azimuth (ϕ_k) about the particle. The polar angle, θ_k, was varied for the entire ring of vectors. The particle had an index of refraction of $1.33 + i0.1$ with radius $a = 1.4\ \mu m$. The incident wavelength $\lambda = 10\ \mu m$. The sphere was divided into 275 cells. The error in the forward peak and backscatter (the difference between the VGF and the Mie solutions) is shown in Figure 17 along with the error Φ. The oscillation in the error curves near $\theta_k = 0\,^\circ$ was to be expected, as was the general trend toward larger absolute errors for the forward scatter and smaller errors for the backscatter as θ_k increases. The two peaks in the error curve at $45\,^\circ$ and $145\,^\circ$ were unexpected. The peaks in the error curve are smaller for a larger particle with radius $1.54\ \mu m$, shown in Figure 18 but still present. Inspection of the

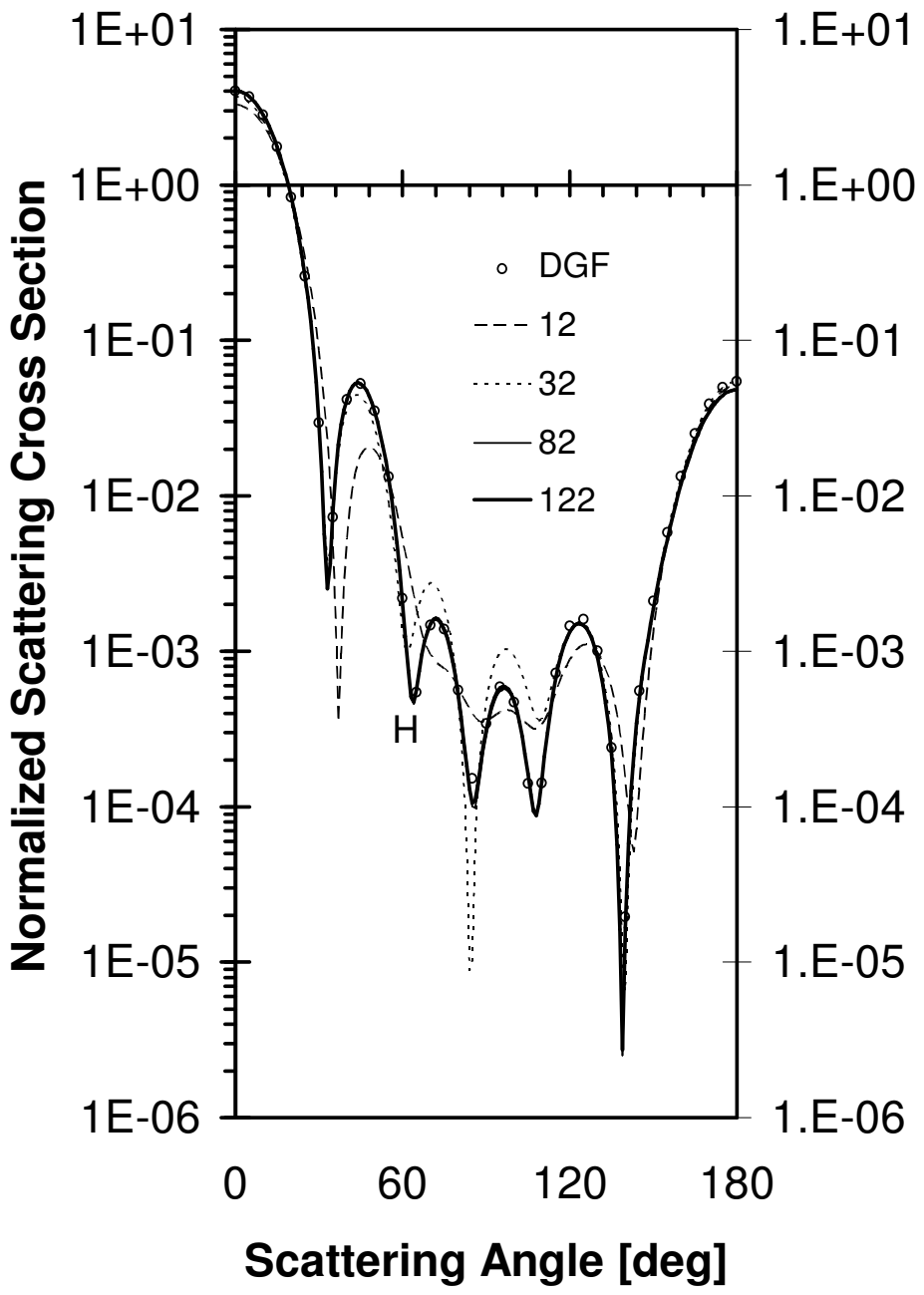


Figure 15. Number of k-Vectors: The solution for the cube presented in figure 14 is again presented. Different curves represent different numbers of k-vectors.

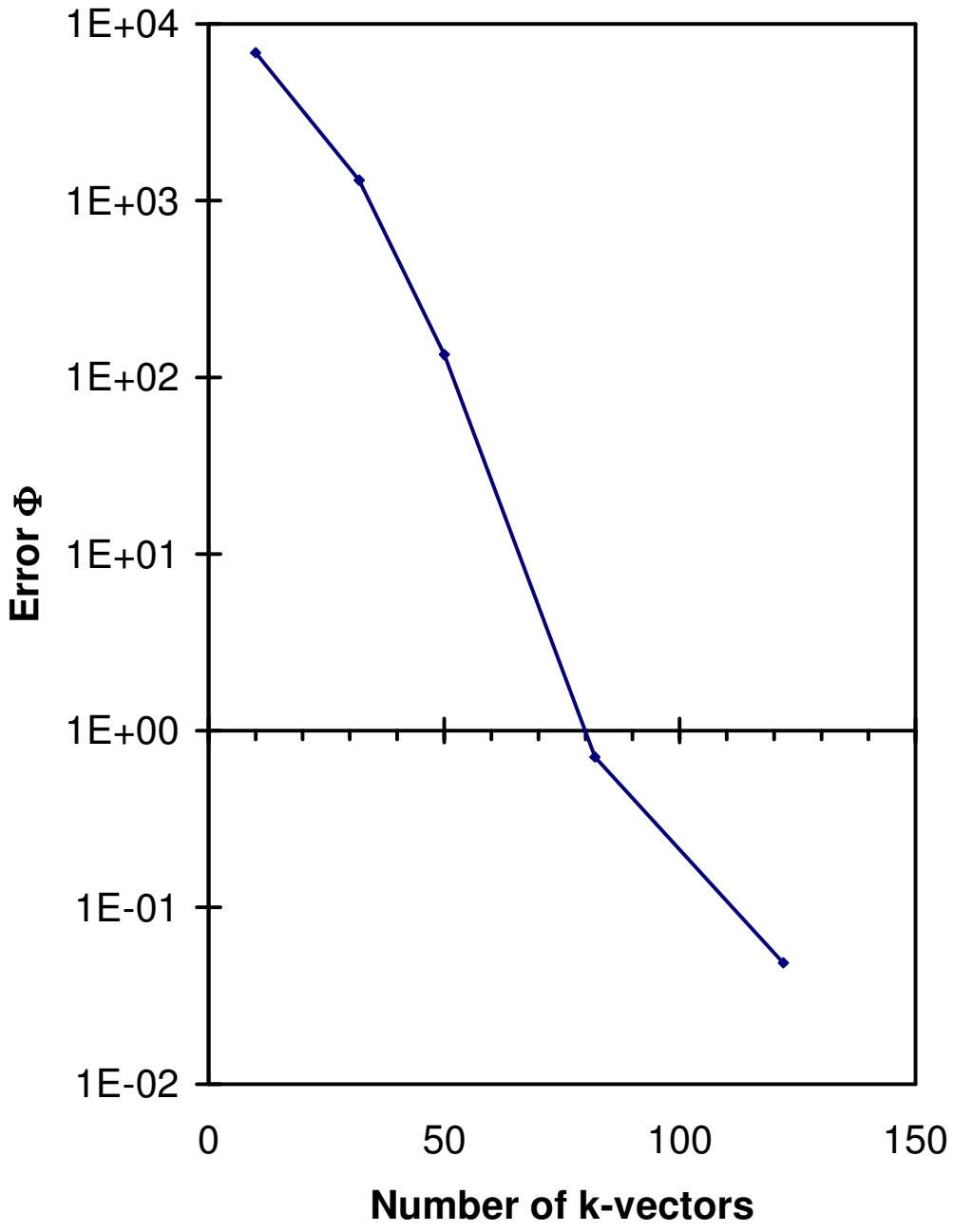


Figure 16. Error Φ for the Case Presented in Figure 15.

error Φ seems to confirm the general features of the error curves. The peaks in the error were smoothed out by adding another ring of vectors. An example is given where one set of vectors was fixed at $15\,^\circ$ while another ring moved from $\theta_k = 10\,^\circ$ to $\theta_k = 170\,^\circ$. Results are shown in Figure 19. Although there was variation, the magnitude of the error was very small for all values of θ_k for the second ring, implying that the two rings of k-vectors may be evenly spaced with little loss of accuracy. The values for the error Φ were correspondingly smaller which seems to confirm this result.

A further test suggested that a fairly even distribution of k-vectors is required if the accuracy is desired in the field in all directions. The particle from Figure 14 was again used. The different lines on the graph represent different configurations of k-vectors. Each configuration has one forward and one backward directed k-vector. The solid curve represents the same configuration of 122 k-vectors (ten rings of twelve vectors plus forward and backward vectors) that was presented in Figure 15. Compared to it are two configurations with 44 k-vectors each. One configuration was obtained by evenly spacing six rings of eight vectors. The other was obtained by taking out the inner most rings from the 122 k-vector configuration. This left vectors with $\theta_k < 20^\circ$ and $\theta_k > 160^\circ$. The results are given in Figure 20. Both smaller configurations represent the curve fairly well within about 40° of the forward scatter and back scatter directions. Near the 90° scatter direction, however, the curves represent the true cross section less well, and the case with no k-vectors from $20^\circ < \theta_k < 160^\circ$ shows large errors. Note that the differential cross section covers nearly six orders of magnitude from forward peak to the 90° scatter. It seems reasonable that without plane waves near

the 90° scatter direction the field expansion will not have sufficient terms to accurately describe the cross section in this direction. Also note that the principal differences in the curves occur at values at least three orders of magnitude lower than the value of the forward scatter. The contribution of this part of the curve to the integrated total cross section is small. Indeed, the values for the total scattering cross section for all three VGF curves in Figure 20 differ by less than 0.07%.

The value of the error Φ depends upon the number of cells. Once the condition in equation (157) is met, the value of Φ/N_c should be small. Thus to test for convergence, the condition

$$\frac{\Phi}{N_c} \ll 1 \tag{187}$$

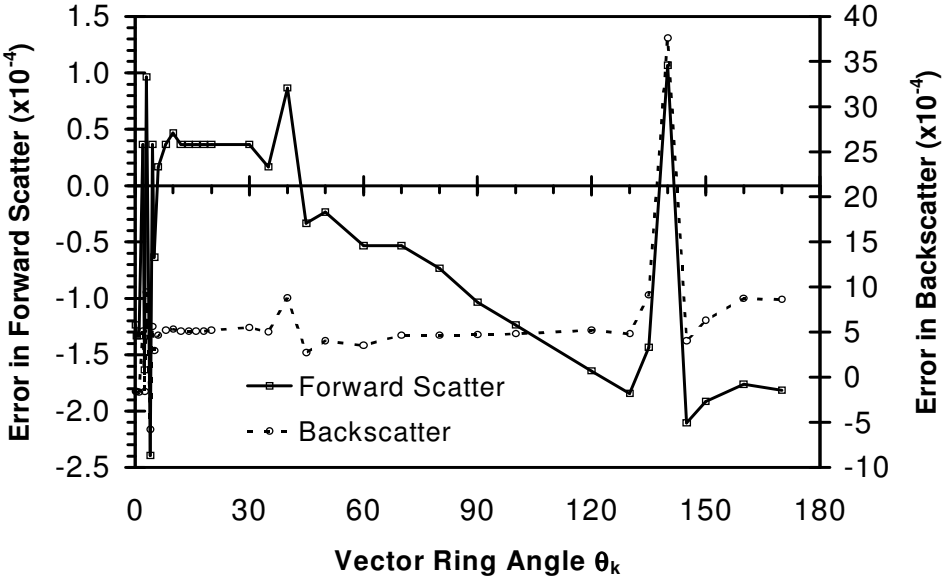
may be applied.

Van de Hulst formula

For non-absorbing spheres, van de Hulst[12] developed a formula that describes the salient features of the extinction efficiency.

$$Q_{ext} = \frac{\sigma_{ext}}{\pi a^2} = 2 - \frac{4}{\rho}\sin\rho + \frac{4}{\rho^2}\left(1 - \cos\rho\right) \tag{188}$$

where $\rho = 2X\left|m - 1\right|$. He demonstrated that, although this formula was supposed to be good for $m \approx 1.0$, it actually described the extinction quite well for values of m as large as 2. Because the forward and backscatter directions give the largest contribution to the scattering cross section, it was expected that calculations of the total extinction cross section for real-valued m could be performed with very few k-vectors arranged near $(\theta_s, \phi_s) = (0, 0)$ and $(\theta_s, \phi_s) = (\pi, 0)$. However, as $m \to 1$, the parameter



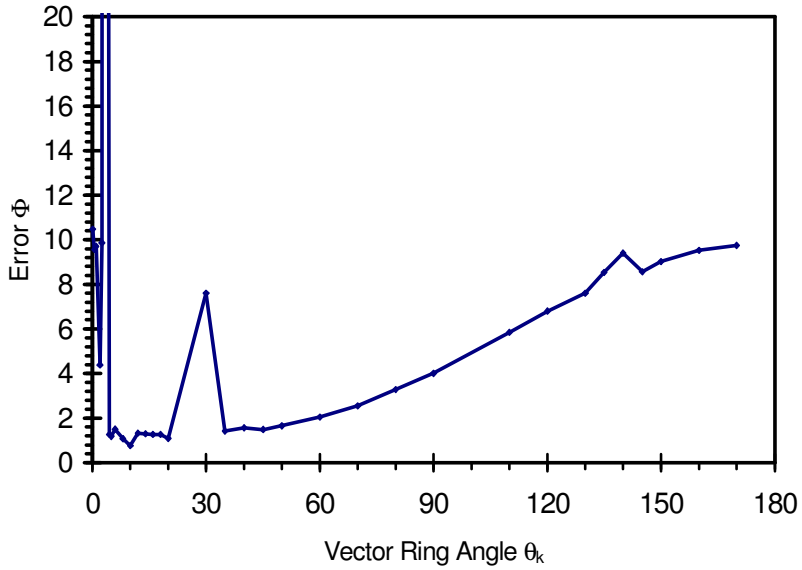


Figure 17. Position of the k-Vectors Example 1: The top graph shows the error in the value of the differential cross section for the forward scatter ($\theta = 0$) and backscatter ($\theta = \pi$) for a spherical particle as compared to the Mie code solution. The particle has radius of $1.4\ \mu m$ and index of refraction of $1.3 + i0.1$ and is illuminated by $10\ \mu m$ light traveling in the $+\hat{z}$-direction. Eighteen k-vectors were placed in a ring evenly spaced in ϕ_k and the value of θ_k was varied for the entire ring. Additional vectors in the forward and backward directions remained fixed. The values of the error are given as a function of the ring angle θ_k. Also shown is the value of Φ for each value of θ_k.

