UNIVERSITY OF CENTRAL OKLAHOMA
Edmond, Oklahoma
Jackson College of Graduate Studies

Electromagnetic Field Distribution and Power Absorption of 3D Spherical Objects

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING PHYSICS-
MECHANICAL ENGINEERING

By

Timothy M. Collins Jr.

Edmond, Oklahoma

MAY 2017

Electromagnetic Field Distribution and Power Absorption of 3D Spherical Objects

A THESIS

APPROVED FOR THE DEPARTMENT OF
ENGINEERING & PHYSICS

By [illegible] 4/26/2017
Dr. Mohammad Hossan Date
Committee Chairperson
Department of Engineering and Physics

Evan C. Lemley 4/26/17
Dr. Evan Lemley Date
Committee Member
Department of Engineering and Physics

[illegible] 4/26/17
Dr. Gang Xu Date
Committee Member
Department of Engineering and Physics

ABTRACT OF THESIS

Electromagnetic Field Distribution and Power Absorption of 3D Spherical Objects

Timothy M. Collins Jr.

Thesis Advisor: Dr. Mohammad R. Hossan

Microwave and radiofrequency heating has great promise in many engineering and biomedical applications because of its non-contact, volumetric heat generation and selective heating. However, the heating patterns and temperature distributions are non-uniform and difficult to control. Electromagnetic power absorption guides the heating pattern which is a complex function of dielectric properties, electromagnetic frequencies, size, and shape of the target object. A closed form expression of power absorption with functional relationship with various parameters is obtained for a spherical shaped dielectric object using Maxwell's equations in spherical coordinate. Maxwell's equations are solved using vector potentials and separation of variables. Mathematical tools such as Bessel functions, Legendre Polynomials, infinite series, and complex number expressions are employed in finding the solution. The electromagnetic power absorption is calculated from the knowledge of electromagnetic field within the object using Poynting theorem. The analytical expression of the electric field, magnetic field, and power generation within the sphere are coded in MATLAB and FORTRAN to get numerical results for spherical shaped meat balls of 1.0, 2.0, 3.0 and 5.0 cm radii with varying properties and electromagnetic frequencies of 2800 MHz, 2450 MHz, 915 MHz, and 300 MHz. Origin Labs is utilized to produce 1-D plots and also 2-D polar plots by reading the data text files generated in the FORTRAN program. Results show that the presence of local maxima of electric and magnetic field strength due to the constructive interference of the electromagnetic wave in the target object. The spatial distribution of microwave power absorption follows the trend of electromagnetic field distribution. The locations of local maxima and minima of power absorption and electromagnetic field distributions vary with the radius of the sphere and applied frequencies. The results also show that the strength of the absorbed electromagnetic wave at the 2450 MHz is most non-uniform at the radius of 3 cm nugget. The smallest (1 cm radius) and largest (5 cm radius) dielectric radii show a lower electromagnetic and power generation peak values but a more even distribution of energy overall. Analysis reveals the correlations of propagating wavelength, penetration depth of electromagnetic waves and size of the beef nuggets. Results indicate that the uniform and effective electromagnetic power absorption can be facilitated by proper design of the object of interest and selection of appropriate frequencies. This rigorous analytic investigation will provide significant insight in understanding the power absorption and temperature distribution mechanism for spherical shaped objects under electromagnetic wave (microwave and radiofrequency) treatment.

ACKNOWLEDGEMENT

I would like to thank Dr. Hossan, Dr. Lemley, Dr. Xu, Dr. Chen, Dr. Wilson, Dr. Hughes, and Dr. Bingabr for their support and assistance during my graduate sojourn at UCO. Completion of this degree would have been impossible without their assistance. I would also like to thank Mr. Rada, Mr. St. John, Ms. Dels, and Ms. Alaee for their friendship, employment opportunities, and assistance as well.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

Electromagnetic Field Distribution and Power Absorption of 3D Spherical Objects

Chapter 1:

Introduction

The use of microwave ovens to heat and cook food is seemingly ubiquitous in modern life. Microwave heating utilizes electromagnetic (EM) waves in the frequency range between 300 MHz and 300 GHz. Heating can be accomplished in lower frequencies of the electromagnetic spectrum as well, such as radiofrequency (3 kHz to 300 MHz) heating. Electromagnetic heating has emerged as one of the most promising heating mechanisms as electromagnetic irradiation imparts several advantages over the use of conventional ovens such as: 1. Electromagnetic heating possesses higher energy efficiency since heating is focused on the target object and surroundings are not greatly affected by the EM radiation[1]; 2. The ability to begin heating all areas of the object instantaneously rather than a transfer of heat energy from the outside to the inside[2]; 3. It does not require physical contact and provides shorter processing time[3]; 4. In mass scale industrial material processing, it provides pollution free, environment friendly heating process[4]; 5. It can be designed for material selective heating for composite material processing[5]. The electromagnetic heating mechanism has been demonstrated not only in food processing but also in many other engineering applications such as polymer processing, contaminated soil remediation, waste processing, minerals processing and activated carbon regeneration, superficial tissue disease treatment, drilling in oil and gas industries, etc[3,4,6,7].

Electromagnetic heating, i.e. microwave and radiofrequency heating, is accomplished by the excitation of polar molecules in dielectric material by the alternating amplitude of the electric component of the electromagnetic wave. These excited molecules exhibit “dipole moments” due to their polar construction that causes them to flip and rotate with the passing of the electromagnetic wave. As the amplitude and orientation of the electromagnetic wave changes, the orientation of the polar molecule also changes in the attempt to minimize the dipole moment. With high frequency waves, the effect causes significant friction as a result of this molecular motion [2]. In foodstuffs, water, in particular, is one of the major source of molecular motion due to its polar nature. Fat molecules also are subject to this molecular motion. Prompt heating can be accomplished with radiofrequency and microwave radiation, as heat is generated internally within the target object. However, due to the nature of how a substance is heated under this method, electromagnetic heating is generally non-uniform, often leading to uneven heating of the host material [2].

1.1 Background of the Study

The non-uniformity of electromagnetic heating is due to several factors, including: constructive/destructive wave interactions within the target object, electromagnetic properties of the specimen such as dielectric losses and electrical permittivity, physical properties such as shape, size and material phase, and strength of incident electromagnetic wave energy in the target object. Non-uniform heating poses a number of challenges in material and food processing. Uneven heating can alter chemical composition and texture of the materials. In foodstuffs, the texturing is often over-emphasized if the extremes between hot and cold regions are significant enough. Overcooking and burning often result due to a higher degree of uneven heating. While overheating can result in distorted taste and burning, undercooking can present its own set of problems. For instances, food processing requires a minimum cooking temperature for safe consumption. Certain foods, such as meat products, are more pathologically active at certain temperatures for bacterial growth or other pathogens; uneven heating can facilitate these temperatures which are not desirable and can pose serious health issues[8]. In addition, the non-uniformity of heating limits widespread use, especially as a replacement for conventional heating methods [9]. Better understanding of the electromagnetic heating mechanism, and the effects of parameters on heating patterns and control, can be put to good use in many industries including material and food processing.

A tremendous effort has been invested in understanding and improving control of electromagnetic heating through experimental and numerical methods[4,10,11]. Experimental techniques, such as measuring real-time temperature at the different parts and depths of the specimen as a function of electromagnetic energy incident on the specimen, reveals information that can be correlated with material and process parameters [8]. Although experiment reveals critical information and helps us understanding the process, sometimes it is more convenient to perform numerical and theoretical study to grasp the mechanism and physics behind it. Especially since electromagnetic heating is mostly dependent on the electro-thermo-physical properties of the specimen, transformation of one set of results to another condition is not possible. Numerical techniques such as Finite Element Analysis (FEA) and/or Finite Volume Methods (FVM) are used to model the interaction of the irradiating electromagnetic radiation with the dielectric material [12,13]. Numerical modeling and simulation allows greater flexibility in changing material compositions, electro-thermo-physical properties, and process parameters to analyze the effect of electromagnetic irradiation[14].

Despite numerous experimental and numerical studies of electromagnetic heating, the fundamental mechanism and protocols for optimization of process parameters have not been developed yet. Analytical methods seek to find a closed form solution that describes the interaction of the electromagnetic waves with the host material. A handful of pure analytical studies have been reported that can enhance the understanding underlying physics of electromagnetic heating for limited geometric shapes and conditions. Lately, Hossan et al. provided closed-form analytical expression for electromagnetic power absorption and temperature distribution in rectangular[15] and cylindrical shaped objects[9].

In the analytical derivation of closed form expressions of power absorption and electromagnetic field distribution, there are two approaches; - i) Lambert’s law, which is based on the assumption that the electromagnetic radiation decays exponentially and there are no reflection and interference in wave propagation and ii) Maxwell's equation for electromagnetic waves. Maxwell's equation is regarded as the most accurate method and it is imperative to use Maxwell's equation to evaluate absorption of electromagnetic wave energy in small sized samples[16]. The solution of Maxwell's equation provides the distribution of the electromagnetic fields. With the knowledge of the electromagnetic field distributions within the object, the Poynting theorem is utilized to determine the distribution of power within the object. The Poynting theorem has been utilized with numerical as well as analytical modeling in several works. Ayappa et al. produced numerical results of power distribution and heating within lossy dielectric cylindrical rods using 2D finite element analysis [12]. Hossan et al. showed that an analytical result for the microwave heating of rectangular shaped foodstuffs could be generated using Poynting theorem [15]. In another work, Hossan et al. showed an equivalent result for cylindrical shaped foodstuffs [9].

For larger objects (especially those that can be modeled as semi-infinite in some of the spatial dimensions), Lambert's law can generate results with acceptable accuracy for some special cases. However, Lambert's law cannot generate accurate results of microwave power generation in small sized foodstuffs (such as nuggets) [17]. Lambert's law predicts microwave power generation as a simple exponential decay model where wave reflections are neglected in the underlying mathematics [14]. Curet et al. published work that highlights the key differences between the two approaches [18]. Modeling with Lambert's law can be implemented in a more straightforward fashion, without needing intimate knowledge of the object's electromagnetic field intensities [18]. Computationally speaking, Maxwell's law is more intensive on siliconic resources, although simplifications can be made in certain cases that permit use of electronic spreadsheets [19].

1.2 Statement of the Problem

Although there are some reported works that provide an analytical expression for the microwave power absorption for rectangular and cylindrical shaped objects, there are no analytical closed form expressions of electric, magnetic and electromagnetic power absorption for a three dimensional spherical shaped object. Spherical shape objects under electromagnetic treatment requires solving Maxwell's equation in the spherical coordinate system. A more common mathematical arrangement involves allowing the electromagnetic waves to impinge radially from all directions, which can be modeled as three dimensional radial direction impingement. Therefore the goal of this thesis is to obtain a closed form solution to the electric field, magnetic field, and power distribution in small, spherical dielectric foodstuffs, utilizing the spherical coordinate system and vertical impingement of electromagnetic waves. Based on Hossan et al.'s previous work with rectangular [15] and cylindrical [9] objects and Balanis' work with dielectric scattering [20], the transverse electric and magnetic (TEM) wave is transformed into spherical coordinates. Three dimensional Maxwell's equations in spherical coordinates are solved using vector potentials and separation of variables to evaluate the distribution of the electric and magnetic fields within the target object[21]. The analytical expression of the electric and magnetic fields are then used to find a closed-form solution of electromagnetic power absorption using Poynting theorem. The expression is evaluated for various sizes of meat balls and varying electromagnetic wave frequencies and properties.

References

1. Tewari G. Microwave and Radio-Frequency Heating. *Advances in thermal and non-thermal food preservation.* 2007:91-98.
2. M. E. C. Oliveira ASF. Microwave heating of foodstuffs. *Journal of Food Engineering.* 2001;53(6):347-359.
3. Stern CH. A Transient Heat Transfer Model for Selectvie Microwave Heating of Multilayer Material Systems. *International Microwave Power Institute.* 1998;33(4):207-215.
4. Jones D, Lelyveld T, Mavrofidis S, Kingman S, Miles N. Microwave heating applications in environmental engineering—a review. *Resources, conservation and recycling.* 2002;34(2):75-90.
5. Kasi Balamurugan Mani MRH, Prashanta Dutta. Thermal analysis of microwave assisted bonding of poly(methyl methacrylate) substrates in mircofluidic devices. *International Journal of Heat and Mass Transfer.* 2013;58:229-239.
6. Jacobsen S, Stauffer PR, Neuman DG. Dual-mode antenna design for microwave heating and noninvasive thermometry of superficial tissue disease. *IEEE Transactions on Biomedical Engineering.* 2000;47(11):1500-1509.
7. Jerby E, Dikhtyar V, Aktushev O, Grosglick U. The microwave drill. *Science.* 2002;298(5593):587-589.
8. Goksoy EO. Non-Uniformity of Surface Temperatures After Microwave Heating of Poultry Meat. *International Microwave Power Institute.* 1999;34(3):149-160.
9. Mohammad Robiul Hossan DB, Prashanta, Dutta. Analysis of microwave heating for cylindrical shaped objects. *International Journal of Heat and Mass Transfer.* 2010:5129-5138.
10. Lidström P, Tierney J, Wathey B, Westman J. Microwave assisted organic synthesis—a review. *Tetrahedron.* 2001;57(45):9225-9283.
11. Vadivambal R, Jayas D. Non-uniform temperature distribution during microwave heating of food materials—a review. *Food and Bioprocess Technology.* 2010;3(2):161-171.
12. K. G. Ayappa HTD, E. A. Davis, J. Gordon. Two-Dimensional Finite Element Analysis of Microwave Heating. *AlChE Journal.* 1992:1577-1592.
13. Joseph B. Keller DG. Exact Non-reflecting Boundary Conditions. *Journal of Computational Physics.* 1988;82:172-192.
14. Ayappa K. Modelling transport processes during microwave heating: a review. *Reviews in chemical engineering.* 1997;13(2):1.
15. Hossan MR, Dutta P. Effects of temperature dependent properties in electromagnetic heating. *International journal of heat and mass transfer.* 2012;55(13):3412-3422.
16. BARRINGER SA, DAVIS EA, GORDON J, AYAPPA KG, DAVIS H. Microwave-Heating Temperature Profiles for Thin Slabs Compared to Maxwell and Lambert Law Predictions. *Journal of Food Science.* 1995;60(5):1137-1142.
17. Ayappa K, Davis H, Crapiste G, Davis E, Gordon J. Microwave heating: an evaluation of power formulations. *Chemical engineering science.* 1991;46(4):1005-1016.
18. Curet S, Rouaud O, Boillereaux L. Microwave tempering and heating in a single-mode cavity: Numerical and experimental investigations. *Chemical Engineering and Processing: Process Intensification.* 2008;47(9-10):1656-1665.
19. Fleischman GJ. Predicting temperature range in food slabs undergoing short-term/high-power microwave heating. *Journal of Food Engineering.* 1999;40(1999):81-88.
20. Balanis CA. *Advanced Engineering Electromagnetics.* 2nd ed. Hoboken, NJ: John Wiley & Sons; 2012.
21. Balanis CA. *Advanced Engineering Electromagnetics*. Hoboken, NJ: John WIley & Sons; 2012:260.

Chapter 2:

Literature Review

In this study, the power generation within small spherical foodstuffs irradiated with dual-source planar electromagnetic waves is investigated. The goal of this research is to follow a completely analytical solution methodology, with the aim of finding a closed-form solution to electromagnetic power absorption as a function of dielectric and physical properties of the target object. Heat generation due to power absorption is studied for various electromagnetic frequencies. After defining key theoretical tenets and mathematical techniques, this chapter provides a brief overview of the current state of electromagnetic heating research, applications, methodologies, limitations and future directions. Finally, the chapter will conclude regarding questions the research aims to answer and how this research will contribute in the field.

2.1 Theoretical Background

Electromagnetic waves, particularly the planar variety, often radiate in particular orientations or polarizations. The specific polarizations typically seen with traveling plane waves are the Transverse Electric (TE), Transverse Magnetic (TM) and Transverse Electromagnetic (TEM). The spatial arrangements of each waveform can be seen in Figure 2.1. Of the three modes, TEM is of the lowest energy configuration (lowest-order mode). A microwave oven can theoretically produce waves of any of the three configurations, but TEM waves are usually simple and convenient to model [1].

A transverse electric wave is a specific field configuration that does not have an electric field component in the direction of wave propagation. As an example, in the Cartesian coordinate system, a TE^z (referred to as transverse electric to z) wave does not have an electric component of the electromagnetic wave that exists in the z-direction; however, the electric field components in the x & y directions, and all of the magnetic field (x, y, & z) components can exist [2]. This sort of waveform is represented in Figure 2.1 (b). A transverse magnetic (TM) wave behaves in a similar fashion, only in this case the magnetic component of the wave is the focus of the attention. For instance, a TM^z (transverse magnetic to z) wave in Cartesian coordinates does not have a magnetic component in wave propagating direction, i.e. in this case the z-direction. However the magnetic field components in the x and y directions, and all of the electric field (x, y, & z) components can exist [3]. Figure 2.1 (c) represents such a waveform. A transverse electromagnetic wave does not have an electric or magnetic component in the direction of propagation [4]. For instance, a TEM^z (transverse electromagnetic to z) wave in the Cartesian coordinate system does not have an electric or magnetic field component in the z direction. Figure 2.1 (a) highlights such a waveform. The classification of electromagnetic wave based on the direction of electric and magnetic field is usually described as *mode.* Waveguides are used to guide different modes of electromagnetic waves. Regardless the electromagnetic wave mode, electromagnetic heating functions within the frequency range of 3 kHz – 300 MHz (radiofrequency heating) and 300 MHz- 300 GHz (microwave heating). In the US, specific frequencies for heating or other purposes is specified by the Federal Communications Commission (FCC).

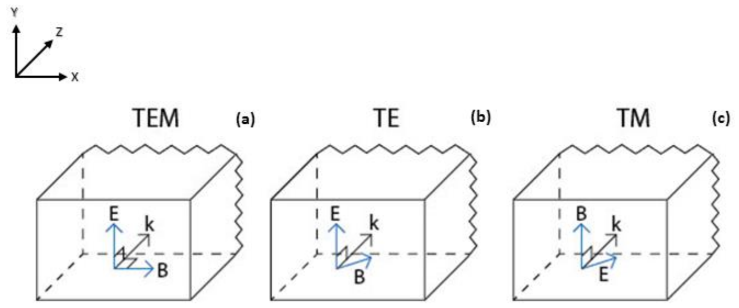

Figure 2.1: A comparison between the three wave modes, (a) TEM, (b) TE, (c), TM, in Cartesian coordinates. k represents the direction of wave travel, and E and M represent the electric and magnetic components of the electromagnetic wave, respectfully.

As a further mathematical simplification, *uniform plane waves* are often employed, as seen in Figure 2.2. While wave propagation is not truly planar in the physical world, making the plane wave or uniform plane wave simplification is often used for convenience in finding a mathematical solution. The electromagnetic wave is represented by Maxwell's equation. The electric and magnetic field distribution of the object subjected to electromagnetic wave heat treatment are predicted by the solution of Maxwell's equation. The knowledge of the electromagnetic field distribution, material properties and wave characteristics (i.e. absorption, penetration and reflection) are used to find the power generation within the target object.

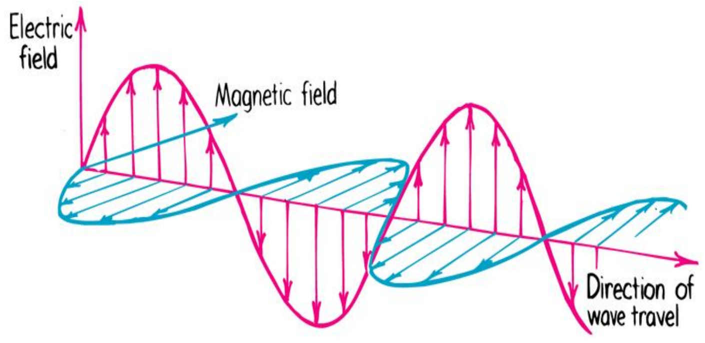

Figure 2.2: An example of a uniform TEM plane wave[1].

2.1.1 Literature Survey Overview

The study of electromagnetic power absorption and heating has become a subject of growing interest in many engineering, food science, and biomedical applications. A tremendous effort has been invested to understand fundamental mechanisms and relations among functional parameters through experimental techniques, computer modeling and simulation, and theoretical investigation with advanced mathematical tools. Experimental studies provide the opportunity of real-time, physical impact of electromagnetic wave impingement, recording behaviors of microwave-dielectric interaction, heat, and temperature distribution of the target object. However, experimental techniques are expensive, require trial and error, and take up significant time and resource commitments. Certain studies, such as those seeking to understand the behaviors of microwave-dielectric interaction that closely model real-world setups, are best studied using experimental methods. Computer modeling and simulation of electromagnetic treatment can optimize experimental protocols and save time and resources. Numerical analysis and simulation can predict the distribution of microwave power absorption and heat generation in various conditions in a short period of time. On the other hand, the theoretical/analytical investigation provides fundamental insight and underlying physics behind a phenomenon. Analytical solutions provide closed-form mathematical expressions of electromagnetic power absorption and heat generation that relates functional relationships among various parameters. It also provides benchmark solutions that can be used to test the accuracy of numerical and computer modeling and simulation. This thesis works presents an analytical expression for the electromagnetic power absorption and heat generation for spherical shaped objects subjected to electromagnetic treatment.

2.2 Literature Survey: Experimental Studies

Knowledge of accurate material properties can greatly influence the quality of microwave heating studies. Detailed dielectric measurements on liquid and solid foodstuffs were performed by E. C. To [5] to establish a predictive model for dielectric properties under the influence of microwave radiation. Several temperatures and frequencies were used in the study. It was determined that dielectric losses for the liquid under study could not be predicted from a linear, additive, composition-based model. Dielectric measurements for solid food products were also reported in various literature [5-8] . Curet et. al [6] studied the microwave heating of tylose subjected to an incident sinusoidal wave. Both experimental and numerical techniques were employed and compared different approaches of modeling microwave heat generation such as Maxwell's equations and Lambert's law. Estimates of dielectric properties for the frozen phase were deduced and gave comparable results between numerical results and experimental data. Resonance phenomena in the frozen state could only be predicted using Maxwell's equations, while good agreement could be found between Maxwell's equations, Lambert's law, and experimental data for thin, non-frozen objects. This study provided the criteria of different approaches of modeling to predict accurate power absorption, heat generation and temperature distribution of frozen and non-frozen objects by comparing experimental results[6]. Distribution of power absorption is uneven in microwave heating of specimens which facilitates non-uniform temperature distribution. Goksoy [7] investigated the ability of microwave ovens and use of various shielding techniques to achieve sufficiently uniform surface temperatures on pieces of poultry meat. The goal of the study was to reduce numbers of surface bacteria without significantly changing the original texture of the poultry. The study concluded that such a surface treatment produced unreliable results. Kelen et. al [9] mapped the heat distribution in corn starch-based granule layers for the purposes of quantitatively evaluating and optimizing the even distribution of microwave energy to facilitate higher quality pharmaceutical microwave vacuum drying. Results show that regulating the hottest areas during the drying process can optimize the distribution of heat and avoid over-heating the hottest regions in the layers. The continuous and pulsed microwave heating techniques had also been investigated to provide uniformity in heat generation and temperature distribution. It is reported that the pulsed microwave heating techniques can provide more uniform heat distribution for certain conditions[10]. Gunasekaran [10] performed temperature distribution studies with agar gel cylinders heated with a microwave oven. Results indicated when the same average power level settings were applied with both pulsed and continuous microwave heating, the temperature distribution was more uniform under pulsed heating, as can be seen for a certain case in Figure 2.3.

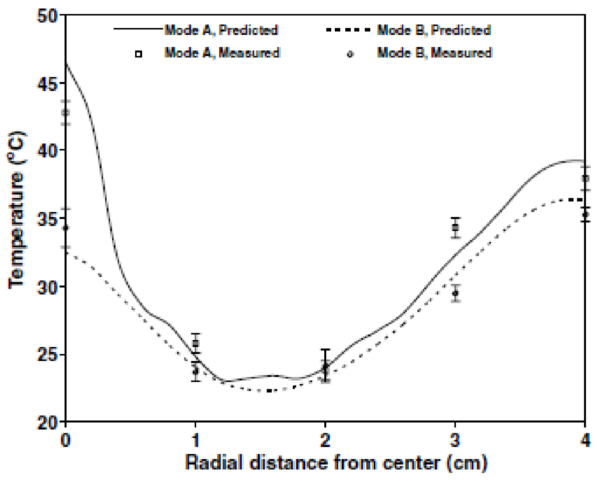

Figure 2.3: Temperature distribution in 2% agar gel cylinders, 4 cm radius, after 4 min of heating by using an average microwave absorbed power of 225 W under continuous (Mode A) and pulsed (Mode B) microwave applications.[10]

The experimental studies show that the electromagnetic heating and temperature distributions are dependent on electromagnetic power absorption which is a complex function of physical and dielectric properties of the target object[7-11].

2.3 Literature Survey: Numerical Studies

Plane wave scattering studies using computers and numerical methods such as Finite Element Methods / Finite Element Analysis can be found as far back as the 1970's. Bussey [12] outlined a theoretical scattering solution for a plane wave irradiating the side of a lossy multilayer dielectric cylinder of infinite length. Numerical values of the modal scattering coefficient for TE and TM modes are given for several single and multilayer cylinders. To allow for the study of dielectrics of complex shape, Chang [13] made use of the unimoment method to calculate the scattered fields of dielectric cylinders of inhomogeneous materials and/or arbitrary cross sections [Figure 2.4].

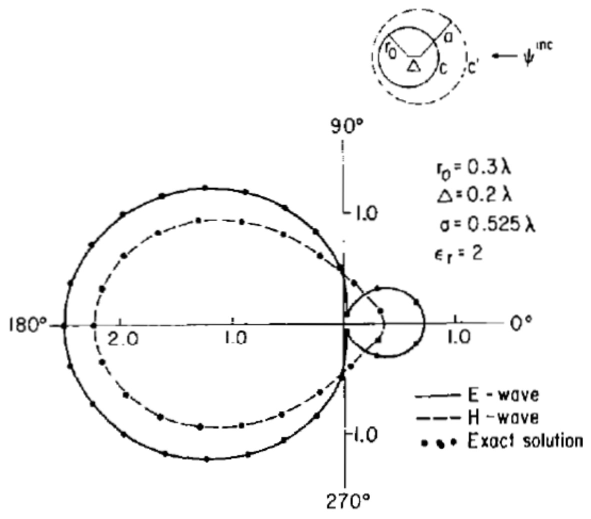

Figure 2.4: Amplitude of E-wave and H-wave scattering far-field pattern for off-centered circular cylinder.[13]

Use of this method involves finite element analysis inside a mathematical circle enclosing the inhomogeneous body, with the goal of greater simplicity and efficiency in programming. Ayappa et. al [14] made use of Galerkin finite elements to investigate power absorption profiles in homogeneous, isotropic, multilayered slabs irradiated with microwave plane waves from opposite sides. An expression for critical slab thickness in which Lambert's law can be substituted for Maxwell's equation in the transient heat equation was determined. For slab thicknesses above the critical thickness value, temperature profiles were within 0.5% of those predicted by Maxwell's equation. To conserve computational resources, and to increase the accuracy of the solutions within the dielectric, Ayappa et. al [15] utilized Galerkin finite elements to predict the distribution of temperature in cylindrical and square dielectric rods exposed to incident plane waves. The numerical approach employed a radiation boundary condition (RBC) to limit the total domain of the analysis. Temperature dependent dielectric properties were incorporated into the modeling. Results showed that, for cylindrical rods, the distribution of power is a strong function of the cylinder's radius. For square rods, the areas of greatest power generation tended to be in the middle, and the corners. When both the cylindrical and square dielectric rods where thicker, a greater effect of heating could be seen on the side/face incident to the irradiating plane wave. Figure 2.5 illustrates this effect.

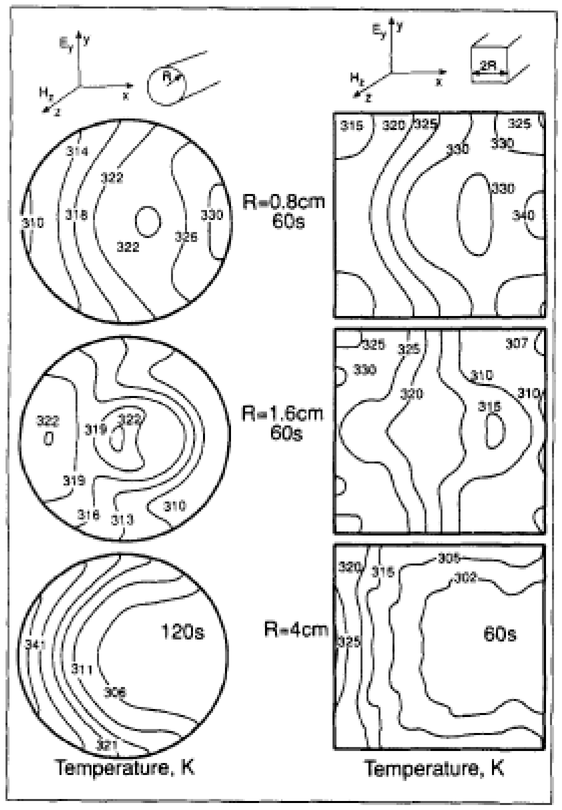

Figure 2.5: Comparison of temperature contours for cylindrical and square rods exposed to TE^z polarized plane waves.[15]

Thermal runaway was observed for materials with high dielectric loss. The polarization of the incident wave also influenced the temperature distribution, for instance, TM^z polarization giving a more pronounced heating effect than TE^z polarization. The utilization of radio frequency waves was also investigated to minimize the non-uniformity of microwave heating. Oliveira [16] performed numerical studies on the distribution of thermal energy within foodstuffs by solving Maxwell's equation and incorporating the solution as a source term in the transient heat equation. The finite element method was used in the simulations. Results showed that the sample size and shape had a significant effect on power distribution and heating within the sample. The radiation penetration was more effective at lower frequencies as opposed to higher frequencies. Romano [17] developed a 3D multi-physics mathematical model to model the microwave heating rate and power distribution within 3D foodstuffs of cubical, cylindrical, and spherical shapes. Numerical modeling was employed, using a dual microwave source with 180 degrees between the emitters. Temperature dependent dielectric and physical properties were employed in the model. Results showed that the shape of objects had a significant effect on the distribution of heat and power within the sample [Figure 2.6]. The cubic shape exhibits fast, uniform heating and good absorption of power, while cylinders responded better when the ends were placed 90 degrees with the incoming radiation. Spherical objects responded the least favorably. Numerical studies also revealed that temperature distribution and heat generation are mostly dependent on the electromagnetic power absorption, which is a complex function of electro-physical parameters of the target object.

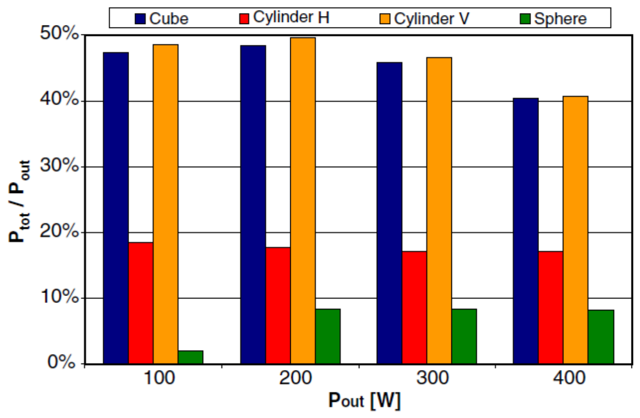

Figure 2.6: Time-average total absorbed power (P_{tot}) by each sample after 900 s of heating, with respect to output power (P_{out}), of 100, 200, 300 and 400 W.[17]

2.4 Literature Survey: Analytical Studies

A variety of analytical solutions have been attempted in 1-D, 2-D, and 3-D configurations, in the Cartesian and Cylindrical coordinate systems. For 1-D dielectric systems modeled in rectangular coordinate systems, Nachman et. al [18] studied the heating patterns of muti-layered slabs irradiated by microwaves.