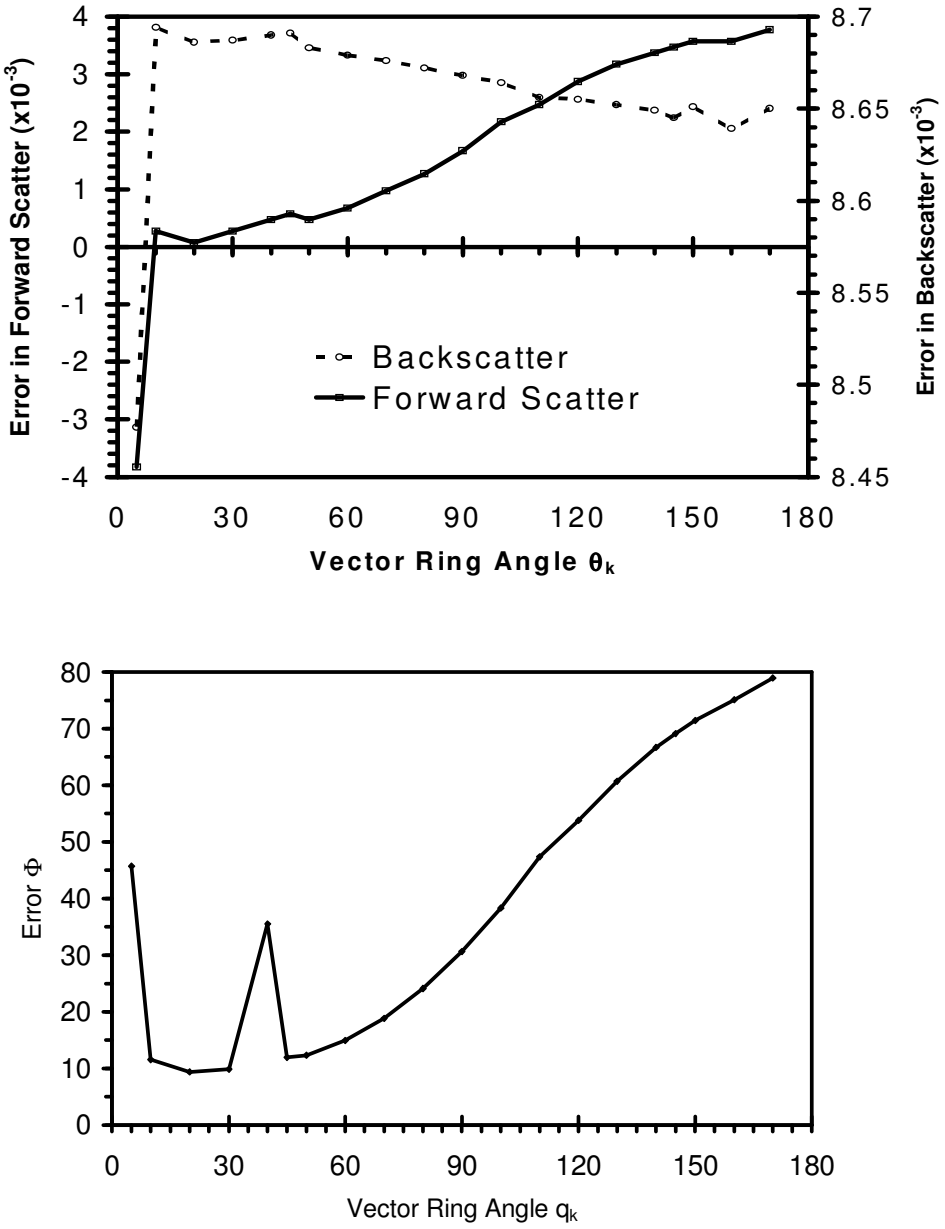


Figure 18. Position of the k-Vectors Example 2. The top graph shows the error in the value of the differential cross section for the forward scatter ($\theta = 0$) and backscatter ($\theta = \pi$) for a spherical particle as compared to the Mie code solution. The particle has radius of $1.5\ \mu m$ and index of refraction of $1.3 + i0.1$ and is illuminated by $10\ \mu m$ light traveling in the $+\hat{z}$-direction. Eighteen k-vectors were placed in a ring evenly spaced in ϕ_k and the value of θ_k was varied for the entire ring. Additional vectors in the forward and backward directions remained fixed. The values of the error are given as a function of the ring angle θ_k. Also shown is the value of Φ for each value of θ_k.

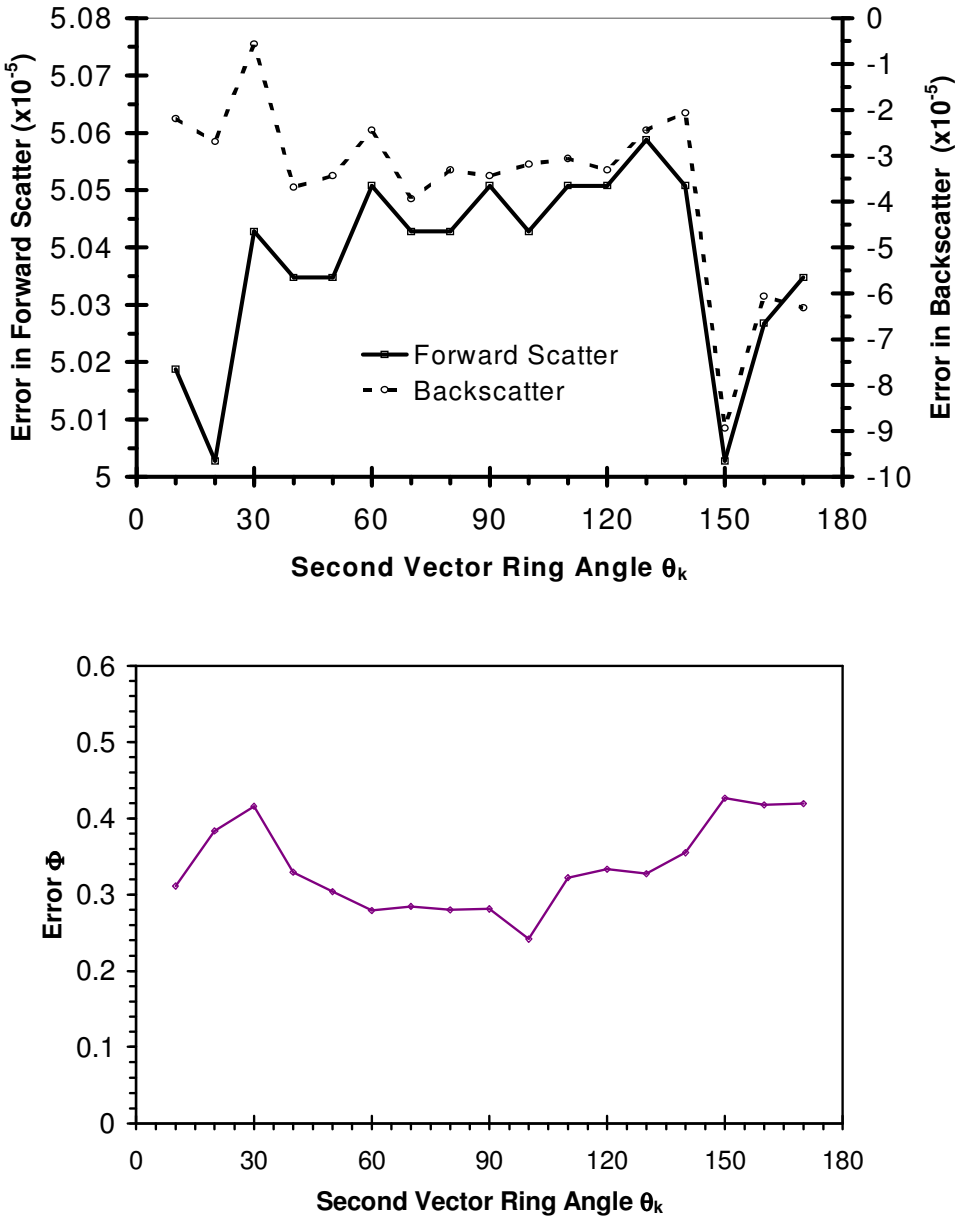


Figure 19. Two Rings of k-Vectors: The top graph shows the error in the value of the differential cross section for the forward scatter ($\theta = 0$) and backscatter ($\theta = \pi$) for a spherical particle as compared to the Mie code solution. The particle has radius of 1.4 μm and index of refraction of $1.3 + i0.1$ and is illuminated by 10 μm light traveling in the $+\hat{z}$-direction. The k-vectors were placed in two rings of 18 k-vectors evenly spaced in ϕ_k. One ring fixed at $\theta_k = 15°$ while the value of θ_k was varied for the other ring. Additional vectors in the forward and backward directions remained fixed. The values of the error are given as a function of the ring angle θ_k. Also shown is the value of Φ for each value of θ_k.

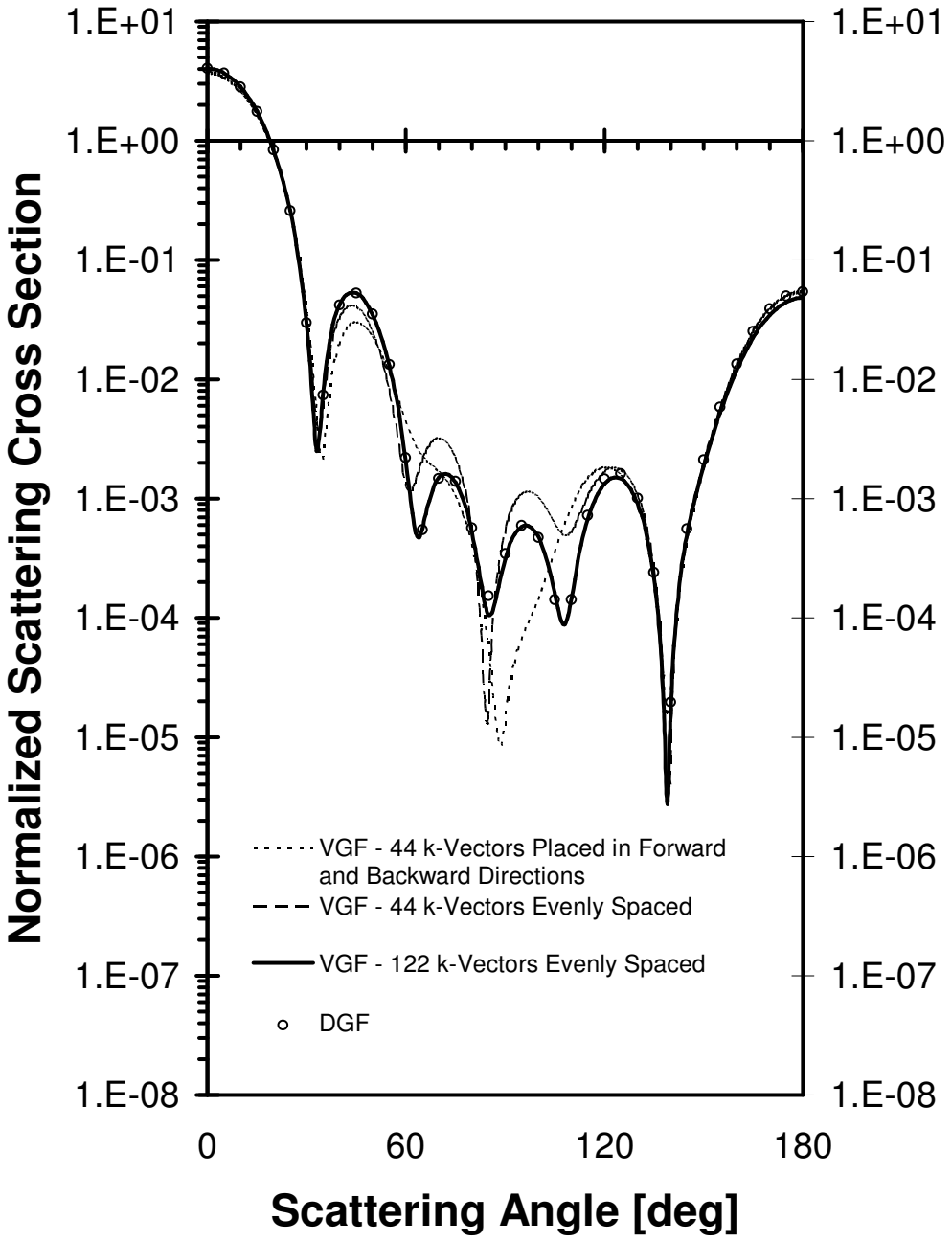


Figure 20. Even vs. Non-Even Placement of k-Vectors: The scattering from the particle from figure 14 is again presented with different configurations of k-vectors. Each configuration has one forward and one backward directed k-vector. The solid curve represents the same configuration of ten rings of twelve vectors that was presented in figure 15. Compared to it are two configurations with 44 k-vectors each. One with even spacing in θ_s of six rings of eight vectors evenly spaced in ϕ_s. The other was obtained by taking the inner most rings from the ten rings of twelve k-vector configuration. This left vectors with $\theta_s < 20^\circ$ and $\theta_s > 160^\circ$.

$\rho \to 0$, so to check this formula for low contrast particles, X tends to become large quickly. Thus the value of N_c becomes large. The curve in equation (188) is graphed in Figure 21.Perhaps more interesting is the fact that this curve describes fairly well the scattering from cubical particles. The line delineated by small squares in Figure 21 gives the function Q_{ext} for cubical particles calculated using the VGF method.

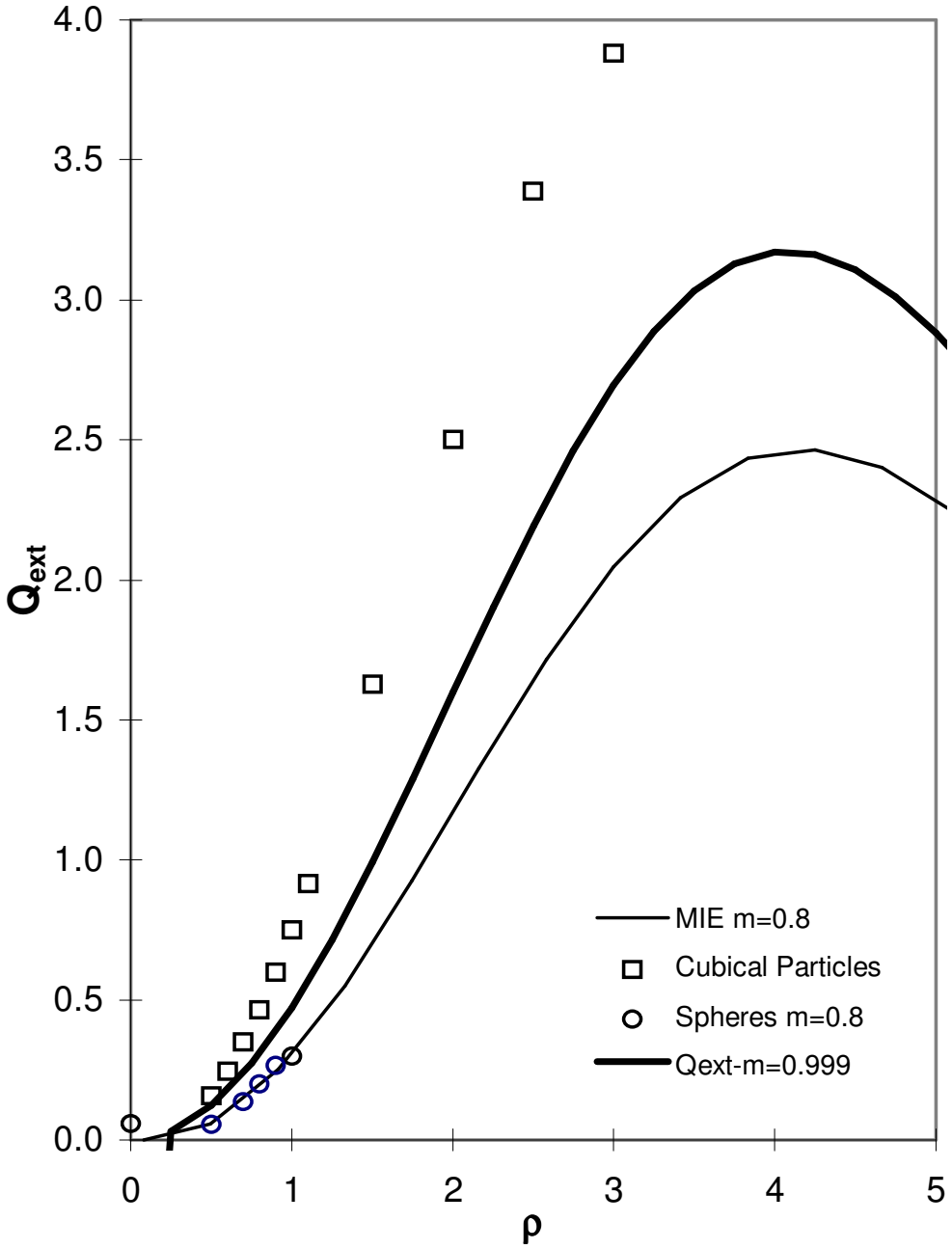


Figure 21. The van de Hulst formula for Q_{ext} for Low Contrast Particles. The heavy curve is the van de Hulst equation, the light solid curve is the result of Mie calculations for $m = 0.8$. The remaining curves give the VGF calculations for spherical and cubical particles.

Chapter 6

SUMMARY AND DISCUSSION

Discussion of Results

It has been shown that the results of the Variational Green Function method for determining the scattering from homogeneous particles are good for all the particles studied. From the results presented, a fairly even distribution of k-vectors is required if accuracy is desired at all scattering angles. If, however, only the total scattering cross section is desired, the fine details of the field off the forward scatter direction are not required, and therefore fewer k-vectors may be used.

It should be noted that for large particles the differential scattering cross section spans many orders of magnitude. Accuracy in computation over many orders of magnitude is difficult in any computer aided solution. For cross section features far below the forward peak value, there is some question of which code will be more accurate. The VGF code and the Mie, T-Matrix, and DGF codes agree quite well on all features presented in this work . However, some care must be taken to ensure sufficient machine precision in all codes. The VGF code achieved sufficient accuracy for the purpose of this work with single precision arithmetic. Certainly the matrix inversion of the $N_k \times N_k$ matrix can be done with sufficient accuracy in single precision. But for larger particles double precision–especially in the complex quantities– should be considered.

Advantages of the VGF Method

The VGF method has the advantage of avoiding computational problems in matrix inversion. The DGF must invert a $3N_c \times 3N_c$ complex matrix. Currently the DGF code uses a simple LU decomposition code which is good for $3N_c$ on the order of a few hundred. Many special matrix inversion techniques could be employed to extend the DGF to the realm of $3N_c$ on the order of several thousand. The VGF method allows the ordinary LU decomposition to apply to particles with $3N_c$ on the order of tens of thousands because the $3N_c \times 3N_c$ complex matrix inversion has been replaced by a much smaller $3N_k \times 3N_k$ complex matrix inversion. The same advanced techniques that could extend the DGF method to $3N_c$ on the order of a thousand could also extend the VGF method perhaps to $3N_c$ on the order of hundreds of thousands. This is certainly a direction for future work.

The VGF method also uses far less memory than the DGF and other CD codes. The implementation here was a compromise between speed and low memory usage, but is not the only possible implementation. Some values were stored in the matrix $T_{\alpha iNj}$ so they would not have to be recalculated. This is the only "optimization" used in the code. The guiding principle in code design was to make the code easily understandable for the purpose of demonstrating the method. Future work could optimize the code for speed and future versions of the code will require much less execution time.

Problems and Limitations

It should be repeated that the VGF method does retain the same requirement on number of cells N_c as in the DGF method. Thus for large N_c the storage requirement for the VGF method may actually be dominated by the need to store the N_c cell locations. The VGF solution is also restricted at present to homogeneous particles.

Other limitations noted for CD methods are also true for the VGF method. For example, the scattering cross section in the backscatter direction is less accurate in DGF calculations as compared to Mie calculations. Likewise, the scattering cross section in the backscatter direction is less accurate in VGF calculations as compared to Mie calculations.

Applications

The product of this work is a new method for obtaining the fields scattered from a homogeneous particle. However, finding the scattered field is often only part of a larger problem. This work began by examining the problem of radiative transfer. The radiative transfer equation for the simple case of an optically thin cloud of particles is given below[13].

$$-\frac{1}{\kappa\rho}\left(\mathbf{s}\cdot\boldsymbol{\nabla}\right)I\left(\mathbf{r},\mathbf{s}\right)=I\left(\mathbf{r},\mathbf{s}\right)-\frac{1}{4\pi}\int P\left(\mathbf{s},\mathbf{s}'\right)I\left(\mathbf{r},\mathbf{s}'\right)d\Omega_{s'} \tag{189}$$

where κ is the mass absorption coefficient, ρ is the density of the material, $\mathbf{s}$ is the direction of travel for the radiation, $I\left(\mathbf{r},\mathbf{s}\right)$ is the specific intensity, and $P\left(\mathbf{s},\mathbf{s}'\right)$ is the

scattering phase function. The dependence on the incident and scattered directions is shown explicitly as $(\mathbf{s}, \mathbf{s}')$. The integration is over solid angle and the element of solid angle is $d\Omega$. The phase function gives the rate at which energy is being scattered into and element of solid angle $d\Omega$. The phase function is defined in terms of the scattering cross section.

$$\sigma_{scat} = \frac{1}{k^2} \int P(\mathbf{s}, \mathbf{s}') \, d\Omega = \frac{1}{k^2} \int \frac{d\sigma_{scat}}{d\Omega} d\Omega \tag{190}$$

Most realistic particle clouds consist of a distribution of particle sizes, $N(a)$ where a is the particle characteristic radius. A simplification may be employed

$$\sigma_{scat,eff} = \int_0^\infty \sigma_{scat}(a) \, N(a) \, da \tag{191}$$

Thus, to model atmospheric particle constituents, the cross section for each species in the distribution may be calculated and combined with the size distribution to give an effective cross section σ_{eff} that represents the combined optical properties of the cloud. This effective cross section may be calculated once for use in radiative transfer models. For example, such effective values are routinely employed for the gaseous components, cloud droplets, ice particles, aerosols, and precipitation in atmospheric radiative transfer models and weather forecast models[12][37][38]. For particulates, with variable size distributions the various $\sigma_{scat}(a)$ values can be calculated, stored, and used to calculate equation (191) as required. Cloud particles are generally spherical, thus Mie theory may be used more efficiently to model the scattering due to cloud particles. Rain[39], snow[25], ice particles[40], aerosols[41] and military obscurants are generally not spherical and require more general methods like the VGF

to calculate their scattering properties. The calculation of the scattering properties for such particles will be the subject of future study.

Future Studies

The principle behind the VGF method is the solution by variational technique. Because the variational technique requires a trial function $\tilde{F}_{\beta j}$, the solution will be affected by the nature of the trial function chosen. The plane wave form, $\exp\left(imk\hat{\mathbf{k}}_N \cdot \mathbf{r}_\beta\right)$, chosen for this work is a good choice for larger particles. It is possible that there is a better choice for specific types of particles. For this reason, other functional forms should be investigated.

Code optimization, as has been discussed, should be a serious area of study for future work. Using compiler optimization can decrease execution time by half. More execution time savings can no doubt be achieved by taking advantage of symmetries in the matrices calculated. Ever increasing computer speeds will also reduce execution time.

Advances in computer languages promise to allow temporary storage to be freed for other use, allowing more time savings in computing H_{MlNj} and Y_{Nj} without tying up storage needed during the matrix inversion.

Advanced matrix inversion techniques are another area for future study. The VGF implementation provided here purposely used a simple and well understood method of solving equation (180). Such techniques may allow very large and complex particles to be studied by researchers with only modest computer resources.

Ultimately, it is hoped that this work will provide a valuable new tool for the betterment of radiative transfer and numerical forecast modeling.

REFERENCES

[1] K.-N. Liou, *An Introduction to Atmospheric Radiation* (Academic Press, NY, 1980).

[2] G. Mie, Ann. Physik **25,** 337 (1908).

[3] S. Asano and G. Yamamoto, "Light Scattering by a Spheroidal Particle," Applied Optics **14,** 29–49 (1975).

[4] M. F. Iskander, A. Lakhtakia, and C. H. Durney, "A New Iterative Procedure to Solve for Scattering and Absorption by Dielectric Objects," Proceedings of the IEEE **70,** 1361–1362 (1982).

[5] J. H. Richmond, IEEE Trans. Ant. Prop. **AP-13,** 334 (1965).

[6] E. M. Purcell and C. R. Pennypacker, "Scattering and Absorption of Light by Nonspherical Dielectric Grains," Astrophys. J. **186,** 705–715 (1973).

[7] L. S. B. Cammack, Master's thesis, New Mexico State University, Las Cruces, New Mexico, 1983.

[8] G. H. Goedecke and S. G. O'Brien, "Scattering by Irregular Inhomogeneous Particles via the Digitized Green's Function Algorithm," Appl. Opt. **27,** 2431–2438 (1988).

[9] B. T. Draine and J. Goodman, "Beyond Clausius-Mossotti: Wave Propagation on a Polarizable Point Lattice and the Discrete Dipole Approximation," Astrophys. J. **405,** 685–697 (1993).

[10] C. Acquista, "Light Scattering by Tenuous Particles: A Generalization of the Rayleigh-Gans-Rocard Approach," Applied Optics **15,** 2932–2936 (1976).

[11] J. R. Wait, Canadian J. of Phys. **33,** 189 (1955).

[12] H. C. van de Hulst, *Light Scattering by Small Particles* (Dover, NY, 1981).

[13] S. Chandrasekhar, *Radiative Transfer* (Dover, NY, 1960).

[14] M. M. Herman, "Développements en fonctions généralisées de Legendre, des terms de la matrice de diffusion d́une particle sphérique, dédutis de la théorie de Mie," C. R. Acad. Sci. Paris **260,** 468–471 (1965).

[15] M. Herman and J. Lenoble, "Asymptotic Radiation in a Scattering and Absorbing Medium," J. Quant. Spectrosc. Radiat. Transfer **8,** 355–367 (1968).

[16] C. Yeh, J. Math. Phys. **4,** 65 (1963).

[17] C. Yeh, J. Opt. Soc. Am. **55,** 309 (1965).

[18] P. C. Waterman, "Matrix Formulation of Electromagnetic Scattering," Proceedings of the IEEE pp. 805–812 (1965).

[19] P. Barber and C. Yeh, "Scattering of electromagnetic waves by arbitrarily shaped dielectric bodies," Applied Optics **14,** 2864–2872 (1975).

[20] M. F. Iskander, A. Lakhtakia, and C. H. Durney, "A New Procedure for Improving the Solution Stability and Extending the Frequency Range of the EBCM," IEEE Trans. Ant. Prop. **AP-31,** 317–324 (1983).

[21] M. F. Iskander and A. Lakhtakia, "Extension of the iterative EBCM to calculate scattering by low-loss or lossless elongated dielectric objects," Applied Optics **23,** 948–953 (1984).

[22] K. A. Fuller, "Optical resonances and two-sphere systems," Applied Optics **30,** 4716–4731 (1991).

[23] P. W. Barber and S. C. Hill, *Light Scattering by Particles: Computational Methods* (World Scientific, New Jersy, 1990).

[24] S. B. Singham and G. C. Salzman, "Evaluation of the Scattering Matrix of an Arbitrary Particle using the Coupled Dipole Approximation," J. Chem. Phys. **84,** 2658–2667 (1986).

[25] S. G. O'Brien and G. H. Goedecke, "Propagation of polarized millimeter waves through falling snow," Applied Optics **27,** 2445–2450 (1988).

[26] S. B. Singham and C. F. Bohren, "Scattering of Unpolarized and Polarized Light by Particle Aggregates of Different Size and Fractal Dimension," Langmuir **9,** 1431–1435 (1993).

[27] B. Millard and R. T. Lines, "Scattering from a thin circular disk," Technical report (1993) .

[28] B. T. Draine, "The Discrete-Dipole Approximation and its Application to INterstellar Graphite Grains," Astrophys. J. **333,** 848–872 (1988).

[29] S. B. Singham and C. F. Bohren, "Light scattering by an arbitrary particle: The scattering-order formulation of the coupled-dipole method," J. Opt. Soc. Am. A **5,** 1867–1872 (1988).

[30] G. H. Goedecke, *Classical Electromagnetic Theory* (Unpublished, 1996).

[31] S. G. O'Brien, Ph.D. thesis, New Mexico State University, Las Cruces, New Mexico 88001, 1985.

[32] P. J. Flatau, G. L. Stephens, and B. T. Draine, "Light scattering by rectangular solids in the discrete-dipole approximation: a new algorithm exploiting the Block-Toepliz structure," Journal of the Optical Society of Americal **7,** 593–600 (1990).

[33] B. T. Draine and P. J. Flatau, "The Discrete-Dipole Approximation for Scattering Calculations," J. Opt. Soc. Am. (1994).

[34] C. F. Bohren and D. R. Huffman, *Absorption and Scattering of Light by Small Particles* (Wiley, NY, 1983).

[35] J. D. Jackson, *Classical Electrodynamics*, 2nd ed. (John Wiley & Sons, New York, 1975).

[36] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 77* (Cambridge University Press, Cambridge, UK, 1992).

[37] R. M. Measures, *Laser Remote Sensing: Fundamentals and Applications* (John Wiley & Sons, New York, 1984).

[38] V. J. F. Jr., L. W. Abreu, and E. P. Shettle, "Atmospheric Attenuation of Millimeter and Submillimeter Waves: Models and Computer Code AFGL-TR-79-0253," Technical report, Hanscome AFB, Massachusetts (1979) .

[39] G. L. Stephens, *Remote Sensing of the Lower Atmosphere* (Oxford University Press, New York, 1994).

[40] D. L. Mitchell and W. P. Arnott, "A Model Predicting the Evolution of Ice Particle Size Spectra and Radiative Properties of Cirrus clouds. Part II: Dependence of Absorption and Extinction on Ice Crystal Morphology," Journal of the Atmospheric Sciences **Vol. 51,** 817–832 (1994).

[41] C. F. Bohren and S. B. Singham, "Backscattering by Nonspherical Particles: A Review of Methods and Suggested New Approaches," Journal of Geophysical Research **96,** 5269–5277 (1991).

[42] C. B. M. Jack J. Dongarra, James R. Bunch and G. W. Stewart, *LINPACK User's Guide* (Society for Industrial and Applied Mathematics, Philadellphia, PA, 1979).

[43] C. B. J. D. J. D. J. D. C. A. G. S. H. A. M. S. O. E. Anderson, Z. Bai and D. Sorens, *LAPACK User's Guide* (Society for Industrial and Applied Mathematics,

Philadelphia, PA, 1992).

APPENDIX A COMPUTER CODE DESCRIPTION

The VGF method was tested using the following computer code. The code is divided into several different programs. The main program calculates the electric field inside the particle. The other programs perform the particle sub division into cells, define k-vectors, and calculate the cross sections. The entire code set is written in FORTRAN77. The code listings follow in Appendix B in the same order as they are described below.

VGFIN

The first code presented , VGFIN, takes in the particle descriptive parameters (index of refraction, orientation, shape, number of cells in bounding cube) and the wavelength of the incident radiation. The program produces the particle cell locations, weighting factors for computing d_ν and passes on the orientation and wavelength to the main program. In each of the codes in the VGF suite the definition of the array dimension parameters NMAX and KNMAX are defined in the file NMAX.INC by way of an include statement.

VGFKV

The next code , VGFKV, is a simple k-vector generation scheme. The code gives k-vectors in the forward and backward directions, and allows the user to specify a

number of rings of vectors and the number of vectors in each ring. The file also includes looping indices for using the Monte Carlo option in the main calculation.

VGFMC

VGFMC is the principal calculation found in equation (181) using equation (186). The input is the set of k-vectors and the cell positions and other information from VGFIN and VGFKV or equivalent. The output is the electric field at each cell position. The output file also contains the cell positions, values of the coefficients a_{Nj}, the k-vectors expressed as Euclidean unit vectors, and the error Φ. The VGFMC does allow a "Monte Carlo" type iteration that moves the k-vectors randomly in an attempt to produce a better answer for the field. The results presented do not use this option. The reason for this is the length of time needed to come to convergence. The answers do improve, however, and if extra precision is needed, this option exists. This iterative approach does not, however, overcome the need to meet the criteria on N_c given in equation (158), and it is not a substitute for using sufficient k-vectors. The output of VGFMC is interesting in itself, but the two codes which follow may be used to make the output more meaningful. The listing of VGFMC is given below. Note that the code was designed to be easily readable (i.e. to follow the equations as they are written in the text). No attempt has been made to optimize the code for speed.

VGFPHZ

VGFPHZ calculates the differential scattering cross section using the fields output from VGFMC. The HH, HV, VH, and VV components of the scattering cross section are output as a function of angle. VGFPHZ also uses the shared subroutines ROT and MV described below.

VGFSIG

VGFSIG calculates the total cross sections using the fields output from VGFMC. The extinction, scattering, and absorption cross sections are calculated directly from the fields, and the extinction cross sections is also calculated using the optical theorem. VGFSIG uses a modified version of the Simpson rule integration routine by Press et. al. [36]. VGFSIG also uses the shared subroutines ROT and MV listed below.

Shared Subroutines

Two subroutines are shared between the programs VGFMC, VGFPHZ, and VGFSIG. The first, ROT, calculates the rotation matrix given the angles α, β, and γ. The second multiplies a 3×3 matrix by a three component vector.

Numerical Linear Algebra Codes from LINPAC and LAPACK

The matrix inversion routines used by the field calculation in VGFMC code come from the LINPAC and LAPACK libraries[42][43]. The source code from the libraries is included below for completeness.