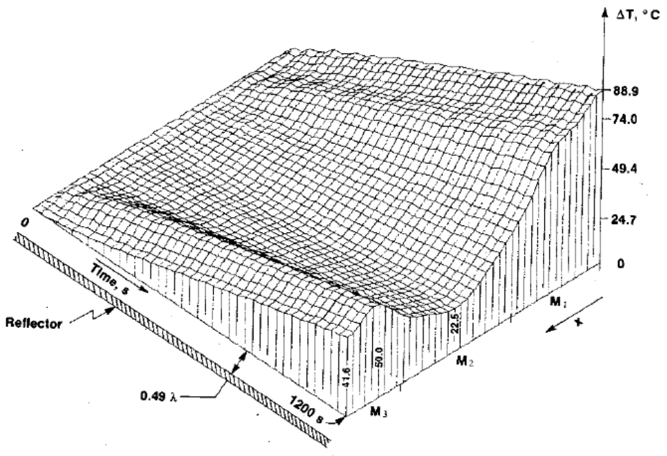

Figure 2.7: Evolution in time of the temperature profile in a thermally insulated three-layered material in the presence of a reflector (L = 0.49λ).[18]

A general expression was derived for volumetric power absorption taking into account transmissions and reflections at the interfaces. A numerical method was used to determine the temperature distribution within the material. Both Dirichlet and Neumann-type boundary conditions were considered. The special case where cooling fluid is circulated inside the slabs was also investigated and was found to significantly alter the heating pattern. It was also found that placing a reflector after the slab and opposite the EM source could alter the temperature distribution within the slab. The region between the slab and the reflector was heated more prominently. In another paper, Fleischman [19] utilized integral transformation techniques to obtain a closed form 1-D solution to the heat equation for short-term/high-power microwave heated food slabs. When using a simplified closed form solution, better uniformity in temperature distribution was obtained when microwave processing time was limited to 40 seconds. Results from the full form solution showed temperature variation within beef slabs to be most sensitive to slab width and heating time. Slabs of 1-3 cm width interval showed the greatest extremes. Lastly, the least variation in temperature values for slabs were observed in the 3-4 cm width interval. Finally, Mani et al. [20] developed a mathematical model for investigating the bonding of multiple PMMA slabs using microwave heated dielectric material. Maxwell's equation was used to map electric field distribution under plane wave configuration. The Poynting theorem was used to volumetrically find the power absorbed by each layer. Results showed dielectric properties, layer thickness, heat transfer coefficient and processing time have great influence on the heating pattern. Yang [11] compared the predicted radial temperature distribution in a 2D cylindrical shaped model food object using finite-difference models based on Maxwell's equations and Lambert's law [Figure 2.8].

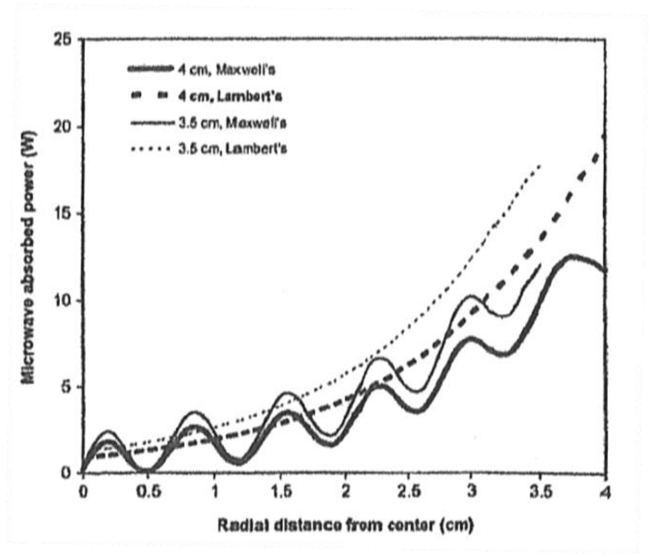

Figure 2.8: Maxwell's and Lambert's model predicted microwave power absorbed during continuous heating of 2% agar gel cylinders (3.5 and 4.0 cm radius) as a function of radial distance for sample center. The electric field is oriented along the vertical z-axis of the cylinder.[11]

The microwave power absorption and temperature distribution were compared with experimental data gathered from microwave heated agar gel cylinders. Results indicated that power-absorption efficiency increased as sample volume increased. Ignoring edge effects, there was also no appreciable variation in temperature along the longitudinal direction of the cylinder. The power absorption results derived from Maxwell's equation showed nodal/anti-nodal wave interactions of the incoming microwave radiation, while Lambert's law results demonstrated exponential decay. The study suggests Maxwell's equations predict temperature generation within the sample most accurately. It was also shown that pulsed microwave heating allows for more uniform heating. Hossan et al.[21] reported microwave power absorption and temperature distribution in 3D cylindrical object subject to microwave radiation.

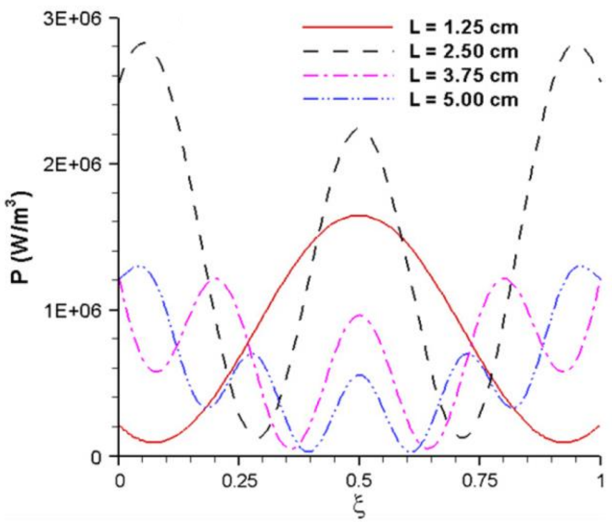

Figure 2.9: Absorbed power distribution along centerline of a cylindrical foodstuff for f=2450 MHz.[21]

An analytical solution for both microwave power absorption and temperature distribution was presented. Such parameters as cylinder length, diameter, heat transfer coefficient, and the frequency of the incoming microwave radiation were varied to study their specific effects on the temperature distribution inside the material. It was determined that [21]: 1. The electric field distribution within the food sample closely mirrored power generation. 2. The length of the food cylinder had a significant effect on the temperature distribution in the material. 3. The change in temperature in the radial direction of the cylinder is significantly affected by the heat transfer coefficient. 4. No clear trend with axial temperature distribution as cylinder length is varied. 5. Internal heat generation exceeds heat lost to the ambient environment. 6. An optimum length may exist for a frequency and food type where the thermodynamic efficiency of the heating system is at an optimum. Lately, Hossan et al. [22] investigated the effects of temperature dependent properties in a three dimensional rectangular food slab undergoing microwave and radio frequency heating. A closed form solution to the temperature distribution within the object was determined by solving Maxwell's equation and utilizing integral transformation techniques. It was found that incident frequency, sample thickness, and processing time have significant influence on the heating pattern. Radio frequency electromagnetic wave radiation provides more uniform heat generation and hence the overall uniformity of the heating within the sample was improved.

2.5 Research Goals

Despite numerous experimental and numerical studies on electromagnetic power absorption and heating, there are only a few handful of pure analytical works that can provide some theoretical insight about the mechanism of this technology. Moreover, to the best of the Author’s knowledge, there are no studies available that present analytical closed-form solutions of electromagnetic power absorption for 3D spherical shaped objects. Therefore, the goal of this thesis is to find a closed-form expression of electromagnetic field and power absorption distribution in a spherical shaped object subjected to planar impingement of electromagnetic radiation. The incoming electromagnetic waves are modeled as transverse electric and magnetic (TEM) waves. The three dimensional Maxwell’s equation in spherical coordinate for TEM wave is solved using vector potentials, and separation of variables. The electromagnetic field distribution and power absorption are plotted for different sizes of beef nugget. The theory, governing equation, assumptions and model domains are described in the next chapter.

References

1. Balanis CA. *Advanced engineering electromagnetics.* John Wiley & Sons; 2012.
2. Balanis CA. Transverse Electric Modes: Source-Free Region. *Advanced Engineering Electromagnetics*. 2nd ed. Hoboken, NJ: John Wiley and Sons; 2012:276.
3. Balanis CA. Transverse Magnetic Modes: Source-Free Region. *Advanced Engineering Electromagnetics*. 2nd ed. Hoboken, NJ: John Wiley and Sons; 2012:272-273.
4. Balanis CA. Transverse Electromagnetic Modes: Source-Free Region. *Advanced Engineering Electromagnetics*. 2nd ed. Hoboken, NJ: John Wiley and Sons; 2012:265.
5. To EC. Dielectric Properties of Food Materials. *Journal of Microwave Power.* 1974;9(4):303-315.
6. Curet S, Rouaud O, Boillereaux L. Microwave tempering and heating in a single-mode cavity: Numerical and experimental investigations. *Chemical Engineering and Processing: Process Intensification.* 2008;47(9-10):1656-1665.
7. Goksoy EO. Non-Uniformity of Surface Temperatures After Microwave Heating of Poultry Meat. *International Microwave Power Institute.* 1999;34(3):149-160.
8. BARRINGER SA, DAVIS EA, GORDON J, AYAPPA KG, DAVIS H. Microwave-Heating Temperature Profiles for Thin Slabs Compared to Maxwell and Lambert Law Predictions. *Journal of Food Science.* 1995;60(5):1137-1142.
9. Kelen Á, Ress S, Nagy T, Pallai E, Pintye-Hódi K. Mapping of temperature distribution in pharmaceutical microwave vacuum drying. *Powder Technology.* 2006;162(2):133-137.
10. Gunasekaran S. Effect of experiemental parameters on temperature distribution during continuous and pulsed microwave heating. *Journal of Food Engineering.* 2006;78:1452-1456.
11. Yang HWaG, S. Comparison of temperature distribution in model food cylinders based on Maxwell's equations and Lambert's law during pulsed microwave heating. *Journal of Food Engineering.* 2004:445-453.
12. Howard E. Bussey JHR. Scattering by a Lossy Dielectric CIrcular Cylindrical Multilayer, Numerical Values. *IEEE Transactions on Antennas and Propagation.* 1975:723-725.
13. Shu-Kong Chang KKM. Application of the Unimoment Method to Electromagnetic Scattering of Dielectric Cylinders. *IEEE Transactions on Antennas and Propagation.* 1976;24(1):35-42.
14. Ayappa KG, Davis HT, Crapiste G, Davis EA, Gordon J. Microwave heating: an evaluation of power formulations. *Chemical Engineering Science.* 1991;46(4):1005-1016.
15. K. G. Ayappa HTD, E. A. Davis, J. Gordon. Two-Dimensional Finite Element Analysis of Microwave Heating. *AlChE Journal.* 1992:1577-1592.
16. M. E. C. Oliveira ASF. Microwave heating of foodstuffs. *Journal of Food Engineering.* 2001;53(6):347-359.
17. Romano V, Marra F. A numerical analysis of radio frequency heating of regular shaped foodstuff. *Journal of Food Engineering.* 2008;84(3):449-457.
18. M. Nachman GT. Heating Pattern in a Multi-layered Material Exposed to Microwaves. *IEEE Transactions on Microwave THeory and Techniques.* 1984;32(5):547-552.
19. Fleischman GJ. Predicting temperature range in food slabs undergoing short-term/high-power microwave heating. *Journal of Food Engineering.* 1999;40(1999):81-88.
20. Kasi Balamurugan Mani MRH, Prashanta Dutta. Thermal analysis of microwave assisted bonding of poly(methyl methacrylate) substrates in mircofluidic devices. *International Journal of Heat and Mass Transfer.* 2013;58:229-239.
21. Hossan MR, Byun D, Dutta P. Analysis of microwave heating for cylindrical shaped objects. *International Journal of Heat and Mass Transfer.* 2010;53(23):5129-5138.
22. M. R. Hossan PD. Effects of temperature dependent properties in electromagnetic heating. *International Journal of Heat and Mass Transfer.* 2012;55:3412-3422.

Chapter 3:

Theory

Generally, electromagnetic heating uses electromagnetic waves within the range of 3 kHz-300 GHz. Radiofrequency heating is in the range of 3 kHz-300 MHz, and microwave heating is in the 300 MHz-300 GHz range. Most materials that respond to radiofrequency or microwave heating contain polar molecules such as water and molecules of low heat capacity such as fat and oil. Polar molecules are molecules that have a positive charge in one side (or pole) of the molecule and have a negative charge on the other side (or pole). When a material composed of polar and/or low heat capacity molecules is subjected to an electromagnetic field in the radio/microwave frequency, the molecules start rotating in the attempt to align with the direction of the incoming electromagnetic field. This rotation creates friction (and therefore heat) in the material. This heat generation and its spatial distribution are dependent on the electromagnetic field distribution within the target object. Hence the electromagnetic heat generation can be explained by examining the electromagnetic field distribution within the target object. The electromagnetic field is governed by Maxwell's equation and the following section presents Maxwell's equation, relevant assumptions, and Poynting theorem for evaluating electromagnetic power absorption or heat generation.

3.1 Governing Equation

Maxwell's equations govern the electromagnetic field distribution within the material, as given below [1]

$$\vec{\nabla} \times \vec{\mathscr{E}} = \vec{\mathscr{M}} - \frac{\partial \vec{\mathscr{B}}}{\partial t} \tag{3-1a}$$

$$\vec{\nabla} \times \vec{\mathcal{H}} = \vec{\mathcal{J}} + \frac{\partial \vec{\mathcal{D}}}{\partial t} \tag{3-1b}$$

$$\vec{\nabla} \cdot \vec{\mathcal{D}} = q_{ev} \tag{3-1c}$$

$$\vec{\nabla} \cdot \vec{\mathcal{B}} = q_{mv} \tag{3-1d}$$

where $\vec{\mathcal{E}}$ is the electric field intensity, $\vec{\mathcal{M}}$ is the magnetic current density, $\vec{\mathcal{B}}$ is the magnetic flux density, $\vec{\mathcal{H}}$ is the magnetic field intensity, $\vec{\mathcal{J}}$ is the electric current density, $\vec{\mathcal{D}}$ is the electric flux density, q_{ev} is the electric charge density, and q_{mv} is the magnetic charge density. The time harmonic version of Maxwell's equations, i.e. using time-harmonic electromagnetic waves, is convenient to represent high frequency (radio and microwave) electromagnetic heating. Many practical systems lend well to the time-harmonic formulation, where the time variation is of cosinusoidal form, represented in this work by $e^{i\omega t}$. Using the time-harmonic formulation, the instantaneous representations of Maxwell's equation can be related to their complex forms by the following expressions

$$\vec{\mathcal{E}}(x,y,z;t) = \mathrm{Re}[\vec{E}(x,y,z)e^{i\omega t}] \tag{3-2a}$$

$$\vec{\mathcal{H}}(x,y,z;t) = \mathrm{Re}[\vec{H}(x,y,z)e^{i\omega t}] \tag{3-2b}$$

$$\vec{\mathcal{D}}(x,y,z;t) = \mathrm{Re}[\vec{D}(x,y,z)e^{i\omega t}] \tag{3-2c}$$

$$\vec{\mathcal{B}}(x,y,z;t) = \mathrm{Re}[\vec{B}(x,y,z)e^{i\omega t}] \tag{3-2d}$$

$$\vec{\mathcal{J}}(x,y,z;t) = \mathrm{Re}[\vec{J}(x,y,z)e^{i\omega t}] \tag{3-2e}$$

$$\vec{\mathcal{M}}(x,y,z;t) = \mathrm{Re}[\vec{M}(x,y,z)e^{i\omega t}] \tag{3-2f}$$

$$q_{ev}(x, y, z; t) = \text{Re}[q_{\varepsilon v}(x, y, z)e^{i\omega t}] \quad \text{(3-2g)}$$

$$q_{mv}(x, y, z; t) = \text{Re}[q_{mv}(x, y, z)e^{i\omega t}] \quad \text{(3-2h)}$$

where $\vec{E}$ is the electric field intensity, $\vec{M}$ is the magnetic current density, $\vec{B}$ is the magnetic flux density, $\vec{H}$ is the magnetic field intensity, $\vec{J}$ is the electric current density, $\vec{D}$ is the electric flux density, q_{ev} is the electric charge density, and q_{mv} is the magnetic charge density, all in complex spatial form. *x*, *y*, and *z* represent the three spatial dimensions, *t* represents time, ω is angular frequency and *i* represents the imaginary number. With substitution, and differentiation for (1a-1b), equations (1a-1d) take on the following form [1]

$$\vec{\nabla} \times \vec{E} = \vec{M} - i\omega\vec{B} \quad \text{(3-3a)}$$

$$\vec{\nabla} \times \vec{H} = \vec{J} + i\omega\vec{D} \quad \text{(3-3b)}$$

$$\vec{\nabla} \cdot \vec{D} = q_{ev} \quad \text{(3-3c)}$$

$$\vec{\nabla} \cdot \vec{B} = q_{mv} \quad \text{(3-3d)}$$

The knowledge of the electromagnetic field is utilized to find the power generation using Poynting theorem. In the following work, Poynting theorem for determining power generation within the material is given by [2]

$$q_{av} = \frac{1}{2}\text{Re}[\vec{E} \times \vec{H}^*] \quad \text{(3-4a)}$$

$$Q_{gen} = \text{Re}[-\vec{\nabla} \cdot (\frac{1}{2}\vec{E} \times \vec{H}^*)] \quad \text{(3-4b)}$$

Where q_{av} is the time-average Poynting vector (average power density) over one period, $\vec{H}^*$ is the complex conjugate of the magnetic field, and Q_{gen} is the generated power in the material.

References

1. Balanis CA. *Advanced Engineering Electromagnetics*. Hoboken, NJ: John Wiley & Sons; 2012:21-22.
2. Meredith RJ. *Engineers' handbook of industrial microwave heating.* IET; 1998.

Chapter 4:

Solution Methodology

Maxwell's equation presented in Chapter 3 governs the general electromagnetic waves. However, when it comes to a specific application of heat generation through radiofrequency and microwave radiation, the consideration of material properties, material constitutive laws, and boundary conditions are critical for finding a solution of the electromagnetic fields within the specimen. Using material constitutive laws and consideration of heating conditions, Maxwell's equation can be simplified into a single partial differential equation. The following sections provides necessary assumptions, boundary conditions and solution methodology.

4.1 Assumptions

In this study, a spherical object of homogeneous, dielectric construction is subjected to electromagnetic heating. Uniform plane waves, also known as transverse electromagnetic (TEM) waves, are utilized to model the incoming electromagnetic (EM) radiation. While the TEM waves generated in a real system are not usually uniform, such a simplification allows an analytical study to be carried out, and gives results that closely approximate the behavior of a real microwave oven. The following assumptions are made for this study

(i) Food system is linear and follows linear material constitutive laws.

(ii) The system satisfies the electroneutrality condition.

(iii) Dielectric properties are temperature independent.

(iv) The incident EM radiation are uniform TEM waves.

(v) Material properties are temperature independent

4.2 Boundary Conditions

In this work, the impingent of uniform TEM plane waves propagate in the +z and –z directions. The waves are transformed from a Cartesian coordinate system representation to a spherical coordinates form so that the target spherical object is equivalently exposed to electromagnetic radiation radially in all directions. For a lossy dielectric sphere, continuity of the tangential electric and magnetic fields are required [1]. Therefore boundary conditions are as follows

$$E_\theta^{t-}(r=a, 0\le\theta\le\pi, 0\le\phi\le 2\pi) = E_\theta^{t+}(r=a, 0\le\theta\le\pi, 0\le\phi\le 2\pi) \quad \text{(4-1a)}$$

$$E_\phi^{t-}(r=a, 0\le\theta\le\pi, 0\le\phi\le 2\pi) = E_\phi^{t+}(r=a, 0\le\theta\le\pi, 0\le\phi\le 2\pi) \quad \text{(4-1b)}$$

$$H_\theta^{t-}(r=a, 0\le\theta\le\pi, 0\le\phi\le 2\pi) = H_\theta^{t+}(r=a, 0\le\theta\le\pi, 0\le\phi\le 2\pi) \quad \text{(4-1c)}$$

$$H_\phi^{t-}(r=a, 0\le\theta\le\pi, 0\le\phi\le 2\pi) = H_\phi^{t+}(r=a, 0\le\theta\le\pi, 0\le\phi\le 2\pi) \quad \text{(4-1d)}$$

where E_θ^{t-} represents the theta component of the electric wave inside the sphere, E_θ^{t+} represents the theta component of the electric wave outside the sphere, E_ϕ^{t-} represents the phi component of the electric wave inside the sphere, E_ϕ^{t+} represents the phi component of the electric wave outside the sphere H_θ^{t-} represents the theta component of the magnetic wave inside the sphere, H_θ^{t+} represents the theta component of the magnetic wave outside the sphere, H_ϕ^{t-} represents the phi component of the magnetic wave inside the sphere, H_{ϕ}^{t+} represents the phi component of the magnetic wave outside the sphere, and *a* is the outer radius of the sphere.

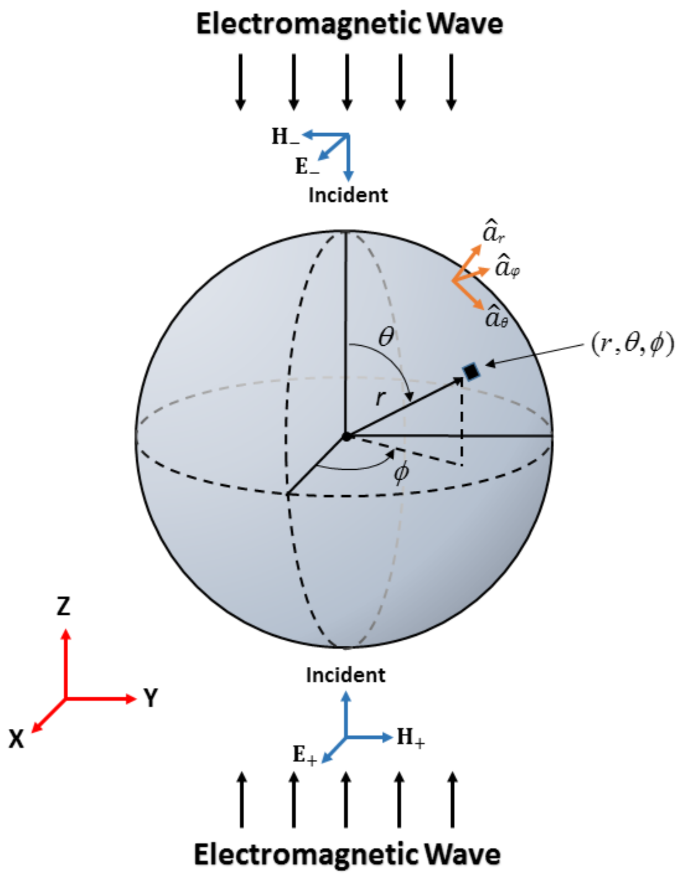

Figure 4.1: Plane wave and dielectric sphere system as modeled in research.

4.3 Analysis of Wave Equation in Spherical Object

Consider a lossy dielectric spherical object subjected to TEM electromagnetic radiation as shown in the Figure 4.1. Based on the assumptions mentioned in Section 4.1, the object under study is electrically neutral (source free) i.e. $\vec{J} = \vec{M} = q_{ev} = q_{mv} = 0$ and the following material constitutive relations are employed[2,3]

$$\vec{D} = \varepsilon \vec{E} \tag{4-2a}$$

$$\vec{B} = \mu\vec{H} \tag{4-2b}$$

Where μ is permeability, and ε is permittivity. Maxwell's equations take on the following form

$$\vec{\nabla} \times \vec{E} = -i\omega\mu\vec{H} \tag{4-3a}$$

$$\vec{\nabla} \times \vec{H} = i\omega\varepsilon\vec{E} \tag{4-3b}$$

$$\vec{\nabla} \cdot \varepsilon\vec{E} = 0 \tag{4-3c}$$

$$\vec{\nabla} \cdot \mu\vec{H} = 0 \tag{4-3d}$$

Where $i = \sqrt{-1}$ is an imaginary number, and ω is angular frequency. Two auxiliary functions known as *vector potentials*: $\vec{A}$ (magnetic vector potential), and $\vec{F}$ (electric vector potential), are employed in finding a solution [4]. To utilize vector potentials, it is helpful to define separate $\vec{A}$ and $\vec{F}$ vector potential components for the $\vec{E}$ and $\vec{H}$ fields. Using vector identities and Lorenz conditions, Maxwell's equations can be expressed in terms of vector potentials as follows

$$\vec{\nabla}^2\vec{A} + \beta^2\vec{A} = 0 \tag{4-4a}$$

$$\vec{\nabla}^2\vec{F} + \beta^2\vec{F} = 0 \tag{4-4b}$$

Where β represents phase constant, and $\beta^2 = \omega^2\mu\varepsilon$. Vector potentials are usually considered strictly mathematical tools, even though the resulting electromagnetic radiated fields ($\vec{E}$, $\vec{H}$) represent physically measurable quantities. The vector potentials ($\vec{A}$ and $\vec{F}$) are defined such a way that each of this vector has both electric and magnetic field component. In other words, the total electric field will have contributions from the magnetic vector potential as well as the electric vector potential. Therefore total electric field and magnetic field can be found in terms of vector potentials using superposition [5] as follows

$$\vec{E} = \vec{E}_A + \vec{E}_F \tag{4-5a}$$

$$\vec{H} = \vec{H}_A + \vec{H}_F \tag{4-5b}$$

where $\vec{E}_A$ is the electrical component of vector potential $\vec{A}$, $\vec{H}_A$ is the magnetic component of vector potential $\vec{A}$, $\vec{E}_F$ is the electric component of vector potential $\vec{F}$ and $\vec{H}_F$ is the magnetic component of vector potential $\vec{F}$. The vector potential component is given by [5]

$$\vec{E}_A = -i\omega\vec{A} - i\frac{1}{\omega\mu\varepsilon}\vec{\nabla}(\vec{\nabla}\cdot\vec{A}) \tag{4-6a}$$

$$\vec{H}_A = \frac{1}{\mu}\vec{\nabla}\times\vec{A} \tag{4-6b}$$

$$\vec{E}_F = -\frac{1}{\varepsilon}\vec{\nabla}\times\vec{F} \tag{4-6c}$$

$$\vec{H}_F = -i\omega\vec{F} - i\frac{1}{\omega\mu\varepsilon}\vec{\nabla}(\vec{\nabla}\cdot\vec{F}) \tag{4-6d}$$

The solution of electric and magnetic field (equation 4-5a and 4-5b) through vector potentials can be further simplified using vector identities and substituting $\vec{\nabla}\times\vec{E}_A = -i\omega\mu\vec{H}_A$ and $\vec{\nabla}\times\vec{E}_F = -i\omega\mu\vec{H}_F$ as follows [6]

$$\vec{E} = -\frac{1}{\varepsilon}\vec{\nabla}\times\vec{F} + \frac{1}{i\omega\mu\varepsilon}\vec{\nabla}\times\vec{\nabla}\times\vec{A} \tag{4-7a}$$

$$\vec{H} = \frac{1}{i\omega\mu\varepsilon}\vec{\nabla}\times\vec{\nabla}\times\vec{F} + \frac{1}{\mu}\vec{\nabla}\times\vec{A} \tag{4-7b}$$

The two step sequence for determining the electromagnetic radiated fields using vector potentials are as follows: 1. $\vec{A}$ and $\vec{F}$ are determined by integration of Maxwell's equations; 2. $\vec{A}$ and $\vec{F}$ are then differentiated to arrive at the solution for the electromagnetic radiated fields ($\vec{E}$ and $\vec{H}$).

To find the expression for $\vec{A}$ and $\vec{F}$, an electric and a magnetic scalar potential functions are defined as, $\psi_e = -\frac{1}{i\omega\mu\varepsilon}\vec{\nabla}\cdot\vec{A}$ and $\psi_m = -\frac{1}{i\omega\mu\varepsilon}\vec{\nabla}\cdot\vec{F}$ respectively. With these two potential scalar functions, Maxwell's equation is rewritten as [6]

$$\vec{\nabla}\times\vec{\nabla}\times\vec{A}-\omega^2\mu\varepsilon\vec{A}=-i\omega\mu\,\varepsilon\vec{\nabla}\psi_e \qquad \text{(4-8a)}$$

$$\vec{\nabla}\times\vec{\nabla}\times\vec{F}-\omega^2\mu\varepsilon\vec{F}=-i\omega\mu\varepsilon\,\vec{\nabla}\psi_m \qquad \text{(4-8b)}$$

Considering TE^r and TM^r modes separately allows ψ_m and ψ_e to be determined in terms of $F_r(r,\theta,\phi)$ and $A_r(r,\theta,\phi)$, respectively [6]. Since propagation is happening in the radial direction, utilizing the *r* components of eqns (4-8a) and (4-8b), the following relations are obtained [6]

$$(\vec{\nabla}^2+\beta^2)\frac{F_r}{r}=0 \qquad \text{(4-9a)}$$

$$(\vec{\nabla}^2+\beta^2)\frac{A_r}{r}=0 \qquad \text{(4-9b)}$$

Solutions to F_r and A_r are found by separation of variables [6] and can be expressed

$$F_r(r,\theta,\phi)=A_r(r,\theta,\phi)=f(r)g(\theta)h(\phi) \qquad \text{(4-10)}$$

where $f(r)$, $g(\theta)$, and $h(\phi)$ must be represented by appropriate wave functions that satisfy the wave equation in spherical coordinates. Solutions to these functions take on the following forms [6]

$$f_1(r) = \mathscr{A}_1 \hat{J}_n(\beta r) \tag{4-11a}$$

$$f_2(r) = B_1 \hat{H}_n^{(2)}(\beta r) \tag{4-11b}$$

$$g(\theta) = C_1 P_n^m(\cos\ \theta) \tag{4-11c}$$

$$h(\phi) = C_2 \cos(\ m\phi) + D_1 \sin(\ m\phi) \tag{4-11d}$$

Where $\mathscr{A}_1$, B_1, C_1, C_2, and D_1 represent arbitrary constants; $\hat{J}_n$ is an alternate form of spherical Bessel functions of the 1st kind, of order *n*, respectively; $\hat{H}_n^{(2)}$ are also an alternate form of spherical Hankel functions of the 2nd kind, of order *n*, respectively; P_n^m is the associated Legendre function of the 1st kind of order *m* and degree *n*, respectively; $C_2\cos\ (m\phi)$ and $D_1\sin(m\phi)$ are “cosinusoids” of order *m,* and m and n are whole, positive integers. Depending on the region the waveform is being modeled, solutions to F_r and A_r take on the following two forms

$$F_r(r,\theta,\phi) = A_r(r,\theta,\phi) = f_1(r)g(\theta)h(\phi) \tag{4-12a}$$

for the incident portion of the wave, and for when the waveform is inside the object

$$F_r(r,\theta,\phi) = A_r(r,\theta,\phi) = f_2(r)g(\theta)h(\phi) \tag{4-12b}$$

for the reflected portion of the incident wave. The spherical Bessel and Hankel functions represented in eqns (4-11a) and (4-11b) can be related to regular Bessel and Hankel functions as follows [6]

$$\hat{B}_n(\beta r) = \beta r b_n(\beta r) = \beta r\sqrt{\frac{\pi}{2\beta r}} B_{n+1/2}(\beta r) = \sqrt{\frac{\pi\beta r}{2}} B_{n+1/2}(\beta r) \tag{4-13}$$

where B_n represents J_n, or $H_n^{(2)}$. This alternative form of the spherical Bessel and Hankel functions satisfy the differential equation below [6]

$$\left[\frac{d^2}{dr^2}+\beta^2-\frac{n(n+1)}{r^2}\right]\hat{B}_n=0 \tag{4-14}$$

4.4 Closed Form Electric and Magnetic Expression for a Solid Sphere

The methodology for determining the electric and magnetic field distributions and power generation within a dielectric sphere involves three main sequences: 1. Making use of vector potentials to find solutions for F_r and A_r [5]; 2. Using the solutions for F_r and A_r to determine the three special components (r,θ,ϕ) of the electric $\vec{E}(r,\theta,\phi)$ and magnetic $\vec{H}(r,\theta,\phi)$ fields ; 3. Utilizing the *Poynting theorem* and the *conservation-of-energy equation* to determine power generation [7]. The physical model used in the research is a dual source uniform TEM plane wave as shown in Figure 4.1, with one source situated directly above the dielectric sphere, and the second source situated below. The electric component of both EM waves are situated along the positive x-axis. The magnetic component of the upper EM source is directed along the negative component of the y-axis, and the magnetic component of the lower EM source is directed along the positive y-axis. As uniform plane waves are natively represented in the rectangular coordinate system, a transformation to spherical coordinates will be utilized to allow their use in the following derivations. An infinite sum of spherical wave functions will be used to represent the electromagnetic plane waves [1]

$$E_x^+=e^{-i\beta z}=e^{-i\beta r\cos\theta}=\sum_{n=0}^{\infty}a_n j_n(\beta r)P_n(\cos\theta) \tag{4-15a}$$

$$E_x^-=e^{i\beta z}=e^{i\beta r\cos\theta}=\sum_{n=0}^{\infty}b_n j_n(\beta r)P_n(\cos\theta) \tag{4-15b}$$

where E_x^+ represents the lower plane wave source with the amplitude of the electric component of the EM wave polarized in the x-direction, E_x^- represents the upper plane wave source with the amplitude of the electric component of the EM wave polarized in the x-direction, *r* is the radial distance from the origin of the spherical polar coordinate system, θ is an angle *0* to π radians as measured from the +z axis, $a_n = i^{-n}(2n+1)$, $b_n = i^n(2n+1)$, $j_n(\beta r)$ is a spherical Bessel function of the 1st kind of order *n*, $P_n(\cos\ \theta)$ is a Legendre polynomial of order *n*, with *cos* θ varying between *-1* and *1*, and n is a positive integer. The polarized electric fields of the upper and lower incident uniform plane waves are expressed in the *r* component within the sphere as follows

$$E_r^{in} = E_o \frac{\cos\phi}{i\beta r}\left[\frac{\partial}{\partial\theta}\left(e^{-i\beta r\cos\theta} - e^{i\beta r\cos\theta}\right)\right] \tag{4-16}$$

where E_o is the amplitude of the electric field, *r* is the radial distance from the origin of the spherical polar coordinate system, θ is an angle *0* to π radians as measured from the +z axis, ϕ is an angle *0* to *2*π radians as measured from the +x axis. Utilizing eqns (4-15a, 4-15b, and 4-16), the *r* component of the incident electric field can be written as follows

$$E_r^{in} = iE_o \frac{\cos\phi}{(\beta r)^2}\sum_{n=1}^{\infty} c_n \hat{J}_n(\beta r) P_n^1(\cos\theta) \tag{4-17}$$

where $c_n = b_n - a_n$, E_o is the amplitude of the electric field, *r* is the radial distance from the origin of the spherical polar coordinate system, $P_n^1(\cos\theta)$ is an associated Legendre function of order 1 and degree *n*, with *cos* θ varying between *-1* and *1,* θ is an angle *0* to π radians as measured from the +z axis, ϕ is an angle *0* to *2*π radians as measured from the +x axis, and *n* is a positive integer.