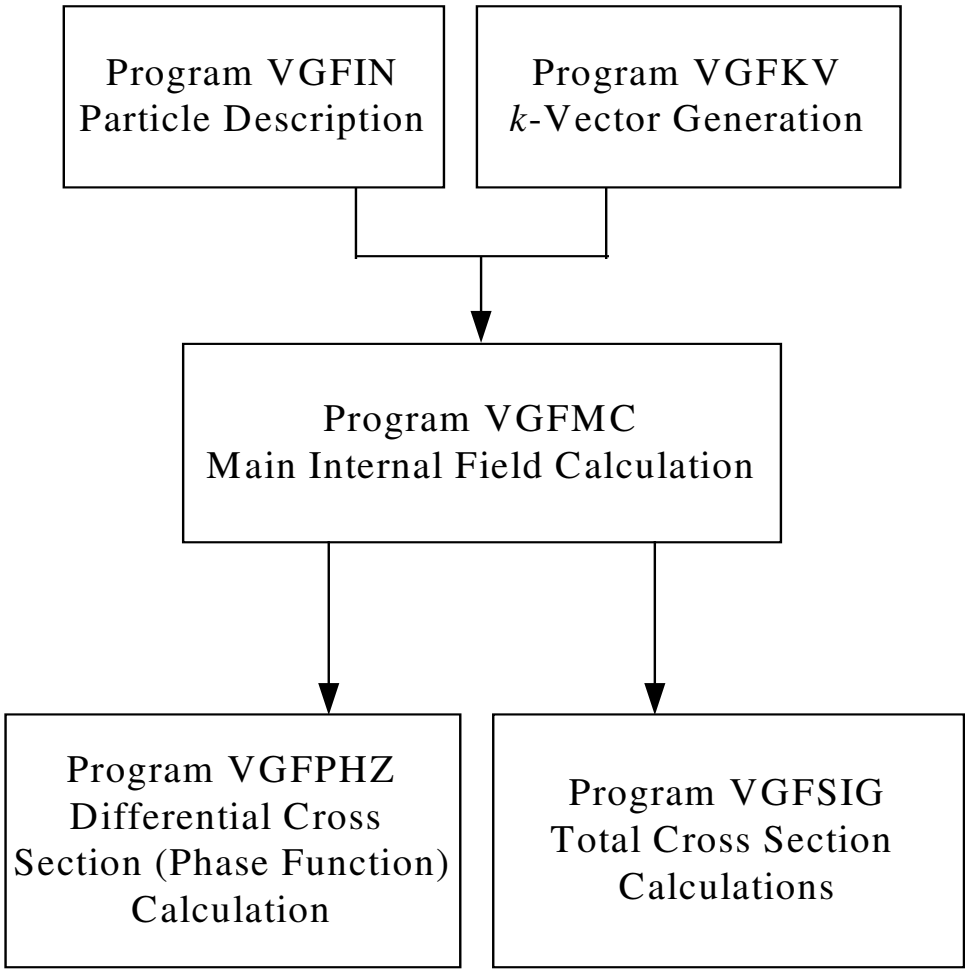


Figure 22. Structure of Programs in the VGF Suite

APPENDIX B CODE LISTINGS

```
C**********************************************************************
      program VGFIN
C**********************************************************************
C**** Program to set up a particle dipole array for the VGF code.      *
C****   A particle (sphere, oblate or prolate spheroid, cubical box or *
C****   spheroidal shell) is split into a cubic array of dipole cells  *
C****   (cubic cells with a dipole at the center).  A coarse array of  *
C****   NSID^3 dipole cells is defined using a cubic lattice. The      *
C****   particle is carved out of a NSID^3 cube of dipole cells. The   *
C****   coarse array cells are divided into NLSID fine array cells.    *
C****   Each of the fine array cells is tested to see if it is within  *
C****   the particle geometry. If so, it is counted.  The coarse array *
C****   cell is weighted  in volume (by adjusting the cell length on a *
C****   side (D) according to how many of its fine array cells are in  *
C****   the particle.                                                  *
C****                                                                  *
C**** Units:                                                           *
C****   Angles are input in degrees and converted to radians.          *
C****   Lengths are all relative, that is, if you input a wavelength   *
C****   of 10 um then all other lengths must be in um. You may use     *
C****   m or cm or furlongs if you wish, as long as all lengths are    *
C****   in the same units.                                             *
C****                                                                  *
C**** Time dependence:                                                 *
C****   The VGF code uses the time dependence exp(-iwt). This code     *
C****   (VGFIN) receives the index of refraction from the user and     *
C****   modifies the sign of the imaginary part to fit the VGF time    *
C****   time dependence convention (i.e. the imaginary part should be  *
C****   positive).                                                     *
C****                                                                  *
C**** Geometry:                                                        *
C****   The particle is aligned with its symmetry axis along           *
C****   the Z-axis.  The incident plane wave may be rotated            *
C****   around the particle by the angles alpha, beta, and gamma       *
C****   (corresponding to the Euler angles as given in Arfkin,         *
C****   Mathematical Methods for Physicists, 2nd Ed., Academic Press,  *
C****   NY, 1970). Gamma is effectively the polarization angle around  *
C****   the incident (k-hat) direction and is input as such.            *
C****                                                                  *
C**** Author: Original code by R. T. Lines                             *
C****                                                                  *
C**** extensive modification by Lines 8 APR 96                         *
C****  Last Modification by Lines 17 JAN 98                            *
C**********************************************************************
C**** Variable Definitions                                          ****
      implicit none
      include 'nmax.inc'
C *** Indices
      integer G, H, I, IX, IY, IZ
```

```
C *** Counters
      integer  NSID, NLSID, NCUB, INCNT, NUSE, IHP, IC
C *** Reals
      real W, RAD, alpha, beta, gamma, psi, MR, MI, AR
      real R(NMAX, 3), RF(3), D(NMAX), ER, EI
      real PI, TD, DRF, LD, FRAC, SDN, SD
      real x,y,z
      real temp
C *** Complex
      complex M, EPS, C1
C *** Logical (function)
      logical ISINSIDE
C *** Character
      character*80 OUTFILE, COMMENTS
C *** Parameters
      parameter (PI = 3.141592654)
      C1=cmplx(1.0,0.0)
      SDN=0.0
C**** Start the program by asking the user for data                ****
      write(6,*) 'Enter (0) for sphereoidal particles, (1) for cubes'
      read(5,*) IC
      write(6, *) 'Enter wavelength:'
      read(5, *) W
      write(6, *) 'Enter number of dipoles along major axes:'
      read(5, *) NSID
      write(6, *) 'Enter number of fine structure divisions'
      write(6, *) 'per dipole cell side:'
      read(5, *) NLSID
      write(6, *) 'Enter particle symmetry semi-axes (z-axes radius):'
      read(5, *) RAD
      write(6, *) 'Enter incident theta, phi, gamma (Degrees):'
      read(5, *) alpha, beta, gamma
      write(6, *) 'Enter polarization angle (Degrees):'
      read(5, *) psi
      write(6, *) 'Enter real & imag parts of index of refraction:'
      read(5, *) MR, MI
      write(6, *) 'Enter aspect ratio (major axis/minor axis)'
      read(5,*) AR
      write(6,*) 'Enter choice of solid sphere (0) or hollow shell (1)'
      read(5,*) IHP
      if (IHP.eq.1) then
        write(6,*) 'Enter number of skin depths for thickness of shell'
        read(5,*) SDN
      end if
      write(6, *) 'Enter file comments:'
      read(5, '(1a80)') COMMENTS
      write(6, *) 'Enter output filename:'
      read(5, '(1a80)') OUTFILE
      write(6, *)
C**** Start calculations                                           ****
C *** Calculate skin depth
```

```
      if (MI.ne.0.0) then
         SD=W/(2*pi*MI)
      else
         SD=0.0
      end if
C *** Get dipole coarse array cell length on a side i                 ***
      TD = 2.0*max(RAD,RAD/AR)/NSID
C *** Make sure the imaginary index of refraction is the right sign   ***
C *     We will use exp(ikr-iwt) in ADMF so MI positive                 *
      if (MI.LT.0.0) MI = -MI
      M = cmplx(MR, MI)
C *** Half a cell dimension ***
      DRF = 0.5*(float(NSID) - 1.0)
C *** Build coarse array... (first let user know what is happening)   ***
      write(6, *) 'Building coarse array...'
      NCUB = 0               ! Number of total cells (N-cubed)
C *** Fine array cell size (little D)                                 ***
      LD = TD/NLSID
C *** 1 over Volume of the fine array cell (used later)               ***
      FRAC = (1.0/float(NLSID))**3
C
      write(6, *) 'Calculating fine structure...'
C *** The number if cells within the particle geometry is counted in
C *** Nuse. This will be equal to Nc from the text.
C
      NUSE = 0
C
C *** Find the center positions of the fine array cells. Test to  see **
C *   if they are in the particle, if so calculate the volume weight   *
C *   if none of the coarse cell's fine array cells are within the     *
C *   particle geometry, then drop the coarse array cell.               *
      do G = 0, NSID-1
        do H = 0, NSID-1
          do I = 0, NSID-1
            x = (float(G) - DRF)*TD
            y = (float(H) - DRF)*TD
            z = (float(I) - DRF)*TD
            INCNT = 0
            do IX = 0, NLSID-1
              do IY = 0, NLSID-1
                do IZ = 0, NLSID-1
                  RF(1) = x + LD*(0.5*(1.0 - NLSID) + IX)
                  RF(2) = y + LD*(0.5*(1.0 - NLSID) + IY)
                  RF(3) = z + LD*(0.5*(1.0 - NLSID) + IZ)
                  if (ISINSIDE(RAD,RF,AR,SD,SDN,IHP,IC)) INCNT=INCNT+1
                end do   !IZ
              end do     !IY
            end do       !IX
            if (INCNT.GT.0) then               ! drop unfilled cells
              NUSE = NUSE + 1
              if (MOD(NUSE, 1000).EQ.0) write(6, *) NUSE
```

```
              R(NUSE, 1) = x
              R(NUSE, 2) = y
              R(NUSE, 3) = z
              D(NUSE) = TD * (INCNT*FRAC)**(1.0/3.0)   !weight cells
            endif
          end do
        end do
      end do
C**** DONE! Write out the cubic array and data to file              ****
      write(6, *) 'Writing file...'
30    open(10, file = OUTFILE)
      write(10, *) COMMENTS
      write(10, *) NUSE
      write(10, *) W, alpha, beta, gamma, psi, RAD
      EPS = M*M
      ER = real(EPS)
      EI = aimag(EPS)
      write(10,*) ER, EI, TD
      do I = 1, NUSE
        write(10, 100) R(I, 1), R(I, 2), R(I, 3), D(I)
        temp=temp+d(I)**3
      end do
      write(10,*) temp
      write(10,*) 4.*PI*RAD**3/3.
      close(10)
100   format(4(1x, 1e12.4))
      end

C*********************************************************************
C*--------------------------------------------------------------------
      logical function ISINSIDE(RAD, RF, AR, SD, SDN, IHP,IC)
C*--------------------------------------------------------------------
C*********************************************************************
C*    Function to determine if a fine array cell is inside the       *
C*      particle. The fine array cell is considered inside if its    *
C*      center point is within the particle.  Only spheroids and     *
C*      cubical boxes are considered here. This version is modified to *
C*      exclude cells within a number of skin depths to allow creation *
C*      of spherical shells if the flag IHP is set.                  *
C*********************************************************************
C**** VAriables                                                     ****
      real RAD, RF(3),AR, SD, SDN
      real RAD2,ARAD2,TRAD
      real SDRAD,SDRAD2,SDARAD2,SDTRAD
      integer IHP, IC
C *** If the particle is a cubical box, then just return            ***
      If (IC .eq. 1) then
         if ((RF(3).le.RAD/AR).and(RF(3).ge.-RAD/AR)) then
            ISINSIDE = .TRUE.
         end if
         return
```

```
      end if
C *** Calculate a semi-major axes that is several (SDN) skin depths  ***
C *     less than the given semi-major axes.                          *
      SDRAD=RAD-SDN*SD
C *** Use the spheroid equation to see if the point falls inside the ***
C *      spheroidal particle. Set up the equation here.               *
      RAD2=RAD*RAD
      ARAD2=RAD2/(AR*AR)
      TRAD = RF(1)*RF(1)/ARAD2                  ! spheroid equation
      TRAD = TRAD + RF(2)*RF(2)/ARAD2
      TRAD = TRAD + RF(3)*RF(3)/RAD2
      TRAD = (TRAD)**0.5
C *** Now if the particle radius is larger than (SDN) skin depths,   ***
C *       use the spheroidal equation to define a smaller spheroid     *
C *       inside the particle and test to see if the point is outside  *
C *       this spheroid.                                               *
      if ((SDRAD.gt.0).and.(IHP.eq.1)) then
         SDRAD2=SDRAD*SDRAD
         SDARAD2=SDRAD2/(AR*AR)
         SDTRAD = RF(1)*RF(1)/SDARAD2           ! spheroid equation
         SDTRAD = SDTRAD + RF(2)*RF(2)/SDARAD2
         SDTRAD = SDTRAD + RF(3)*RF(3)/SDRAD2
         SDTRAD = (SDTRAD)**0.5
      else
         SDTRAD=1.0
      end if
C *** See if it is between the two spheroids, watch out for roundoff.***
      ISINSIDE = ((TRAD.LE.1 + 1.0e-6).AND.(SDTRAD.GE.1 - 1.0e-6))
      return
      end
C**********************************************************************

C**********************************************************************
C*    Include file for maximum array size parameter.
C**********************************************************************
      integer NMAX
      parameter (NMAX = 3000)
      integer KNMAX
      parameter (KNMAX = 200)
C**********************************************************************
```

```
C*********************************************************************
C*--------------------------------------------------------------------
      Program VGFKV
C*--------------------------------------------------------------------
C*    Program to set up a kvector file for the VGF scattering code     *
C*    Author:  Origonal code  by Lines                                 *
C*********************************************************************
C *** Set the value of NMAX via an included file                     ***
      implicit none
      include 'nmax.inc'
      character*80 FNAME
C****  Variables
      real kth(KNMAX),kph(KNMAX),TH, PH,PI
      real delTh, delPh
      integer NTH,NPH,i,j,count ,N
      parameter (PI=3.141592654)
      write(*,*) 'Enter the number of angles theta (1-5)'
      read (*,*) NTH
      write(*,*) 'Enter the number of angles phi'
      read (*,*) NPH
      write(*,*) ' Enter the file name '
      read(*,'(1a80)') FNAME
        count=1
        Write (*,*) 'Define the Kn vectors'
        delTh=PI/(NTH+1)
        delPh=2*PI/NPH
c ***   Straight up first
        kTh(count)=0.0
        kPH(count)=0.0
        count=count+1
c ***   now the rest
        do i=1,NTH
           Th=i*delTH
           do j=1,NPH
              PH=j*delPH
              kth(count)= Th
              kph(count)= PH
              count=count+1
              print *, TH*180/PI,PH*180/PI
           end do
        end do
c ***   now down
        kTh(count)=PI
        kPH(count)=0.0
C ***
        open(UNIT=20,file=FNAME,status='UNKNOWN')
        write(20,200) count
        do N=1,count
           write (20,210) kth(N),kph(N)
        end do
        write(20,*) 99999,99999,1,1
```

```
        close (20)
      stop
 200  format(I5)
 210  format(3f15.12)
      end
C*********************************************************************
C*--------------------------------------------------------------------
      Program VGFKV
C*--------------------------------------------------------------------
C*    Program to set up a kvector file for the VGF scattering code     *
C*    Author:  Origonal code  by Lines                                 *
C*********************************************************************
C *** Set the value of NMAX via an included file                    ***
      implicit none
      include 'nmax.inc'
      character*80 FNAME
C****  Variables
      real kth(KNMAX),kph(KNMAX),TH, PH,PI
      real delTh, delPh
      integer NTH,NPH,i,j,count ,N
      parameter (PI=3.141592654)
      write(*,*) 'Enter the number of angles theta (1-5)'
      read (*,*) NTH
      write(*,*) 'Enter the number of angles phi'
      read (*,*) NPH
      write(*,*) ' Enter the file name '
      read(*,'(1a80)') FNAME
        count=1
        Write (*,*) 'Define the Kn vectors'
        delTh=PI/(NTH+1)
        delPh=2*PI/NPH
c ***   Straight up first
        kTh(count)=0.0
        kPH(count)=0.0
        count=count+1
c ***   now the rest
        do i=1,NTH
           Th=i*delTH
           do j=1,NPH
              PH=j*delPH
              kth(count)= Th
              kph(count)= PH
              count=count+1
              print *, TH*180/PI,PH*180/PI
           end do
        end do
c ***   now down
        kTh(count)=PI
        kPH(count)=0.0
C ***
        open(UNIT=20,file=FNAME,status='UNKNOWN')
```

```
        write(20,200) count
        do N=1,count
           write (20,210) kth(N),kph(N)
        end do
        write(20,*) 99999,99999,1,1
        close (20)
      stop
 200  format(I5)
 210  format(3f15.12)
      end
C**********************************************************************
```

```
C***********************************************************************
      Program VGFMC
C***********************************************************************
C**** Program to use the variational formalizm to solve the scattering *
C****   problem for a particle of arbitrary shape.  The particle is    *
C****   divided into an array of dipoles on a cubic lattice (by        *
C****   program VGFIN).  The scattering is computed through a plane    *
C****   wave expansion of the field inside the particle.  From this    *
C****   the external field and phase function are calculated (in       *
C****   program VGFPHZ)                                                *
C****                                                                  *
C**** Units:                                                           *
C****   All equations are in Gausian units.                            *
C****   Lengths are all relative, that is, if you input a wavelength   *
C****   of 10 um then all other lengths must be in um. You may use     *
C****   m or cm or furlongs if you wish as long as all lengths are     *
C****   in the same units.                                             *
C****                                                                  *
C**** Time dependence:                                                 *
C****   The VGFMC code uses the time dependence exp(-iwt).             *
C****                                                                  *
C**** Geometry:                                                        *
C****   The particle is aligned with its symmetry axis along           *
C****   the Z-axis.  The incident plane wave may be rotated            *
C****   around the particle by the angles alpha, beta, and gamma       *
C****   (corresponding to the Euler angles as given in Arfkin,         *
C****   Mathematical Methods for Physicists, 3nd Ed., Academic Press,  *
C****   NY, 1970). All calculations are done in this 'body frame'      *
C****   The polarization is given relative to the x-axis in the lab or *
C****   global frame. In this frame the incident plane wave travels    *
C****   in the z-hat direction and the polarization is in the x-y      *
C****   plane. The direction of the electric field is measured by an   *
C****   angle psi from the x-axis.                                     *
C****                                                                  *
C**** Author: Original code by Lines                                   *
C****                                                                  *
C****  Last Modification by Lines 21 APR 97                            *
C****                                                                  *
C**** List of subroutines and their signatures                         *
C****                                                                  *