A similar solution methodology yields the following equations for the *r* component of the incident wave of the magnetic field

$$H_r^{in} = -iH_o \frac{\sin\phi}{(\beta r)^2} \sum_{n=1}^{\infty} d_n \hat{J}_n(\beta r) P_n^1(\cos\theta) \tag{4-18a}$$

$$H_o = \frac{E_o}{\eta} \tag{4-18b}$$

where $d_n = a_n + b_n$, H_o is the amplitude of the magnetic field, *r* is the radial distance from the origin of the spherical polar coordinate system, $\eta = \sqrt{\frac{\mu}{\varepsilon}}$. A_r^{in} is obtained by equating (4-17) with (4-7a) and considering only TM^r modes ($\vec{A} = \hat{a}_r A_r(r,\theta,\phi)$ and $\vec{F} = 0$)

$$A_r^{in} = -\frac{E_o \cos\phi}{\omega} \sum_{n=1}^{\infty} c_n \hat{J}_n(\beta r) P_n^1(\cos\theta) \tag{4-19}$$

where $c_n = b_n - a_n$, E_o is the amplitude of the electric field, $\hat{J}_n$ is an alternate form of the spherical Bessel function of the 1st kind and order *n*, *r* is the radial distance from the origin of the spherical polar coordinate system, $P_n^1(\cos\theta)$ is an associated Legendre function of order 1 and degree *n*, with *cos* θ varying between *-1* and *1,* θ is an angle *0* to π radians as measured from the +z axis*,* ϕ is an angle *0* to 2π radians as measured from the +x axis, and *n* is a positive integer. F_r^{in} is obtained by equating (4-18a) with (4-7b) and considering only TE^r modes ($\vec{F} = \hat{a}_r F_r(r,\theta,\phi)$ and $\vec{A} = 0$)

$$F_r^{in} = \frac{E_o \sin\phi}{\eta\omega} \sum_{n=1}^{\infty} d_n \hat{J}_n(\beta r) P_n^1(\cos\theta) \tag{4-20}$$

where $d_n = a_n + b_n$, E_0 is the amplitude of the electric field, *r* is the radial distance from the origin of the spherical polar coordinate system, with *cos* θ varying between *-1* and *1,* θ is an angle *0* to π radians as measured from the +z axis, ϕ is an angle *0* to 2π radians as measured from the +x axis, and *n* is a positive integer. As some of the incoming EM radiation is reflected by the surface of the sphere, the reflected portion of the wave also must be considered. The reflected portions of the incoming EM radiation, in terms of magnetic and electric vector potentials, are as follows

$$A_r^s = -\frac{E_o \cos\phi}{\omega}\sum_{n=1}^{\infty} e_n \hat{H}_n^{(2)}(\beta r) P_n^1(\cos\theta) \tag{4-21a}$$

$$F_r^s = \frac{E_o \sin\phi}{\eta\omega}\sum_{n=1}^{\infty} f_n \hat{H}_n^{(2)}(\beta r) P_n^1(\cos\theta) \tag{4-21b}$$

$$\hat{H}_n^{(2)}(\beta r) = \hat{J}_n(\beta r) - i\hat{Y}_n(\beta r) \tag{4-21c}$$

where e_n and f_n will be found using appropriate boundary conditions, and *r* is the radial distance from the origin of the spherical polar coordinate system; Eqns (4-21a) and (4-21b) differ from eqns (4-19) and (4-20) by the replacement of the spherical Bessel function, $\hat{J}_n$, with the Hankel function of the second kind, $\hat{H}_n^{(2)}$, in order to represent outward traveling waves. The complete representation of the magnetic and electric vector potential, as it exists outside of the dielectric sphere, is a summation of the incident and reflected fields

$$A_r^{t+} = A_r^{in} + A_r^s = -\frac{E_o \cos\phi}{\omega}\sum_{n=1}^{\infty}(c_n \hat{J}_n(\beta r) + e_n \hat{H}_n^{(2)}(\beta r)) P_n^1(\cos\theta) \tag{4-22a}$$

$$F_r^{t+} = F_r^{in} + F_r^s = \frac{E_o \sin\phi}{\eta\omega}\sum_{n=1}^{\infty}(d_n \hat{J}_n(\beta r) + f_n \hat{H}_n^{(2)}(\beta r)) P_n^1(\cos\theta) \tag{4-22b}$$

where A_r^{t+} represents the magnetic vector potential outside of the spherical dielectric, F_r^{t+} represents the electric vector potential outside of the spherical dielectric, and *r* is the radial distance outside of the dielectric sphere, with the origin being set at the center of the spherical polar coordinate system. Considering the TE^r and TM^r modes, and making use of eqns (4-7a) and (4-7b), the spherical vector components of the electric and magnetic fields, for the space outside the sphere, can be determined to be the following

$$E_r^{t+} = \frac{1}{i\omega\mu\varepsilon}\left(\frac{\partial^2}{\partial r^2} + \beta^2\right)A_r^{t+} \tag{4-23a}$$

$$E_\theta^{t+} = \frac{1}{i\omega\mu\varepsilon}\frac{1}{r}\frac{\partial^2 A_r^{t+}}{\partial r\partial\theta} - \frac{1}{\varepsilon}\frac{1}{r\sin\theta}\frac{\partial F_r^{t+}}{\partial\phi} \tag{4-23b}$$

$$E_\phi^{t+} = \frac{1}{i\omega\mu\varepsilon}\frac{1}{r\sin\theta}\frac{\partial^2 A_r^{t+}}{\partial r\partial\phi} + \frac{1}{\varepsilon}\frac{1}{r}\frac{\partial F_r^{t+}}{\partial\theta} \tag{4-23c}$$

$$H_r^{t+} = \frac{1}{i\omega\mu\varepsilon}\left(\frac{\partial^2}{\partial r^2} + \beta^2\right)F_r^{t+} \tag{4-23d}$$

$$H_\theta^{t+} = \frac{1}{\mu}\frac{1}{r\sin\theta}\frac{\partial A_r^{t+}}{\partial\phi} + \frac{1}{i\omega\mu\varepsilon}\frac{1}{r}\frac{\partial^2 F_r^{t+}}{\partial r\partial\theta} \tag{4-23e}$$

$$H_\phi^{t+} = -\frac{1}{\mu}\frac{1}{r}\frac{\partial A_r^{t+}}{\partial\theta} + \frac{1}{i\omega\mu\varepsilon}\frac{1}{r\sin\theta}\frac{\partial^2 F_r^{t+}}{\partial r\partial\phi} \tag{4-23f}$$

E_r^{t+} and H_r^{t+} are unique from the other spherical components of the electric and magnetic fields in that the radial component of the total electric field outside the dielectric sphere depends solely on the radial component of the total magnetic vector potential outside the dielectric sphere, and the radial component of the total magnetic field outside the dielectric sphere depends solely on the radial component of the total electric vector potential outside the dielectric sphere. In the solution to follow, β in eqns (4-23a – 4-23f) will be replaced with β_o, the free space phase constant, when representing EM waves outside the sphere. For the portion of the incident EM wave that penetrates the dielectric sphere, all of the wave can be considered absorbed and so the magnetic and electric vector potentials take on simplified forms of eqns. (4-21a) and (4-21b)

$$A_r^{t-} = -\frac{E_o \cos\phi}{\omega} \sum_{n=1}^{\infty} g_n \hat{J}_n(\beta_d r) P_n^1(\cos\theta) \qquad \text{(4-24a)}$$

$$F_r^{t-} = \frac{E_o \sin\phi}{\eta_d \omega} \sum_{n=1}^{\infty} h_n \hat{J}_n(\beta_d r) P_n^1(\cos\theta) \qquad \text{(4-24b)}$$

$$\eta_d = \sqrt{\frac{\mu_d}{\varepsilon_d}} = \eta_o \sqrt{\frac{\dot{\mu}_r}{\dot{\varepsilon}_r}} \qquad \beta_d = \omega\sqrt{\mu_d \varepsilon_d} = \beta_o \sqrt{\dot{\mu}_r \dot{\varepsilon}_r} \qquad \text{(4-24c, 4-24d)}$$

$$\mu_d = \dot{\mu}_r \mu_o \qquad \varepsilon_d = \dot{\varepsilon}_r \varepsilon_o \qquad \text{(4-24e, 4-24f)}$$

$$\eta_o = \frac{\mu_o}{\varepsilon_o} \qquad \beta_o = \omega\sqrt{\mu_o \varepsilon_o} \qquad \text{(4-24g, 4-24h)}$$

$$\dot{\varepsilon}_r = \varepsilon_r' - i\varepsilon_r'' \qquad \dot{\mu}_r = \mu_r' - i\mu_r'' \qquad \text{(4-24i, 4-24j)}$$

Where A_r^{t-} represents the magnetic vector potential inside the spherical dielectric, F_r^{t-} represents the electric vector potential inside the spherical dielectric, g_n and h_n are constants to be determined by boundary conditions, β_d is the lossy dielectric phase constant, η_d is the lossy dielectric wave impedance, η_0 is the free space wave impedance, ε_0 is the free space permittivity, $\dot{\varepsilon}_r$ is the relative complex permittivity, ε_r' is the real part of the relative complex permittivity, ε_r'' is the imaginary part of the relative complex permittivity, μ_0 is the free space permeability, $\dot{\mu}_r$ is the relative complex permeability, μ_r' is the real part of the relative complex permeability, and μ_r'' is the imaginary part of the relative complex permeability. The spherical vector components of the electric and magnetic fields, for the space inside the sphere, take on a familiar form

$$E_r^{t-} = \frac{1}{i\omega\mu_d\varepsilon_d}\left(\frac{\partial^2}{\partial r^2} + \beta_d^2\right)A_r^{t-} \tag{4-25a}$$

$$E_\theta^{t-} = \frac{1}{i\omega\mu_d\varepsilon_d}\frac{1}{r}\frac{\partial^2 A_r^{t-}}{\partial r\partial\theta} - \frac{1}{\varepsilon_d}\frac{1}{r\sin\theta}\frac{\partial F_r^{t-}}{\partial\phi} \tag{4-25b}$$

$$E_\phi^{t-} = \frac{1}{i\omega\mu_d\varepsilon_d}\frac{1}{r\sin\theta}\frac{\partial^2 A_r^{t-}}{\partial r\partial\phi} + \frac{1}{\varepsilon_d}\frac{1}{r}\frac{\partial F_r^{t-}}{\partial\theta} \tag{4-25c}$$

$$H_r^{t-} = \frac{1}{i\omega\mu_d\varepsilon_d}\left(\frac{\partial^2}{\partial r^2} + \beta_d^2\right)F_r^{t-} \tag{4-25d}$$

$$H_\theta^{t-} = \frac{1}{\mu_d}\frac{1}{r\sin\theta}\frac{\partial A_r^{t-}}{\partial\phi} + \frac{1}{i\omega\mu_d\varepsilon_d}\frac{1}{r}\frac{\partial^2 F_r^{t-}}{\partial r\partial\theta} \tag{4-25e}$$

$$H_\phi^{t-} = -\frac{1}{\mu_d}\frac{1}{r}\frac{\partial A_r^{t-}}{\partial\theta} + \frac{1}{i\omega\mu_d\varepsilon_d}\frac{1}{r\sin\theta}\frac{\partial^2 F_r^{t-}}{\partial r\partial\phi} \tag{4-25f}$$

Determining the three spherical components of the electric and magnetic fields, for the regions outside and inside the dielectric sphere, is accomplished by the following sequence: Plugging the solutions for A_r^{i+} (4-22a) and F_r^{i+} (4-22b) into eqns. (4-23a – 4-23f), and inserting A_r^{i-} (4-24a) and F_r^{i-} (4-24b) into eqns. (4-25a – 4-25f) allows the electric and magnetic field components, for the regions outside and inside the sphere, respectively, to be expressed solely in terms of the spherically transformed incident plane waves

For regions outside the dielectric sphere, the three spatial components of the electric and magnetic fields, respectively, take on the following form

$$E_r^{t+} = iE_o \cos\phi \sum_{n=1}^{\infty} \left\{ c_n \left[\hat{J}_n''(\beta_o r) + \hat{J}_n(\beta_o r) \right] + e_n \left[\hat{H}_n''^{(2)}(\beta_o r) + \hat{H}_n^{(2)}(\beta_o r) \right] \right\} P_n^1(\cos\theta) \quad \text{(4-26a)}$$

$$E_\theta^{t+} = i \frac{E_o \cos\phi}{\beta_o r} \sum_{n=1}^{\infty} \left[c_n \hat{J}_n'(\beta_o r) + e_n \hat{H}_n'^{(2)}(\beta_o r) \right] P_n'^1(\cos\theta)$$

$$- \frac{E_o \cos\phi}{\beta_o r \sin\theta} \sum_{n=1}^{\infty} \left[d_n \hat{J}_n(\beta_o r) + f_n \hat{H}_n^{(2)}(\beta_o r) \right] P_n^1(\cos\theta) \quad \text{(4-26b)}$$

$$E_\phi^{t+} = -i \frac{E_o \sin\phi}{\beta_o r \sin\theta} \sum_{n=1}^{\infty} \left[c_n \hat{J}_n'(\beta_o r) + e_n \hat{H}_n'^{(2)}(\beta_o r) \right] P_n^1(\cos\theta)$$

$$+ \frac{E_o \sin\phi}{\beta_o r} \sum_{n=1}^{\infty} \left[d_n \hat{J}_n(\beta_o r) + f_n \hat{H}_n^{(2)}(\beta_o r) \right] P_n'^1(\cos\theta) \quad \text{(4-26c)}$$

$$H_r^{t+} = -i \frac{E_o \sin\phi}{\eta_o} \sum_{n=1}^{\infty} \left\{ d_n \left[\hat{J}_n''(\beta_o r) + \hat{J}_n(\beta_o r) \right] + f_n \left[\hat{H}_n''^{(2)}(\beta_o r) + \hat{H}_n^{(2)}(\beta_o r) \right] \right\} P_n^1(\cos\theta) \quad \text{(4-26d)}$$

$$H_\theta^{t+} = \frac{E_o \sin\phi}{\beta_o \eta_o r \sin\theta} \sum_{n=1}^{\infty} \left[c_n \hat{J}_n(\beta_o r) + e_n \hat{H}_n^{(2)}(\beta_o r) \right] P_n^1(\cos\theta)$$

$$- i \frac{E_o \sin\phi}{\beta_o \eta_o r} \sum_{n=1}^{\infty} \left[d_n \hat{J}_n'(\beta_o r) + f_n \hat{H}_n'^{(2)}(\beta_o r) \right] P_n'^1(\cos\theta) \quad \text{(4-26e)}$$

$$H_\phi^{t+} = \frac{E_o \cos\phi}{\beta_o \eta_o r} \sum_{n=1}^{\infty} \left[c_n \hat{J}_n(\beta_o r) + e_n \hat{H}_n^{(2)}(\beta_o r)\right] P_n'^1(\cos\theta)$$

$$-i \frac{E_o \cos\phi}{\beta_o \eta_o r \sin\theta} \sum_{n=1}^{\infty} \left[d_n \hat{J}_n'(\beta_o r) + f_n \hat{H}_n'^{(2)}(\beta_o r)\right] P_n^1(\cos\theta) \qquad \text{(4-26f)}$$

For regions inside the dielectric sphere, the three spatial components of the electric and magnetic fields, respectively, take on the following form

$$E_r^{t-} = iE_o \cos\phi \sum_{n=1}^{\infty} \left\{g_n \left[\hat{J}_n''(\beta_d r) + \hat{J}_n(\beta_d r)\right]\right\} P_n^1(\cos\theta) \qquad \text{(4-26g)}$$

$$E_\theta^{t-} = i \frac{E_o \cos\phi}{\beta_d r} \sum_{n=1}^{\infty} \left[g_n \hat{J}_n'(\beta_d r)\right] P_n'^1(\cos\theta) - \frac{E_o \cos\phi}{\beta_d r \sin\theta} \sum_{n=1}^{\infty} \left[h_n \hat{J}_n(\beta_d r)\right] P_n^1(\cos\theta) \qquad \text{(4-26h)}$$

$$E_\phi^{t-} = -i \frac{E_o \cos\phi}{\beta_d r \sin\theta} \sum_{n=1}^{\infty} \left[g_n \hat{J}_n'(\beta_d r)\right] P_n^1(\cos\theta) - \frac{E_o \cos\phi}{\beta_d r} \sum_{n=1}^{\infty} \left[h_n \hat{J}_n(\beta_d r)\right] P_n'^1(\cos\theta) \qquad \text{(4-26i)}$$

$$H_r^{t-} = -i \frac{E_o \sin\phi}{\eta_d} \sum_{n=1}^{\infty} \left\{h_n \left[\hat{J}_n''(\beta_d r) + \hat{J}_n(\beta_d r)\right]\right\} P_n^1(\cos\theta) \qquad \text{(4-26h)}$$

$$H_\theta^{t-} = \frac{E_o \sin\phi}{\beta_d \eta_d r \sin\theta} \sum_{n=1}^{\infty} \left[g_n \hat{J}_n(\beta_d r)\right] P_n^1(\cos\theta) - i \frac{E_o \sin\phi}{\beta_d \eta_d r} \sum_{n=1}^{\infty} \left[h_n \hat{J}_n'(\beta_d r)\right] P_n'^1(\cos\theta) \qquad \text{(4-26i)}$$

$$H_\phi^{t-} = \frac{E_o \cos\phi}{\beta_d \eta_d r} \sum_{n=1}^{\infty} \left[g_n \hat{J}_n(\beta_d r)\right] P_n'^1(\cos\theta) - i \frac{E_o \cos\phi}{\beta_d \eta_d r \sin\theta} \sum_{n=1}^{\infty} \left[h_n \hat{J}_n'(\beta_d r)\right] P_n^1(\cos\theta) \qquad \text{(4-26j)}$$

where $\hat{B}_n''(\beta r) = \frac{\partial^2}{\partial(\beta r)^2} \hat{B}_n(\beta r)$, $\hat{B}_n'(\beta r) = \frac{\partial}{\partial(\beta r)} \hat{B}_n(\beta r)$, $P_n'^1(\cos\theta) = \frac{\partial}{\partial(\cos\theta)} P_n^1(\cos\theta)$,

$\frac{\partial}{\partial(\cos\theta)} = -\sin\theta \frac{\partial}{\partial\theta}$, $\hat{B}_n''(\beta r) = \hat{J}_n''(\beta r)$ or $\hat{H}_n''(\beta r)$, $\hat{B}_n'(\beta r) = \hat{J}_n'(\beta r)$ or $\hat{H}_n'(\beta r)$, and

$\hat{B}_n(\beta r) = \hat{J}_n(\beta r)$ or $\hat{H}_n(\beta r)$.

Treating the real and imaginary terms of the tangential components of the electric and magnetic fields separately allows the coefficients e_n, f_n, g_n and h_n to be expressed in terms of known quantities

$$e_n = -c_n \frac{\hat{J}'_n(\beta_d a)\hat{J}_n(\beta_o a)\eta_d - \hat{J}_n(\beta_d a)\hat{J}'_n(\beta_o a)\eta_o}{\hat{H}^{(2)}_n(\beta_o a)\hat{J}'_n(\beta_d a)\eta_d - \hat{H}'^{(2)}_n(\beta_o a)\hat{J}_n(\beta_d a)\eta_o} \tag{4-27a}$$

$$f_n = -d_n \frac{\hat{J}_n(\beta_d a)\hat{J}'_n(\beta_o a)\eta_d - \hat{J}'_n(\beta_d a)\hat{J}_n(\beta_o a)\eta_o}{\hat{H}'^{(2)}_n(\beta_o a)\hat{J}_n(\beta_d a)\eta_d - \hat{H}^{(2)}_n(\beta_o a)\hat{J}'_n(\beta_d a)\eta_o} \tag{4-27b}$$

$$g_n = c_n \frac{\beta_d \eta_d \left[\hat{H}^{(2)}_n(\beta_o a)\hat{J}'_n(\beta_o a) - \hat{H}'^{(2)}_n(\beta_o a)\hat{J}_n(\beta_o a)\right]}{\beta_o \left[\hat{H}^{(2)}_n(\beta_o a)\hat{J}'_n(\beta_d a)\eta_d - \hat{H}'^{(2)}_n(\beta_o a)\hat{J}_n(\beta_d a)\eta_o\right]} \tag{4-27c}$$

$$h_n = -d_n \frac{\beta_d \eta_d \left[\hat{H}^{(2)}_n(\beta_o a)\hat{J}'_n(\beta_o a) - \hat{H}'^{(2)}_n(\beta_o a)\hat{J}_n(\beta_o a)\right]}{\beta_o \left[\hat{H}'^{(2)}_n(\beta_o a)\hat{J}_n(\beta_d a)\eta_d - \hat{H}^{(2)}_n(\beta_o a)\hat{J}'_n(\beta_d a)\eta_o\right]} \tag{4-27d}$$

where $c_n = b_n - a_n$, $d_n = a_n + b_n$, $a_n = i^{-n}(2n+1)$, and $b_n = i^n(2n+1)$.

4.5 Power Generation Term

To determine the power generation within the sphere, the following *conservation of energy equation* in differential form is employed [8]

$$-\vec{\nabla}\cdot\left(\frac{1}{2}\vec{E}\times\vec{H}^*\right) = \frac{1}{2}\vec{H}^*\cdot\vec{M}_i + \frac{1}{2}\vec{E}\cdot\vec{J}^*_i + \frac{1}{2}\sigma\left|\vec{E}\right|^2 + i2\omega\left(\frac{1}{4}\mu\left|\vec{H}\right|^2 - \frac{1}{4}\varepsilon\left|\vec{E}\right|^2\right) \tag{4-28}$$

where $\vec{M}_i$ represents impressed (source) magnetic current density, $\vec{J}^*_i$ is the complex conjugate impressed electric current density, σ is the conductivity of the material, and $\sigma = \omega\varepsilon_o\varepsilon''_r$.

Since $\vec{M}_i = \vec{J}^*_i = 0$, and the power generated within the sphere is a real quantity, the following relation is arrived upon

$$Q_{gen} = \frac{1}{2}\omega\varepsilon_o\varepsilon_r''\left|\vec{E}\right|^2 + \frac{1}{2}\omega\mu_o\mu_r''\left|\vec{H}\right|^2 \tag{4-29}$$

where Q_{gen} represents the power generated within the sphere, $\left|\vec{E}\right|^2 = \vec{E}\cdot\vec{E}^*$, $\left|\vec{H}\right|^2 = \vec{H}\cdot\vec{H}^*$,

$\vec{E} = E_r^{t-}\hat{r} + E_\theta^{t-}\hat{\theta} + E_\phi^{t-}\hat{\phi}$, $\vec{H} = H_r^{t-}\hat{r} + H_\theta^{t-}\hat{\theta} + H_\phi^{t-}\hat{\phi}$, $\vec{E}^*$ is the complex conjugate of $\vec{E}$, and $\vec{H}^*$ is the complex conjugate of $\vec{H}$. Since the region outside the sphere is treated as free space (free space wave number β_o) and the region inside the sphere is modeled as a lossy dielectric (wave number β_d), only the portion of the electric and magnetic field that propagates within the sphere need be considered.

References

1. Balanis CA. *Advanced Engineering Electromagnetics*. 2nd ed. Hoboken, NJ: John Wiley & Sons; 2012:654.
2. G. Rossy JAP. *Foundations and Industrial Applications of Microwaves and Radio Frequency Fields: Physical and Chemical Processes*. West Sussex, England: John Wiley & Sons; 1995:10-12.
3. Frenske KM, Devendra. Dielectric Materials at Microwave Frequencies. *Applied Microwave & Wireless*.
4. Balanis CA. Radiation Integrals and Auxiliary Potential Functions. *Antenna Theory: Analysis and Design,* 3rd ed. Hoboken, NJ: John Wiley & Sons; 2005:133-135.
5. Balanis CA. The Vector Potential A. *Advanced Engineering Electromagnetics*. 2nd ed. Hoboken, NJ: John Wiley & Sons; 2012:260-265.
6. Balanis CA. Spherical Transmission Lines and Cavities. *Advanced Engineering Electromagnetics*. 2nd ed. Hoboken, NJ: John Wiley & Sons; 2012:549-557.
7. Balanis CA. Power and Energy. *Advanced Engineering Electromagnetics*. 2nd ed. Hoboken, NJ: John Wiley & Sons; 2012:25-29.
8. Balanis CA. *Advanced Engineering Electromagnetics, 2nd Ed.* Hoboken, NJ: John Wiley & Sons, Inc.; 2012:25-29.

Chapter 5:

Results and Discussion

The analytical expressions for the electric field, magnetic field, and power distribution are obtained by solving Maxwell's equation for TEM waves in spherical coordinates. The expressions are evaluated for typical beef nuggets which are usually in spherical shape. The sizes and electromagnetic frequencies are varied to help understand heat generation distribution. The dielectric properties of beef nuggets for various electromagnetic frequencies are found from literature [1]. The incident electromagnetic energy flux ($I = c\varepsilon_0 E_0^2$) is kept constant and is considered to be 3 W/cm^2. This is equivalent to a 1.2 kW household microwave oven and the equivalent incident electric field strength E_0 is found to be 4754.3 V/m. Table 1 lists the four microwave heating frequencies used in this study with corresponding dielectric constant and dielectric loss.

Table 1: Dielectric properties of spherical beef nugget at different frequencies.

Properties/frequency (MHz)	2800	2450	915	300
Dielectric constant, κ' [1]	33.6	30.5	35.4	38
Dielectric loss, κ'' [1]	12.6	9.6	16	47

Four separate beef nugget radii were studied: 1.0 cm, 2.0 cm, 3.0 cm, and 5.0 cm. These radii were chosen to correspond to common nugget sizes for use in frozen foods.

Table 2: Properties and input parameters.

Parameters	Values
Incidence microwave energy flux, I (W/cm^2)	3
Equivalent microwave power level (kW)	1.2
Electric field strength, E_0 (V/m)	4754.3
Radii of spherical beef nuggets, r_0 (cm)	1.0, 2.0, 3.0, 5.0

In the sub sections to follow, results from various combinations of radii, frequency, and cross sections are presented.

5.1 Effect of Sphere Sizes on Electric Field and Power Absorption Distribution along Centerline

The absolute Electric field strength along the centerline of spherical beef nuggets for radii 0.01 m, 0.02 m, 0.03 m, and 0.05 m are presented in Figure 5.1. The irradiating frequency is 2450 MHz which is the frequency for household microwave ovens. The radii of each sphere is non-dimensionalized to allow ease of comparison.

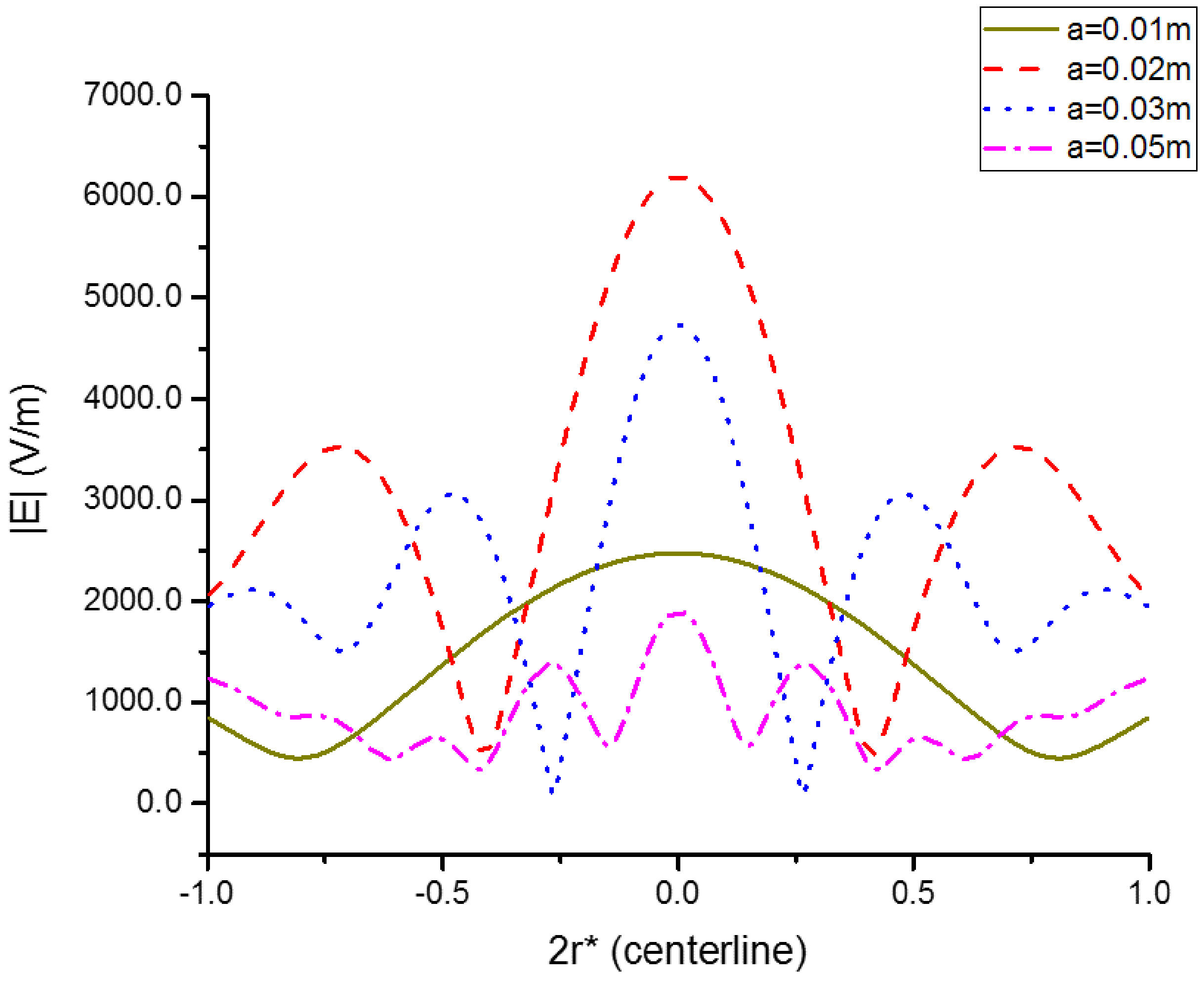

Figure 5.1: Electric field strength along centerline of spherical beef nugget.

It can be observed that the number of peaks in the electric field distribution increase in number as the radius of the beef sphere increases. In all cases, the highest peak of the electric field is found at the center of the sphere. When the electromagnetic field is propagating in the opposite direction, the superposition of waves are taking place and resonance of the wave is happening at the center. At the frequency of 2450 MHz, the propagation wavelength in the beef nugget is calculated from the equations reported in [2] and found to be 2.18 cm. The locations of the peaks are dependent on the wavelength and radius of the sphere [1]. The greatest peaks of the electric field strength is seen in 0.02 m radius sphere because it is closer to the propagation wavelength and positive interference, i.e. resonance, is happening between the two waves propagating towards the center. Only one peak is seen in the smallest sphere because the radius of the sphere is much smaller than the incident wavelength of the electromagnetic radiation. Similar trends of electric field distribution and peaks are reported in previous works for rectangular and cylindrical shaped objects under electromagnetic heating [3,4]. The corresponding electromagnetic power absorption along the centerline of spherical beef nuggets for 2450 MHz electromagnetic radiation are shown in Figure 5.2. The radii of each sphere is non-dimensionalized for better comparison among the nuggets. The power distribution follows the trend of electric field distribution. From the distribution, it is evident that along the centerline, the smallest and largest sphere provides more uniform heat generation compared to the other sizes. The overall power absorption, i.e. heat generation patterns, are discussed the following sections.