c****  Subroutine kvectors(NTH,khatN,count)                            *
C****  complex Function GAM(d,k,EPS)                                   *
C****  complex function CPSI(R,KhatN,k,mm,N,b)                         *
C****  complex Function Wcalc(X,d,k,EPS)                               *
C****  subroutine Gcalc(R,k,Gmn,d,EPS,m,n)                             *
C****  real function delta(alpha,beta)                                 *
C****  real function dd(alpha,i,beta,k)                                *
C****  subroutine ROT(RRR, alpha, beta, gamma)                         *
C****  subroutine MV(M,V,U)                                            *
C***********************************************************************
C**** Declarations                                                  ****
```

```
      implicit none
C *** Set the value of NMAX via an included file                      ***
      include 'nmax.inc'
C**** Varables
      integer  ipvt(3*KNMAX)
      complex  z(3*KNMAX)
      real     rcond
C *** Integers
      integer  NUSE, I, J, COUNT,mcount, kcount
      integer  N,M,l,a,b,NK,np,mp,NTH,NPH,iseed,ikcount,itemp
C *** Real
      real     Wave, alpha, beta, gamma, psi, RAD
      real     R(NMAX,3), ER, EI, D(NMAX),  RDK, TD
      real     k, PI, DEG, Khat(3)
      real     RRR(3,3),V(3), E0hat(3)
      real     khatN(KNMAX,3),dtemp
      real     dsum,divd
      real     kth(KNMAX),kph(KNMAX),ERR,ERRlast,ERR0
C *** Complex
      complex  aa(3*KNMAX,3*KNMAX)
      complex  bb(3*KNMAX)
      complex  mm, EPS, X, W, C, CI, temp
      complex  E0(NMAX,3),E(NMAX,3)
      complex  T1(NMAX,3,KNMAX,3)
      complex  F(NMAX,3)
      complex  H(KNMAX,3,KNMAX,3),Y(KNMAX,3),An(KNMAX,3)
      complex  PHI
C *** Character
      character*80 INFILE, OUTFILE, COMMENTS,KFILE
C**** Function types
      complex  Wcalc, CPSI, GG, CPHI
      real     dd
C**** Typed Parameters ****
      parameter (PI=3.141592654,DEG=PI/180.0,CI=(0.0,1.0))
C**** DEFINITIONS                                                    ****
      iseed=234564
C *** Read the input and output file names and No. of iterations      ***
      write(*,*)       'Enter input file name:'
      read(*,'(1a80)')  INFILE
      write(*,*)       'Enter output file name:'
      read(*,'(1a80)')  OUTFILE
      write (*,*)       OUTFILE
      write(*,*)       'Enter input kvector file name:'
      read(*,'(1a80)')  KFILE
      write(*,*)       'Enter exit criteria ERR0, and increment divisor'
      read(*,*)         ERR0,divd
C *** Read in the input file                                          ***
      write (*,*)      'Reading input file...'
      open (10,file=INFILE)
      read(10,'(1a80)') Comments
      read(10, *)       NUSE
```

```
      read(10, *)       Wave,alpha,beta,gamma,psi,RAD
C**** CALCULATIONS                                                  ****
      read(10,*)        ER, EI, TD
      EPS=cmplx(ER,EI)
      mm=sqrt(EPS)
      X=(EPS-1.0)/(4.0*PI)
C**** Input the cell positions
      do I=1,NUSE
         read(10,*) R(I,1),R(I,2),R(I,3),D(I)
         dsum=dsum+D(I)**3
      end do
      write (*,*) 'total volume = ', dsum
      close(10)
C**** Calculate the K-hat direction and K, the wavenumber in vacuum ****
      write (*,*) 'Calculating incident fields...'
      k     = 2.0*PI/Wave
      alpha = alpha*DEG
      beta  = beta*DEG
      gamma = gamma*DEG
      psi   = psi*DEG
C *** set up rotation matrix with call to Rot                       ***
      call Rot(RRR,alpha,beta,gamma)
C *** Khat is in the z direction in the lab frame                   ***
      V(1)  = 0.0
      V(2)  = 0.0                              ! V is temporary storage
      V(3)  = 1.0
      call MV(RRR,V,Khat)
C *** Thus E0hat must be in the x-y plane in the lab frame          ***
      V(1)  = cos(psi)
      V(2)  = sin(psi)
      v(3)  = 0.0
      call MV(RRR,V,E0hat)
C *** Assign incident and initial fields...                         ***
      W     = Wcalc(X)
      do m  = 1,NUSE
         RDK = 0.0
         do I = 1,3
            RDK = RDK+Khat(I)*R(m,I)
         end do
         temp=CI*k*RDK
         C=cexp(temp)
         do I=1,3
            E0(m,I)=C*E0hat(I)                    ! incident E-field
         end do
      end do
C**** Setup the khatN directions.
      Write (*,*) 'Define the Kn vectors'
      call getkv(kth,kph,NK,KFILE,ERR,ERRlast,mcount,ikcount)
      ERR=ERRlast
      print *,'last ',NK
      call kvectors3(khatN,kth,kph,NK)
```

```
C**** HERE STARTS THE MONTE CARLO LOOP                                  ***
      open(50, file = 'ERRlist', STATUS=''UNKNOWN'')
      write(50,*) INFILE
      write(50,*) 'ERROR LIST'
      do 10 while ((ERR.gt.ERR0) .and. (mcount.lt.100))
      do 15 kcount=ikcount,NK-1
C *** Internal field calculation                                        ***
      write(*,*) 'Calculating internal fields...'
      print *, 'calculating T1 '
      do a=1,NUSE
         do i=1,3
            do N =1,NK
               do j=1,3
                  T1(a,i,N,j)=(0.0,0.0)
                  do b=1,NUSE
                     dtemp=d(b)
                     T1(a,i,N,j)=T1(a,i,N,j)+
     &               (dd(a,i,b,j)-d(b)**3*W*GG(R,k,dtemp,EPS,a,i,b,j))
     &               *CPSI(R,KhatN,k,mm,N,b)
                  end do
               end do
            end do
         end do
      end do
 22   format(3F17.10)
C
      write (*,*) 'calculate H and Y'
      do 1 M=1,NK
      do 2 l=1,3
          Y(M,l)=(0.0,0.0)
          do a=1,NUSE
             do i=1,3
               Y(M,l)=Y(M,l)+conjg(T1(a,i,M,l))*E0(a,i)
             end do
          end do
          do 3 N=1,NK
          do 4 j=1,3
          H(M,l,N,j)=(0.0,0.0)
             do a=1,NUSE
                do i=1,3
                  H(M,l,N,j)=H(M,l,N,j)+conjg(T1(a,i,M,l))*T1(a,i,N,j)
                end do
             end do
 4        continue !end do
 3        continue !end do
 2    continue !end do
 1    continue !end do
C *** Now do the matrix invertion and solve the system of equations
      write(*,*) 'Now solve the set of equations'
C *** reform H into a N*3 by N*3 complex matrix *****
      do n=1,NK
```

```
         do i=1,3
            m =3*(n -1)+i
            bb(m)=Y(n,i)
            do np=1,NK
               do j=1,3
                  mp=3*(np-1)+j
                  aa(m,mp)=H(n,i,np,j)
                  if (aa(m,mp).eq.(0.0,0.0)) then
                      print *, m,mp,aa(m,mp)
                      print *, n,np,i,j,H(n,np,i,j)
                  end if
               end do
            end do
         end do
      end do
C ***
      do i=1,3*KNMAX
         ipvt(i)=0
         z(i)=(0.0,0.0)
      end do
      rcond=0.0
      call cgeco(aa,3*KNMAX,3*NK,ipvt,rcond,z)
      call cgesl(aa,3*KNMAX,3*NK,ipvt,bb,0)
      do N=1,NK
          do i=1,3
             M=3*(N-1)+i
             An(N,i)=bb(M)
          end do
      end do
C *** Here we end the Monte Carlo Loop by testing the error         ****
      PHI=CPHI(An,H,Y,E0,NK,NUSE)
      ERR = PHI*conjg(PHI)
      print *, ' test ERR ', ERR,' > ',ERR0
      write (50,*) mcount, ERR
      if (ERR.gt.ERR0) then
      print *, ' test ERR ', ERR,' > ',ERRlast
          if (ERR.gt.ERRlast) then
             print *,'reject'
             call getkv(kth,kph,NK,KFILE,ERR,ERRlast,itemp,itemp)
             ERRlast=ERR
             call kvectors3(khatN,kth,kph,NK)
          else
             print *, 'accept'
             call Ecalc(NUSE,R,E,F,An,khatN,X,NK,K,mm)
             call printout(COMMENTS,ERR,ERR0,NUSE,wave,alpha,beta,gamma,
     &               psi, RAD, EPS,R,D,E,NK,An,mcount,OUTFILE,khatN,divd)
             call printkv(kth,kph,NK,KFILE,ERR,ERRlast,mcount,kcount)
             ERRlast=ERR
          end if
C
          if (kcount.lt.NK-1) then
```

```
            call move1kv(kcount+1,khatN,kth,kph,NK,iseed,divd)
         end if
      else
         call printkv(kth,kph,NK,KFILE,ERR,ERRlast,mcount,kcount)
         call Ecalc(NUSE,R,E,F,An,khatN,X,NK,K,mm)
         call printout(COMMENTS,ERR,ERR0,NUSE,wave,alpha,beta,gamma,
     &              psi, RAD, EPS,R,D,E,NK,An,mcount,OUTFILE,khatN,divd)
         goto 10                                              ! Drop out
      end if
      print *, ERR,ERR0, ERRlast ,mcount,kcount
  15  continue                                             ! kcount loop
      mcount=mcount+1
  10  continue
C**** DONE WITH THE MONTE CARLO LOOP                                  ****
      close(50)
      print *, ERR, ERR0, mcount
      stop
C
      end
C
C****** FUNCTIONS AND SUBROUTINES ************************************
C
C*********************************************************************
C*-------------------------------------------------------------------
      subroutine Ecalc(NUSE,R,E,F,An,khatN,X,NK,K,mm)
C*-------------------------------------------------------------------
C*********************************************************************
C     subroutine to print out a copy of the kvectkors to the file      *
C       KFILE                                                          *
C*********************************************************************
C *** Set the value of NMAX via an included file                     ***
       implicit none
       include 'nmax.inc'
C****  Variables
       integer NUSE, b,j,m,i,N,NK
       complex  F(NMAX,3),E(NMAX,3),X,CPSI
       complex  An(KNMAX,3),mm
       real khatN(KNMAX,3),PI,R(NMAX,3),K
       parameter (PI=3.141592654)
C****  Now form the f's                                             ****
       do b=1,NUSE
          do j=1,3
             F(b,j)=(0.0,0.0)
             do N=1,NK
                F(b,j)=F(b,j)+An(N,j)*CPSI(R,KhatN,k,mm,N,b)
             end do
          end do
       end do
C****  Done, Calculate the internal E-field to output it.           ****
       do m = 1,NUSE
          do I = 1,3
```

```
            E(m,I)=F(m,I)/(1.+(4.*PI)*X/3.)
          end do
        end do
       return
       end
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine printout(COMMENTS,ERR,ERR0,NUSE,wave,alpha,beta,gamma,
     & psi, RAD, EPS,R,D,E,NK,An,mcount,OUTFILE,khatN,divd)
C*--------------------------------------------------------------------
C*********************************************************************
C     subroutine to print out a copy of the kvectkors to the file      *
C       KFILE                                                          *
C*********************************************************************
C *** Set the value of NMAX via an included file                     ***
       implicit none
       include 'nmax.inc'
C****  Variables
C ***  Integers
       integer  NUSE, J, mcount
       integer NK,m,n
C ***  Real
       real     Wave, alpha, beta, gamma, psi, RAD
       real     R(NMAX,3),  D(NMAX),khatN(KNMAX,3)
       real     ERR,ERR0,divd
C ***  Complex
       complex  EPS
       complex  E(NMAX,3)
       complex  An(KNMAX,3)
C ***  Character
       character*80 OUTFILE, COMMENTS
C****  And output to file                                           ****
       open(10, file = OUTFILE, STATUS=''UNKNOWN'')
       write(10, *) COMMENTS
       write(10, *) ERR,ERR0
       write(10, *) NUSE
       write(10, *) wave, alpha, beta, gamma, psi ,RAD
       write(10, *) real(EPS), aimag(EPS)
       do m = 1, NUSE
          write(10, 100) R(m, 1), R(m, 2), R(m, 3), D(m)**3
       end do
       do m = 1, NUSE
          write(10, 110) (real(E(m,J)), aimag(E(m,J)), J=1,3)
       end do
       do N=1,NK
          write(10,105) N, (real(An(N,J)),aimag(An(N,J)), J=1,3)
     &     ,real(An(N,1)*conjg(An(N,1))+An(N,2)*conjg(An(N,2))
     &    +An(N,3)*conjg(An(N,3)))
       end do
       do N=1,NK
          write (10,115) khatN(N,1),khatN(N,2),khatN(N,3)
```

```
       end do
       write(10,*) 'ERR = ',ERR, 'mcount =', mcount,' divd ',divd
       close(10)
      return
 100  format(4(1x, 1g18.8))
 105  format(I5,7F8.3)
 110  format(6(1x, 1g18.8))
 115  format(3f17.9)
      end
C*********************************************************************
C*--------------------------------------------------------------------
      complex function CPHI(An,H,Y,E0,NK,NUSE)
C*--------------------------------------------------------------------
C*********************************************************************
C *** Set the value of NMAX via an included file                   ***
       implicit none
       include 'nmax.inc'
C****  Variables
       complex  E0(NMAX,3)
       complex  H(KNMAX,3,KNMAX,3),Y(KNMAX,3),An(KNMAX,3)
       integer N,j,M,l,a,i,NK,NUSE
C
       CPHI=(0.0,0.0)
       do 5 N=1,NK
       do 6 j=1,3
       do 7 M=1,NK
       do 8 l=1,3
          CPHI = CPHI+conjg(An(M,l))*H(M,l,N,j)*An(N,j)
  8    continue
  7    continue
  6    continue
  5    continue
       do j=1,3
          do N=1,NK
             CPHI=CPHI-(conjg(Y(N,j))*An(N,j)+Y(N,j)*conjg(An(N,j)))
          end do
       end do
       do a=1,NUSE
          do i=1,3
             CPHI=CPHI+E0(a,i)*conjg(E0(a,i))
          end do
       end do
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine printkv(kth,kph,NK,KFILE,ERR,ERRlast,mcount,kcount)
C*--------------------------------------------------------------------
C*********************************************************************
C     subroutine to print out a copy of the kvectkors to the file      *
C       KFILE                                                          *
```

```
C*********************************************************************
C *** Set the value of NMAX via an included file                    ***
       implicit none
       include 'nmax.inc'
C****  Variables
       real kth(KNMAX),kph(KNMAX),ERR,ERRlast
       integer NK, N ,mcount,kcount
       character*80 KFILE
C
       open(UNIT=45,FILE=KFILE,status='UNKNOWN')
         write(45,200) NK
         do N=1,NK
            write(45,210) kth(N),kph(N)
         end do
       write(45,*)  ERR, ERRlast,mcount, kcount
       close(45)
       return
 200   format(I5)
 210   format(3f15.12)
      end
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine getkv(kth,kph,NK,KFILE,ERR,ERRlast,mcount,kcount)
C*--------------------------------------------------------------------
C*********************************************************************
C     subroutine to read in the kvectkors from the file KFILE          *
C*********************************************************************
C *** Set the value of NMAX via an included file                    ***
       implicit none
       include 'nmax.inc'
C****  Variables
       real kth(KNMAX),kph(KNMAX),ERR,ERRlast
       integer NK, N,mcount,kcount
       character*80 KFILE
C
       open(UNIT=45,FILE=KFILE,status='UNKNOWN')
         read(45,200) NK
         do N=1,NK
            read(45,210) kth(N),kph(N)
         end do
       read(45,*) ERR, ERRlast, mcount,kcount
       close(45)
       return
 200   format(I5)
 210   format(3f15.7)
      end
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine move1kv(mcount,khatN,kth,kph,NK,iseed,divd)
C*--------------------------------------------------------------------
C*********************************************************************
```

```
C *** Set the value of NMAX via an included file                      ***
       implicit none
       include 'nmax.inc'
C****  Variables
       real khatN(KNMAX,3), kth(KNMAX), kph(KNMAX),PI,PI2,rannum
       real THinc, PHinc, divd
       real temp1,temp2
       integer N,iseed ,NK, mcount
       parameter (PI=3.141592654)
       parameter (PI2=6.28318530718)
       THinc=PI/divd
       PHinc=2*THinc
       temp1=THinc*rannum(iseed)
       temp2=PHinc*rannum(iseed)
       kth(mcount)=kth(mcount)+temp1
       kph(mcount)=kph(mcount)+temp2
       if (kth(mcount).gt.PI) then
          kth(mcount)=2*PI-kth(mcount)
       end if
       if (kph(mcount).gt.PI2) then
          kph(mcount)=kph(mcount)-PI2
       end if
       if(kth(mcount).lt.0.0) then
          kth(mcount)=-kth(mcount)
       end if
       if(kph(mcount).lt.0.0) then
          kph(mcount)=-kph(mcount)
       end if
       call kvectors4(mcount,khatN,kth,kph,NK)
       return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      real function rannum(iseed)
C*--------------------------------------------------------------------
C*********************************************************************
C *** Set the value of NMAX via an included file                      ***
       implicit none
       include 'nmax.inc'
       real ran1
C****  Variables
       integer iseed
       rannum=2.0*ran1(iseed)-1.0
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine kvectors4(N,khatN,kth,kph,NK)
C*--------------------------------------------------------------------
C*    Subroutine to set up the  khatN directions.                     *
C*********************************************************************
```

```
C *** Set the value of NMAX via an included file                      ***
       implicit none
       include 'nmax.inc'
C****  Variables
       real khatN(KNMAX,3), kth(KNMAX), kph(KNMAX)
       integer N ,NK
C
       khatN(N,1)=sin(kth(N))*cos(kph(N))
       khatN(N,2)=sin(kth(N))*sin(kph(N))
       khatN(N,3)=cos(kth(N))
C
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine kvectors3(khatN,kth,kph,NK)
C*--------------------------------------------------------------------
C*    Subroutine to set up the  khatN directions.                     *
C*********************************************************************
C *** Set the value of NMAX via an included file                     ***
       implicit none
       include 'nmax.inc'
C****  Variables
       real khatN(KNMAX,3), kth(KNMAX), kph(KNMAX)
       integer N, NK
         do N=1,NK
            khatN(N,1)=sin(kth(N))*cos(kph(N))
            khatN(N,2)=sin(kth(N))*sin(kph(N))
            khatN(N,3)=cos(kth(N))
         end do
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      complex Function GAM(d,k,EPS)
C*--------------------------------------------------------------------
C*********************************************************************
C*    Function to calculate the self term contribution termed GAMMA    *
C*      in the text. The form for GAM  is taken from Goedecke and      *
C*      O'Brian. Note the sign change due the time dependance          *
C*      exp(-iwt). This differs from Goedecke and O'Brien's choice.    *
C*       GAM=(3./(4.*PI))**(2./3.)*(kd)**2 + CI*kd**3/(2.*PI)          *
C*********************************************************************
C**** Variables                                                      ****
      implicit none
       complex EPS,CI
       real k,d,kd
       real PI
       parameter (PI=3.141592654,CI=(0.0,1.0))
       real b1
       kd=k*d
```

```
       b1=(3./(4.*PI))**(2./3.)
       GAM=b1*(kd)**2 + CI*kd**3/(2.*PI)
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      complex function CPSI(R,KhatN,k,mm,N,b)
C*--------------------------------------------------------------------
C*********************************************************************
C*    Function to calculate the trial function expansion functions    *
C*      CPSI=cexp(i*k*khatN.R(b))                                     *
C*********************************************************************
C *** Set the value of NMAX via an included file                    ***
       implicit none
       include 'nmax.inc'
C****  Variables                                                   ****
       real KDR,KhatN(KNMAX,3),R(NMAX,3),k,PI
       complex temp,CI,mm
       integer i,b,N
       parameter (PI=3.141592654,CI=(0.0,1.0))
C
       KDR=0.0
       do i = 1,3
          KDR=KDR+KhatN(N,i)*R(b,i)
       end do
       temp=CI*mm*k*KDR
       CPSI=cexp(temp)
       return
       end
C*********************************************************************
C*--------------------------------------------------------------------
      complex Function Wcalc(X)
C*--------------------------------------------------------------------
C*********************************************************************
C*    Function to calculate the W-factor                              *
C*      W(nu)=X/(1+(4*pi/3)*X)                                        *
C*********************************************************************
C****  Variables                                                   ****
       implicit none
       complex X,CI
       real PI
       parameter (PI=3.141592654,CI=(0.0,1.0))
C
       Wcalc=X/(1.+(4.*PI/3.)*X)
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      complex function GG(R,k,d,EPS,a,i,b,j)
C*--------------------------------------------------------------------
C*********************************************************************
```

```
C*    Function to calculate the dyadic Green's function for a dipole   *
C**********************************************************************
C *** Set the value of NMAX via an included file                      ***
       implicit none
       include 'nmax.inc'
C****  Variables                                                      ****
       complex PHZ,t1,t2,temp,CI,GAM,EPS
       real RMAG,Rhat(3),R(NMAX,3),Rab(3),k,K2,d,PI
       real delta
       integer a,b,i,j
       Parameter(PI=3.141592654,CI=(0.0,1.0))
C
       K2=k*k
       if(b.ne.a) then
C         calculate separation distance Rmn=Rn-Rm and RMAG=|Rmn|
          Rab(1)=R(a,1)-R(b,1)
          Rab(2)=R(a,2)-R(b,2)
          Rab(3)=R(a,3)-R(b,3)
          RMAG=Rab(1)**2+Rab(2)**2+Rab(3)**2
          RMAG=RMAG**0.5
C         Make a unit vector in the Rmn direction                    ***
          Rhat(1)=Rab(1)/RMAG
          Rhat(2)=Rab(2)/RMAG
          Rhat(3)=Rab(3)/RMAG
C
          temp=CI*k*RMAG
          PHZ=cexp(temp)
C
          t1=(K2/RMAG)*(delta(i,j)-Rhat(i)*Rhat(j))
          t2=(ci*k/RMAG**2-1.0/RMAG**3)*(delta(i,j)-3.*Rhat(i)*Rhat(j))
          GG=PHZ*(t1+t2)
        else
          GG=4.*PI*GAM(d,k,EPS)/(3.0*d**3)
       end if
       return
      end
C**********************************************************************
C*---------------------------------------------------------------------
      real function delta(alpha,beta)
C*---------------------------------------------------------------------
C**********************************************************************
C     This is Kronecker delta                                         *
C**********************************************************************
C****  Variables
       implicit none
       integer alpha, beta
       delta=0.0
       if (alpha.eq.beta) delta=1.0
      return
      end
```

```
C*********************************************************************
C*--------------------------------------------------------------------
      real function dd(alpha,i,beta, j)
C*--------------------------------------------------------------------
C*********************************************************************
C     This is really two delta functions,                            *
C        delta(alpha,beta) * delta(i,j)                              *
C*********************************************************************
C****  Variables
       implicit none
       integer alpha,i, beta,j
       dd=0.0
       if ((alpha.eq.beta).and.(i.eq.j)) dd=1.0
      return
      end
C*********************************************************************
```

```
************************************************************************
*  Include file for maximum array size parameter.
************************************************************************
      integer NMAX
      parameter (NMAX = 3000)
      integer KNMAX
      parameter (KNMAX = 200)
************************************************************************
```

```
C*********************************************************************
      program VGFPHZ
C*********************************************************************
C**** Program to calculate the differential scattering cross section  *
C****    for an arbitrary particle using the electric fields generated *
C****    by the VGF program.  First the scattering amplitude is       *
C****    calculated as a function of the scattering angle theta for a *
C****    particular value of the azmuthal angle phi.  Theta and phi   *
C****    are lab or global frame coordinates.  The VGF code performs  *
C****    its calculations in the particles body frame coordinates.    *
C****    Numerical calculation is more efficient if theta and phi are *
C****    expressed in the body frame cordinates as well. Thus, the    *
C****    subroutine Rot from VGFMC is employed to do the              *
C****    transformations. Like VGFMC the assumed time dependence is   *
C****    given by the expression exp(ikr-iwt).  The program takes as  *
C****    its input the output files from VGFMC containing the         *
C****    electric fields.                                             *
C****                                                                 *
C**** Geometry:                                                       *
C****   The particle is aligned with its principal axis along         *
C****   the Z-axis (for particles with symmetry this is usually the   *
C****   symmetry axis).  The incident plane wave may be rotated in the *
C****   field calculation (performed by the program VGFMC)            *
C****   around the particle by the angles alpha, beta, and gamma      *
C****   (corresponding to the Euler angles as given in Arfkin,        *
C****   Mathematical Methods for Physicists, 3nd Ed., Academic Press, *
C****   NY, 1970). These angles are passed on to VGFPHZ for use in its *
C****   calculations. All calculations are done in the 'body frame.'  *
C****   The polarization is given relative to the x-axis in the lab or *
C****   global frame. In this frame the incident plane wave travels   *
C****   in the z-hat direction and the polarization is in the x-y     *
C****   plane. The direction of the electric field is measured by an  *
C****   angle psi from the x-axis.                                    *
C****                                                                 *
C**** Author: Original code by Lines                                  *
C****                                                                 *
C**** extensive modification by Lines 28 JAN 97                       *
C****  Last Modification by Lines 28 JAN 97 to match theory           *
C*********************************************************************
C**** Variable diffinitions                                        ****
      implicit none
      include 'nmax.inc'
C *** loop counters
      integer IANG,mu
      integer I, J, NDATA, NUSE ,NTH,NPH
C *** Reals
      real*8 PI, K, K2, K3, DVOL(NMAX), R(NMAX, 3)
      real*8 DEG, alpha, beta, gamma, psi, RDOT, Rhat(3), RAD, W
      real*8 TH, PH, RRR(3,3), V(3),THhat(3),PHhat(3)
      real*8 ER, EI, EXR, EXI, EYR, EYI, EZR, EZI, PA2, HOR, VER
      real*8 dsum,ERR,ERR0
```

```
C *** complex quantities
      complex E(NMAX, 3), EPS, X, CI, C,temp
      complex FH, FV ,TDE, PDE
C *** File names and comments
      character*80 INFILE, OUTFILE, COMMENTS
C *** Parameters
      parameter (PI = 3.141592654, DEG = PI/180.0, CI = (0.0, 1.0))
      parameter (NDATA = 181)
C *** Read input filename from standard in...                          ***
      write(6, *) 'Enter input file name:'
      read(5, '(1a80)') INFILE
      write(6, *) 'Enter output file name:'
      read(5, '(1a80)') OUTFILE
C *** Read input file data... First alert user                         ***
      write(6, *) 'Reading input file...'
C *** read in file name, comments, wavelength, angles, radius          ***
      open(10, file = INFILE)
      read(10, '(1a80)') COMMENTS
      read(10, *) ERR,ERR0
      read(10, *) NUSE
      print *,NUSE
      read(10, *) W, alpha, beta, gamma, psi, RAD
      print *,W, alpha, beta, gamma, psi, RAD
      print *, 'input cell positions'
      print *,NUSE
C *** read in cell positions and adjusted volume
C *** read in epsilon and compute chi (X)                              ***
C
      read(10, *) ER, EI
      EPS = cmplx(ER, EI)
      X = (EPS - 1.0)/(4.0*PI)
C
      dsum=0.0
      do I = 1, NUSE
         read(10, *) R(I, 1), R(I, 2), R(I, 3), DVOL(I)
         dsum=dsum+DVOL(I)
      end do
      write(*,*) 'total vol =', dsum
      print *, 'input eps',NUSE
      print *, 'input fields',NUSE
C *** read in the field at each cell, three complex components         ***
C
      do mu = 1, NUSE
         read(10, *) EXR, EXI, EYR, EYI, EZR, EZI
         E(mu, 1) = cmplx(EXR, EXI)
         E(mu, 2) = cmplx(EYR, EYI)
         E(mu, 3) = cmplx(EZR, EZI)
      end do
      close(10)
C *** Calculate rotation matrix                                       ***
C
```

```
      call ROT(RRR, alpha,beta,gamma)
C *** Calculate the normalized differential cross section           ***
C
      write(6, *) 'Calculating phase function...'
      K = 2.0*PI/W
      K2=K*K
      K3=K2*K
C *** normalization constant, this seems to be the standard where   ***
C *      RAD is the symmetry semi axis.                               *
C
      PA2 = 1/(PI*RAD*RAD)
C *** open  the output file                                         ***
      open(10, file = OUTFILE, status = 'UNKNOWN')
C
C *** Do the main calculation of the scattering amplitude           ***
      PH=0.0 ! for now
C *** Loop over angle (rad).                                        ***
      do IANG = 0, NDATA-1
         TH = PI*IANG/(NDATA - 1.0)
C ***    define THhat and PHhat, and rotate them into prime frame  ***
         V(1)= cos(TH)*cos(PH)
         V(2)= cos(TH)*sin(PH)
         V(3)=-sin(TH)
         call MV(RRR,V,THhat)
         V(1)=-sin(PH)
         V(2)= cos(PH)
         V(3)= 0
         call MV(RRR,V,PHhat)
C ***    THhat and PHhat are now really primed,but drop prime       ***
C ***    calculate r-hat prime direction for the given theta        ***
         V(1) = sin(TH)*cos(PH)
         V(2) = sin(TH)*sin(PH)
         V(3) = cos(TH)
         call MV(RRR,V,Rhat)
C ***    loop over cell number mu                                   ***
         fh=(0.0,0.0)
         fv=(0.0,0.0)
         do mu = 1, NUSE
            RDOT = 0.0
C ***       calculate rhatprime dot rprime for the exponential      ***
            do j = 1, 3
               RDOT = RDOT + R(mu, j)*Rhat(j)
            end do
C ***       calculate all factors that don't depend on i and j      ***
            temp = -CI*K*RDOT
            C = CI*K3*X*DVOL(mu)*cexp(temp)
C ***       now put the scattering amplitude together               ***
            TDE=(0.0,0.0)
            PDE=(0.0,0.0)
C ***       calculate THhat dot E and PHhat dot E
            do I = 1, 3
```

```
               TDE=TDE+THhat(I)*E(mu,I)
               PDE=PDE+PHhat(i)*E(mu,I)
            end do
C ***       now put the scattering amplitude together                 ***
            fh=fh+C*TDE
            fv=fv+C*PDE
         end do
C
C ***    separate into _H_orizontal and _V_ertical components and    ***
C *        at the same time convert to diff. scat. cross section (/K2) *
         HOR = (Fh*conjg(Fh))/K2
         VER = (Fv*conjg(Fv))/K2
C
C ***    normalize and and output to a file                          ***
         write(10, 100) TH/DEG, PH/DEG, VER*PA2, HOR*PA2
100      format(1x, 4(1e12.4, 2x))
C
      end do
      close(10)
      stop
      end
C*********************************************************************
C*--------------------------------------------------------------------
-*
      subroutine ROT(RRR, alpha, beta, gamma)
C*--------------------------------------------------------------------
-*
C*********************************************************************
C*    subroutine to return the Euler rotation matrix where the three   *
C*      rotation angles are alpha, a rotation about the z-axis, beta,  *
C*      a rotation about the new y-axis, and gamma, a rotation about   *
C*      the new z-axis.                                                *
C*********************************************************************
C**** Veriables                                                    ****
      real*8 RRR(3,3),alpha, beta, gamma
C
      RRR(1,1)= cos(alpha)*cos(beta)*cos(gamma)-sin(alpha)*sin(gamma)
      RRR(1,2)= sin(alpha)*cos(beta)*cos(gamma)+cos(alpha)*sin(gamma)
      RRR(1,3)=-sin(beta)*cos(gamma)
C
      RRR(2,1)=-cos(alpha)*cos(beta)*sin(gamma)-sin(alpha)*cos(gamma)
      RRR(2,2)=-sin(alpha)*cos(beta)*sin(gamma)+cos(alpha)*cos(gamma)
      RRR(2,3)= sin(beta)*sin(gamma)
C
      RRR(3,1)= cos(alpha)*sin(beta)
      RRR(3,2)= sin(alpha)*sin(beta)
      RRR(3,3)= cos(beta)
C
      return
      end
C*********************************************************************
```

```
C*--------------------------------------------------------------------
-*
      subroutine MV(M,V,U)
C*--------------------------------------------------------------------
-*
C**********************************************************************
C*    subroutine to multiply a 3x3 matrix by a vector with three       *
C*      components                                                     *
C**********************************************************************
C**** Veriables                                                    ****
      real*8 M(3,3), V(3), U(3), temp
      integer i,j
      do i=1,3
         temp=0.0
         do j=1,3
            temp=temp+M(i,j)*V(j)
         end do
         U(i)=temp
      end do
      return
      end
C**********************************************************************
```

```
C***********************************************************************
      program VGFSIG
C***********************************************************************
C**** Program to calculate total cross sections of an arbitrary        *
C****    particle.  The scattering and absorption cross sections are   *
C****    calculated  directly and summed to find the extinction cross  *
C****    section.  The extinction cross section is also calculated     *
C****    using the Optical Theorem, although do to computational error *
C****    the Optical Theorem result may be very wrong!  The program    *
C****    uses the VGF  output files for its input files. Like VGF the  *
C****    assumed time dependence is given by the expression            *
C****    exp(ikr-iwt). Each cross section involves an integration over *
C****    all angles.  Thus the integration can be done in the body     *
C****    frame just as well as the global frame.                       *
C****                                                                  *
C**** Geometry:                                                        *
C****   The particle is aligned with its symmetry axis along           *
C****   the Z-axis.  The incident plane wave may be rotated            *
C****   around the particle by the angles alpha, beta, and gamma       *
C****   (corresponding to the Euler angles as given in Arfkin,         *
C****   Mathematical Methods for Physicists, 3nd Ed., Academic Press,  *
C****   NY, 1970). All calculations are done in this 'body frame'      *
C****   The polarization is given relative to the x-axis in the lab or *
C****   global frame. In this frame the incident plane wave traveles   *
C****   in the z-hat direction and the polarization is in the x-y      *
C****   plane. The direction of the electric field is measured by an   *
C****   angle psi from the x-axis.                                     *
C****                                                                  *