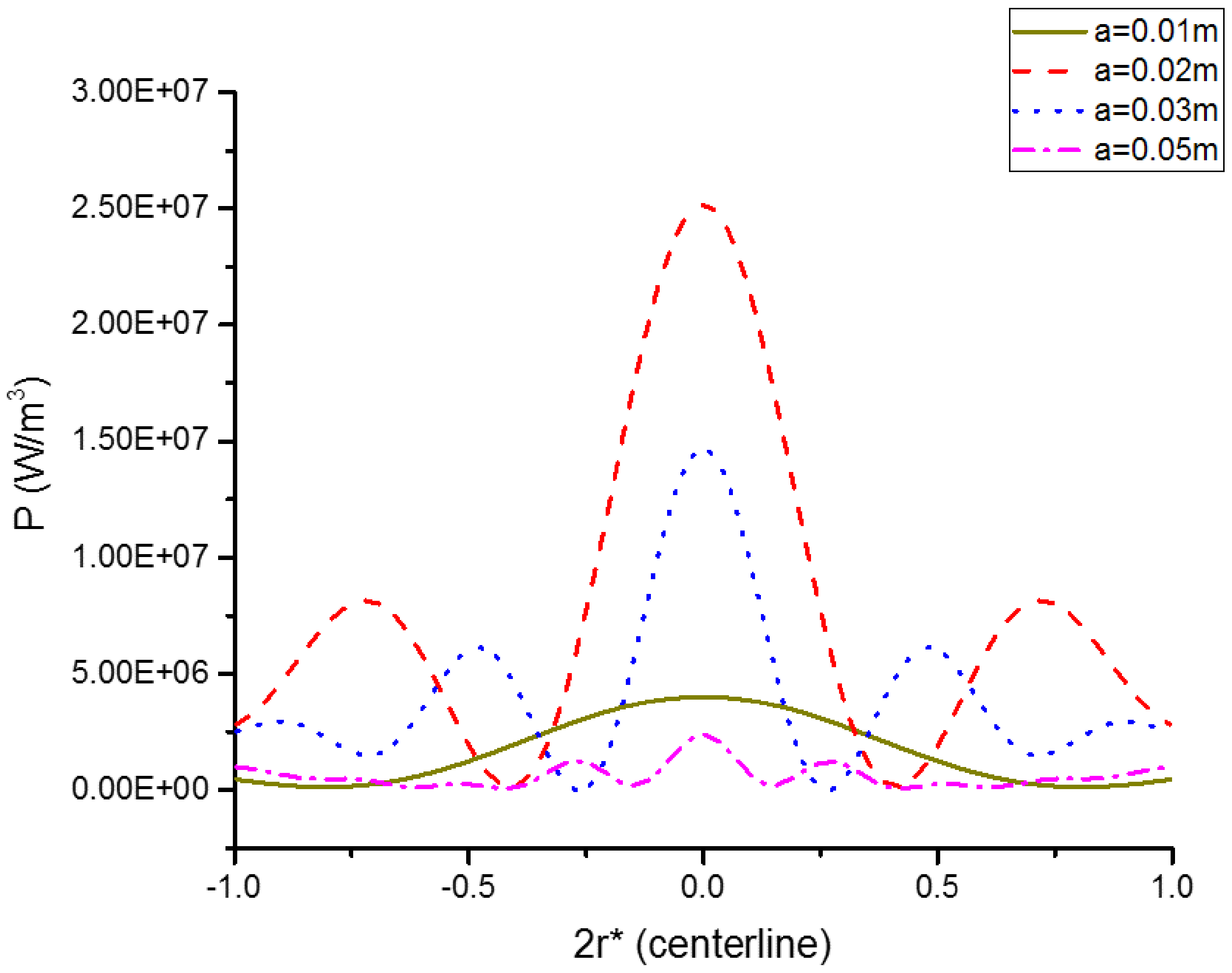

Figure 5.2: Power generation along centerline of spherical beef nugget.

5.2 Effect of Sphere Sizes on Planar Electric Field and Power Absorption Distribution

Figure 5.3 depicts the absolute strength of the E-field generated within spherical beef nuggets of four differing radii as mentioned in the above section for a typical microwave heating frequency of 2450 MHz. The results are presented as a 2D slice running through the center of the nuggets vertically and looking down the +y axis. For ease of comparison, each radii are non-dimensionalized. Results show the nature of the electric field intensity, especially in the case of **(a)**, **(b)**, and **(c)**, takes on the form of horizontal layers or bands, with the greatest intensity taking place in the inner oval region. In **(b)**, the greatest region of activity can be seen as a vertical two-ring structure, with an especially active region where the two "rings" meet in the center of the plot. The greatest value of the electric field in **(b)** is over two times that of **(a)**. For **(c)**, the most active regions are something of a continuation of what was seen in the case of **(b)**, with the addition of two extra "stacks" on top of the previously seen vertical two-ring structure. The greatest value of the electric field in **(c)** is less than that of **(b)**, but is still nearly two times that of **(a)**. A much more complex and diverse pattern can be observed for **(d)**. The hot "tips" at the end of the large X can be seen at theta = 60, 120, 240, and 300 degrees, are strongest at phi = 0 and 180 degrees, and least powerful at phi = 90 and 270 degrees. It is at the tips that the greatest electric field intensity is observed.

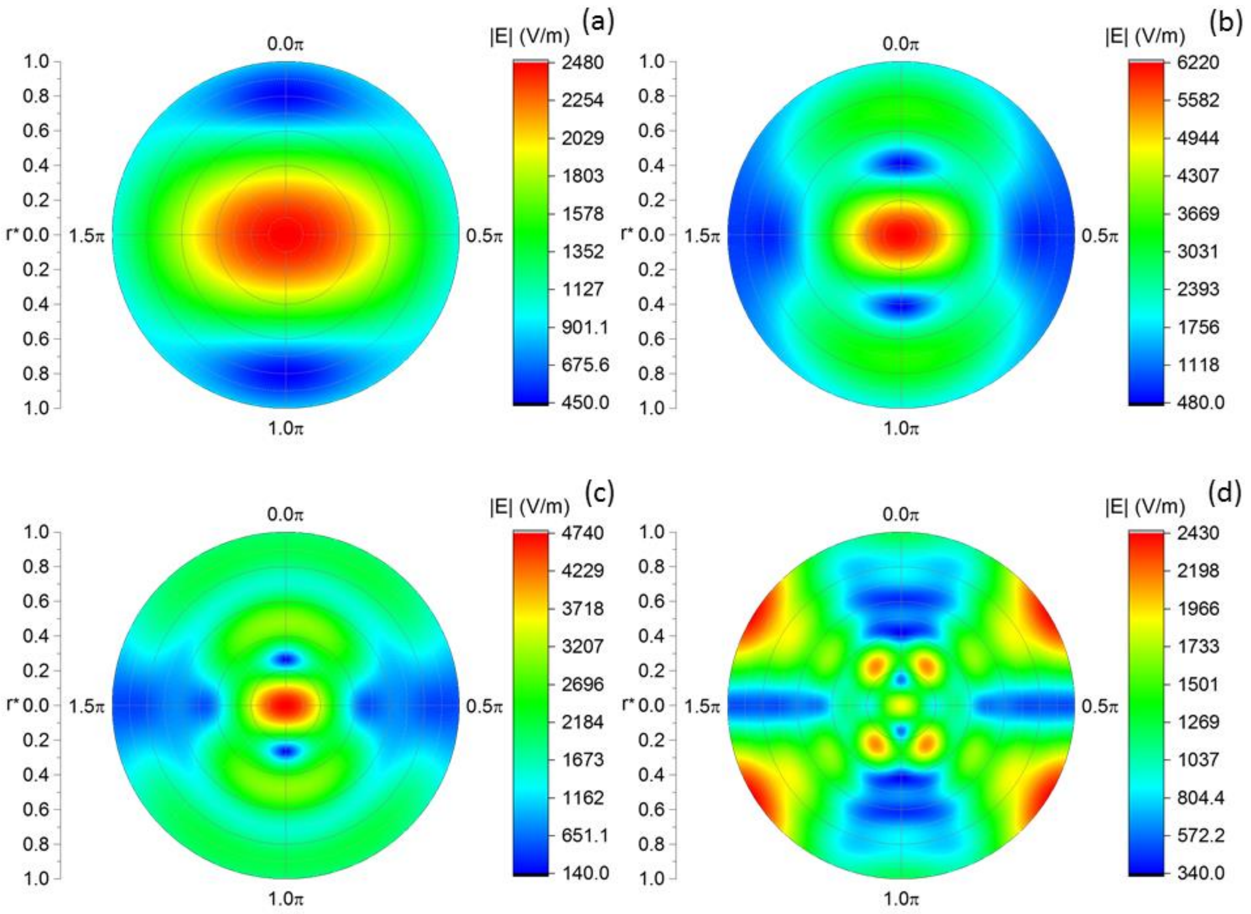

Figure 5.3: Electric Field Distribution Cross Section within Spherical Beef Nuggets of radii (a) 1.0 cm, (b) 2.0 cm, (c) 3.0 cm, and (d) 5.0 cm. Here f = 2450 MHz.

The corresponding electromagnetic power absorption for four different nugget sizes are shown in Figure 5.4. The results again show that the power absorption distribution closely follow the trends of electric field distribution. The peak electromagnetic power absorption is taking place at the inner core of the sphere for 1.0 cm, 2.0 cm and 3.0 cm radius of beef nuggets (i.e. Figures 5.4a, 5.4b and 5.4c respectively), however the largest size of nugget, i.e. 5.0 cm radius, experiences the highest energy absorption at the surface in Figure 5.4d. So at the frequency of 2450 MHz, the 5.0 cm nugget will most likely to have surface burning and the 3.0 cm nugget will experience repetitive hot and cold zones throughout the nugget.

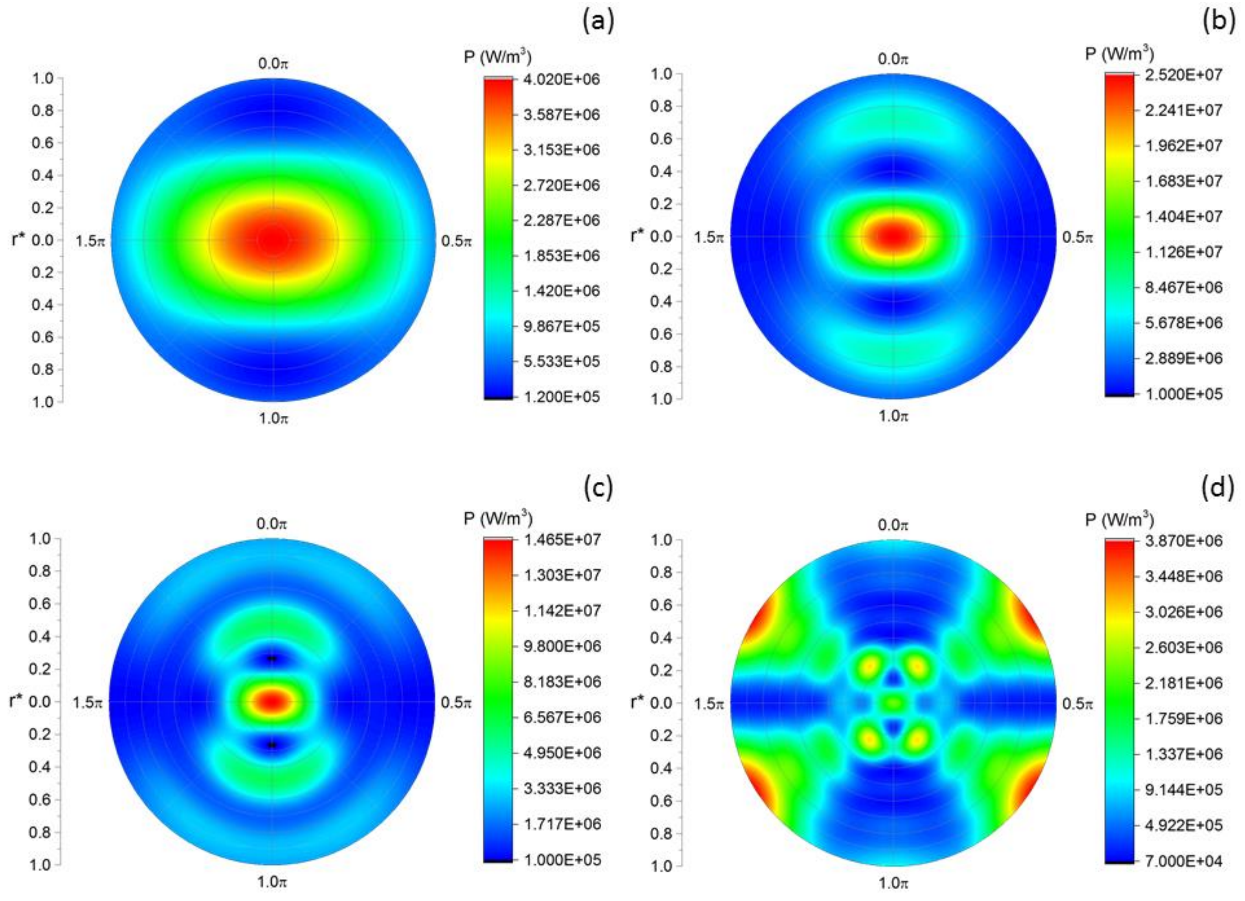

Figure 5.4: Power Distribution Cross Section within Spherical Beef Nuggets of radii (a) 1.0cm, (b) 2.0cm, (c) 3.0cm, and (d) 5.0cm. Here f = 2450 MHz.

At the frequency of 2450 MHz, the penetration depth for the beef nugget is calculated using equation presented in [2] and found to be 1.58 cm. At this depth, the electromagnetic radiation decays exponentially. So at the larger size of nugget, i.e. 5.0 cm radius beef sphere, the highest absorption happens at the surface instead of the core of the nugget. This is also evident from the electric field and power absorption distribution along the centerline in Figures 5.1 and 5.2. For **(a)** two circular zones of low power generation can be seen situated at the top and bottom of the nugget, i.e. core heating is most likely for smaller nuggets. The maximum recorded generated power for any of the radii is observed for 2.0 cm nugget shown in figure 5.4b since the wavelength of electromagnetic wave is close to the radius of the nugget as explained in the previous section.

5.3 Electric Field and Power Absorption Distribution at Different Cutting Lines

Figure 5.5 depicts the strength of the absolute electric field as a line beginning at the center of the sphere and extending to the outer surface of the sphere for six separate cutting lines at electromagnetic frequency of 2450 MHz to elucidate the orientation effect of TEM wave impingement. The results are presented for all four beef nuggets of 1.0 cm, 2.0 cm, 3.0 cm, and 5.0 cm radii in non-dimensionless form. A general trend of peaks and valleys increasing with greater radii can be observed, particularly for **(a)**, **(b)**, and **(d)**, and to a lesser extent, **(c)**. For all results, the maximum peak value for the electric field strength observed with the 0.02 m radius sphere at the center of the sphere as seen in previous sections. It is interesting to note the more pronounced damping of the electric field strength near the surface of the sphere for the 0.02 m and 0.03 m radii at the chosen angles for **(e)** and **(f)**. Apart from what is seen for the radii of 0.02 m, the observed values for the remaining radii are very similar to the preceding figure. Even more damping of the electric field strength near the surface of the sphere, for all studied radii, can be observed for **(c)**, **(e)**, and **(f)**.

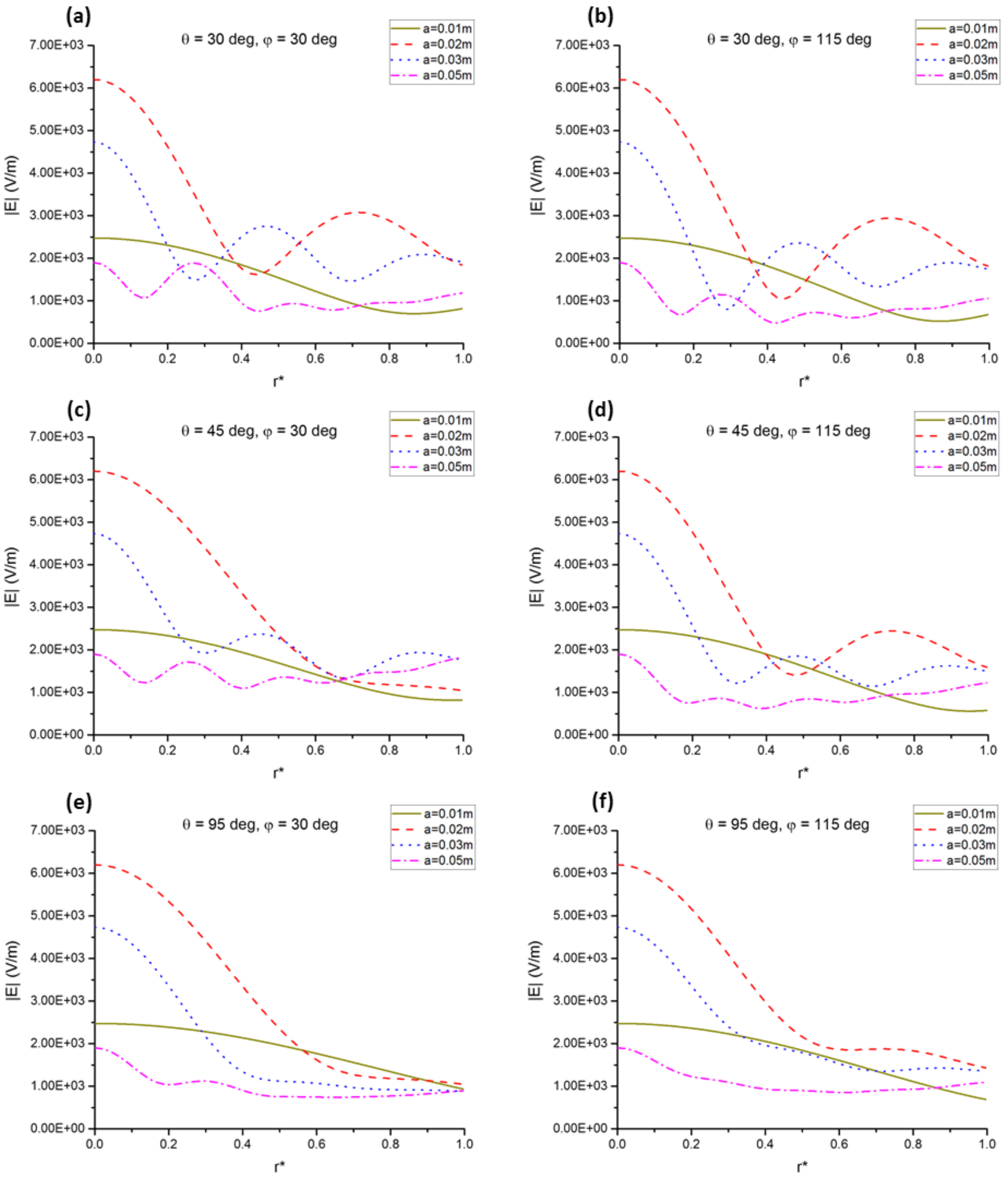

Figure 5.5: Electric Field Strength along six Line Paths within Spherical Beef Nuggets of radii 1.0 cm, 2.0 cm, 3.0 cm, and 5.0 cm. Here f = 2450 MHz.

Figure 5.6 traces the strength of the generated power as a line beginning at the center of the sphere and extending to the outer surface of the sphere for six separate combinations of theta and phi at electromagnetic frequency of 2450 MHz. The trend of the differences in strength between peaks and valleys increasing in spheres with greater radii can be observed for the generated power as well. The maximum peak value for the strength of the generated power observed for the 0.02 m radius sphere in **(a)**, **(b)** and **(d)**. This maximum value is generated at the center of the sphere for all radii. A similar pattern can be seen with phi set to 115 degrees. The more pronounced damping of the strength of the generated power near the surface of the sphere for the 0.02 m and 0.03 m radii can be observed for **(e)** and **(f)**. A similar trend can be seen in this case as well. More pronounced damping of the strength generated power, near the surface of the sphere, for all studied radii, can be observed for the chosen angles for theta and phi **(a)** – **(f)**. Compared to **(a)** – **(d)**, a lessening of damping can be seen for the 0.02 m and 0.03 m radii towards the outer surface of the sphere in **(e)** and **(f)**.

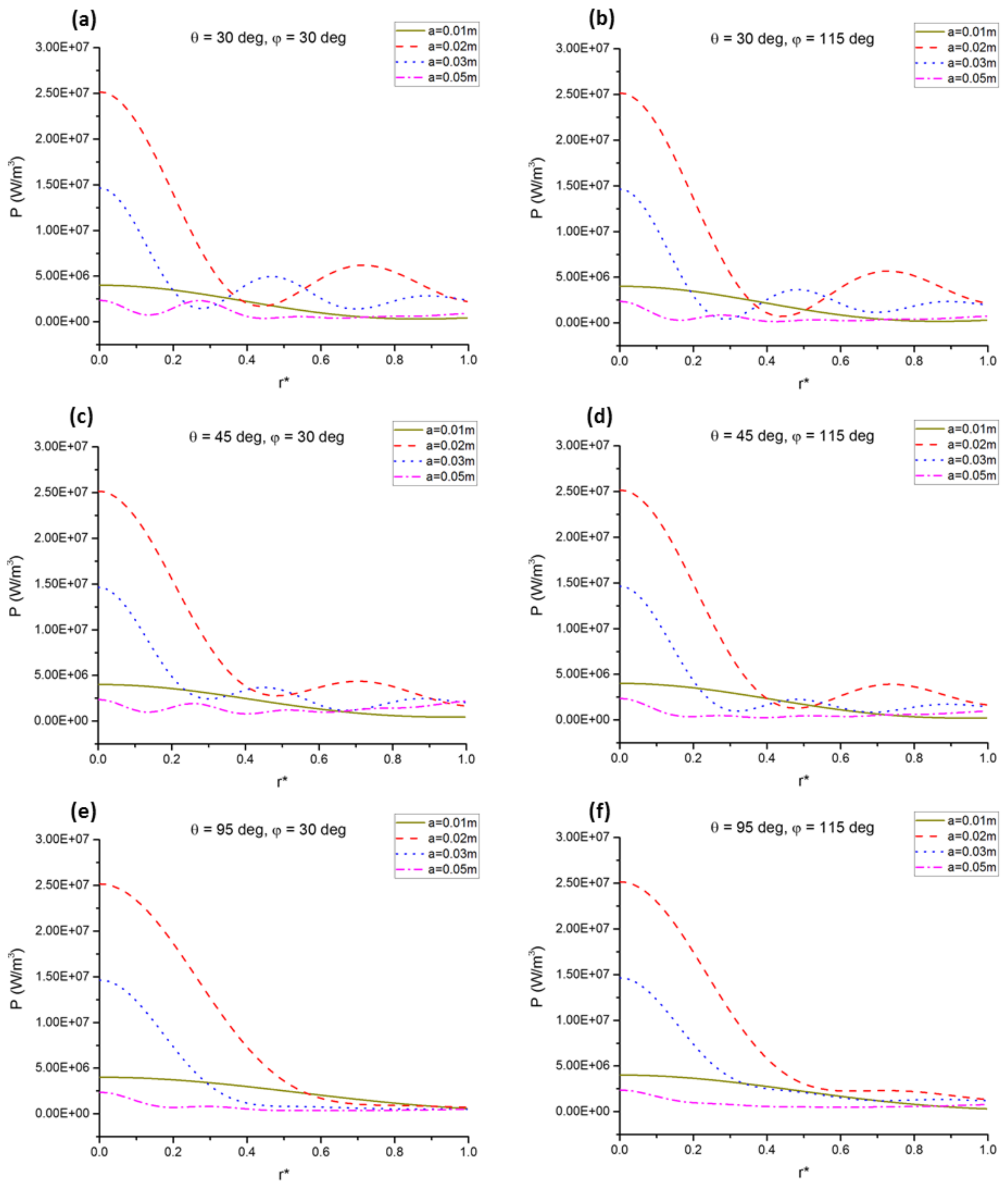

Figure 5.6: Power Generation strength along six Line Paths within Spherical Beef Nuggets of radii 1.0 cm, 2.0 cm, 3.0 cm, and 5.0 cm. Here f = 2450 MHz.

5.4 Effect of Frequencies on Electric Field and Power Absorption Distribution

Three different frequencies are employed to elucidate the effect of frequencies on electromagnetic power absorption within the beef nugget of 2.0 cm radius. A 2.0 cm radius beef nugget is selected from the previous results of the 2450 MHz treatment because at this size, it generates highest power absorption. Also at 2450 MHz, the propagation wavelength within the beef nugget is 2.18 cm, which is close to the size of the nugget. Figure 5.7 depicts the absolute strength of the E-field generated in the spherical nugget of the 2.0 cm radii and corresponding power absorption. The incident electromagnetic energy flux is kept same which is 3 W/cm^2. It is interesting to note that the distribution of the electric field drastically changes at the different frequencies. At the lower frequency, electric field distribution is much more uniform. Also, the maximum strength of the electric field increases with frequency, which can be observed looking at **(a)**, then **(c)**, and then **(e)**.

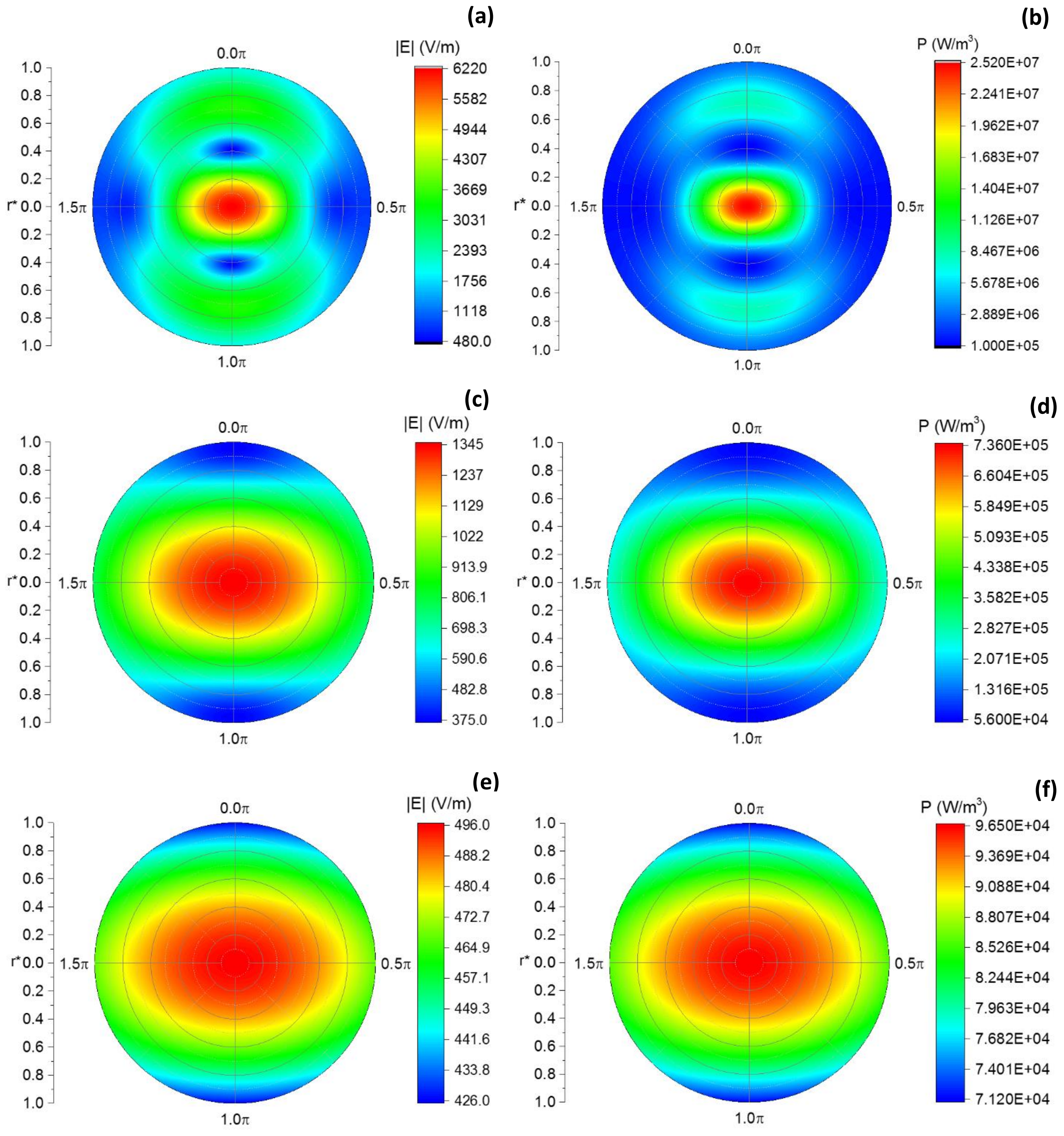

Figure 5.7: Electric Field Distribution and Power Generation Distribution Cross Sections within Spherical Beef Nuggets of radius 2.0cm for three different frequencies. (a) E field for f = 2450 MHz, (b) P gen for f = 2450 MHz, (c) E field for f = 915 MHz, (d) P gen for f = 915 MHz, (e) E field for f = 300 MHz, and (f) P gen for f = 300 MHz.

The corresponding power absorption distribution follows the pattern of electric field distributions as seen all other cases. Although power absorption in lower frequencies are smaller compared to the higher frequencies, the distribution are much more uniform. This will lead to more uniform temperature distributions when the nuggets are treated with lower electromagnetic frequencies. This is because at the lower frequencies the dielectric constant and dielectric loss increases and hence they change the electromagnetic wavelength within the nugget. For instance, the propagation wavelength at 300 MHz is 14.25 cm, which is much larger than the size of the nugget.

5.5 Electric Field and Power Generation Strength along Sphere Centerline

The effect of frequencies of the four different nugget sizes along the center line are described in Figure 5.8. Each of the four spheres is irradiated with electromagnetic radiation in four different frequencies, and the electric field distribution of each frequency, for each sphere radius, are displayed in an overlay fashion. Lower frequencies in general showcase more uniform electric field strength along the centerline, while the higher frequencies show greater extremes between maximum and minimum field strength, with higher peak values increasing with frequency **(a)** – **(d)**. The greatest value for the electric field can be observed in **(a)**, **(b)**, and **(c)** at the sphere center for all frequencies. For **(d)** the electric field results follow a similar trend, except for the 2800 MHz frequency, which shows the outer surface to have the largest recorded electric field values. For **(b)** and **(c)**, the 2450 MHz frequencies record the highest electric field strength. For **(a)** and **(d)**, 2800 MHz frequencies register the largest electric field strengths. It is interesting to note the greater variation in overall waveform and maximum and minimum values when comparing **(a)** and **(b)** to **(c)** and **(d)** for the 915 MHz frequency. For all other frequencies, the overall waveform remains the same, with variations in the strength and the total number of peaks and valleys. It is interesting to note a general trend of a more even distribution of the electric field strength along the total length of the centerline of the sphere as the radius of the sphere increases. This trend is especially apparent in **(d)**.

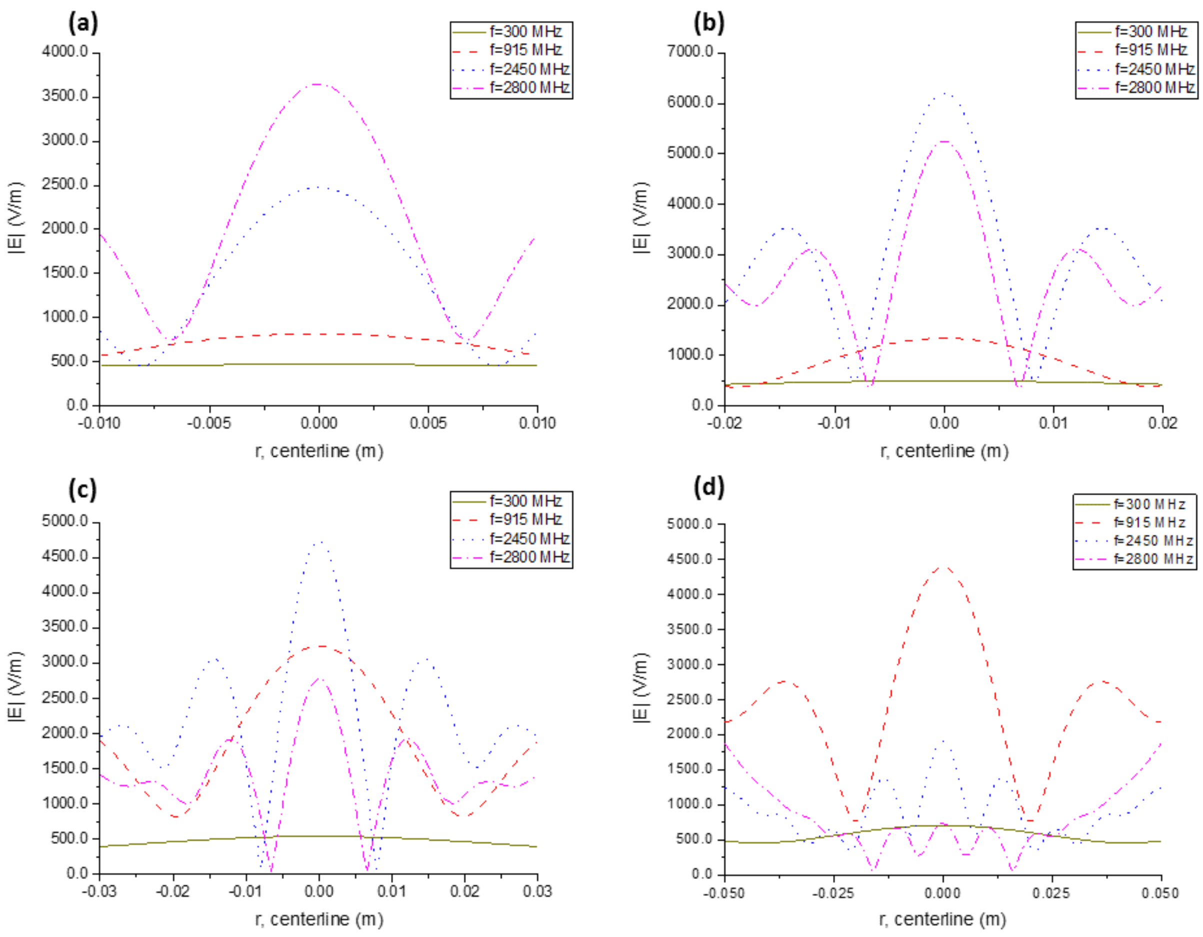

Figure 5.8: Electric Field Strength along Sphere Centerline within Spherical Beef Nuggets radii (a) 1.0 cm, (b) 2.0 cm, (c) 3.0 cm, and (d) 5.0 cm.

Figure 5.9 visualizes the absolute strength of the power generated along the centerline of beef spheres of varying radii. As before, each of the four spheres are irradiated with four different electromagnetic frequencies, and the results of each frequency, for a specific sphere radius, are overlaid on each sub-figure **(a)** – **(d)** for ease of comparison. Lower frequencies in general showcase more uniform electric field strength along the centerline (the exception being the 915 MHz frequency reading **(d)**), while the higher frequencies commonly show greater extremes between maximum and minimum field strength, with peak power generation values increasing with frequency. It is interesting to note that the maximum power generation is seen at 2800 MHz for **(a)** and **(b)**, and 2450 MHz for **(c)**. In **(d)**, the 915 MHz frequency produces the maximum power generation value. The largest power generation values are observed for **(b)** at the center of the sphere for the 2800 MHz frequency. The next largest power generation value is seen in **(c)** for the 2450 MHz frequency. The third and fourth highest power generation values can be observed in **(a)** and **(d)**, respectively. In **(d)**, a general trend of enhanced power generation at the top and bottom of the sphere centerline is noted. The highest recorded value for power generation of any of the four frequencies is at the 915 MHz value.

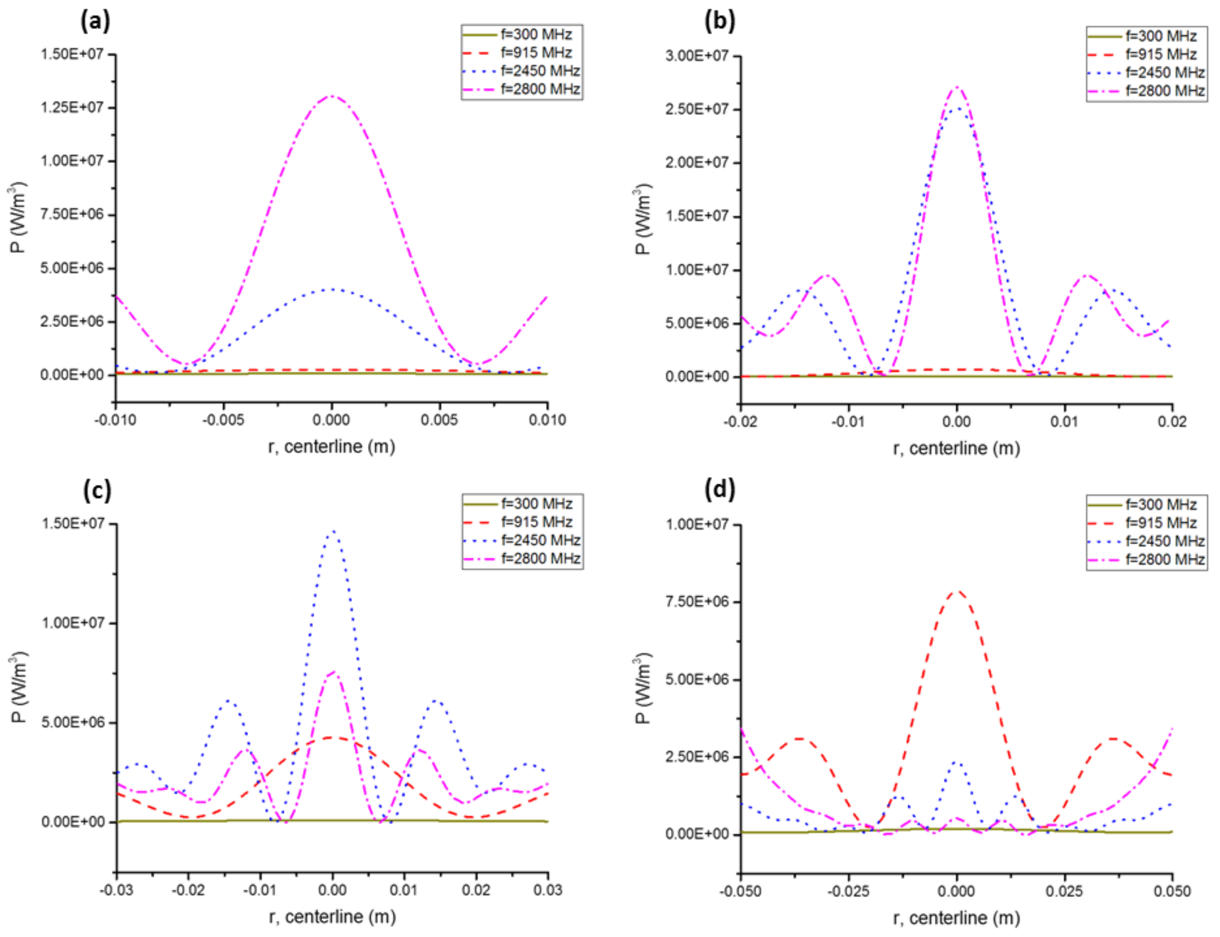

Figure 5.9: Power Generation strength along Sphere Centerline of Spherical Beef Nuggets radii (a) 1.0 cm, (b) 2.0 cm, (c) 3.0 cm, and (d) 5.0 cm.

References

1. Ayappa KG, Davis HT, Crapiste G, Davis EA, Gordon J. Microwave heating: an evaluation of power formulations. *Chemical Engineering Science.* 1991;46(4):1005-1016.
2. K. G. Ayappa HTD, E. A. Davis, J. Gordon. Two-Dimensional Finite Element Analysis of Microwave Heating. *AlChE Journal.* 1992:1577-1592.
3. M. R. Hossan PD. Effects of temperature dependent properties in electromagnetic heating. *International Journal of Heat and Mass Transfer.* 2012;55:3412-3422.
4. Mohammad Robiul Hossan DB, Prashanta, Dutta. Analysis of microwave heating for cylindrical shaped objects. *International Journal of Heat and Mass Transfer.* 2010:5129-5138.

Chapter 6:

Summary and Conclusions

A closed form solution is obtained for the electric field, magnetic field, and power generation distributions within a spherical shaped dielectric object using Maxwell's equation. The transverse electric and magnetic (TEM) wave in spherical coordinate is solved using vector potentials and separation of variables. Mathematical tools such as Bessel functions, Legendre Polynomials, Infinite series, and complex number expressions are employed in finding a closed form expression. The continuity boundary conditions from outside to the inside of the object for the tangential components of the electric and magnetic field are used. The electromagnetic power absorption is obtained from the knowledge of the electric and magnetic field distributions using Poynting theorem.

The closed form expression of the electric field and power absorption are evaluated for beef nuggets of four different sizes (radii of 1.0 cm, 2.0 cm, 3.0 cm and 5 .0 cm) and frequencies of 2800 MHz, 2450 MHz, 915 MHz, and 300 MHz. Numerical tools such as Maple, MATLAB, and the FORTRAN coding language were used as a CAS, graphing, and as a means of generating data, respectfully. Origin Labs was utilized to produce 1-D plots and also 2-D polar plots by reading in the data text files generated in the FORTRAN program.

Results show the presence of local maxima and minima of electric strength within the target object due to the constructive interference of electromagnetic wave impingement throughout the sphere. The local maxima and minima of electric field strength vary depending on the sizes and applied frequencies. The lower frequencies, i.e. longer wavelengths, have less peaks and valleys in the electric field distribution for the same sized nugget exposed to different applied frequencies. The spatial distribution of microwave power absorption follows the trend of the electromagnetic field distribution. The number and locations of local maxima and minima of power absorption also depend on the radius of the sphere and applied frequencies. For instance, the results show that that the strength of the absorbed electromagnetic wave at the 2450 MHz range has 3 weak peaks close to the center at the vertical plane of the 5.0 cm radius nugget while it has only one peak at the center of the 1.0 cm radius nugget for the same vertical plane passing through the center of the nugget.

Results indicate that there is a correlation among the electric field and power absorption distribution, propagating wavelength within the nugget, and penetration depth and size of the nugget. For instance, the 300 MHz frequency provides uniform electric field and power absorption distribution for all sizes of nuggets in this study. This is because the propagating wavelength is about 15 cm, which is much larger than all nugget radii considered in this study. Results also show that 2800 MHz and 2450 MHz can provide core heating for 2.0 cm and 3.0 cm radii nuggets while they can facilitate surface burning for the 5.0 cm radius nugget. The analysis shows that the uniform and effective electromagnetic power absorption can be facilitated by proper design of the object of interest and selection of appropriate electromagnetic frequencies. The variations of the fundamental parameters (dielectric properties, size, frequency, etc.) could affect a profound change on the electromagnetic distribution within the chosen dielectric sphere.

Future Work:

The following future work can be recommended to elucidate the effect of other parameters on heat generation and temperature distribution in spherical shaped object:

1. Generate results and study for different applications such as microwave heat treatment of cancer tissues or cells.
2. Solve three dimensional transient heat equation with electromagnetic heat generation in spherical coordinate for spherical shaped objects.
3. Incorporation of temperature dependent properties in numerical algorithm for evaluating heat generation and temperature distribution.

Appendix:

FORTRAN Source Code to Generate 3D Spherical Data for Electric Field, Magnetic Field, and Power Generation

```
!
! File:   E_H_P_Dbl_spherical_calculator.f95
! Author: Timothy M. Collins Jr.
!
! Started on August 24, 2015
! Finished on September 14, 2015
! Transcription of original MATLAB code

program EHPSphCalc
!this program calculates and stores, in spherical polar coordinates, the
!absolute Electric and Magnetic fields and the (absolute) Power generated
!in a lossy spherical dielectric irradiated by two planar EM radiation
!sources. These sources are arranged 180 deg apart and both point at the
!sphere. One source is placed directly above the sphere, and the other placed
!directly below the sphere. The E components of both EM waves are polarized
!along the "+" x-axis.
!One H component of the EM wave (top side) is polarized along the "+" y-axis,
!while the other component (bottom side) is polarized along the "-" y-axis.
!values are calculated (and stored) at specific r, theta, and phi values;
!these steps are regulated by pre-arranged step sizes. This version of the
!program only calculates E, H, and P for the region inside the sphere.

implicit none



!<<<< user modifiable values >>>>

! E-field constant (V/m)
real (kind=8) :: Econst_o = 4754.3

!number of summed terms for bessel, hankel and legendre functions
integer :: MaxN = 50

!radius of sphere (m)
```

```
real (kind=8) :: a = 0.01D+00

!max radius E and H-field is calculated to (m)
real (kind=8) :: b = 0.01D+00

!frequency of EM wave
real (kind=8) :: f = 2800.0D+6

!<<<<     --------------     >>>>



!more constants
!moved PI to here to fix NaN issue with results.
real (kind=8) :: PI = 4.0D+00*datan(1.0D+00)
real (kind=8) :: uo, w
real (kind=8) :: eo = 8.8541878176D-12



!<<<< user modifiable values >>>>

!dielectric constant
real (kind=8) :: erp = 33.6D+00
!dielectric loss
real (kind=8) :: erdp = 12.6D+00
!set to "1" if material other than ferrite
real (kind=8) :: urp = 1.0D+00
!set to "0" if material other than ferrite
real (kind=8) :: urdp = 0.0D+00

!<<<<  ------------------   >>>>



!characters for printing data description fields for display in data files
character :: radiusTxt*5 = 'r (m)'
character :: ndRadiusTxt*8 = 'r (n.d.)'
character :: thetaTxt*8 = 'Th (deg)'
character :: thetaRtxt*8 = 'Th (rad)'
character :: phiTxt*8 = 'Ph (deg)'
character :: phiRtxt*8 = 'Ph (rad)'

character :: EabsTxt*11 = 'E abs (V/m)'

character :: HabsTxt*11 = 'H abs (A/m)'

character :: PabsTxt*13 = 'P abs (W/m^3)'



!declaring more vars

complex (kind=8) :: j = (0.0D+00,1.0D+00)

complex (kind=8) :: er, ed

complex (kind=8) :: ur, ud

complex (kind=8) :: Bo, Bd

complex (kind=8) :: no, nd



!vars for calc E, H, and P abs at specific r, theta, phi

real (kind=8) :: rStep, thetaStep, thetaMin, thetaMax, phiStep

real (kind=8) :: r, theta, thetaR, phi, phiR, x

real (kind=8) :: Esquared, Hsquared, Pabs, ans_EsquaredDblReal, ans_HsquaredDblReal, ans_Pabs

integer :: rIntStep, rMaxStep, thetaIntStep, thetaMaxStep, phiIntStep, phiMaxStep

integer :: aMax



!vars for E, H, and P abs estimation at r=0

integer :: counter, rIntStep2, thetaIntStep2, phiIntStep2

integer :: rIntStep3, thetaIntStep3, phiIntStep3

real (kind=8) :: EabsZero, HabsZero, PabsZero

real (kind=8) :: EabsTemp, HabsTemp, PabsTemp

real (kind=8) :: EabsZeroFinal, HabsZeroFinal, PabsZeroFinal



!vars and arrays for building printable 2-dim arrays for E, H, and P abs

integer :: maxRow, maxCol, col_E, col_H, col_P

integer :: counterE, counterH, counterP

integer :: rIntStepE, thetaIntStepE, phiIntStepE

integer :: rIntStepH, thetaIntStepH, phiIntStepH

integer :: rIntStepP, thetaIntStepP, phiIntStepP

real (kind=8), allocatable, dimension(:,:) :: EabsDblWrite

real (kind=8), allocatable, dimension(:,:) :: HabsDblWrite
real (kind=8), allocatable, dimension(:,:) :: PabsDblWrite

!additional vars for loop control when writing results to text files
integer :: wCE, wCH, wCP


!declaring 3-dim arrays for storing E, H and P results. size is variable.
real (kind=8), allocatable, dimension(:,:,:) :: EabsDbl3Darray
real (kind=8), allocatable, dimension(:,:,:) :: HabsDbl3Darray
real (kind=8), allocatable, dimension(:,:,:) :: PabsDbl3Darray


!declaring 1-dim arrays for storing r, theta, phi values as they are stepped from min
!to maximum values
real (kind=8), allocatable, dimension(:) :: rArray
real (kind=8), allocatable, dimension(:) :: rNoDimArray
real (kind=8), allocatable, dimension(:) :: thetaArray
real (kind=8), allocatable, dimension(:) :: phiArray


!<<<< user modifiable values >>>>

!this dimension is in meters. this variable sets the step size for "r" as it progresses
!from the center of the sphere (r=0) to the max radius that E, H, and P is calculated to (r=b).
!try to select step size that divides "a" without a remainder.
rStep = 0.002D+00

!this dimension is in degrees. this is the size of the steps that the E and H
!calc will step through as phi progresses from 0 to 180 deg.
thetaStep = 30.0D+00

!this is the absolute min value for theta that will be considered 0 deg in
!calcs. choose thetaStep, thetaMin, and thetaMax carefully! cannot = 0 exactly!
thetaMin = 0.01D+00

!this is the absolute max value for theta that will be considered 180 deg
!in calcs. choose thetaStep, thetaMin, and thetaMax carefully! cannot = 180 exactly!
thetaMax = 179.99D+00
```

```
!this dimension is in degrees. this is the size of the steps that the E and H
!calc will step through as phi progresses from 0 to 360 deg.
phiStep = 60.0D+00

!<<<< -------------------- >>>>



!assigning values needed to send to calc subroutines


uo = 4*PI*1.0D-7
!angular frequency
w = 2*PI*f
!another error the compiler didn't catch!
!PI = 4.0D+00*datan(1.0D+00)



!further assignments needed to generate values to send to calc subroutines


er = erp - j*erdp
ur = urp - j*urdp
ed = er*eo
ud = ur*uo
Bo = w*dsqrt(eo*uo)
Bd = Bo*zsqrt(er*ur)
no = dsqrt(uo/eo)
nd = no*zsqrt(ur/er)



!determines the total number of steps (nearest integer [round up]) from
!r-min (r=0) to r-max (r=b)
rMaxStep = CEILING(b/rStep);

!needed as MaxStep=1 when r=0.
rMaxStep = rMaxStep + 1;

!determines the total number of steps (nearest integer) from
!r-min (r=0) to r-sphere (r=a)
```

```
aMax = NINT(a/rStep);


!needed since aMax=1 when r=0.

aMax = aMax+1;


!number of (integer) steps to step theta through calculations. rounding

!down to ensure total angle range is <= 180.

thetaMaxStep = FLOOR((180.0D+00)/thetaStep);


!this is so theta = 0 is included

thetaMaxStep = thetaMaxStep + 1;


!number of (integer) steps to step phi through calculations. rounding

!down to ensure total angle range is <= 360.

phiMaxStep = FLOOR((360.0D+00)/phiStep);


!this is so phi = 0 is included

phiMaxStep = phiMaxStep + 1;




!settting sizes for arrays


!three 3D arrays to hold calc'd values.

allocate(EabsDbl3Darray(rMaxStep,thetaMaxStep,phiMaxStep))

allocate(HabsDbl3Darray(rMaxStep,thetaMaxStep,phiMaxStep))

allocate(PabsDbl3Darray(rMaxStep,thetaMaxStep,phiMaxStep))


!these arrays will be used later when formatting the data from the above 3D arrays to allow
writing to files.

allocate(rArray(rMaxStep))

allocate(rNoDimArray(rMaxStep))

allocate(thetaArray(thetaMaxStep))

allocate(phiArray(phiMaxStep))




!set values at r=0


rArray(1) = 0.0D+00
```

```
rNoDimArray(1) = 0.0D+00




!<<<< the following sequence of code would be good to move to a subroutine in a future version of
this program >>>>


!hoping for more precision using integer rather than r (real number) directly

do rIntStep = 2,aMax,1


    !using integer to step through radius sizes from r-min to r-max. easier

    !to store integer steps in 3D array. deliberately avoiding case of r=0

    !for now (will address later in program)

    r = 0.0D+00 + rStep*(rIntStep - 1);


    !recording actual value of r to its own array

    rArray(rIntStep) = r;


    !recording actual value of dimensionless r to its own array. since the

    !dielectric sphere is the main item of intrest, the true radius (a) is

    !used to non-dimensionalize.

    rNoDimArray(rIntStep) = r/a;


    !using integers to progress theta angle from 0 to 180 deg (much like for r).

    do thetaIntStep = 1,thetaMaxStep,1


        !using integer to step through radius sizes from theta-min to

        !theta-max. easier to store integer steps in 3D array. reducing int

        !stepper by 1 in below eqn to make math correct.

        theta = 0.0D+00 + thetaStep*(thetaIntStep - 1); !angle in deg


        !recording actual value of theta to its own array

        thetaArray(thetaIntStep) = theta;


        !actual value used in following E/H calcs

        thetaR = theta*(PI/(180.0D+00)); !radians
```

```
!the following two if statements adjust the value of theta used in
!the calc equations when the true value of theta actually equals 0
!or 180. this is done to keep theta values from reaching the
!calculating equations that would cause the associated function to
!crash. in the case of theta, 0 and 180 will cause the
!calc equations to fail. a sufficient estimate for theta in
!these cases is a slight off centering from the true value of theta.
!this slight variation can be adjusted by thetaMin and thetaMax
!vars.

!allowing some wiggle room in case theta does not exactly equal 0.
if (theta < (0.0D+00 + (0.5D+00)*thetaStep)) then

    !this is to approximate theta = 0 as close as possible.
    !following E/H calcs blows up if theta = 0 (exactly).
    theta = thetaMin;

    !actual value used in following E/H calcs
    thetaR = theta*(PI/(180.0D+00)); !radians

end if

!allowing some wiggle room in case theta does not exactly equal 180.
if (theta > (180.0D+00 - (0.5D+00)*thetaStep)) then

    !this is to approximate theta = 180 as close as possible.
    !following E/H calcs blows up if theta = 180 (exactly).
    theta = thetaMax;

    !actual value used in following E/H calcs
    thetaR = theta*(PI/(180.0D+00)); !radians

end if

! -1 < x < 1. used for associated legendre polynomials.
x = dcos(thetaR);

!using integers to progress phi angle from 0 to 360 deg (much
        !like for r and theta).
        do phiIntStep = 1,phiMaxStep,1


            !using integer to step through radius sizes from phi-min to
            !phi-max. easier to store integer steps in 3D array. reducing
            !int stepper by 1 in below eqn to make math correct.
            phi = 0.0D+00 + phiStep*(phiIntStep - 1); !angle in deg


            !storing the current value of phi (in deg)
            phiArray(phiIntStep) = phi;


            phiR = phi*(PI/(180.0D+00)); !conversion to radians





            !!test for math errors
            !print*, ' r is: ',r
            !calculating E squared before calc E abs. using temp var to hold
            !answer
            call
EdblSquaredStep(PI,MaxN,Bo,Bd,er,ur,r,thetaR,phiR,x,Econst_o,a,b,rMaxStep,aMax,rIntStep,ans_Esqua
redDblReal)


            !!test for math errors
            !print*, 'before assignment, Esquared is: ',Esquared, 'ans_EsquaredDblReal is:
',ans_EsquaredDblReal


            Esquared = ans_EsquaredDblReal


            !!test for math errors
            !print*, 'after assignment, Esquared is: ',Esquared


            !calculating H squared before calc H abs. using temp var to hold
            !answer.
            call
HdblSquaredStep(PI,MaxN,Bo,Bd,er,ur,no,nd,r,thetaR,phiR,x,Econst_o,a,b,rMaxStep,aMax,rIntStep,ans
_HsquaredDblReal)


            !!test for math errors
            !print*, 'before assignment, Hsquared is: ',Hsquared, 'ans_HsquaredDblReal is:
',ans_HsquaredDblReal
```

```
Hsquared = ans_HsquaredDblReal

!!test for math errors
!print*, 'after assignment, Hsquared is: ',Hsquared

!very straight forward! again, using temp var.
call PowerAbsoluteSphereStep( w,eo,erdp,uo,urdp,Esquared,Hsquared,ans_Pabs)

!!test for math errors
!print*, 'ans_Pabs is: ',ans_Pabs

Pabs = ans_Pabs

!!test for math errors
!print*, 'Pabs is: ',Pabs



!the actual E abs value being stored in 3D array. this is for
!specific r, theta, phi value.
EabsDbl3Darray(rIntStep,thetaIntStep,phiIntStep) = dsqrt(Esquared);

!!test for math errors
!print*, 'final E abs value: ', EabsDbl3Darray(rIntStep,thetaIntStep,phiIntStep)

!the actual H abs value being stored in 3D array. this is for
!specific r, theta, phi value.
HabsDbl3Darray(rIntStep,thetaIntStep,phiIntStep) = dsqrt(Hsquared);

!!test for math errors
!print*, 'final H abs value: ', HabsDbl3Darray(rIntStep,thetaIntStep,phiIntStep)

!trivial, but simpler this way.
PabsDbl3Darray(rIntStep,thetaIntStep,phiIntStep) = Pabs;

!!test for math errors
!print*, 'final P abs value: ',PabsDbl3Darray(rIntStep,thetaIntStep,phiIntStep)
!print*, ' r is: ',r
```

```
        end do

    end do

end do

!<<<< ---------------------------------------------------------------------------- >>>>



!<<<< time to estimate E, H, and P at r=0. this section of code would also make a good candidate
!for placement in a subroutine, at a future date. >>>>



!this var keeps running total of all the data points, holding E, H, and P values,
!that are used in estimating E, H, and P abs at r=0.

counter = 0

EabsZero = 0.0D+00

!+++++ KILL ++++++++
!HabsZero = 0.0D+00
!+++++ END KILL ++++

PabsZero = 0.0D+00



!needed to access correct r value (one step out from r=0) in 3D arrays
rIntStep2 = 2



    !theta bounded to a certain range (all values not 0 or 180 deg) in this
    !portion of code.

    do thetaIntStep2 = 2,(thetaMaxStep - 1),1
```

```
        do phiIntStep2 = 1,(phiMaxStep - 1),1 !don't need dup. value at 360 deg.

            counter = counter + 1;



            !reading and storing E value as running total

            EabsTemp = EabsDbl3Darray(rIntStep2,thetaIntStep2,phiIntStep2);


            EabsZero = EabsZero + EabsTemp;


            !++++++++++++++++++++ KILL ++++++++++++++++++++

            !!reading and storing H value as running total

            !HabsTemp = HabsDbl3Darray(rIntStep2,thetaIntStep2,phiIntStep2);

            !

            !HabsZero = HabsZero + HabsTemp;

            !++++++++++++++++++ END KILL ++++++++++++++++++


            !reading and storing P value as running total

            PabsTemp = PabsDbl3Darray(rIntStep2,thetaIntStep2,phiIntStep2);


            PabsZero = PabsZero + PabsTemp;


        end do


    end do



!special case for when theta = 0 deg


rIntStep3 = 2


thetaIntStep3 = 1


phiIntStep3 = 1



!keep reading and storing in running totals
```

```
EabsTemp = EabsDbl3Darray(rIntStep3,thetaIntStep3,phiIntStep3)

EabsZero = EabsZero + EabsTemp

!++++++++++++++++++++ KILL ++++++++++++++++++++++++++++++++++++
!HabsTemp = HabsDbl3Darray(rIntStep3,thetaIntStep3,phiIntStep3)
!
!HabsZero = HabsZero + HabsTemp
!+++++++++++++++++++ END KILL +++++++++++++++++++++++++++++++++

PabsTemp = PabsDbl3Darray(rIntStep3,thetaIntStep3,phiIntStep3)

PabsZero = PabsZero + PabsTemp

counter = counter + 1


!special case for when theta = 180 deg

rIntStep3 = 2

thetaIntStep3 = thetaMaxStep

phiIntStep3 = 1


!keep reading and storing in running totals

EabsTemp = EabsDbl3Darray(rIntStep3,thetaIntStep3,phiIntStep3)

EabsZero = EabsZero + EabsTemp

!++++++++++++++++++++++++ KILL ++++++++++++++++++++++++++++++++
!HabsTemp = HabsDbl3Darray(rIntStep3,thetaIntStep3,phiIntStep3)
!
!HabsZero = HabsZero + HabsTemp
!++++++++++++++++++++++ END KILL ++++++++++++++++++++++++++++++
PabsTemp = PabsDbl3Darray(rIntStep3,thetaIntStep3,phiIntStep3)

PabsZero = PabsZero + PabsTemp

counter = counter + 1


!finally, time to determine best estimate of E, H, and P abs r=0 values.

EabsZeroFinal = EabsZero/counter

!correction to code. H = 0 (always) when r = 0 for this EM wave configuration.
!HabsZeroFinal = HabsZero/counter
HabsZeroFinal = 0.0D+00

PabsZeroFinal = PabsZero/counter

!<<<< -------------------------------------------------------------------- >>>>



!<<<< the following sequence of code should be moved to a subroutine at a later date. >>>>


!this var determines the max number of rows needed to store the E, H, and P abs data in a
!2D array.
maxRow = (aMax - 1)*(thetaMaxStep)*(phiMaxStep) + 1

!this var only partially functional as total # of data columns have to be manually adjusted and
!the data writen to the files are based on what the user wants to see.
maxCol = 7

!sizing the 2-dim arrays for use later in the program
allocate(EabsDblWrite(maxRow,maxCol))
allocate(HabsDblWrite(maxRow,maxCol))
allocate(PabsDblWrite(maxRow,maxCol))

!<<<< ---------------------------------------------------------------- >>>>
```

```
!<<<<this sequence of code will appropriately fill the EabsDblW array with data and then
!write the results to a file. said code should be moved to a subroutine in future. >>>>


!this bit of code takes care of the case when r=0. all values are "0" at r=0, except for E.
do col_E = 1,(maxCol-1),1
    EabsDblWrite(1,col_E) = 0.0D+00
end do

!value for Eabs when r=0
EabsDblWrite(1,maxCol) = EabsZeroFinal

!this var ensures the data from the various original arrays is placed in
!the correct row of the final 2D array
counterE = 1;

!writing the data to the E abs 2D array for eventual storage as a text file
do rIntStepE = 2,aMax,1

    do thetaIntStepE = 1,thetaMaxStep,1

        do phiIntStepE = 1,phiMaxStep,1

            counterE = counterE + 1

            !these series of commands writes the data collected in the
            !various arrays and places them in a 2D array. not able to realy automate column
            !selection as final output in file is dependent on what user wants to see.
            EabsDblWrite(counterE,1) = rArray(rIntStepE)
            EabsDblWrite(counterE,2) = rNoDimArray(rIntStepE)
            EabsDblWrite(counterE,3) = thetaArray(thetaIntStepE)
            EabsDblWrite(counterE,4) = thetaArray(thetaIntStepE)*(PI/(180.0D+00))
            EabsDblWrite(counterE,5) = phiArray(phiIntStepE)
            EabsDblWrite(counterE,6) = phiArray(phiIntStepE)*(PI/(180.0D+00))
            EabsDblWrite(counterE,7) = EabsDbl3Darray(rIntStepE,thetaIntStepE,phiIntStepE)
        end do

    end do

end do



!code sequence to write results to file

open(20, file='EabsDblSphData_f2800_r0p01_coarse_test.txt')
 write(20,10) radiusTxt, ndRadiusTxt, thetaTxt, thetaRtxt, phiTxt, phiRtxt, EabsTxt
 10 format(a9, a9, a9, a9, a9, a9, a20)

do wCE = 1,maxRow,1
    !have to keep total width of one line of code under 132 characters
   write(20,11) EabsDblWrite(wCE,1), EabsDblWrite(wCE,2), EabsDblWrite(wCE,3), &
   EabsDblWrite(wCE,4), EabsDblWrite(wCE,5), EabsDblWrite(wCE,6), EabsDblWrite(wCE,7)
   11 format(f9.5, f9.6, f9.4, f9.6, f9.4, f9.6, f20.10)
end do

close(20)



!<<<< ----------------------------------------------------------------- >>>>




!<<<<this sequence of code will appropriately fill the HabsDblW array with data and then
!write the results to a file. said code should be moved to a subroutine in future. >>>>



!this bit of code takes care of the case when r=0. all values are "0" at r=0, except for H.
do col_H = 1,(maxCol-1),1
    HabsDblWrite(1,col_H) = 0.0D+00
end do
```

```
!value for Habs when r=0
HabsDblWrite(1,maxCol) = HabsZeroFinal


!this var ensures the data from the various original arrays is placed in
!the correct row of the final 2D array
counterH = 1;


!writing the data to the H abs 2D array for eventual storage as a text file
do rIntStepH = 2,aMax,1

    do thetaIntStepH = 1,thetaMaxStep,1

        do phiIntStepH = 1,phiMaxStep,1

            counterH = counterH + 1

            !these series of commands writes the data collected in the
            !various arrays and places them in a 2D array. not able to realy automate column
            !selection as final output in file is dependent on what user wants to see.
            HabsDblWrite(counterH,1) = rArray(rIntStepH)
            HabsDblWrite(counterH,2) = rNoDimArray(rIntStepH)
            HabsDblWrite(counterH,3) = thetaArray(thetaIntStepH)
            HabsDblWrite(counterH,4) = thetaArray(thetaIntStepH)*(PI/(180.0D+00))
            HabsDblWrite(counterH,5) = phiArray(phiIntStepH)
            HabsDblWrite(counterH,6) = phiArray(phiIntStepH)*(PI/(180.0D+00))
            HabsDblWrite(counterH,7) = HabsDbl3Darray(rIntStepH,thetaIntStepH,phiIntStepH)

        end do

    end do

end do


!code sequence to write results to file
```

```
open(21, file='HabsDblSphData_f2800_r0p01_coarse_test.txt')

 write(21,12) radiusTxt, ndRadiusTxt, thetaTxt, thetaRtxt, phiTxt, phiRtxt, HabsTxt

 12 format(a9, a9, a9, a9, a9, a9, a20)


do wCH = 1,maxRow,1

    !have to keep total width of one line of code under 132 characters

   write(21,13) HabsDblWrite(wCH,1), HabsDblWrite(wCH,2), HabsDblWrite(wCH,3),
HabsDblWrite(wCH,4), &

   HabsDblWrite(wCH,5), HabsDblWrite(wCH,6), HabsDblWrite(wCH,7)

   13 format(f9.5, f9.6, f9.4, f9.6, f9.4, f9.6, f20.10)

end do


close(21)



!<<<< ------------------------------------------------------------------ >>>>




!<<<<this sequence of code will appropriately fill the PabsDblW array with data and then

!write the results to a file. said code should be moved to a subroutine in future. >>>>



!this bit of code takes care of the case when r=0. all values are "0" at r=0, except for P.

do col_P = 1,(maxCol-1),1

    PabsDblWrite(1,col_P) = 0.0D+00

end do



!value for Pabs when r=0

PabsDblWrite(1,maxCol) = PabsZeroFinal



!this var ensures the data from the various original arrays is placed in

!the correct row of the final 2D array

counterP = 1;
```

```
!writing the data to the H abs 2D array for eventual storage as a text file
do rIntStepP = 2,aMax,1


    do thetaIntStepP = 1,thetaMaxStep,1


        do phiIntStepP = 1,phiMaxStep,1


            counterP = counterP + 1


            !these series of commands writes the data collected in the
            !various arrays and places them in a 2D array. not able to realy automate column
            !selection as final output in file is dependent on what user wants to see.
            PabsDblWrite(counterP,1) = rArray(rIntStepP)
            PabsDblWrite(counterP,2) = rNoDimArray(rIntStepP)
            PabsDblWrite(counterP,3) = thetaArray(thetaIntStepP)
            PabsDblWrite(counterP,4) = thetaArray(thetaIntStepP)*(PI/(180.0D+00))
            PabsDblWrite(counterP,5) = phiArray(phiIntStepP)
            PabsDblWrite(counterP,6) = phiArray(phiIntStepP)*(PI/(180.0D+00))
            PabsDblWrite(counterP,7) = PabsDbl3Darray(rIntStepP,thetaIntStepP,phiIntStepP)


        end do


    end do


end do



!code sequence to write results to file


open(22, file='PabsDblSphData_f2800_r0p01_coarse_test.txt')
 write(22,14) radiusTxt, ndRadiusTxt, thetaTxt, thetaRtxt, phiTxt, phiRtxt, PabsTxt
 14 format(a9, a9, a9, a9, a9, a9, a21)


do wCP = 1,maxRow,1
    !have to keep total width of one line of code under 132 characters
   write(22,15) PabsDblWrite(wCP,1), PabsDblWrite(wCP,2), PabsDblWrite(wCP,3),
PabsDblWrite(wCP,4), &
   PabsDblWrite(wCP,5), PabsDblWrite(wCP,6), PabsDblWrite(wCP,7)
   15 format(f9.5, f9.6, f9.4, f9.6, f9.4, f9.6, 2E21.10)
end do

close(22)


!<<<< ----------------------------------------------------------------- >>>>



end program EHPSphCalc



!+++++++++++++++++++++++++++++++++++++++++++++++++++++++
subroutine PowerAbsoluteSphereStep(w,eo,erdp,uo,urdp,Esquared,Hsquared, ans_Pabs)
!this function calculates the absolute (generated) power within a lossy
!dielectric irradiated by a planar EM wave (of any configuration).
!The power is calculated at a
!specific r, theta and phi value (a point value), either inside or outside
!the sphere.

implicit none

!setting up the vars
real (kind = 8) :: w, eo, erdp, uo, urdp, Esquared, Hsquared, ans_Pabs

!the squared terms must have no imaginary components to them as this power
!equation only can work with real numbers.
ans_Pabs = ((1.0D+00)/(2.0D+00))*w*eo*erdp*Esquared + ((1.0D+00)/(2.0D+00))*w*uo*urdp*Hsquared

return
end subroutine PowerAbsoluteSphereStep



!+++++++++++++++++++++++++++++++++++++++++++++++++++++++
subroutine
EdblSquaredStep(PI,n,Bo,Bd,er,ur,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_EsquaredDblReal)
```

```
!calc's the absolute value of the complete E-field, at a specific r, theta, and phi, for the case
!of a spherical dielectric being irradiated by two planar EM waves, both facing the sphere, but
!spaced 180 degrees apart. The E components
!of the incoming EM waves are both assumed to be polarized along the positive x axis.

implicit none

!setting up the vars
real (kind = 8) :: PI, r, ThR, PhR, x, Eo, a, b
integer :: n, MaxStep, aMax, step
complex (kind = 8) :: er, ur, Bo, Bd

!final values for each component of E (r, theta, phi)
complex (kind = 8) :: ans_ErDbl, ans_EthetaDbl, ans_EphiDbl, EsquaredDbl

!final result must be real
real (kind = 8) :: ans_EsquaredDblReal

!calling the functions
call ErDblSphereStep(PI,n,Bo,Bd,er,ur,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_ErDbl)
call EthetaDblSphereStep(PI,n,Bo,Bd,er,ur,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_EthetaDbl)
call EphiDblSphereStep(PI,n,Bo,Bd,er,ur,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_EphiDbl)

!!test for computational errors
!print*, 'r = ', r
!print*, 'ThR = ', ThR
!print*, 'PhR = ', PhR
!print*, 'ans_ErDbl = ',ans_ErDbl
!print*, 'ans_EthetaDbl = ',ans_EthetaDbl
!print*, 'ans_EphiDbl = ',ans_EphiDbl

!finding the squared result from the three (complex) components of E (r, theta, phi)
EsquaredDbl =
(ans_ErDbl*DCONJG(ans_ErDbl)+ans_EthetaDbl*DCONJG(ans_EthetaDbl)+ans_EphiDbl*DCONJG(ans_EphiDbl))



!!test for computational errors
!print*, 'EsquaredDbl = ',EsquaredDbl
```

```
!only need the real component

ans_EsquaredDblReal = REAL((EsquaredDbl),8)


!!test for computational errors

!print*, 'ans_EsquaredDblReal = ',ans_EsquaredDblReal



return

end subroutine EdblSquaredStep




!+++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine
HdblSquaredStep(PI,n,Bo,Bd,er,ur,no,nd,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_HsquaredDblReal)

!calc's the absolute value of the complete H-field, at a specific r, theta, and phi, for the case

!of a spherical dielectric being irradiated by two planar EM waves, both facing the sphere, but

!spaced 180 degrees apart. One H component

!of the EM wave (top side) is polarized along the "+" y-axis, while the other component (bottom
side)

!is polarized along the "-" y-axis.


implicit none


!setting up the vars

real (kind = 8) :: PI, r, ThR, PhR, x, Eo, a, b

integer :: n, MaxStep, aMax, step

complex (kind = 8) :: er, ur, no, nd, Bo, Bd


!final values for each component of H (r, theta, phi)

complex (kind = 8) :: ans_HrDbl, ans_HthetaDbl, ans_HphiDbl, HsquaredDbl


!final result must be real

real (kind = 8) :: ans_HsquaredDblReal


!calling the functions

call HrDblSphereStep(PI,n,Bo,Bd,er,ur,no,nd,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_HrDbl)
```

```
call
HthetaDblSphereStep(PI,n,Bo,Bd,er,ur,no,nd,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_HthetaDbl)

call HphiDblSphereStep(PI,n,Bo,Bd,er,ur,no,nd,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_HphiDbl)


!!test for computational errors

!print*, 'r = ', r

!print*, 'ThR = ', ThR

!print*, 'PhR = ', PhR

!print*, 'ans_HrDbl = ',ans_HrDbl

!print*, 'ans_HthetaDbl = ',ans_HthetaDbl

!print*, 'ans_HphiDbl = ',ans_HphiDbl


!finding the squared result from the three (complex) components of H (r, theta, phi)

HsquaredDbl =
(ans_HrDbl*DCONJG(ans_HrDbl)+ans_HthetaDbl*DCONJG(ans_HthetaDbl)+ans_HphiDbl*DCONJG(ans_HphiDbl))



!!test for computational errors

!print*, 'HsquaredDbl = ',HsquaredDbl


!only need the real component

ans_HsquaredDblReal = REAL((HsquaredDbl),8)


!!test for computational errors

!print*, 'ans_HsquaredDblReal = ',ans_HsquaredDblReal



return

end subroutine HdblSquaredStep





!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine ErDblSphereStep(PI,n,Bo,Bd,er,ur,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_ErDbl)

!calc's r component of E-field for the case of a spherical dielectric being irradiated by

!two planar EM waves, both facing the sphere, but spaced 180 degrees apart. Both E components

!of the EM wave are assumed to be polarized along the positive x axis.


implicit none
```

```
!setting up the vars

real (kind = 8) :: PI, r, ThR, PhR, x, Eo, a, b

integer :: m, n, p, MaxStep, aMax, step

complex (kind = 8) :: er, ur, Bo, Bd, j


!final value to send out of subroutine is a single value of E for a single value of r

complex (kind = 8) :: ans_ErDbl


!these subroutine calls have to be formatted in arrays

complex (kind = 8) :: dBr2MSBJ_Bo(0:n)

complex (kind = 8) :: dBr2MSH2_Bo(0:n)

complex (kind = 8) :: MSBJ_Bo(0:n)

complex (kind = 8) :: MSH2_Bo(0:n)

complex (kind = 8) :: MSBJ_Bd(0:n)

complex (kind = 8) :: dBr2MSBJ_Bd(0:n)

!the commented out subroutine calls, as seen below, are to make it easier to adapt this
subroutine

!to other related subroutines

complex (kind = 8) :: ans_cn(0:n), ans_en(0:n), ans_gn(0:n)!, ans_dn(0:n), ans_fn(0:n),
ans_hn(0:n)

real (kind = 8) :: ans_P1n(0:n)!, ans_dx_P1n(0:n), ans_P1n_div_SinThR(0:n)


!temporary holding locations for running total of E (Etotr) and each value of E at

!each instance of "n" (Ern).

complex (kind = 8) :: Etotr, Ern


!for complex values

j = (0.0D+00, 1.0D+00)


!calling the subroutines needed to determine Er

call ModSphBesJ(PI,n,Bd,r,MSBJ_Bd(0:n))

call dBr2ModSphBesJ(PI,n,Bd,r,dBr2MSBJ_Bd(0:n))


call ModSphBesJ(PI,n,Bo,r,MSBJ_Bo(0:n))

call dBr2ModSphBesJ(PI,n,Bo,r,dBr2MSBJ_Bo(0:n))

call ModSphHnk2(PI,n,Bo,r,MSH2_Bo(0:n))

call dBr2ModSphHnk2(PI,n,Bo,r,dBr2MSH2_Bo(0:n))
call cn_dbl(n,ans_cn(0:n))

call en_dbl(n,PI,Bo,Bd,a,er,ur,ans_en(0:n))

call gn_dbl(n,PI,Bo,Bd,a,er,ur,ans_gn(0:n))


call AscLegendre(n,x,ans_P1n(0:n))


!!test for math errors

!print*,'Inside ErDblSphereStep subroutine'

!print*,'r = ',r,' ThR = ',ThR,' PhR = ', PhR


!!test for NaN error

!print*, 'Bo in ErdblSphereStep = ',Bo,'. Bd in ErdblSphereStep = ',Bd


!this if statement is for when "r" is less than or equal to the radius of the sphere.

!"aMax" is the number of sequential steps from r=0 to r=a and "MaxStep" is the total number of
"r" steps

!that the E-field is calculated at each instance of "r", and may be greater than the number of
"r" steps that

!count out to the radius of the sphere.

if (step <= aMax .and. step <= MaxStep) then

    Etotr = (0.0D+00,0.0D+00)


    do m = 1,n,1

        Ern = (j*Eo*dcos(PhR))*ans_gn(m)*(dBr2MSBJ_Bd(m) + MSBJ_Bd(m))*ans_P1n(m)


        Etotr = Etotr + Ern


        !!test for math errors

        !print*,'n = ',m,' dBr2MSBJ_Bd = ',dBr2MSBJ_Bd(m)

        !print*,'n = ',m,' MSBJ_Bd = ',MSBJ_Bd(m)


        !!test to find Nan error

        !print*, 'gn for n = ',m,' in inner loop is: ',ans_gn(m)

        !print*, 'Etotr for n = ',m,' in inner loop is: ',Etotr


    end do


end if
```

```
!this if statement is for when r is greater than the radius of the sphere
if (step > aMax .and. step <= MaxStep) then
    Etotr = (0.0D+00,0.0D+00)

    do p = 1,n,1
        Ern = (j*Eo*dcos(PhR))*(ans_cn(p)*(dBr2MSBJ_Bo(p) + MSBJ_Bo(p)) &
        + ans_en(p)*(dBr2MSH2_Bo(p) + MSH2_Bo(p)))*ans_P1n(p)

        Etotr = Etotr + Ern

        !!test for math errors
        !print*,'n = ',p,' dBr2MSBJ_Bo = ',dBr2MSBJ_Bo(p)
        !print*,'n = ',p,' MSBJ_Bo = ',MSBJ_Bo(p)
        !print*,'n = ',p,' dBr2MSH2_Bo = ',dBr2MSH2_Bo(p)
        !print*,'n = ',p,' MSH2_Bo = ',MSH2_Bo(p)

        !!test to find Nan error
        !print*, 'cn for n = ',p,' in inner loop is: ',ans_cn(p)
        !print*, 'en for n = ',p,' in inner loop is: ',ans_en(p)
        !print*, 'Etotr for n = ',p,' in outer loop is: ',Etotr

    end do

end if

!the final result!
ans_ErDbl = Etotr

!!test for math errors
!print*,'Exiting ErDblSphereStep subroutine'

return
end subroutine ErDblSphereStep




!+++++++++++++++++++++++++++++++++++++++++++++++++++++++
subroutine HrDblSphereStep(PI,n,Bo,Bd,er,ur,no,nd,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_HrDbl)