C**** Author: R. Todd Lines                                            *
C****                                                                  *
C***********************************************************************
C**** Variable diffinitions                                        ****
      implicit none
      include 'nmax.inc'
C *** loop counters
      integer mu
      integer I, NDATA, NUSE
C *** Reals
      real PI, K, K2, K3, DVOL(NMAX), R(NMAX, 3)
      real DEG, alpha, beta, gamma, psi, RAD, W
      real ER, EI, EXR, EXI, EYR, EYI, EZR, EZI, PA2
      real SIGe, SIGa, SIGs, FPIK
      real dsum
      real RRR(3,3),ERR,ERR0
C     complex quantities
      complex E(NMAX, 3), EPS, X, CI
C *** File names and comments
      character*80 INFILE, OUTFILE, COMMENTS
C *** Parameters
      parameter (PI = 3.141592654, DEG = PI/180.0, CI = (0.0, 1.0))
      parameter (NDATA = 181)
```

```
C *** Read input filename from standard in...                          ***
      write(6, *) 'Enter input file name:'
      read(5, '(1a80)') INFILE
      write(6, *) 'Enter output file name:'
      read(5, '(1a80)') OUTFILE
C *** Read input file data... First allert user                        ***
      write(6, *) 'Reading input file...'
C *** read in file name, comments, wavelength, angles, radius         ***
      open(10, file = INFILE)
      read(10, '(1a80)') COMMENTS
      print *, comments
      read(10, *) ERR,ERR0
      print *, ERR,ERR0
      read(10, *) NUSE
      print *,NUSE
      read(10, *) W, alpha, beta, gamma, psi, RAD
      print *,W, alpha, beta, gamma, psi, RAD
      print *, 'input cell positions'
C *** read in cell positions and adjusted volume
C *** read in epsilon and compute chi (X)                              ***
      read(10, *) ER, EI
      print *, ER, EI
      EPS = cmplx(ER, EI)
      X = (EPS - 1.0)/(4.0*PI)

      dsum=0.0
      do I = 1, NUSE
         read(10, *) R(I, 1), R(I, 2), R(I, 3), DVOL(I)
c         print *, I, R(I, 1), R(I, 2), R(I, 3), DVOL(I)
         dsum=dsum+DVOL(I)
      end do
      write(*,*) 'total vol =', dsum
      print *, 'input eps',NUSE

      print *, 'input fields',NUSE
C *** read in the field at each cell, three complex components         ***
      do mu = 1, NUSE
         read(10, *) EXR, EXI, EYR, EYI, EZR, EZI
         E(mu, 1) = cmplx(EXR, EXI)
         E(mu, 2) = cmplx(EYR, EYI)
         E(mu, 3) = cmplx(EZR, EZI)
      end do
      close(10)
C *** Calculate rotation matrix                                       ***
      call ROT(RRR, alpha,beta,gamma)
      close(10)
C *** Begin Calculation                                               ***
      write(6, *) 'Calculating ...'
      K = 2.0*PI/W
      K2=K*K
      K3=K2*K
```

```
      FPIK=4*PI*K
C**** Calculate the absorption corss section                          ****
      call sigAbs(SIGa, E, X, DVOL, FPIK, NUSE)
C**** Calculate the extinction cross section using the optical therem **
      call sigExOT(SIGe,K3,K,R,DVOL,X,E,alpha,beta,gamma,psi,NUSE)
C**** Calculate the scattering cross section                          ****
      call simpson (SIGs,R,k,X,DVOL,E,NUSE,RRR)
C *** normalization constant, this seems to be the standard where     ***
C ***   RAD is the symmetry semi axis.                                 *
      PA2 = 1/(PI*RAD*RAD)
C**** Output the efficiency sig/(pi*a**2) to a file                   ****
C *** open  the output file                                           ***
      open(10, file = OUTFILE, status = 'UNKNOWN')
      write(*,*)
      write(*,*) 'size parameter:    ',K*RAD
      write(*,*) 'Qext (opt. ther):  ',SIGe*PA2
      write(*,*) 'Qext (Qscat+Qabs): ',(SIGs+SIGa)*PA2
      write(*,*) 'Qscat (calcuated): ',SIGs*PA2
      write(*,*) 'Qabs (calculated): ',SIGa*PA2
C
      write(10,*) 'size parameter:    ',K*RAD
      write(10,*) 'Qext (opt. ther):  ',SIGe*PA2
      write(10,*) 'Qext (Qscat+Qabs): ',(SIGs+SIGa)*PA2
      write(10,*) 'Qscat (calcuated): ',SIGs*PA2
      write(10,*) 'Qabs (calculated): ',SIGa*PA2
      close(10)
 100  format (5f6.4)
      stop
      end

C*********************************************************************
C*--------------------------------------------------------------------
      subroutine sigAbs(SIGa, E, X, DVOL, FPIK, NUSE)
C*--------------------------------------------------------------------
C*********************************************************************
C*    subroutine to calculate the absorption cross section.           *
C*********************************************************************
C**** Variables                                                       ****
      include 'nmax.inc'
      integer mu, i, NUSE
      real SIGa, EM, DVOL(NMAX), FPIK
      complex E(NMAX,3), X
C *** Calculate the absorption corss section                           ***
       SIGa=0.0
       do mu=1,NUSE
          EM=0.0
          do i=1,3
             EM=EM+E(mu,i)*conjg(E(mu,i))
          end do
          SIGa=SIGa+aimag(X)*DVOL(mu)*EM
       end do
```

```
       SIGa=FPIK*SIGa
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine sigExOT(SIGe,K3,K,R,DVOL,X,E,alpha,beta,gamma,psi,NUSE)
C*--------------------------------------------------------------------
C*********************************************************************
C*    Subroutine to calculate the extintion cross section using the    *
C*       Optical theorem.                                              *
C*********************************************************************
C**** Variables                                                      ****
      include 'nmax.inc'
      integer mu,i,NUSE
      real SIGe,K,K3,R(NMAX,3),DVOL(NMAX)
      real PI,Rhat(3),RRR(3,3),V(3),E0hat(3)
      real alpha,beta,gamma
      complex F(3), temp, CI, X,E(NMAX,3),C
      complex FE(3)
      parameter (PI = 3.141592654, CI = (0.0, 1.0))
C *** Calculate the extinction cross section using the optical therem **
C *    First find the scattering amplitude in the forward direction    *
       call ROT(RRR,alpha,beta,gamma)
C *** Thus E0hat must be in the x-y plane in the lab frame           ***
      V(1)=cos(psi)
      V(2)=sin(psi)
      v(3)=0.0
      call MV(RRR,V,E0hat)
C *** calculate r-hat direction for the given the incident direction ***
       V(1) = 0
       V(2) = 0
       V(3) = 1
       call MV(RRR,V,Rhat)
C ***  Initialize F                                                 ***
       do I = 1, 3
          F(I) = cmplx(0.0, 0.0)
       end do
C ***  loop over cell number mu                                     ***
       do mu = 1, NUSE
C ***     calculate all factors that don't depend on i              ***
          temp = -CI*K*R(mu,3)
          C = CI*K3*X*cexp(temp)*DVOL(mu)
C ***     calculate the factor that depensd on Emu,i                ***
          do I=1,2
             FE(I)=E(mu,I)
          end do
          FE(3)=cmplx(0.0,0.0)
C ***     now put the scattering amplitude together                 ***
          do I = 1, 3
             F(I) = F(I)+C*FE(I)
          end do
```

```
       end do
       SIGe=0.0
       do I=1,3
          SIGe=SIGe+E0hat(I)*aimag(F(I))
       end do
       SIGe=4*PI*SIGe
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      real function FM(TH,PH,R,k,X,DVOL,E,NUSE,RRR)
C*--------------------------------------------------------------------
C*********************************************************************
C* Function to do the main calculation of the magnitude of the         *
C*     scattering amplitude. Angles are in radians                      *
C*********************************************************************
C**** Variable diffinitions                                          ****
      implicit none
      include 'nmax.inc'
C *** loop counters
      integer mu
      integer I, J, NUSE
C *** Reals
      real PI, K, K2,K3, DVOL(NMAX), R(NMAX, 3)
      real TH, PH, RDOT, Rhat(3),THhat(3),PHhat(3)
      real RRR(3,3),V(3),HOR,VER
C     complex quantities
      complex E(NMAX, 3), X, CI, C,temp,fh,fv,TDE,PDE
C *** Parameters
      parameter (PI = 3.141592654, CI = (0.0, 1.0))
C *** Calculations                                                   ***
c      print *, ' FM '  ,TH,PH
      K2=K*K
      K3=K2*K
C         print *, 'calculating FM'
C ***    define THhat and PHhat, and rotate them into prime frame    ***
         V(1)= cos(TH)*cos(PH)
         V(2)= cos(TH)*sin(PH)
         V(3)=-sin(TH)
         call MV(RRR,V,THhat)
         V(1)=-sin(PH)
         V(2)= cos(PH)
         V(3)= 0
         call MV(RRR,V,PHhat)
C ***    THhat and PHhat are now really primed,but drop prime        ***
C ***    calculate r-hat prime direction for the given theta         ***
         V(1) = sin(TH)*cos(PH)
         V(2) = sin(TH)*sin(PH)
         V(3) = cos(TH)
         call MV(RRR,V,Rhat)
c         print *, Rhat(1), Rhat(2), Rhat(3)
```

```
C ***    loop over cell number mu                                        ***
         fh=(0.0,0.0)
         fv=(0.0,0.0)
         do mu = 1, NUSE
            RDOT = 0.0
C ***       calculate rhatprime dot rprime for the exponential           ***
            do j = 1, 3
               RDOT = RDOT + R(mu, j)*Rhat(j)
            end do
C ***       calculate all factors that don't depend on i and j           ***
            temp = -CI*K*RDOT
            C = CI*K3*X*DVOL(mu)*cexp(temp)
C ***       now put the scattering amplitude together                    ***
            TDE=(0.0,0.0)
            PDE=(0.0,0.0)
C ***       calculate THhat dot E and PHhat dot E
            do I = 1, 3
               TDE=TDE+THhat(I)*E(mu,I)
               PDE=PDE+PHhat(i)*E(mu,I)
            end do
C ***       now put the scattering amplitude together                    ***
            fh=fh+C*TDE
            fv=fv+C*PDE
         end do
C ***    separate into _H_orizontal and _V_ertical components and        ***
C *        at the same time convert to diff. scat. cross section (/K2)  *
         HOR = (Fh*conjg(Fh))/K2
         VER = (Fv*conjg(Fv))/K2
         FM  = HOR+VER
c         print *, TH, PH, FM  ,HOR,VER
      return
      end

C*********************************************************************
C*--------------------------------------------------------------------
      Subroutine simpson(Res,R,k,X,DVOL,E,NUSE,RRR)
C*--------------------------------------------------------------------
C*********************************************************************
C*                                                                   *
C* Subroutine to perform a Simpson's rule integration in two         *
C*    dimensions by performing a one dimensional simpson's rule to   *
C*    all rows of data in a grid, and then performing the Simpson's  *
C*    rule again on the array of results.                            *
C*    Sub Simpson is set up for spherical cordinates theta an phi    *
C*    where the theta is used as the inner most integration.  The    *
C*    integragration is carried out in three sections to try to      *
C*    spend the most time evaluating the integral where it is most   *
C*    important.                                                     *
C*********************************************************************
C**** Variables                                                    ****
      implicit none
```

```
      include 'nmax.inc'
C *** Declarations for use in FM
      real R(NMAX, 3),K,DVOL
      complex X(NMAX),E(NMAX, 3)
      integer NUSE
C *** Declaration for use in simpson
      real RRR(3,3)
      real PI
      real TH1,TH2,PH1,PH2
      real Res
      real qsimp1
      parameter (PI = 3.141592654)
      RES=0.0
C***  Set up limits of integration for each region and integrate.    ***
      write(6,*) 'Integrating region...'
      TH1=0.0                ! 0 degrees
      TH2=pi !30.0*PI/180.0      ! 30 degrees
      PH1=0.0
      PH2=2.0*PI
      RES=qsimp1(TH1,TH2,PH1,PH2,R,K,X,DVOL,E,NUSE,RRR)
      print *, 'Result so far ', RES
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      real function qsimp1(TH1,TH2,PH1,PH2,R,K,X,DVOL,E,NUSE,RRR)
C*--------------------------------------------------------------------
C*********************************************************************
C *** Declarations for use in FM
      include 'nmax.inc'
      real R(NMAX, 3),K,DVOL
      complex X(NMAX),E(NMAX, 3)
      integer NUSE
C *** Other declarations
      INTEGER JMAX
      REAL TH1,TH2,PH1,PH2,s,EPS
      real RRR(3,3)
      PARAMETER (EPS=1.e-2, JMAX=200)
CU    USES trapzd1
      INTEGER j
      REAL os,ost,st
      ost=-1.e30
      os= -1.e30
      do 11 j=1,JMAX
        call trapzd1(TH1,TH2,PH1,PH2,st,j,R,K,X,DVOL,E,NUSE,RRR)
        s=(4.*st-ost)/3.
        if (abs(s-os).lt.EPS*abs(os)) then
           qsimp1=s
           return
        end if
        print *, j,s,os,s-os
```

```
        os=s
        ost=st
11    continue
      pause 'too many steps in qsimp1'
      END
C*********************************************************************
C*--------------------------------------------------------------------
      SUBROUTINE trapzd1(TH1,TH2,PH1,PH2,s,n,R,K,X,DVOL,E,NUSE,RRR)
C*--------------------------------------------------------------------
C*********************************************************************
C *** Declarations for use in FM
      include 'nmax.inc'
      real R(NMAX, 3),K,DVOL
      real RRR(3,3)
      complex X(NMAX),E(NMAX, 3)
      integer NUSE
C *** Other declarations
      INTEGER n
      REAL TH1,TH2,PH1,PH2,s,qsimp2
      INTEGER it,j
      REAL del,sum,tnm,PH
      if (n.eq.1) then
        print *, 'call to qsimp2'
        s=0.5*(PH2-PH1)*(qsimp2(TH1,TH2,PH1,R,K,X,DVOL,E,NUSE,RRR)
     &    +qsimp2(TH1,TH2,PH2,R,K,X,DVOL,E,NUSE,RRR))
      else
        it=2**(n-2)
        tnm=it
        del=(PH1-PH2)/tnm
        PH=PH1+0.5*del
        sum=0.
        do 11 j=1,it
          sum=sum+qsimp2(TH1,TH2,PH,R,K,X,DVOL,E,NUSE,RRR)
          PH=PH+del
11      continue
        s=0.5*(s+(PH2-PH1)*sum/tnm)
      endif
      return
      END
C*********************************************************************
C*--------------------------------------------------------------------
      real function qsimp2(TH1,TH2,PH,R,K,X,DVOL,E,NUSE,RRR)
C*--------------------------------------------------------------------
C*********************************************************************
C *** Declarations for use in FM
      include 'nmax.inc'
      real R(NMAX, 3),K,DVOL
      complex X(NMAX),E(NMAX, 3)
      integer NUSE
C *** Other declarations
      INTEGER JMAX
```

```
      REAL TH1,TH2,s,EPS,PH
      real RRR(3,3)
      PARAMETER (EPS=1.e-2, JMAX=200)
CU    USES trapzd2
      INTEGER j
      REAL os,ost,st
c      print *, 'TH ',TH1, TH2,' PH ',ph
      ost=-1.e30
      os= -1.e30
      do 11 j=1,JMAX
c        print *, 'call trapzd2 ', TH1, TH2,  PH
        call trapzd2(TH1,TH2,st,j,PH,R,K,X,DVOL,E,NUSE,RRR)
        s=(4.*st-ost)/3.
        if (abs(s-os).lt.EPS*abs(os)) then
           qsimp2=s
           return
        end if
        os=s
        ost=st
11    continue
      pause 'too many steps in qsimp2'
      END
C*********************************************************************
C*--------------------------------------------------------------------
      SUBROUTINE trapzd2(TH1,TH2,s,n,PH,R,K,X,DVOL,E,NUSE,RRR)
C*--------------------------------------------------------------------
C*********************************************************************
C*    Declarations for use in FM
      include 'nmax.inc'
      real R(NMAX, 3),K,DVOL
      complex X(NMAX),E(NMAX, 3)
      integer NUSE
C*    Other declarations
      INTEGER n
      REAL TH1,TH2,s,FM,PH
      real RRR(3,3)
      INTEGER it,j
      REAL del,sum,tnm,TH
      if (n.eq.1) then
        s=0.5*sin(TH)*(TH2-TH1)*(FM(TH1,PH,R,K,X,DVOL,E,NUSE,RRR)
     &    +FM(TH2,PH,R,K,X,DVOL,E,NUSE,RRR))
      else
        it=2**(n-2)
        tnm=it
        del=(TH2-TH1)/tnm
        TH=TH1+0.5*del
        sum=0.
        do 11 j=1,it
          sum=sum+sin(TH)*FM(TH,PH,R,K,X,DVOL,E,NUSE,RRR)
          TH=TH+del
11      continue
```

```
        s=0.5*(s+(TH2-TH1)*sum/tnm)
      endif
      return
      end
C*********************************************************************
C*********************************************************************
C*-------------------------------------------------------------------
      subroutine ROT(RRR, alpha, beta, gamma)
C*-------------------------------------------------------------------
C*********************************************************************
C*    subroutine to return the Euler rotation matrix where the three   *
C*      rotation angles are alpha, a rotation about the z-axis, beta,  *
C*      a rotation about the new y-axis, and gamma, a rotation about   *
C*      the new z-axis.                                                *
C*********************************************************************
C****  Variables                                                   ****
       real RRR(3,3),alpha, beta, gamma
C
       RRR(1,1)= cos(alpha)*cos(beta)*cos(gamma)-sin(alpha)*sin(gamma)
       RRR(1,2)= sin(alpha)*cos(beta)*cos(gamma)+cos(alpha)*sin(gamma)
       RRR(1,3)=-sin(beta)*cos(gamma)
C
       RRR(2,1)=-cos(alpha)*cos(beta)*sin(gamma)-sin(alpha)*cos(gamma)
       RRR(2,2)=-sin(alpha)*cos(beta)*sin(gamma)+cos(alpha)*cos(gamma)
       RRR(2,3)= sin(beta)*sin(gamma)
C
       RRR(3,1)= cos(alpha)*sin(beta)
       RRR(3,2)= sin(alpha)*sin(beta)
       RRR(3,3)= cos(beta)
C
      return
      end
C*********************************************************************
C*-------------------------------------------------------------------
      subroutine MV(M,V,U)
C*-------------------------------------------------------------------