```

```
!calc's r component of H-field for the case of a spherical dielectric being irradiated by

!two planar EM waves, both facing the sphere, but spaced 180 degrees apart. One H component

!of the EM wave (top side) is polarized along the "+" y-axis, while the other component (bottom
side)

!is polarized along the "-" y-axis.


implicit none


!setting up the vars

real (kind = 8) :: PI, r, ThR, PhR, x, Eo, a, b

integer :: m, n, p, MaxStep, aMax, step

complex (kind = 8) :: er, ur, no, nd, Bo, Bd, j


!final value to send out of subroutine is a single value of H for a single value of r

complex (kind = 8) :: ans_HrDbl


!these subroutine calls have to be formatted in arrays

complex (kind = 8) :: dBr2MSBJ_Bo(0:n)

complex (kind = 8) :: dBr2MSH2_Bo(0:n)

complex (kind = 8) :: MSBJ_Bo(0:n)

complex (kind = 8) :: MSH2_Bo(0:n)

complex (kind = 8) :: MSBJ_Bd(0:n)

complex (kind = 8) :: dBr2MSBJ_Bd(0:n)

!the commented out subroutine calls, as seen below, are to make it easier to adapt this
subroutine

!to other related subroutines

complex (kind=8) :: ans_dn(0:n), ans_fn(0:n), ans_hn(0:n)!, ans_dn(0:n), ans_fn(0:n), ans_hn(0:n)

real (kind=8) :: ans_P1n(0:n)!, ans_dx_P1n(0:n), ans_P1n_div_SinThR(0:n)


!temporary holding locations for running total of H (Htotr) and each value of H at

!each instance of "n" (Hrn).

complex (kind=8) :: Htotr, Hrn


!for complex values

j = (0.0D+00, 1.0D+00)


!calling the subroutines needed to determine Hr

call ModSphBesJ(PI,n,Bd,r,MSBJ_Bd(0:n))

call dBr2ModSphBesJ(PI,n,Bd,r,dBr2MSBJ_Bd(0:n))
call ModSphBesJ(PI,n,Bo,r,MSBJ_Bo(0:n))

call dBr2ModSphBesJ(PI,n,Bo,r,dBr2MSBJ_Bo(0:n))

call ModSphHnk2(PI,n,Bo,r,MSH2_Bo(0:n))

call dBr2ModSphHnk2(PI,n,Bo,r,dBr2MSH2_Bo(0:n))


call dn_dbl(n,ans_dn(0:n))

call fn_dbl(n,PI,Bo,Bd,a,er,ur,ans_fn(0:n))

call hn_dbl(n,PI,Bo,Bd,a,er,ur,ans_hn(0:n))


call AscLegendre(n,x,ans_P1n(0:n))



!this if statement is for when "r" is less than or equal to the radius of the sphere.

!"aMax" is the number of sequential steps from r=0 to r=a and "MaxStep" is the total number of
"r" steps

!that the H-field is calculated at each instance of "r", and may be greater than the number of
"r" steps that

!count out to the radius of the sphere.

if (step <= aMax .and. step <= MaxStep) then

    Htotr = (0.0D+00,0.0D+00)


    do m = 1,n,1

        Hrn = -(j*Eo*dsin(PhR)/nd)*ans_hn(m)*(dBr2MSBJ_Bd(m) + MSBJ_Bd(m))*ans_P1n(m)


        Htotr = Htotr + Hrn


    end do


end if


!this if statement is for when r is greater than the radius of the sphere

if (step > aMax .and. step <= MaxStep) then

    Htotr = (0.0D+00,0.0D+00)


    do p = 1,n,1

        Hrn = -(j*Eo*dsin(PhR)/no)*(ans_dn(p)*(dBr2MSBJ_Bo(p) + MSBJ_Bo(p)) &

        + ans_fn(p)*(dBr2MSH2_Bo(p) + MSH2_Bo(p)))*ans_P1n(p)
        Htotr = Htotr + Hrn

    end do

end if

!the final result!
ans_HrDbl = Htotr

return
end subroutine HrDblSphereStep




!+++++++++++++++++++++++++++++++++++++++++++++++++++++++
subroutine
EthetaDblSphereStep(PI,n,Bo,Bd,er,ur,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_EthetaDbl)
!calc's theta component of E-field for the case of a spherical dielectric being irradiated by
!two planar EM waves, both facing the sphere, but spaced 180 degrees apart. Both E components
!of the EM wave are assumed to be polarized along the positive x-axis.

implicit none

!setting up the vars
real (kind = 8) :: PI, r, ThR, PhR, x, Eo, a, b
integer :: m, n, p, MaxStep, aMax, step
complex (kind = 8) :: er, ur, Bo, Bd, j

!final value to send out of subroutine is a single value of E for a single value of r
complex (kind = 8) :: ans_EthetaDbl

!these subroutine calls have to be formatted in arrays
complex (kind = 8) :: dBrMSBJ_Bo(0:n)
complex (kind = 8) :: dBrMSH2_Bo(0:n)
complex (kind = 8) :: MSBJ_Bo(0:n)
complex (kind = 8) :: MSH2_Bo(0:n)
complex (kind = 8) :: MSBJ_Bd(0:n)
complex (kind = 8) :: dBrMSBJ_Bd(0:n)
```

```
!any commented out subroutine calls, if seen below, are to make it easier to adapt this
subroutine

!to other related subroutines

complex (kind = 8) :: ans_cn(0:n), ans_en(0:n), ans_gn(0:n), ans_dn(0:n), ans_fn(0:n),
ans_hn(0:n)

real (kind = 8) :: ans_dxP1n(0:n), ans_P1n_div_sinThR(0:n)!, ans_P1n(0:n)


!temporary holding locations for running total of E (Etotr) and each value of E at

!each instance of "n" (Ern).

complex (kind = 8) :: EtotTheta, Etheta_n


!for complex values

j = (0.0D+00, 1.0D+00)


!calling the subroutines needed to determine Er

call ModSphBesJ(PI,n,Bd,r,MSBJ_Bd(0:n))

call dBrModSphBesJ(PI,n,Bd,r,dBrMSBJ_Bd(0:n))


call ModSphBesJ(PI,n,Bo,r,MSBJ_Bo(0:n))

call dBrModSphBesJ(PI,n,Bo,r,dBrMSBJ_Bo(0:n))

call ModSphHnk2(PI,n,Bo,r,MSH2_Bo(0:n))

call dBrModSphHnk2(PI,n,Bo,r,dBrMSH2_Bo(0:n))


call cn_dbl(n,ans_cn(0:n))

call dn_dbl(n,ans_dn(0:n))

call en_dbl(n,PI,Bo,Bd,a,er,ur,ans_en(0:n))

call fn_dbl(n,PI,Bo,Bd,a,er,ur,ans_fn(0:n))

call gn_dbl(n,PI,Bo,Bd,a,er,ur,ans_gn(0:n))

call hn_dbl(n,PI,Bo,Bd,a,er,ur,ans_hn(0:n))


!call AscLegendre(n,x,ans_P1n(0:n))

call dxAscLegendre(n,x,ans_dxP1n(0:n))

call AscLegDivSinThR(n,x,ans_P1n_div_sinThR(0:n))



!!test for math errors

!print*,'Inside EthetaDblSphereStep subroutine'

!print*,'r = ',r,' ThR = ',ThR,' PhR = ', PhR
```

```
!!test for NaN error

!print*, 'Bo in EthetadblSphereStep = ',Bo,'. Bd in EthetadblSphereStep = ',Bd


!this if statement is for when "r" is less than or equal to the radius of the sphere.

!"aMax" is the number of sequential steps from r=0 to r=a and "MaxStep" is the total number of
"r" steps

!that the E-field is calculated at each instance of "r", and may be greater than the number of
"r" steps that

!count out to the radius of the sphere.

if (step <= aMax .and. step <= MaxStep) then

    EtotTheta = (0.0D+00,0.0D+00)


    do m = 1,n,1

        Etheta_n = (-j*Eo*dcos(PhR)/(Bd*r))*(ans_gn(m)*dBrMSBJ_Bd(m)*dsin(ThR)* &

        ans_dxP1n(m)) - (Eo*dcos(PhR)/(Bd*r))*(ans_hn(m)*MSBJ_Bd(m)*ans_P1n_div_sinThR(m))


        EtotTheta = EtotTheta + Etheta_n


        !!test for math errors

        !print*,'n = ',m,' dBrMSBJ_Bd = ',dBrMSBJ_Bd(m)

        !print*,'n = ',m,' MSBJ_Bd = ',MSBJ_Bd(m)


    end do


end if


!this if statement is for when r is greater than the radius of the sphere

if (step > aMax .and. step <= MaxStep) then

    EtotTheta = (0.0D+00,0.0D+00)


    do p = 1,n,1

        Etheta_n = (-j*Eo*dcos(PhR)/(Bo*r))*(ans_cn(p)*dBrMSBJ_Bo(p) + ans_en(p)*dBrMSH2_Bo(p)) &

        *dsin(ThR)*ans_dxP1n(p) - (Eo*dcos(PhR)/(Bo*r))*(ans_dn(p)*MSBJ_Bo(p) +
ans_fn(p)*MSH2_Bo(p)) &

        *ans_P1n_div_sinThR(p)


        EtotTheta = EtotTheta + Etheta_n


        !!test for math errors
        !print*,'n = ',p,' dBrMSBJ_Bo = ',dBrMSBJ_Bo(p)

        !print*,'n = ',p,' MSBJ_Bo = ',MSBJ_Bo(p)

        !print*,'n = ',p,' dBrMSH2_Bo = ',dBrMSH2_Bo(p)

        !print*,'n = ',p,' MSH2_Bo = ',MSH2_Bo(p)


    end do


end if


!the final result!

ans_EthetaDbl = EtotTheta


!!test for math errors

!print*,'Exiting EthetaDblSphereStep subroutine'


return

end subroutine EthetaDblSphereStep





!+++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine
HthetaDblSphereStep(PI,n,Bo,Bd,er,ur,no,nd,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_HthetaDbl)

!calc's theta component of H-field for the case of a spherical dielectric being irradiated by

!two planar EM waves, both facing the sphere, but spaced 180 degrees apart. One H component

!of the EM wave (top side) is polarized along the "+" y-axis, while the other component (bottom
side)

!is polarized along the "-" y-axis.


implicit none


!setting up the vars

real (kind = 8) :: PI, r, ThR, PhR, x, Eo, a, b

integer :: m, n, p, MaxStep, aMax, step

complex (kind = 8) :: er, ur, no, nd, Bo, Bd, j


!final value to send out of subroutine is a single value of E for a single value of r

complex (kind = 8) :: ans_HthetaDbl
```

```
!these subroutine calls have to be formatted in arrays

complex (kind = 8) :: dBrMSBJ_Bo(0:n)

complex (kind = 8) :: dBrMSH2_Bo(0:n)

complex (kind = 8) :: MSBJ_Bo(0:n)

complex (kind = 8) :: MSH2_Bo(0:n)

complex (kind = 8) :: MSBJ_Bd(0:n)

complex (kind = 8) :: dBrMSBJ_Bd(0:n)

!any commented out subroutine calls, if seen below, are to make it easier to adapt this
subroutine

!to other related subroutines

complex (kind = 8) :: ans_cn(0:n), ans_en(0:n), ans_gn(0:n), ans_dn(0:n), ans_fn(0:n),
ans_hn(0:n)

real (kind = 8) :: ans_dxP1n(0:n), ans_P1n_div_sinThR(0:n)!,ans_P1n(0:n)


!temporary holding locations for running total of E (Etotr) and each value of E at

!each instance of "n" (Ern).

complex (kind = 8) :: HtotTheta, Htheta_n


!for complex values

j = (0.0D+00, 1.0D+00)


!calling the subroutines needed to determine Er

call ModSphBesJ(PI,n,Bd,r,MSBJ_Bd(0:n))

call dBrModSphBesJ(PI,n,Bd,r,dBrMSBJ_Bd(0:n))


call ModSphBesJ(PI,n,Bo,r,MSBJ_Bo(0:n))

call dBrModSphBesJ(PI,n,Bo,r,dBrMSBJ_Bo(0:n))

call ModSphHnk2(PI,n,Bo,r,MSH2_Bo(0:n))

call dBrModSphHnk2(PI,n,Bo,r,dBrMSH2_Bo(0:n))


call cn_dbl(n,ans_cn(0:n))

call dn_dbl(n,ans_dn(0:n))

call en_dbl(n,PI,Bo,Bd,a,er,ur,ans_en(0:n))

call fn_dbl(n,PI,Bo,Bd,a,er,ur,ans_fn(0:n))

call gn_dbl(n,PI,Bo,Bd,a,er,ur,ans_gn(0:n))

call hn_dbl(n,PI,Bo,Bd,a,er,ur,ans_hn(0:n))


!call AscLegendre(n,x,ans_P1n(0:n))

call dxAscLegendre(n,x,ans_dxP1n(0:n))
call AscLegDivSinThR(n,x,ans_P1n_div_sinThR(0:n))



!this if statement is for when "r" is less than or equal to the radius of the sphere.

!"aMax" is the number of sequential steps from r=0 to r=a and "MaxStep" is the total number of
"r" steps

!that the E-field is calculated at each instance of "r", and may be greater than the number of
"r" steps that

!count out to the radius of the sphere.

if (step <= aMax .and. step <= MaxStep) then

    HtotTheta = (0.0D+00,0.0D+00)


    do m = 1,n,1

        Htheta_n = (Eo*dsin(PhR)/(Bd*nd*r))*(ans_gn(m)*MSBJ_Bd(m)*ans_P1n_div_sinThR(m)) &

         + (j*Eo*dsin(PhR)/(Bd*nd*r))*(ans_hn(m)*dBrMSBJ_Bd(m)*dsin(ThR)*ans_dxP1n(m))


        HtotTheta = HtotTheta + Htheta_n


    end do


end if


!this if statement is for when r is greater than the radius of the sphere

if (step > aMax .and. step <= MaxStep) then

    HtotTheta = (0.0D+00,0.0D+00)


    do p = 1,n,1

        Htheta_n = (Eo*dsin(PhR)/(Bo*no*r))*(ans_cn(p)*MSBJ_Bo(p) + ans_en(p)*MSH2_Bo(p)) &

        *ans_P1n_div_sinThR(p) + (j*Eo*dsin(PhR)/(Bo*no*r))*(ans_dn(p)*dBrMSBJ_Bo(p) +
ans_fn(p)*dBrMSH2_Bo(p)) &

        *dsin(ThR)*ans_dxP1n(p)


        HtotTheta = HtotTheta + Htheta_n


    end do


end if


!the final result!
```

```
ans_HthetaDbl = HtotTheta


return

end subroutine HthetaDblSphereStep





!++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine EphiDblSphereStep(PI,n,Bo,Bd,er,ur,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_EphiDbl)

!calc's phi component of E-field for the case of a spherical dielectric being irradiated by

!two planar EM waves, both facing the sphere, but spaced 180 degrees apart. Both E components

!of the EM wave are assumed to be polarized along the positive x-axis.


implicit none


!setting up the vars

real (kind = 8) :: PI, r, ThR, PhR, x, Eo, a, b

integer :: m, n, p, MaxStep, aMax, step

complex (kind = 8) :: er, ur, Bo, Bd, j


!final value to send out of subroutine is a single value of E for a single value of r

complex (kind=8) :: ans_EphiDbl


!these subroutine calls have to be formatted in arrays

complex (kind = 8) :: dBrMSBJ_Bo(0:n)

complex (kind = 8) :: dBrMSH2_Bo(0:n)

complex (kind = 8) :: MSBJ_Bo(0:n)

complex (kind = 8) :: MSH2_Bo(0:n)

complex (kind = 8) :: MSBJ_Bd(0:n)

complex (kind = 8) :: dBrMSBJ_Bd(0:n)

!any commented out subroutine calls, if seen below, are to make it easier to adapt this
subroutine

!to other related subroutines

complex (kind = 8) :: ans_cn(0:n), ans_en(0:n), ans_gn(0:n), ans_dn(0:n), ans_fn(0:n),
ans_hn(0:n)

real (kind = 8) :: ans_dxP1n(0:n), ans_P1n_div_sinThR(0:n)!,ans_P1n(0:n)


!temporary holding locations for running total of E (Etotr) and each value of E at

!each instance of "n" (Ern).
```

```
complex (kind = 8) :: EtotPhi, Ephi_n


!for complex values

j = (0.0D+00, 1.0D+00)


!calling the subroutines needed to determine Er

call ModSphBesJ(PI,n,Bd,r,MSBJ_Bd(0:n))

call dBrModSphBesJ(PI,n,Bd,r,dBrMSBJ_Bd(0:n))


call ModSphBesJ(PI,n,Bo,r,MSBJ_Bo(0:n))

call dBrModSphBesJ(PI,n,Bo,r,dBrMSBJ_Bo(0:n))

call ModSphHnk2(PI,n,Bo,r,MSH2_Bo(0:n))

call dBrModSphHnk2(PI,n,Bo,r,dBrMSH2_Bo(0:n))


call cn_dbl(n,ans_cn(0:n))

call dn_dbl(n,ans_dn(0:n))

call en_dbl(n,PI,Bo,Bd,a,er,ur,ans_en(0:n))

call fn_dbl(n,PI,Bo,Bd,a,er,ur,ans_fn(0:n))

call gn_dbl(n,PI,Bo,Bd,a,er,ur,ans_gn(0:n))

call hn_dbl(n,PI,Bo,Bd,a,er,ur,ans_hn(0:n))


!call AscLegendre(n,x,ans_P1n(0:n))

call dxAscLegendre(n,x,ans_dxP1n(0:n))

call AscLegDivSinThR(n,x,ans_P1n_div_sinThR(0:n))


!!test for NaN error

!print*, 'Bo in EphidblSphereStep = ',Bo,' Bd in EphidblSphereStep = ',Bd


!this if statement is for when "r" is less than or equal to the radius of the sphere.

!"aMax" is the number of sequential steps from r=0 to r=a and "MaxStep" is the total number of
"r" steps

!that the E-field is calculated at each instance of "r", and may be greater than the number of
"r" steps that

!count out to the radius of the sphere.

if (step <= aMax .and. step <= MaxStep) then

    EtotPhi = (0.0D+00,0.0D+00)


    do m = 1,n,1

        Ephi_n = (-j*Eo*dsin(PhR)/(Bd*r))*(ans_gn(m)*dBrMSBJ_Bd(m)*ans_P1n_div_sinThR(m)) &
        - (Eo*dsin(PhR)/(Bd*r))*(ans_hn(m)*MSBJ_Bd(m)*dsin(ThR)*ans_dxP1n(m))


        EtotPhi = EtotPhi + Ephi_n


    end do


end if


!this if statement is for when r is greater than the radius of the sphere

if (step > aMax .and. step <= MaxStep) then

    EtotPhi = (0.0D+00,0.0D+00)


    do p = 1,n,1

        Ephi_n = (-j*Eo*dsin(PhR)/(Bo*r))*(ans_cn(p)*dBrMSBJ_Bo(p) + ans_en(p)*dBrMSH2_Bo(p)) &

        *ans_P1n_div_sinThR(p) - (Eo*dsin(PhR)/(Bo*r))*(ans_dn(p)*MSBJ_Bo(p) +
ans_fn(p)*MSH2_Bo(p)) &

        *dsin(ThR)*ans_dxP1n(p)


        EtotPhi = EtotPhi + Ephi_n


    end do


end if


!the final result!

ans_EphiDbl = EtotPhi


return

end subroutine EphiDblSphereStep





!+++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine
HphiDblSphereStep(PI,n,Bo,Bd,er,ur,no,nd,r,ThR,PhR,x,Eo,a,b,MaxStep,aMax,step,ans_HphiDbl)

!calc's phi component of H-field for the case of a spherical dielectric being irradiated by

!two planar EM waves, both facing the sphere, but spaced 180 degrees apart. One H component

!of the EM wave (top side) is polarized along the "+" y-axis, while the other component (bottom
side)
```

```
!is polarized along the "-" y-axis.

implicit none

!setting up the vars
real (kind = 8) :: PI, r, ThR, PhR, x, Eo, a, b
integer :: m, n, p, MaxStep, aMax, step
complex (kind = 8) :: er, ur, no, nd, Bo, Bd, j

!final value to send out of subroutine is a single value of E for a single value of r
complex (kind = 8) :: ans_HphiDbl

!these subroutine calls have to be formatted in arrays
complex (kind = 8) :: dBrMSBJ_Bo(0:n)
complex (kind = 8) :: dBrMSH2_Bo(0:n)
complex (kind = 8) :: MSBJ_Bo(0:n)
complex (kind = 8) :: MSH2_Bo(0:n)
complex (kind = 8) :: MSBJ_Bd(0:n)
complex (kind = 8) :: dBrMSBJ_Bd(0:n)
!any commented out subroutine calls, if seen below, are to make it easier to adapt this
subroutine
!to other related subroutines
complex (kind = 8) :: ans_cn(0:n), ans_en(0:n), ans_gn(0:n), ans_dn(0:n), ans_fn(0:n),
ans_hn(0:n)
real (kind = 8) :: ans_dxP1n(0:n), ans_P1n_div_sinThR(0:n)!,ans_P1n(0:n)

!temporary holding locations for running total of E (Etotr) and each value of E at
!each instance of "n" (Ern).
complex (kind = 8) :: HtotPhi, Hphi_n

!for complex values
j = (0.0D+00, 1.0D+00)

!calling the subroutines needed to determine Er
call ModSphBesJ(PI,n,Bd,r,MSBJ_Bd(0:n))
call dBrModSphBesJ(PI,n,Bd,r,dBrMSBJ_Bd(0:n))

call ModSphBesJ(PI,n,Bo,r,MSBJ_Bo(0:n))
call dBrModSphBesJ(PI,n,Bo,r,dBrMSBJ_Bo(0:n))
call ModSphHnk2(PI,n,Bo,r,MSH2_Bo(0:n))

call dBrModSphHnk2(PI,n,Bo,r,dBrMSH2_Bo(0:n))


call cn_dbl(n,ans_cn(0:n))

call dn_dbl(n,ans_dn(0:n))

call en_dbl(n,PI,Bo,Bd,a,er,ur,ans_en(0:n))

call fn_dbl(n,PI,Bo,Bd,a,er,ur,ans_fn(0:n))

call gn_dbl(n,PI,Bo,Bd,a,er,ur,ans_gn(0:n))

call hn_dbl(n,PI,Bo,Bd,a,er,ur,ans_hn(0:n))


!call AscLegendre(n,x,ans_P1n(0:n))

call dxAscLegendre(n,x,ans_dxP1n(0:n))

call AscLegDivSinThR(n,x,ans_P1n_div_sinThR(0:n))




!this if statement is for when "r" is less than or equal to the radius of the sphere.

!"aMax" is the number of sequential steps from r=0 to r=a and "MaxStep" is the total number of
"r" steps

!that the E-field is calculated at each instance of "r", and may be greater than the number of
"r" steps that

!count out to the radius of the sphere.

if (step <= aMax .and. step <= MaxStep) then

    HtotPhi = (0.0D+00,0.0D+00)


    do m = 1,n,1

        Hphi_n = (-Eo*dcos(PhR)/(Bd*nd*r))*(ans_gn(m)*MSBJ_Bd(m)*dsin(ThR)*ans_dxP1n(m)) &

         - (j*Eo*dcos(PhR)/(Bd*nd*r))*(ans_hn(m)*dBrMSBJ_Bd(m)*ans_P1n_div_sinThR(m))


        HtotPhi = HtotPhi + Hphi_n


    end do


end if


!this if statement is for when r is greater than the radius of the sphere

if (step > aMax .and. step <= MaxStep) then

    HtotPhi = (0.0D+00,0.0D+00)


    do p = 1,n,1
```

```
        Hphi_n = (-Eo*dcos(PhR)/(Bo*no*r))*(ans_cn(p)*MSBJ_Bo(p) + ans_en(p)*MSH2_Bo(p)) &

        *dsin(ThR)*ans_dxP1n(p) - (j*Eo*dcos(PhR)/(Bo*no*r))*(ans_dn(p)*dBrMSBJ_Bo(p) +
ans_fn(p)*dBrMSH2_Bo(p)) &

        *ans_P1n_div_sinThR(p)


        HtotPhi = HtotPhi + Hphi_n


    end do


end if


!the final result!

ans_HphiDbl = HtotPhi


return

end subroutine HphiDblSphereStep





!+++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine ModSphBesJ(PI,n,k,r,ans_MSBJ)

    !calc's modified spherical bessel funct of 1st kind

    implicit none


    real (kind = 8), intent(in) :: PI

    integer, intent(in) :: n

    integer :: l


    real (kind = 8) :: v,vm

    real (kind = 8), intent(in) :: r

    complex (kind = 8) :: cbj(0:n)

    complex (kind = 8) :: cdj(0:n)

    complex (kind = 8) :: cby(0:n)

    complex (kind = 8) :: cdy(0:n)

    complex (kind = 8), intent(out) :: ans_MSBJ(0:n)

    complex (kind = 8) :: z

    complex (kind = 8), intent(in) :: k
    v = n + 0.5D+00

    z = k*r

   ! !test for math errors

   ! print*,'in ModSphBesJ subroutine'

   ! print*,'k = ',k,' r = ',r

    call cjyva(v,z,vm, cbj(0:n), cdj(0:n), cby(0:n), cdy(0:n))

    do l = 0,n

        !D+00 added to "2"

        ans_MSBJ(l) = zsqrt(PI*k*r/(2.0D+00))*cbj(l)

       ! !test for math errors

       ! print*, 'n is: ',l

       ! print*,'J(n+1/2) = ',cbj(l)

       ! print*,'MSBJ(n) = ', ans_MSBJ(l)

    end do

   ! !test for math errors

   ! print*, 'exiting subroutine ModSphBesJ.'

    return

end subroutine ModSphBesJ

!+++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine ModSphHnk2(PI,n,k,r,ans_MSH2)

    !calc's modified spherical hankel function of 2nd kind

    !MSHankel(2) = MSBJ -j*MSBY

    implicit none

    real (kind = 8), intent(in) :: PI

    integer, intent(in) :: n

    integer :: l

    complex (kind = 8) j

    complex (kind = 8), intent(out) :: ans_MSH2(0:n)
    complex (kind = 8) cbj(0:n)
    complex (kind = 8) cdj(0:n)
    complex (kind = 8) cby(0:n)
    complex (kind = 8) cdy(0:n)
    real (kind = 8) v,vm
    real (kind = 8), intent(in) :: r
    complex (kind = 8) z
    complex (kind = 8), intent(in) :: k

    j=(0.0D+00,1.0D+00)
    v = n + 0.5D+00
    z = k*r

   ! !testing for math errors
   ! print*,'in ModSphHnk2 subroutine'
   ! print*,'k = ',k,' r = ',r

    call cjyva(v,z,vm, cbj(0:n), cdj(0:n), cby(0:n), cdy(0:n))
    do l = 0,n
        !D+00 added to "2"
        ans_MSH2(l) = zsqrt(PI*k*r/(2.0D+00))*(cbj(l) - (j*cby(l)))

       ! !test for math errors
       ! print*, 'n is ',l, ' ans_MSH2 is ', ans_MSH2(l)
       ! print*, 'J(n+1/2) = ',cbj(l)
       ! print*, 'Y(n+1/2) = ',cby(l)
       ! print*, 'H2(n) = ',(cbj(l) - (j*cby(l)))
       ! print*, 'ans_MSH2(n) = ',ans_MSH2(l)

    end do

   ! !testing for math errors
   ! print*, 'exiting subroutine ModSphHnk2.'


    return
end subroutine ModSphHnk2
```

```
!++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine dBrModSphBesJ(PI,n,k,r,ans_dkrMSBJ)

    !calc's derivative of modified spherical bessel funct of 1st kind

    implicit none

    real (kind = 8) :: PI

    integer :: n

    integer :: m,l_0,l_1, l_2, l_3


    real (kind = 8) :: v, vm, w

    real (kind = 8) :: r

    complex (kind = 8) :: cbj(0:n)

    complex (kind = 8) :: cdj(0:n)

    complex (kind = 8) :: cby(0:n)

    complex (kind = 8) :: cdy(0:n)


    !these four complex arrays are needed for the second component of the derivative of MSBJ
routine.

    !they are related to n by: m=n+1

    complex (kind = 8), allocatable, dimension(:) :: cbj2

    complex (kind = 8), allocatable, dimension(:) :: cdj2

    complex (kind = 8), allocatable, dimension(:) :: cby2

    complex (kind = 8), allocatable, dimension(:) :: cdy2


    complex (kind = 8) :: dkrMSBJ1(0:n)

    complex (kind = 8) :: dkrMSBJ2(0:n)

    complex (kind = 8) :: dkrMSBJ3(0:n)

    complex (kind = 8) :: ans_dkrMSBJ(0:n)

    complex (kind = 8) :: z

    complex (kind = 8) :: k

    !!test

    !print*, 'in subroutine. after var declarations.'

    m = n + 1

    !now that m is known, exact array sizes can be assigned to these four complex arrays.

    allocate(cbj2(0:m))

    allocate(cdj2(0:m))

    allocate(cby2(0:m))

    allocate(cdy2(0:m))

    !!test
    !print*, 'in subroutine. after allocations for "m".'

    v = n + 0.5D+00
    z = k*r

    !print*,'PI is: ',PI,' k is: ',k,' r is: ',r,' z is: ',z

   ! !testing for math errors
   ! print*,'in dBrModSphBesJ subroutine'
   ! print*,' k = ',k,' r = ',r

    call cjyva(v,z,vm, cbj(0:n), cdj(0:n), cby(0:n), cdy(0:n))
    !broke derivative of besselJ into three parts for ease of programming
    !!test
    !print*, 'in subroutine. after calling cjyva (v=n+0.5)'

    !first part
    do l_1 = 0,n
        !D+00 added to "0.25" and "2"
        dkrMSBJ1(l_1) = ((0.25D+00)*(dsqrt(2.0D+00)*PI)/zsqrt(PI*k*r))*cbj(l_1)
        !!test
        !print*, 'dkrMSBJ1 numerator is: ',(1/4)*(dsqrt(2.0D+00)*PI)
        !print*, 'dkrMSBJ1 denominator is: ',zsqrt(PI*k*r)

       ! !testing for math errors
       ! print*, 'n = ', l_1,' Jn+1/2 = ',cbj(l_1)

    end do
    !!test
    !print*, 'in subroutine. after determining dkrMSBJ1.'
    !third part
    do l_3 = 0,n

        !!!!!!! <<<<<need to re-familarize with cjyva sub routine>>>>>

        !D+00 added to "0.5" and "2"
        dkrMSBJ3(l_3) = (0.5D+00)*zsqrt((2.0D+00)*PI*k*r)*((l_3 +
((1.0D+00)/(2.0D+00)))*cbj(l_3))/(k*r)
```

```
        !!!!!!! <<<<<need to re-familarize with cjyva sub routine>>>>>


        !!test

        !print*, 'dkrMSBJ3 numerator is: ',(1/2)*zsqrt(2*PI*k*r)

        !print*, 'dkrMSBJ3 denominator is: ',(k*r)

        !print*, 'l_3 = ', l_3,' dkrMSBJ3 = ', dkrMSBJ3(l_3),' cbj = ',cbj(l_3)

    end do

   !!test

    !print*, 'in subroutine. after determining dkrMSBJ3.'

    w = n + 1.5D+00

    !!test

    !w=m+0.5

    call cjyva(w,z,vm,cbj2(0:m),cdj2(0:m),cby2(0:m),cdy2(0:m))

    !!test

    !print*, 'in subroutine. after calling cjyva for w=n+1.5.'

    !second part


    !because w=n+1.5, and because of the way cbj2 (and the others) calcuate values for each
member of the array, the array values

    !from cbj2 (and the others) must be shifted over by one value when assigning the results to
ans_dkrMSBJ2 array.

    do l_2 = 1,m

        !D+00 added to "0.5" and "2"

        dkrMSBJ2(l_2 - 1) = -(0.5D+00)*zsqrt((2.0D+00)*PI*k*r)*cbj2(l_2) !note deliberate shift
in array.


      ! !test for math errors

      ! print*, 'n = ', l_2 - 1,' Jn+3/2 = ',cbj2(l_2)


    end do


    !!test

    !print*, 'in subroutine. after determining dkrMSBJ2.'

    !putting all three parts together for final answer

    do l_0 = 0,n

        ans_dkrMSBJ(l_0) = dkrMSBJ1(l_0) + dkrMSBJ2(l_0) + dkrMSBJ3(l_0)


       ! !test
```

```
        ! print*, 'n = ', l_0,' dkrMSBJ(n) = ', ans_dkrMSBJ(l_0)



    end do

    !!test

    !print*, 'in subroutine. after determining dkrMSBJ.'

    deallocate(cbj2)

    !!test

    !print*, 'in subroutine. after deallocating cbj2'

    deallocate(cdj2)

    !!test

    !print*, 'in subroutine. after deallocating cdj2'

    deallocate(cby2)

    !!test

    !print*, 'in subroutine. after deallocating cby2'

    deallocate(cdy2)

    !!test

    !print*, 'in subroutine. after deallocating cdy2'


   ! !test

   ! print*, 'exiting subroutine dBrModSphBesJ.'


    return

end subroutine dBrModSphBesJ


!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine dBrModSphHnk2(PI,n,k,r,ans_dkrMSH2)

    !calc's derivative of modified spherical hankel funct of 2nd kind

    !MSHankel(2) = MSBJ -j*MSBY

    implicit none

    real (kind = 8), intent(in) :: PI

    integer :: m, l_0, l_1, l_2, l_3

    integer, intent(in) :: n


    real (kind = 8) v,vm,w

    real (kind = 8), intent(in) :: r

    complex (kind = 8) cbj(0:n)

    complex (kind = 8) cdj(0:n)
    complex (kind = 8) cby(0:n)

    complex (kind = 8) cdy(0:n)


    !these four complex arrays are needed for the second component of the derivative of MSBJ
routine.

    !they are related to n by: m=n+1

    complex (kind = 8), allocatable, dimension(:) :: cbj2

    complex (kind = 8), allocatable, dimension(:) :: cdj2

    complex (kind = 8), allocatable, dimension(:) :: cby2

    complex (kind = 8), allocatable, dimension(:) :: cdy2


    complex (kind = 8) j

    complex (kind = 8), intent(out) :: ans_dkrMSH2(0:n)

    complex (kind = 8) dkrMSH2_1(0:n)

    complex (kind = 8) dkrMSH2_2(0:n)

    complex (kind = 8) dkrMSH2_3(0:n)

    complex (kind = 8) z

    complex (kind = 8), intent(in) :: k


    m = n + 1

    !now that m is known, exact array sizes can be assigned to these four complex arrays.

    allocate(cbj2(0:m))

    allocate(cdj2(0:m))

    allocate(cby2(0:m))

    allocate(cdy2(0:m))


    j = (0.0D+00,1.0D+00)

    v = n + 0.5D+00

    z = k*r


   ! !testing for math errors

   ! print*,'in dBrModSphHnk2 subroutine'

   ! print*,'k = ',k,' r = ',r


    call cjyva(v,z,vm, cbj(0:n), cdj(0:n), cby(0:n), cdy(0:n))

    !broke derivative of hankel2 into three parts for ease of programming


    !first part
    do l_1 = 0,n

        !D+00 added to "0.25" and "2"

        dkrMSH2_1(l_1) = ((0.25D+00)*(PI*dsqrt(2.0D+00))/zsqrt(PI*k*r))*(cbj(l_1) - (j*cby(l_1)))

       ! !test

       ! print*, 'n = ',l_1,' H2(n+1/2) = ',(cbj(l_1) - (j*cby(l_1)))

    end do


    !third part

    do l_3 = 0,n

        !D+00 added to "0.5" and "2"

        dkrMSH2_3(l_3) = (0.5D+00)*zsqrt((2.0D+00)*PI*k*r)*(l_3+(0.5D+00))*(cbj(l_3) -
(j*cby(l_3)))/(k*r)

        !!test

        !print*, 'l3=', l_3,' dkrMSH2_3=', dkrMSH2_3(l_3)

    end do


    w = n + 1.5D+00

    call cjyva(w,z,vm,cbj2(0:m),cdj2(0:m),cby2(0:m),cdy2(0:m))


    !second part


    !because w=n+1.5, and because of the way cbj2 (and the others) calcuate values for each
member of the array, the array values

    !from cbj2 (and the others) must be shifted over by one value when assigning the results to
ans_dkrMSH2_2 array.

    do l_2 = 1,m

        !D+00 added to "0.5" and "2"

        dkrMSH2_2(l_2 - 1) = -(0.5D+00)*zsqrt((2.0D+00)*PI*k*r)*(cbj2(l_2) - (j*cby2(l_2)))

       ! !test

       ! print*, 'n = ',l_2 - 1,' H2(n+3/2) = ',(cbj2(l_2) -(j*cby(l_2)))

    end do


    !putting all three parts together for final answer

    do l_0 = 0,n

        ans_dkrMSH2(l_0) = dkrMSH2_1(l_0)+dkrMSH2_2(l_0)+dkrMSH2_3(l_0)

       ! !test

       ! print*, 'n = ', l_0,' dkrMSH2(n) = ', ans_dkrMSH2(l_0)

    end do
    deallocate(cbj2)

    deallocate(cdj2)

    deallocate(cby2)

    deallocate(cdy2)


   ! !test

   ! print*, 'exiting subroutine dBrModSphHnk2.'



    return

end subroutine dBrModSphHnk2


!+++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine dBr2ModSphBesJ(PI,n,k,r,ans_dkr2MSBJ)

    !calc's 2nd derivative of modified spherical bessel funct of 1st kind

    implicit none

    real (kind = 8) :: PI

    integer, intent(in) :: n

    integer :: m, l_0, l_1, l_2


    real (kind = 8) :: v, vm, w

    real (kind = 8) :: r

    complex (kind = 8) cbj(0:n)

    complex (kind = 8) cdj(0:n)

    complex (kind = 8) cby(0:n)

    complex (kind = 8) cdy(0:n)


    !these four complex arrays are needed for the second component of the derivative of MSBJ
routine.

    !they are used for the "n+3/2" case and are related to n by: m=n+1

    complex (kind = 8), allocatable, dimension(:) :: cbj2

    complex (kind = 8), allocatable, dimension(:) :: cdj2

    complex (kind = 8), allocatable, dimension(:) :: cby2

    complex (kind = 8), allocatable, dimension(:) :: cdy2


    complex (kind = 8) BesselJ12(0:n)

    complex (kind = 8) BesselJ32(0:n)

    !complex (kind = 8) dkr2MSBJ3(0:n)
```

```
complex (kind = 8) :: ans_dkr2MSBJ(0:n)

complex (kind = 8) :: z

complex (kind = 8) :: k

!!test

!print*, 'in subroutine. after var declarations.'

m = n + 1

!now that m is known, exact array sizes can be assigned to these four complex arrays.

allocate(cbj2(0:m))

allocate(cdj2(0:m))

allocate(cby2(0:m))

allocate(cdy2(0:m))

!!test

!print*, 'in subroutine. after allocations for "m".'


v = n + 0.5D+00

z = k*r


call cjyva(v,z,vm, cbj(0:n), cdj(0:n), cby(0:n), cdy(0:n))

!broke derivative of besselJ into three parts for ease of programming

!!test

!print*, 'in subroutine. after calling cjyva (v=n+0.5)'


    !!test for math errors

    !print*,'r = ',r,' k = ',k,' z = k*r = ',k*r

    !print*,'PI = ',PI,' z = ',z



   ! !test for math errors

   ! print*, 'in subroutine dBr2ModSphBesJ.'

   ! print*,' k = ',k,' r = ',r





!first part

do l_1 = 0,n

    BesselJ12(l_1) = cbj(l_1)


    !!test for math error
```

```
        !print*, 'l1 = ', l_1,' BesselJ12 = ', BesselJ12(l_1)


    end do

    !!test

    !print*, 'in subroutine. after determining dkr2MSBJ12.'


    w = n + 1.5D+00

    !!test

    !w = m + 0.5

    call cjyva(w,z,vm,cbj2(0:m),cdj2(0:m),cby2(0:m),cdy2(0:m))

    !!test

    !print*, 'in subroutine. after calling cjyva for w=n+1.5.'

    !second part


    !because w=n+1.5, and because of the way cbj2 (and the others) calcuate values for each
member of the array, the array values

    !from cbj2 (and the others) must be shifted over by one value when assigning the results to
ans_dkrMSBJ2 array.

    do l_2 = 1,m

        BesselJ32(l_2 - 1) = cbj2(l_2) !note deliberate shift in array.


        !!test for math error

        !print*, 'l2 - 1 = ', l_2 - 1,' BesselJ32 = ', BesselJ32(l_2-1)


    end do

    !!test

    !print*, 'in subroutine. after determining dkrMSBJ2.'

    !putting all parts together for final answer

    do l_0=0,n

        ans_dkr2MSBJ(l_0) = -
(0.125D+00)*((dsqrt(2.0D+00)*BesselJ12(l_0)*(PI**2.0D+00))/((PI*k*r)**(3.0D+00/2.0D+00))) &

        + (1.0D+00/2.0D+00)*dsqrt(2.0D+00)*( -BesselJ32(l_0)+(((l_0 +
1.0D+00/2.0D+00)*BesselJ12(l_0))/ &

        (k*r)) )*PI/(zsqrt(PI*k*r)) + (1.0D+00/2.0D+00)*dsqrt(2.0D+00)*zsqrt(PI*k*r)* &

        ( -BesselJ12(l_0) + (((l_0 + 3.0D+00/2.0D+00)*BesselJ32(l_0))/(k*r)) &

        - (((l_0 + 1.0D+00/2.0D+00)*BesselJ12(l_0))/((k*r)**2.0D+00)) + ( (l_0 + 1.0D+00/2.0D+00)
&

        *( -BesselJ32(l_0) + (((l_0 + 1.0D+00/2.0D+00)*BesselJ12(l_0))/(k*r)) ) )/(k*r) )


       ! !test
```

```
        ! print*, 'n = ', l_0,' dkr2MSBJ(n) = ', ans_dkr2MSBJ(l_0)

    end do
    !!test
    !print*, 'in subroutine. after determining dkr2MSBJ.'
    deallocate(cbj2)
    !!test
    !print*, 'in subroutine. after deallocating cbj2'
    deallocate(cdj2)
    !!test
    !print*, 'in subroutine. after deallocating cdj2'
    deallocate(cby2)
    !!test
    !print*, 'in subroutine. after deallocating cby2'
    deallocate(cdy2)
    !!test
    !print*, 'in subroutine. after deallocating cdy2'

   ! !test
   ! print*, 'exiting subroutine dBr2ModSphBesJ.'


    return
end subroutine dBr2ModSphBesJ