C*********************************************************************
C*    subroutine to multiply a 3x3 matrix by a vector with three       *
C*      components                                                     *
C*********************************************************************
C****  Variables                                                   ****
       real M(3,3), V(3), U(3), temp
       integer i,j
C
       do i=1,3
          temp=0.0
          do j=1,3
             temp=temp+M(i,j)*V(j)
          end do
          U(i)=temp
       end do
```

```
      return
      end
C*********************************************************************
```

```
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine ROT(RRR, alpha, beta, gamma)
C*--------------------------------------------------------------------
C*********************************************************************
C*    subroutine to return the Euler rotation matrix where the three   *
C*      rotation angles are alpha, a rotation about the z-axis, beta,  *
C*      a rotation about the new y-axis, and gamma, a rotation about   *
C*      the new z-axis.                                                *
C*********************************************************************
C****  Variables                                                   ****
       implicit none
       real RRR(3,3),alpha, beta, gamma
C
       RRR(1,1)= cos(alpha)*cos(beta)*cos(gamma)-sin(alpha)*sin(gamma)
       RRR(1,2)= sin(alpha)*cos(beta)*cos(gamma)+cos(alpha)*sin(gamma)
       RRR(1,3)=-sin(beta)*cos(gamma)
C
       RRR(2,1)=-cos(alpha)*cos(beta)*sin(gamma)-sin(alpha)*cos(gamma)
       RRR(2,2)=-sin(alpha)*cos(beta)*sin(gamma)+cos(alpha)*cos(gamma)
       RRR(2,3)= sin(beta)*sin(gamma)
C
       RRR(3,1)= cos(alpha)*sin(beta)
       RRR(3,2)= sin(alpha)*sin(beta)
       RRR(3,3)= cos(beta)
C
      return
      end
C*********************************************************************
C*--------------------------------------------------------------------
      subroutine MV(M,V,U)
C*--------------------------------------------------------------------
C*********************************************************************
C*    subroutine to multiply a 3x3 matrix by a vector with three       *
C*      components                                                     *
C*********************************************************************
C****  Variables                                                   ****
       implicit none
       real M(3,3), V(3), U(3), temp
       integer i,j
C
       do i=1,3
          temp=0.0
          do j=1,3
             temp=temp+M(i,j)*V(j)
          end do
          U(i)=temp
       end do
      return
      end
C*********************************************************************
```

```
      subroutine cgeco(a,lda,n,ipvt,rcond,z)
      integer lda,n,ipvt(1)
      complex a(lda,1),z(1)
      real rcond
c
c     cgeco factors a complex matrix by gaussian elimination
c     and estimates the condition of the matrix.
c
c     if  rcond  is not needed, cgefa is slightly faster.
c     to solve  a*x = b , follow cgeco by cgesl.
c     to compute  inverse(a)*c , follow cgeco by cgesl.
c     to compute  determinant(a) , follow cgeco by cgedi.
c     to compute  inverse(a) , follow cgeco by cgedi.
c
c     on entry
c
c        a       complex(lda, n)
c                the matrix to be factored.
c
c        lda     integer
c                the leading dimension of the array  a .
c
c        n       integer
c                the order of the matrix  a .
c
c     on return
c
c        a       an upper triangular matrix and the multipliers
c                which were used to obtain it.
c                the factorization can be written  a = l*u  where
c                l  is a product of permutation and unit lower
c                triangular matrices and  u  is upper triangular.
c
c        ipvt    integer(n)
c                an integer vector of pivot indices.
c
c        rcond   real
c                an estimate of the reciprocal condition of  a .
c                for the system  a*x = b , relative perturbations
c                in  a  and  b  of size  epsilon  may cause
c                relative perturbations in  x  of size  epsilon/rcond .
c                if  rcond  is so small that the logical expression
c                           1.0 + rcond .eq. 1.0
c                is true, then  a  may be singular to working
c                precision.  in particular,  rcond  is zero  if
c                exact singularity is detected or the estimate
c                underflows.
c
c        z       complex(n)
c                a work vector whose contents are usually unimportant.
c                if  a  is close to a singular matrix, then  z  is
```

```
c                an approximate null vector in the sense that
c                norm(a*z) = rcond*norm(a)*norm(z) .
c
c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.
c
c     subroutines and functions
c
c     linpack cgefa
c     blas caxpy,cdotc,csscal,scasum
c     fortran abs,aimag,amax1,cmplx,conjg,real
c
c     internal variables
c
      complex cdotc,ek,t,wk,wkm
      real anorm,s,scasum,sm,ynorm
      integer info,j,k,kb,kp1,l
c
      complex zdum,zdum1,zdum2,csign1
      real cabs1
      cabs1(zdum) = abs(real(zdum)) + abs(aimag(zdum))
      csign1(zdum1,zdum2) = cabs1(zdum1)*(zdum2/cabs1(zdum2))
c
c     compute 1-norm of a
c
      anorm = 0.0e0
      do 10 j = 1, n
         anorm = amax1(anorm,scasum(n,a(1,j),1))
   10 continue
c
c     factor
c
      call cgefa(a,lda,n,ipvt,info)
c
c     rcond = 1/(norm(a)*(estimate of norm(inverse(a)))) .
c     estimate = norm(z)/norm(y) where  a*z = y  and  ctrans(a)*y = e .
c     ctrans(a)  is the conjugate transpose of a .
c     the components of  e  are chosen to cause maximum local
c     growth in the elements of w  where  ctrans(u)*w = e .
c     the vectors are frequently rescaled to avoid overflow.
c
c     solve ctrans(u)*w = e
c
      ek = (1.0e0,0.0e0)
      do 20 j = 1, n
         z(j) = (0.0e0,0.0e0)
   20 continue
      do 100 k = 1, n
         if (cabs1(z(k)) .ne. 0.0e0) ek = csign1(ek,-z(k))
         if (cabs1(ek-z(k)) .le. cabs1(a(k,k))) go to 30
            s = cabs1(a(k,k))/cabs1(ek-z(k))
```

```
            call csscal(n,s,z,1)
            ek = cmplx(s,0.0e0)*ek
   30    continue
         wk = ek - z(k)
         wkm = -ek - z(k)
         s = cabs1(wk)
         sm = cabs1(wkm)
         if (cabs1(a(k,k)) .eq. 0.0e0) go to 40
            wk = wk/conjg(a(k,k))
            wkm = wkm/conjg(a(k,k))
         go to 50
   40    continue
            wk = (1.0e0,0.0e0)
            wkm = (1.0e0,0.0e0)
   50    continue
         kp1 = k + 1
         if (kp1 .gt. n) go to 90
            do 60 j = kp1, n
               sm = sm + cabs1(z(j)+wkm*conjg(a(k,j)))
               z(j) = z(j) + wk*conjg(a(k,j))
               s = s + cabs1(z(j))
   60       continue
            if (s .ge. sm) go to 80
               t = wkm - wk
               wk = wkm
               do 70 j = kp1, n
                  z(j) = z(j) + t*conjg(a(k,j))
   70          continue
   80       continue
   90    continue
         z(k) = wk
  100 continue
      s = 1.0e0/scasum(n,z,1)
      call csscal(n,s,z,1)
c
c     solve ctrans(l)*y = w
c
      do 120 kb = 1, n
         k = n + 1 - kb
         if (k .lt. n) z(k) = z(k) + cdotc(n-k,a(k+1,k),1,z(k+1),1)
         if (cabs1(z(k)) .le. 1.0e0) go to 110
            s = 1.0e0/cabs1(z(k))
            call csscal(n,s,z,1)
  110    continue
         l = ipvt(k)
         t = z(l)
         z(l) = z(k)
         z(k) = t
  120 continue
      s = 1.0e0/scasum(n,z,1)
      call csscal(n,s,z,1)
```

```
c
      ynorm = 1.0e0
c
c     solve l*v = y
c
      do 140 k = 1, n
         l = ipvt(k)
         t = z(l)
         z(l) = z(k)
         z(k) = t
         if (k .lt. n) call caxpy(n-k,t,a(k+1,k),1,z(k+1),1)
         if (cabs1(z(k)) .le. 1.0e0) go to 130
            s = 1.0e0/cabs1(z(k))
            call csscal(n,s,z,1)
            ynorm = s*ynorm
  130    continue
  140 continue
      s = 1.0e0/scasum(n,z,1)
      call csscal(n,s,z,1)
      ynorm = s*ynorm
c
c     solve  u*z = v
c
      do 160 kb = 1, n
         k = n + 1 - kb
         if (cabs1(z(k)) .le. cabs1(a(k,k))) go to 150
            s = cabs1(a(k,k))/cabs1(z(k))
            call csscal(n,s,z,1)
            ynorm = s*ynorm
  150    continue
         if (cabs1(a(k,k)) .ne. 0.0e0) z(k) = z(k)/a(k,k)
         if (cabs1(a(k,k)) .eq. 0.0e0) z(k) = (1.0e0,0.0e0)
         t = -z(k)
         call caxpy(k-1,t,a(1,k),1,z(1),1)
  160 continue
c     make znorm = 1.0
      s = 1.0e0/scasum(n,z,1)
      call csscal(n,s,z,1)
      ynorm = s*ynorm
c
      if (anorm .ne. 0.0e0) rcond = ynorm/anorm
      if (anorm .eq. 0.0e0) rcond = 0.0e0
      return
      end

    subroutine cgesl(a,lda,n,ipvt,b,job)
      integer lda,n,ipvt(1),job
      complex a(lda,1),b(1)
c
c     cgesl solves the complex system
c     a * x = b  or  ctrans(a) * x = b
```

```
c     using the factors computed by cgeco or cgefa.
c
c     on entry
c
c        a       complex(lda, n)
c                the output from cgeco or cgefa.
c
c        lda     integer
c                the leading dimension of the array  a .
c
c        n       integer
c                the order of the matrix  a .
c
c        ipvt    integer(n)
c                the pivot vector from cgeco or cgefa.
c
c        b       complex(n)
c                the right hand side vector.
c
c        job     integer
c                = 0         to solve  a*x = b ,
c                = nonzero   to solve  ctrans(a)*x = b  where
c                            ctrans(a)  is the conjugate transpose.
c
c     on return
c
c        b       the solution vector  x .
c
c     error condition
c
c        a division by zero will occur if the input factor contains a
c        zero on the diagonal.  technically this indicates singularity
c        but it is often caused by improper arguments or improper
c        setting of lda .  it will not occur if the subroutines are
c        called correctly and if cgeco has set rcond .gt. 0.0
c        or cgefa has set info .eq. 0 .
c
c     to compute  inverse(a) * c  where  c  is a matrix
c     with  p  columns
c           call cgeco(a,lda,n,ipvt,rcond,z)
c           if (rcond is too small) go to ...
c           do 10 j = 1, p
c              call cgesl(a,lda,n,ipvt,c(1,j),0)
c        10 continue
c
c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.
c
c     subroutines and functions
c
c     blas caxpy,cdotc
```

```
c     fortran conjg
c
c     internal variables
c
      complex cdotc,t
      integer k,kb,l,nm1
c
      nm1 = n - 1
      if (job .ne. 0) go to 50
c
c        job = 0 , solve  a * x = b
c        first solve  l*y = b
c
         if (nm1 .lt. 1) go to 30
         do 20 k = 1, nm1
            l = ipvt(k)
            t = b(l)
            if (l .eq. k) go to 10
               b(l) = b(k)
               b(k) = t
   10       continue
            call caxpy(n-k,t,a(k+1,k),1,b(k+1),1)
   20    continue
   30    continue
c
c        now solve  u*x = y
c
         do 40 kb = 1, n
            k = n + 1 - kb
            b(k) = b(k)/a(k,k)
            t = -b(k)
            call caxpy(k-1,t,a(1,k),1,b(1),1)
   40    continue
      go to 100
   50 continue
c
c        job = nonzero, solve  ctrans(a) * x = b
c        first solve  ctrans(u)*y = b
c
         do 60 k = 1, n
            t = cdotc(k-1,a(1,k),1,b(1),1)
            b(k) = (b(k) - t)/conjg(a(k,k))
   60    continue
c
c        now solve ctrans(l)*x = y
c
         if (nm1 .lt. 1) go to 90
         do 80 kb = 1, nm1
            k = n - kb
            b(k) = b(k) + cdotc(n-k,a(k+1,k),1,b(k+1),1)
            l = ipvt(k)
```

```
            if (l .eq. k) go to 70
               t = b(l)
               b(l) = b(k)
               b(k) = t
   70       continue
   80    continue
   90    continue
  100 continue
      return
      end

      subroutine cgefa(a,lda,n,ipvt,info)
      integer lda,n,ipvt(1),info
      complex a(lda,1)
c
c     cgefa factors a complex matrix by gaussian elimination.
c
c     cgefa is usually called by cgeco, but it can be called
c     directly with a saving in time if  rcond  is not needed.
c     (time for cgeco) = (1 + 9/n)*(time for cgefa) .
c
c     on entry
c
c        a       complex(lda, n)
c                the matrix to be factored.
c
c        lda     integer
c                the leading dimension of the array  a .
c
c        n       integer
c                the order of the matrix  a .
c
c     on return
c
c        a       an upper triangular matrix and the multipliers
c                which were used to obtain it.
c                the factorization can be written  a = l*u  where
c                l  is a product of permutation and unit lower
c                triangular matrices and  u  is upper triangular.
c
c        ipvt    integer(n)
c                an integer vector of pivot indices.
c
c        info    integer
c                = 0  normal value.
c                = k  if  u(k,k) .eq. 0.0 .  this is not an error
c                     condition for this subroutine, but it does
c                     indicate that cgesl or cgedi will divide by zero
c                     if called.  use  rcond  in cgeco for a reliable
c                     indication of singularity.
c
```

```
c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.
c
c     subroutines and functions
c
c     blas caxpy,cscal,icamax
c     fortran abs,aimag,real
c
c     internal variables
c
      complex t
      integer icamax,j,k,kp1,l,nm1
c
      complex zdum
      real cabs1
      cabs1(zdum) = abs(real(zdum)) + abs(aimag(zdum))
c
c     gaussian elimination with partial pivoting
c
      info = 0
      nm1 = n - 1
      if (nm1 .lt. 1) go to 70
      do 60 k = 1, nm1
         kp1 = k + 1
c
c        find l = pivot index
c
         l = icamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l
c
c        zero pivot implies this column already triangularized
c
         if (cabs1(a(l,k)) .eq. 0.0e0) go to 40
c
c           interchange if necessary
c
            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue
c
c           compute multipliers
c
            t = -(1.0e0,0.0e0)/a(k,k)
            call cscal(n-k,t,a(k+1,k),1)
c
c           row elimination with column indexing
c
            do 30 j = kp1, n
               t = a(l,j)
```

```
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call caxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
   70 continue
      ipvt(n) = n
      if (cabs1(a(n,n)) .eq. 0.0e0) info = n
      return
      end

      subroutine caxpy(n,ca,cx,incx,cy,incy)
c
c     constant times a vector plus a vector.
c     jack dongarra, linpack, 3/11/78.
c     modified 12/3/93, array(1) declarations changed to array(*)
c
      complex cx(*),cy(*),ca
      integer i,incx,incy,ix,iy,n
c
      if(n.le.0)return
      if (abs(real(ca)) + abs(aimag(ca)) .eq. 0.0 ) return
      if(incx.eq.1.and.incy.eq.1)go to 20
c
c        code for unequal increments or equal increments
c          not equal to 1
c
      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        cy(iy) = cy(iy) + ca*cx(ix)
        ix = ix + incx
        iy = iy + incy
   10 continue
      return
c
c        code for both increments equal to 1
c
   20 do 30 i = 1,n
        cy(i) = cy(i) + ca*cx(i)
   30 continue
      return
      end
```

```
      complex function cdotc(n,cx,incx,cy,incy)
c
c     forms the dot product of two vectors, conjugating the first
c     vector.
c     jack dongarra, linpack,  3/11/78.
c     modified 12/3/93, array(1) declarations changed to array(*)
c
      complex cx(*),cy(*),ctemp
      integer i,incx,incy,ix,iy,n
c
      ctemp = (0.0,0.0)
      cdotc = (0.0,0.0)
      if(n.le.0)return
      if(incx.eq.1.and.incy.eq.1)go to 20
c
c        code for unequal increments or equal increments
c          not equal to 1
c
      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        ctemp = ctemp + conjg(cx(ix))*cy(iy)
        ix = ix + incx
        iy = iy + incy
   10 continue
      cdotc = ctemp
      return
c
c        code for both increments equal to 1
c
   20 do 30 i = 1,n
        ctemp = ctemp + conjg(cx(i))*cy(i)
   30 continue
      cdotc = ctemp
      return
      end

      subroutine  csscal(n,sa,cx,incx)
c
c     scales a complex vector by a real constant.
c     jack dongarra, linpack, 3/11/78.
c     modified 3/93 to return if incx .le. 0.
c     modified 12/3/93, array(1) declarations changed to array(*)
c
      complex cx(*)
      real sa
      integer i,incx,n,nincx
c
```

```
      if( n.le.0 .or. incx.le.0 )return
      if(incx.eq.1)go to 20
c
c        code for increment not equal to 1
c
      nincx = n*incx
      do 10 i = 1,nincx,incx
        cx(i) = cmplx(sa*real(cx(i)),sa*aimag(cx(i)))
   10 continue
      return
c
c        code for increment equal to 1
c
   20 do 30 i = 1,n
        cx(i) = cmplx(sa*real(cx(i)),sa*aimag(cx(i)))
   30 continue
      return
      end

     real function scasum(n,cx,incx)
c
c     takes the sum of the absolute values of a complex vector and
c     returns a single precision result.
c     jack dongarra, linpack, 3/11/78.
c     modified 3/93 to return if incx .le. 0.
c     modified 12/3/93, array(1) declarations changed to array(*)
c
      complex cx(*)
      real stemp
      integer i,incx,n,nincx
c
      scasum = 0.0e0
      stemp = 0.0e0
      if( n.le.0 .or. incx.le.0 )return
      if(incx.eq.1)go to 20
c
c        code for increment not equal to 1
c
      nincx = n*incx
      do 10 i = 1,nincx,incx
        stemp = stemp + abs(real(cx(i))) + abs(aimag(cx(i)))
   10 continue
      scasum = stemp
      return
c
c        code for increment equal to 1
c
   20 do 30 i = 1,n
        stemp = stemp + abs(real(cx(i))) + abs(aimag(cx(i)))
   30 continue
      scasum = stemp
```

```
return
end
```