!++++++++++++++++++++++++++++++++++++++++++++++++++++++
subroutine dBr2ModSphHnk2(PI,n,k,r,ans_dkr2MSH2)
    !calc's 2nd derivative of modified spherical hankel funct of 2nd kind
    implicit none
    real (kind = 8), intent(in) :: PI
    integer, intent(in) :: n
    integer :: m, l_0, l_1, l_2

    real (kind = 8) v, vm, w
    real (kind = 8), intent(in) :: r
    complex (kind = 8) cbj(0:n)
    complex (kind = 8) cdj(0:n)
    complex (kind = 8) cby(0:n)

    complex (kind = 8) cdy(0:n)


    !these four complex arrays are needed for the second component of the derivative of MSBJ
routine.

    !they are used for the "n+3/2" case and are related to n by: m=n+1

    complex (kind = 8), allocatable, dimension(:) :: cbj2

    complex (kind = 8), allocatable, dimension(:) :: cdj2

    complex (kind = 8), allocatable, dimension(:) :: cby2

    complex (kind = 8), allocatable, dimension(:) :: cdy2


    complex (kind = 8) Hankel212(0:n)

    complex (kind = 8) Hankel232(0:n)

    !complex (kind = 8) dkr2MSH23(0:n)

    complex (kind = 8), intent(out) :: ans_dkr2MSH2(0:n)

    complex (kind = 8) z, j

    complex (kind = 8), intent(in) :: k

    !!test

    !print*, 'in subroutine. after var declarations.'

    m = n + 1

    !now that m is known, exact array sizes can be assigned to these four complex arrays.

    allocate(cbj2(0:m))

    allocate(cdj2(0:m))

    allocate(cby2(0:m))

    allocate(cdy2(0:m))


   ! !test for math errors

   ! print*, 'in subroutine dBr2ModSphHnk2.'


    j = (0.0D+00,1.0D+00)

    v = n + 0.5D+00

    z = k*r


   ! !testing for math errors

   ! print*,'k = ',k,' r = ',r


    call cjyva(v,z,vm, cbj(0:n), cdj(0:n), cby(0:n), cdy(0:n))

    !broke derivative of besselJ into three parts for ease of programming
    !!test

    !print*, 'in subroutine. after calling cjyva (v=n+0.5)'


    !first part

    do l_1 = 0,n

        Hankel212(l_1) = (cbj(l_1) - (j*cby(l_1)))


       ! !testing for math errors

       ! print*, 'n = ', l_1,' Hankel212(n) = ', Hankel212(l_1)


    end do

    !!test

    !print*, 'in subroutine. after determining dkr2MSH212.'


    w = n + 1.5D+00

    !!test

    !w = m + 0.5

    call cjyva(w,z,vm,cbj2(0:m),cdj2(0:m),cby2(0:m),cdy2(0:m))

    !!test

    !print*, 'in subroutine. after calling cjyva for w=n+1.5.'

    !second part


    !because w=n+1.5, and because of the way cbj2 (and the others) calcuate values for each
member of the array, the array values

    !from cbj2 (and the others) must be shifted over by one value when assigning the results to
ans_dkrMSBJ2 array.

    do l_2 = 1,m

        Hankel232(l_2-1) = (cbj2(l_2) - (j*cby2(l_2))) !note deliberate shift in array.


       ! !test for math errors

       ! print*, 'n = ', l_2 - 1,' Hankel232(n) = ', Hankel232(l_2 - 1)


    end do

    !!test

    !print*, 'in subroutine. after determining dkrMSBJ2.'

    !putting all parts together for final answer

    do l_0 = 0,n

        ans_dkr2MSH2(l_0) = -
(0.125D+00)*((dsqrt(2.0D+00)*Hankel212(l_0)*(PI**2.0D+00))/((PI*k*r)**(3.0D+00/2.0D+00))) &
```

```
        + (1.0D+00/2.0D+00)*dsqrt(2.0D+00)*( -Hankel232(l_0)+(((l_0 +
(1.0D+00/2.0D+00))*Hankel212(l_0))/ &

        (k*r)) )*PI/(zsqrt(PI*k*r)) + (1.0D+00/2.0D+00)*dsqrt(2.0D+00)*zsqrt(PI*k*r)* &

        ( -Hankel212(l_0) + (((l_0 + (3.0D+00/2.0D+00))*Hankel232(l_0))/(k*r)) &

        - (((l_0 + (1.0D+00/2.0D+00))*Hankel212(l_0))/((k*r)**2.0D+00)) + ( (l_0 +
(1.0D+00/2.0D+00)) &

        *( -Hankel232(l_0) + (((l_0 + (1.0D+00/2.0D+00))*Hankel212(l_0))/(k*r)) ) )/(k*r) )


       ! !test for math errors

       ! print*, 'n = ', l_0,' dkr2MSH2(n) = ', ans_dkr2MSH2(l_0)


    end do

    !!test

    !print*, 'in subroutine. after determining dkr2MSBJ.'

    deallocate(cbj2)

    !!test

    !print*, 'in subroutine. after deallocating cbj2'

    deallocate(cdj2)

    !!test

    !print*, 'in subroutine. after deallocating cdj2'

    deallocate(cby2)

    !!test

    !print*, 'in subroutine. after deallocating cby2'

    deallocate(cdy2)

    !!test

    !print*, 'in subroutine. after deallocating cdy2'


   ! !test for math errors

   ! print*, 'exiting subroutine dBr2ModSphHnk2.'


    return

end subroutine dBr2ModSphHnk2




!++++++++++++++++++++++++++++++++++++++++++++++++++++

!changed from function to subroutine

subroutine cn_dbl(n, ans_cn)

    implicit none

    !complex (kind = 8), allocatable, dimension(:) :: ans_an
    integer, intent(in) :: n

    integer :: l

    complex (kind = 8) :: j

    complex (kind = 8), intent(out) :: ans_cn(1:n)  !start with ans_an at 1.

    j=(0.0D+00,1.0D+00)

    !allocate(ans_an(0:n))

    do l=1,n

        ans_cn(l) = ((j**l) - (j**(-l)))*(((2.0D+00)*l + 1)/(l*(l + 1)))


        !!test for Nan errors

        !print*, 'ans_cn for n= ',l,' is: ',ans_cn(l)


       ! !test for math errors

       ! print*,'cn for n = ',l,' is: ',ans_cn(l)


    end do

    !!deallocate(ans_an)

    !print*, 'just about to exit subroutine cn_dbl'

    return

end subroutine cn_dbl



!+++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine dn_dbl(n, ans_dn)

    implicit none

    !complex (kind=8), allocatable, dimension(:) :: ans_an

    integer, intent(in) :: n

    integer :: l

    complex (kind = 8) :: j

    complex (kind = 8), intent(out) :: ans_dn(1:n)  !start with ans_an at 1.

    j = (0.0D+00,1.0D+00)

    !allocate(ans_an(0:n))

    do l = 1,n

        ans_dn(l) = ((j**l) + (j**(-l)))*(((2.0D+00)*l + 1)/(l*(l + 1)))


        !!test for Nan errors

        !print*, 'ans_dn for n= ',l,' is: ',ans_dn(l)
        ! !test for math errors

        ! print*,'dn for n = ',l,' is: ',ans_dn(l)


    end do

    !deallocate(ans_an)

    !print*, 'just about to exit subroutine dn_dbl'

    return

end subroutine dn_dbl





!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine en_dbl(n,PI,Bo,Bd,a,er,ur,ans_en)

    implicit none


    !specifically for en

    integer :: n, l_0, l_1, l_2

    real (kind = 8) a,PI

    complex (kind = 8) ur,er,Bo,Bd !Bo is technically real, but leaving complex for simplicity
sake

    complex (kind = 8) ans_cn(1:n) !test with ans_an starting at 1.

    complex (kind = 8) ans_en(0:n)

    complex (kind = 8) ans_en_top(0:n)

    complex (kind = 8) ans_en_bot(0:n)


   !!try this modification to avoid SIGABRT error at end of S.R. bn

    !complex (kind=8), allocatable, dimension(:) :: ans_an

    !complex (kind=8), allocatable, dimension(:) ans_bn

    !complex (kind=8), allocatable, dimension(:) :: ans_bn_top

    !complex (kind=8), allocatable, dimension(:) :: ans_bn_bot




    !specifically for function calls

    complex (kind = 8) ans_MSBJ_Boa(0:n)

    complex (kind = 8) ans_MSBJ_Bda(0:n)

    complex (kind = 8) ans_dBrMSBJ_Boa(0:n)

    complex (kind = 8) ans_dBrMSBJ_Bda(0:n)

    complex (kind = 8) ans_MSH2_Boa(0:n)
```

```
 complex (kind = 8) ans_dBrMSH2_Boa(0:n)

 !!try this modification to avoid SIGABRT error at end of S.R. bn
 !!specifically for function calls
 !complex (kind=8), allocatable, dimension(:) ::  ans_MSBJ_k1a
 !complex (kind=8), allocatable, dimension(:) ::  ans_MSBJ_k2a
 !complex (kind=8), allocatable, dimension(:) ::  ans_dkrMSBJ_k1a
 !complex (kind=8), allocatable, dimension(:) ::  ans_dkrMSBJ_k2a
 !complex (kind=8), allocatable, dimension(:) ::  ans_MSH2_k2a
 !complex (kind=8), allocatable, dimension(:) ::  ans_dkrMSH2_k2a

 !!try this modification to avoid SIGABRT error at end of S.R. bn
 !allocate(ans_an(1:n))
 !allocate(ans_bn(0:n))
 !allocate(ans_bn_top(0:n))
 !allocate(ans_bn_bot(0:n))
 !allocate(ans_MSBJ_k1a(0:n))
 !allocate(ans_MSBJ_k2a(0:n))
 !allocate(ans_dkrMSBJ_k1a(0:n))
 !allocate(ans_dkrMSBJ_k2a(0:n))
 !allocate(ans_MSH2_k2a(0:n))
 !allocate(ans_dkrMSH2_k2a(0:n))

 !note ans_cn array starts at "1"
 call cn_dbl(n, ans_cn(1:n))

 ! calling  modified spherical bessel and hankel functions and derivatives of said functions
 call ModSphBesJ(PI,n,Bo,a,ans_MSBJ_Boa(0:n))
 call ModSphBesJ(PI,n,Bd,a,ans_MSBJ_Bda(0:n))
 call dBrModSphBesJ(PI,n,Bo,a,ans_dBrMSBJ_Boa(0:n))
 call dBrModSphBesJ(PI,n,Bd,a,ans_dBrMSBJ_Bda(0:n))
 call ModSphHnk2(PI,n,Bo,a,ans_MSH2_Boa(0:n))
 call dBrModSphHnk2(PI,n,Bo,a,ans_dBrMSH2_Boa(0:n))
 !!test
 !print*, 'in bn. after MSB and MSH subroutine calls'


! !testing for math errors
```

```
   ! print*,'a = ',a,' Bo = ',Bo,' Bd = ',Bd




    !split ans_en into top and bottom equations

    do l_1 = 0,n

        if (l_1 == 0) then

            !var below was, in error, originally assigned value as if real

            ans_en_top(l_1) = (0.0D+00, 0.0D+00) !this is needed to kill zeroth order ans_bn term
as ans_an is undefined at l_1=0

        else

            !pesky 132 char limit!

            ans_en_top(l_1) = (-ans_cn(l_1))*(ans_dBrMSBJ_Bda(l_1)*ans_MSBJ_Boa(l_1)*zsqrt(ur) &

            - ans_MSBJ_Bda(l_1)*ans_dBrMSBJ_Boa(l_1)*zsqrt(er))

        end if

        !!test

        !print*, 'l_1 is ',l_1,'en top is: ', ans_en_top(l_1)


    end do

    !!test

    !print*, 'in bn. after ans_bn_top assignments.'

    do l_2 = 0,n

        if (l_2 == 0) then

            !var below was, in error, originally assigned value as if real

            ans_en_bot(l_2) = (1.0D+00, 1.0D+00) !exact value not important as will ultimately be
multiplied by zero for l_2=0.

        else

            ans_en_bot(l_2) = ans_MSH2_Boa(l_2)*ans_dBrMSBJ_Bda(l_2)*zsqrt(ur) &

            - ans_dBrMSH2_Boa(l_2)*ans_MSBJ_Bda(l_2)*zsqrt(er)

        end if

        !!test

        !print*, 'l_2 is ',l_2,'en bot is: ', ans_en_bot(l_2)


    end do

    !!test

    !print*, 'in bn. after ans_bn_bot assignments.'

    !final expression

    do l_0 = 0,n

        ans_en(l_0) = ans_en_top(l_0)/ans_en_bot(l_0)
        !test for math errors

        !print*, 'en is ',l_0,'en is: ', ans_en(l_0)


       ! !test for math errors

       ! print*,'en for n = ',l_0,' is: ',ans_en(l_0)


    end do

    !!test

    !print*, 'after ans_bn calculated'


    !try this modification to avoid SIGABRT error at end of subroutine bn

    !deallocate(ans_an)

    !deallocate(ans_bn)

    !deallocate(ans_bn_top)

    !deallocate(ans_bn_bot)

    !deallocate(ans_MSBJ_k1a)

    !deallocate(ans_MSBJ_k2a)

    !deallocate(ans_dkrMSBJ_k1a)

    !deallocate(ans_dkrMSBJ_k2a)

    !deallocate(ans_MSH2_k2a)

    !deallocate(ans_dkrMSH2_k2a)


    !test

    !print*, 'just before leaving subroutine en_dbl.'

    !!<><><><><> E-field program crashes HERE <><><><><>!

    return

end subroutine en_dbl


!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


subroutine fn_dbl(n,PI,Bo,Bd,a,er,ur,ans_fn)

    implicit none


    !specifically for fn

    integer :: n, l_0, l_1, l_2

    real (kind = 8) a,PI

    complex (kind = 8) ur,er,Bo,Bd !Bo is technically real, but leaving complex for simplicity
sake
```

```
    complex (kind = 8) ans_dn(1:n) !test with ans_an starting at 1.
    complex (kind = 8) ans_fn(0:n)
    complex (kind = 8) ans_fn_top(0:n)
    complex (kind = 8) ans_fn_bot(0:n)

    !specifically for function calls
    complex (kind = 8) ans_MSBJ_Boa(0:n)
    complex (kind = 8) ans_MSBJ_Bda(0:n)
    complex (kind = 8) ans_dBrMSBJ_Boa(0:n)
    complex (kind = 8) ans_dBrMSBJ_Bda(0:n)
    complex (kind = 8) ans_MSH2_Boa(0:n)
    complex (kind = 8) ans_dBrMSH2_Boa(0:n)

    !note ans_an array starts at "1"
    call dn_dbl(n, ans_dn(1:n))
    !calling modified spherical bessel and hankel functions and derivatives of said functions
    call ModSphBesJ(PI,n,Bo,a,ans_MSBJ_Boa(0:n))
    call ModSphBesJ(PI,n,Bd,a,ans_MSBJ_Bda(0:n))
    call dBrModSphBesJ(PI,n,Bo,a,ans_dBrMSBJ_Boa(0:n))
    call dBrModSphBesJ(PI,n,Bd,a,ans_dBrMSBJ_Bda(0:n))
    call ModSphHnk2(PI,n,Bo,a,ans_MSH2_Boa(0:n))
    call dBrModSphHnk2(PI,n,Bo,a,ans_dBrMSH2_Boa(0:n))


   ! !testing for math errors
   ! print*,'a = ',a,' Bo = ',Bo,' Bd = ',Bd


    !split ans_fn into top and bottom equations
    do l_1 = 0,n
        if (l_1 == 0) then
            ans_fn_top(l_1) = (0.0D+00, 0.0D+00) !needed to kill ans_cn(0) as ans_an undefined at
l_1=0
        else
            !pesky 132 char limit!
            ans_fn_top(l_1) = -ans_dn(l_1)*(ans_MSBJ_Bda(l_1)*ans_dBrMSBJ_Boa(l_1)*zsqrt(ur) &
            - ans_dBrMSBJ_Bda(l_1)*ans_MSBJ_Boa(l_1)*zsqrt(er))
        !!test
        !print*, 'l_1 is ',l_1,'fn top is: ', ans_fn_top(l_1)


        end if

    end do


    do l_2 = 0,n

        if (l_2 == 0) then

            ans_fn_bot(l_2) = (1.0D+00, 1.0D+00) !exact value not important as will be multiplied
by zero anyway

        else

            ans_fn_bot(l_2) = ans_dBrMSH2_Boa(l_2)*ans_MSBJ_Bda(l_2)*zsqrt(ur) &

            - ans_MSH2_Boa(l_2)*ans_dBrMSBJ_Bda(l_2)*zsqrt(er)

        !!test

        !print*, 'l_2 is ',l_2,'fn bot is: ', ans_fn_bot(l_2)


        end if

    end do


    !final expression

    do l_0 = 0,n

        ans_fn(l_0) = ans_fn_top(l_0)/ans_fn_bot(l_0)

        !!test

        !print*, 'l_0 is ',l_0,'fn is: ', ans_fn(l_0)


       ! !test for math errors

       ! print*,'fn for n = ',l_0,' is: ',ans_fn(l_0)


    end do


    !!test

    !print*, 'just before leaving subroutine fn_dbl'


    return

end subroutine fn_dbl


!+++++++++++++++++++++++++++++++++++++++++++++++++++++++


subroutine gn_dbl(n,PI,Bo,Bd,a,er,ur,ans_gn)
```

```
    implicit none



    !specifically for fn
    integer :: n, l_0, l_1, l_2
    real (kind = 8) a,PI
    complex (kind = 8) ur,er,Bo,Bd !Bo is technically real, but leaving complex for simplicity
sake
    complex (kind = 8) ans_cn(1:n) !test with ans_an starting at 1.
    complex (kind = 8) ans_gn(0:n)
    complex (kind = 8) ans_gn_top(0:n)
    complex (kind = 8) ans_gn_bot(0:n)

    !specifically for function calls
    complex (kind = 8) ans_MSBJ_Boa(0:n)
    complex (kind = 8) ans_MSBJ_Bda(0:n)
    complex (kind = 8) ans_dBrMSBJ_Boa(0:n)
    complex (kind = 8) ans_dBrMSBJ_Bda(0:n)
    complex (kind = 8) ans_MSH2_Boa(0:n)
    complex (kind = 8) ans_dBrMSH2_Boa(0:n)

    !note ans_an array starts at "1"
    call cn_dbl(n, ans_cn(1:n))
    call ModSphBesJ(PI,n,Bo,a,ans_MSBJ_Boa(0:n))
    call ModSphBesJ(PI,n,Bd,a,ans_MSBJ_Bda(0:n))
    call dBrModSphBesJ(PI,n,Bo,a,ans_dBrMSBJ_Boa(0:n))
    call dBrModSphBesJ(PI,n,Bd,a,ans_dBrMSBJ_Bda(0:n))
    call ModSphHnk2(PI,n,Bo,a,ans_MSH2_Boa(0:n))
    call dBrModSphHnk2(PI,n,Bo,a,ans_dBrMSH2_Boa(0:n))



   ! !testing for math errors
   ! print*,'a = ',a,' Bo = ',Bo,' Bd = ',Bd



    !split ans_gn into top and bottom equations
    do l_1 = 0,n
        if (l_1 == 0) then
```

```
            ans_gn_top(l_1) = (0.0D+00, 0.0D+00) !needed since ans_cn(0) is undefined

        else

            !pesky 132 char limit!

            ans_gn_top(l_1) = ans_cn(l_1)*ur*zsqrt(er)*(ans_MSH2_Boa(l_1)*ans_dBrMSBJ_Boa(l_1) &

            - ans_dBrMSH2_Boa(l_1)*ans_MSBJ_Boa(l_1))

        !!test

        !print*, 'l_1 is ',l_1,'gn top is: ', ans_gn_top(l_1)


        end if

    end do


    do l_2 = 0,n

        if (l_2 == 0) then

            ans_gn_bot(l_2) = (1.0D+00, 1.0D+00) !only needs non-zero value. will be multiplied
by zero anyway

        else

            ans_gn_bot(l_2) = ans_MSH2_Boa(l_2)*ans_dBrMSBJ_Bda(l_2)*zsqrt(ur) &

            - ans_dBrMSH2_Boa(l_2)*ans_MSBJ_Bda(l_2)*zsqrt(er)

        !!test

        !print*, 'l_2 is ',l_2,'gn bot is: ', ans_gn_top(l_2)


        end if

    end do


    !final expression

    do l_0 = 0,n

        ans_gn(l_0) = ans_gn_top(l_0)/ans_gn_bot(l_0)

        !!test

        !print*, 'l_0 is ',l_0,'gn is: ', ans_gn(l_0)


       ! !test for math errors

       ! print*,'gn for n = ',l_0,' is: ',ans_gn(l_0)


    end do


    !!test

    !print*, 'just before leaving subroutine gn_dbl'
    return

end subroutine gn_dbl


!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


subroutine hn_dbl(n,PI,Bo,Bd,a,er,ur,ans_hn)

    implicit none


      !specifically for hn

    integer :: n, l_0, l_1, l_2

    real (kind = 8) a,PI

    complex (kind = 8) ur,er,Bo,Bd !Bo is technically real, but leaving complex for simplicity
sake

    complex (kind = 8) ans_dn(1:n) !test with ans_an starting at 1.

    complex (kind = 8) ans_hn(0:n)

    complex (kind = 8) ans_hn_top(0:n)

    complex (kind = 8) ans_hn_bot(0:n)


    !specifically for function calls

    complex (kind = 8) ans_MSBJ_Boa(0:n)

    complex (kind = 8) ans_MSBJ_Bda(0:n)

    complex (kind = 8) ans_dBrMSBJ_Boa(0:n)

    complex (kind = 8) ans_dBrMSBJ_Bda(0:n)

    complex (kind = 8) ans_MSH2_Boa(0:n)

    complex (kind = 8) ans_dBrMSH2_Boa(0:n)


    !note ans_an begins at "1"

    call dn_dbl(n, ans_dn(1:n))

    call ModSphBesJ(PI,n,Bo,a,ans_MSBJ_Boa(0:n))

    call ModSphBesJ(PI,n,Bd,a,ans_MSBJ_Bda(0:n))

    call dBrModSphBesJ(PI,n,Bo,a,ans_dBrMSBJ_Boa(0:n))

    call dBrModSphBesJ(PI,n,Bd,a,ans_dBrMSBJ_Bda(0:n))

    call ModSphHnk2(PI,n,Bo,a,ans_MSH2_Boa(0:n))

    call dBrModSphHnk2(PI,n,Bo,a,ans_dBrMSH2_Boa(0:n))



   ! !testing for math errors

   ! print*,'a = ',a,' Bo = ',Bo,' Bd = ',Bd
```

```
    !split ans_hn into top and bottom equations

    do l_1 = 0,n
        if (l_1 == 0) then
            ans_hn_top(l_1) = (0.0D+00, 0.0D+00) !needed since ans_dn(0) undefined
        else
            !pesky 132 char limit!
            ans_hn_top(l_1) = -ans_dn(l_1)*ur*zsqrt(er)*(ans_MSH2_Boa(l_1)*ans_dBrMSBJ_Boa(l_1) &
            - ans_dBrMSH2_Boa(l_1)*ans_MSBJ_Boa(l_1))
        !!test
        !print*, 'l_1 is ',l_1,'hn top is: ', ans_hn_top(l_1)

        end if
    end do

    do l_2 = 0,n
        if (l_2 == 0) then
            ans_hn_bot(l_2) = (1.0D+00, 1.0D+00) !any value will do since will be multiplied by
zero anyway
        else
            ans_hn_bot(l_2) = ans_dBrMSH2_Boa(l_2)*ans_MSBJ_Bda(l_2)*zsqrt(ur) &
            - ans_MSH2_Boa(l_2)*ans_dBrMSBJ_Bda(l_2)*zsqrt(er)
        !!test
        !print*, 'l_2 is ',l_2,'hn bot is: ', ans_hn_top(l_2)

        end if
    end do

    !final expression
    do l_0 = 0,n
        ans_hn(l_0) = ans_hn_top(l_0)/ans_hn_bot(l_0)
        !!test
        !print*, 'l_0 is ',l_0,'hn is: ', ans_hn(l_0)

       ! !test for math errors
       ! print*,'hn for n = ',l_0,' is: ',ans_hn(l_0)
    end do

    !!test
    !print*, 'just before leaving subroutine hn_dbl'

    return
end subroutine hn_dbl

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine AscLegendre(n,x,ans_P1n_corr)
    implicit none

    integer :: m,n,q, p!for math error testing
    real (kind = 8) x
    !only ans_P1n returned
    !real (kind=8), allocatable, dimension(:) :: ans_P1n
    !real (kind=8), allocatable, dimension(:) :: ans_dxP1n
    !allocate(ans_P1n(0:n))
    !allocate(ans_dxP1n(0:n))
    real (kind = 8) ans_P1n(0:n)
    real (kind = 8) ans_P1n_corr(0:n)
    real (kind = 8) ans_dxP1n(0:n)


    !only order of m=1 used
    m=1

call lpmns ( m, n, x, ans_P1n(0:n), ans_dxP1n(0:n))


!T-MATT-> 9/11/15-> this loop is to correct for the sign error present in the original
!source code.
do q = 0,n

    ans_P1n_corr(q) = -ans_P1n(q)
```

```
end do



!!testing for math errors
!print*,'x = ',x

!!testing for math errors
!do p = 0,n
!
!    if (x < 0.6) then
!
!    print*,'x = ',x,' n = ',p,' ans_P1n_corr = ',ans_P1n_corr(p)
!
!    end if
!
!end do

!tidy up memory allocations
!deallocate(ans_P1n)
!deallocate(ans_dxP1n)

    !!test
    !print*, 'just before leaving subroutine AscLegendre'

    return
end subroutine AscLegendre

!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine dxAscLegendre(n,x,ans_dxP1n_corr)
    implicit none

    integer :: m,n,q, p!to test for math errors
    real (kind = 8) x
    !only ans_dxP1n returned
    !real (kind=8), allocatable, dimension(:) :: ans_P1n
    !real (kind=8), allocatable, dimension(:) :: ans_dxP1n
    !allocate(ans_P1n(0:n))
    !allocate(ans_dxP1n(0:n))

    real (kind = 8) ans_P1n(0:n)

    real (kind = 8) ans_dxP1n(0:n)

    real (kind = 8) ans_dxP1n_corr(0:n)


    !only order of m=1 used

    m=1

call lpmns ( m, n, x, ans_P1n(0:n), ans_dxP1n(0:n))



!T-MATT-> 9/11/15-> this loop is to correct for the sign error present in the original

!source code.

do q = 0,n


    ans_dxP1n_corr(q) = -ans_dxP1n(q)


end do



!!testing for math errors

!print*,'x = ',x


!!testing for math errors

!do p = 0,n

!

!    if (x < 0.6) then

!

!   print*,'x = ',x,' n = ',p,' ans_dxP1n_corr = ',ans_dxP1n_corr(p)

!

!    end if

!

!end do


!tidy up memory allocations

!deallocate(ans_P1n)

!deallocate(ans_dxP1n)
```

```
    !!test
    !print*, 'just before leaving subroutine dxAscLegendre'

    return
end subroutine dxAscLegendre

!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine AscLegDivSinThR(n,x,ans_P1n_div_sinThR)
!this subroutine greatly simplifies the mathematics involved in determeining
!the answer. It also corrects a sign error present in the original
!AscLegDivSinThR code.

    implicit none

    integer :: m,n,q, p!to test for math errors
    real (kind = 8) x, sinThR
    real (kind = 8) ans_P1n(0:n)
    real (kind = 8) ans_dxP1n(0:n)
    real (kind = 8) ans_p1n_div_sinThR(0:n)

    !only order of m=1 used
    m=1

call lpmns ( m, n, x, ans_P1n(0:n), ans_dxP1n(0:n))

sinThR = (1.0D+00 - x*x)**(1.0D+00/2+0D+00)

!!testing for math errors
!print*,'x = ',x

do q = 0,n

    !T-MATT->9/11/15->"-" added to correct for sign error in ans_P1n subroutine
    ans_P1n_div_sinThR(q) = -(ans_P1n(q))/sinThR

end do
```

```
!!testing for math errors
!do p = 0,n
!
!    if (x < 0.6) then
!
!    print*,'x = ',x,' n = ',p,' ans_P1n_div_sinThR = ',ans_P1n_div_sinThR(p)
!
!    end if
!
!end do

    !!test
    !print*, 'just before leaving subroutine dxAscLegendre'

    return
end subroutine AscLegDivSinThR


!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine AscLegDivSinThR_old(n,x,ans_P1n_div_sinThR)
 !this subroutine is equivalent to "AscLegendre", divided by sin(theta).
 !for high quality results, this subroutine works a little differently
 implicit none

    integer :: q,n,l
    real (kind = 8) x
    !only ans_P1n_div_sinThR returned
    real (kind = 8), allocatable, dimension(:) :: pn
    real (kind = 8), allocatable, dimension(:) :: pd
    !real (kind = 8), allocatable, dimension(:) :: ans_P1n_div_sinThR
    real (kind = 8) ans_p1n_div_sinThR(0:n)

    q=n+1
    allocate(pn(0:q))
    allocate(pd(0:q))
    !allocate(ans_P1n_div_sinThR(0:n))
    !only order of m=1 used
    !q=n+1

    !!testing for math errors
    !print*,'x = ',x

call lpn ( q, x, pn(0:q), pd(0:q) )
do l=0,n
    !q needs to equal to n+1 in order to use formula below
    ans_P1n_div_sinThR(l) = (-1)*((l + 1)*pn(l + 1)-(l + 1)*x*pn(l))/((x**2.0D+00) - 1)

!    if (x < 0.5) then
!
!        !testing for math errors
!        print*,'x = ',x,' n = ',l,' ans_P1n_div_sinThR = ',ans_P1n_div_sinThR(l)
!        print*,'n = ',l,' legendreP = ',pn(l)
!
!    end if


end do

!tidy up memory allocations
deallocate(pn)
deallocate(pd)
!deallocate(ans_P1n_div_sinThR)

    !!test
    !print*, 'just before leaving subroutine AscLegDivSinThR'

    return
end subroutine AscLegDivSinThR_old

!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

subroutine cjyva ( v, z, vm, cbj, cdj, cby, cdy )
```

```
!*****************************************************************************80
!
!! CJYVA: Bessel functions and derivatives, Jv(z) and Yv(z) of complex argument.
!
!  Licensing:
!
!    This routine is copyrighted by Shanjie Zhang and Jianming Jin.  However,
!    they give permission to incorporate this routine into a user program
!    provided that the copyright is acknowledged.
!
!  Modified:
!
!    03 August 2012
!
!  Author:
!
!    Shanjie Zhang, Jianming Jin
!
!  Reference:
!
!    Shanjie Zhang, Jianming Jin,
!    Computation of Special Functions,
!    Wiley, 1996,
!    ISBN: 0-471-11963-6,
!    LC: QA351.C45.
!
!  Parameters:
!
!    Input, real ( kind = 8 ) V, the order of Jv(z) and Yv(z).
!
!    Input, complex ( kind = 8 ) Z, the argument.  !modification by T-MATT
!
!    Output, real ( kind = 8 ) VM, the highest order computed.
!
!    Output, real ( kind = 8 ) CBJ(0:*), CDJ(0:*), CBY(0:*), CDY(0:*),
!    the values of Jn+v0(z), Jn+v0'(z), Yn+v0(z), Yn+v0'(z).
!
  implicit none
```

```
real ( kind = 8 ) a0

complex ( kind = 8 ) ca

complex ( kind = 8 ) ca0

complex ( kind = 8 ) cb

complex ( kind = 8 ),intent(out) :: cbj(0:*) !modification by T-MATT

complex ( kind = 8 ),intent(out) :: cby(0:*) !modification by T-MATT

complex ( kind = 8 ) cck

complex ( kind = 8 ),intent(out) :: cdj(0:*) !modification by T-MATT

complex ( kind = 8 ),intent(out) :: cdy(0:*) !modification by T-MATT

complex ( kind = 8 ) cec

complex ( kind = 8 ) cf

complex ( kind = 8 ) cf0

complex ( kind = 8 ) cf1

complex ( kind = 8 ) cf2

complex ( kind = 8 ) cfac0

complex ( kind = 8 ) cfac1

complex ( kind = 8 ) cg0

complex ( kind = 8 ) cg1

complex ( kind = 8 ) ch0

complex ( kind = 8 ) ch1

complex ( kind = 8 ) ch2

complex ( kind = 8 ) ci

complex ( kind = 8 ) cju0

complex ( kind = 8 ) cju1

complex ( kind = 8 ) cjv0

complex ( kind = 8 ) cjv1

complex ( kind = 8 ) cjvl

complex ( kind = 8 ) cp11

complex ( kind = 8 ) cp12

complex ( kind = 8 ) cp21

complex ( kind = 8 ) cp22

complex ( kind = 8 ) cpz

complex ( kind = 8 ) cqz

complex ( kind = 8 ) cr

complex ( kind = 8 ) cr0

complex ( kind = 8 ) cr1

complex ( kind = 8 ) crp
complex ( kind = 8 ) crq

complex ( kind = 8 ) cs

complex ( kind = 8 ) cs0

complex ( kind = 8 ) cs1

complex ( kind = 8 ) csk

complex ( kind = 8 ) cyk

complex ( kind = 8 ) cyl1

complex ( kind = 8 ) cyl2

complex ( kind = 8 ) cylk

complex ( kind = 8 ) cyv0

complex ( kind = 8 ) cyv1

real ( kind = 8 ) ga

real ( kind = 8 ) gb

integer ( kind = 4 ) j

integer ( kind = 4 ) k

integer ( kind = 4 ) k0

integer ( kind = 4 ) l

integer ( kind = 4 ) lb

integer ( kind = 4 ) lb0

integer ( kind = 4 ) m

integer ( kind = 4 ) msta1

integer ( kind = 4 ) msta2

integer ( kind = 4 ) n

real ( kind = 8 ) pi

real ( kind = 8 ) pv0

real ( kind = 8 ) pv1

real ( kind = 8 ) rp2

real ( kind = 8 ),intent(in) :: v !modification by T-MATT

real ( kind = 8 ) v0

real ( kind = 8 ) vg

real ( kind = 8 ) vl

real ( kind = 8 ),intent(out) :: vm !modification by T-MATT

real ( kind = 8 ) vv

real ( kind = 8 ) w0

real ( kind = 8 ) w1

real ( kind = 8 ) wa

real ( kind = 8 ) ya0

real ( kind = 8 ) ya1
real ( kind = 8 ) yak

complex ( kind = 8 ),intent(in) :: z !modification by T-MATT

complex ( kind = 8 ) z1

complex ( kind = 8 ) z2

complex ( kind = 8 ) zk


pi = 3.141592653589793D+00

rp2 = 0.63661977236758D+00

ci = cmplx ( 0.0D+00, 1.0D+00, kind = 8 )

a0 = abs ( z )

z1 = z

z2 = z * z

n = int ( v )

v0 = v - n

pv0 = pi * v0

pv1 = pi * ( 1.0D+00 + v0 )


if ( a0 < 1.0D-100 ) then


  do k = 0, n

    cbj(k) = cmplx ( 0.0D+00, 0.0D+00, kind = 8 )

    cdj(k) = cmplx ( 0.0D+00, 0.0D+00, kind = 8 )

    cby(k) = - cmplx ( 1.0D+30, 0.0D+00, kind = 8 )

    cdy(k) = cmplx ( 1.0D+30, 0.0D+00, kind = 8 )

  end do


  if ( v0 == 0.0D+00 ) then

    cbj(0) = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )

    cdj(1) = cmplx ( 0.5D+00, 0.0D+00, kind = 8 )

  else

    cdj(0) = cmplx ( 1.0D+30, 0.0D+00, kind = 8 )

  end if


  vm = v

  return


end if
```

```
if ( real ( z, kind = 8 ) < 0.0D+00 ) then
  z1 = -z
end if

if ( a0 <= 12.0D+00 ) then

  do l = 0, 1
    vl = v0 + l
    cjvl = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
    cr = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
    do k = 1, 40
      cr = -0.25D+00 * cr * z2 / ( k * ( k + vl ) )
      cjvl = cjvl + cr
      if ( abs ( cr ) < abs ( cjvl ) * 1.0D-15 ) then
        exit
      end if
    end do

    vg = 1.0D+00 + vl
    call gamma ( vg, ga )
    ca = ( 0.5D+00 * z1 ) ** vl / ga

    if ( l == 0 ) then
      cjv0 = cjvl * ca
    else
      cjv1 = cjvl * ca
    end if

  end do

else

  if ( a0 < 35.0D+00 ) then
    k0 = 11
  else if ( a0 <50.0D+00 ) then
    k0 = 10
  else
    k0 = 8
  end if

  do j = 0, 1
    vv = 4.0D+00 * ( j + v0 ) * ( j + v0 )
    cpz = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
    crp = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
    do k = 1, k0
      crp = - 0.78125D-02 * crp &
        * ( vv - ( 4.0D+00 * k - 3.0D+00 ) ** 2 ) &
        * ( vv - ( 4.0D+00 * k - 1.0D+00 ) ** 2 )  &
        / ( k * ( 2.0D+00 * k - 1.0D+00 ) * z2 )
      cpz = cpz + crp
    end do
    cqz = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
    crq = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
    do k = 1, k0
      crq = -0.78125D-02 * crq &
        * ( vv - ( 4.0D+00 * k - 1.0D+00 ) ** 2 ) &
        * ( vv - ( 4.0D+00 * k + 1.0D+00 ) ** 2 ) &
        / ( k * ( 2.0D+00 * k + 1.0D+00 ) * z2 )
      cqz = cqz + crq
    end do
    cqz = 0.125D+00 * ( vv - 1.0D+00 ) * cqz / z1
    zk = z1 - ( 0.5D+00 * ( j + v0 ) + 0.25D+00 ) * pi
    ca0 = sqrt ( rp2 / z1 )
    cck = cos ( zk )
    csk = sin ( zk )
    if ( j == 0 ) then
      cjv0 = ca0 * ( cpz * cck - cqz * csk )
      cyv0 = ca0 * ( cpz * csk + cqz * cck )
    else if ( j == 1 ) then
      cjv1 = ca0 * ( cpz * cck - cqz * csk )
      cyv1 = ca0 * ( cpz * csk + cqz * cck )
    end if
  end do

end if
```

```
if ( a0 <= 12.0D+00 ) then

  if ( v0 .ne. 0.0D+00 ) then

    do l = 0, 1
      vl = v0 + l
      cjvl = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
      cr = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
      do k = 1, 40
        cr = -0.25D+00 * cr * z2 / ( k * ( k - vl ) )
        cjvl = cjvl + cr
        if ( abs ( cr ) < abs ( cjvl ) * 1.0D-15 ) then
          exit
        end if
      end do

      vg = 1.0D+00 - vl
      call gamma ( vg, gb )
      cb = ( 2.0D+00 / z1 ) ** vl / gb
      if ( l == 0 ) then
        cju0 = cjvl * cb
      else
        cju1 = cjvl * cb
      end if
    end do
    cyv0 = ( cjv0 * cos ( pv0 ) - cju0 ) / sin ( pv0 )
    cyv1 = ( cjv1 * cos ( pv1 ) - cju1 ) / sin ( pv1 )

  else

    cec = log ( z1 / 2.0D+00 ) + 0.5772156649015329D+00
    cs0 = cmplx ( 0.0D+00, 0.0D+00, kind = 8 )
    w0 = 0.0D+00
    cr0 = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
    do k = 1, 30
      w0 = w0 + 1.0D+00 / k
      cr0 = -0.25D+00 * cr0 / ( k * k ) * z2
      cs0 = cs0 + cr0 * w0
    end do

    cyv0 = rp2 * ( cec * cjv0 - cs0 )

    cs1 = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )

    w1 = 0.0D+00

    cr1 = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )

    do k = 1, 30

      w1 = w1 + 1.0D+00 / k

      cr1 = -0.25D+00 * cr1 / ( k * ( k + 1 ) ) * z2

      cs1 = cs1 + cr1 * ( 2.0D+00 * w1 + 1.0D+00 / ( k + 1.0D+00 ) )

    end do

    cyv1 = rp2 * ( cec * cjv1 - 1.0D+00 / z1 - 0.25D+00 * z1 * cs1 )


  end if


end if


if ( real ( z, kind = 8 ) < 0.0D+00 ) then


  cfac0 = exp ( pv0 * ci )

  cfac1 = exp ( pv1 * ci )


  if ( imag ( z ) < 0.0D+00 ) then

    cyv0 = cfac0 * cyv0 - 2.0D+00 * ci * cos ( pv0 ) * cjv0

    cyv1 = cfac1 * cyv1 - 2.0D+00 * ci * cos ( pv1 ) * cjv1

    cjv0 = cjv0 / cfac0

    cjv1 = cjv1 / cfac1

  else if ( 0.0D+00 < imag ( z ) ) then

    cyv0 = cyv0 / cfac0 + 2.0D+00 * ci * cos ( pv0 ) * cjv0

    cyv1 = cyv1 / cfac1 + 2.0D+00 * ci * cos ( pv1 ) * cjv1

    cjv0 = cfac0 * cjv0

    cjv1 = cfac1 * cjv1

  end if


end if


cbj(0) = cjv0

cbj(1) = cjv1
```

```
if ( 2 <= n .and. n <= int ( 0.25D+00 * a0 ) ) then

  cf0 = cjv0
  cf1 = cjv1
  do k = 2, n
    cf = 2.0D+00 * ( k + v0 - 1.0D+00 ) / z * cf1 - cf0
    cbj(k) = cf
    cf0 = cf1
    cf1 = cf
  end do

else if ( 2 <= n ) then

  m = msta1 ( a0, 200 )
  if ( m < n ) then
    n = m
  else
    m = msta2 ( a0, n, 15 )
  end if
  cf2 = cmplx ( 0.0D+00, 0.0D+00, kind = 8 )
  cf1 = cmplx ( 1.0D-30, 0.0D+00, kind = 8 )
  do k = m, 0, -1
    cf = 2.0D+00 * ( v0 + k + 1.0D+00 ) / z * cf1 - cf2
    if ( k <= n ) then
      cbj(k) = cf
    end if
    cf2 = cf1
    cf1 = cf
  end do
  if ( abs ( cjv1 ) < abs ( cjv0 ) ) then
    cs = cjv0 / cf
  else
    cs = cjv1 / cf2
  end if

  do k = 0, n
    cbj(k) = cs * cbj(k)
  end do
end if

  cdj(0) = v0 / z * cbj(0) - cbj(1)
  do k = 1, n
    cdj(k) = - ( k + v0 ) / z * cbj(k) + cbj(k-1)
  end do

  cby(0) = cyv0
  cby(1) = cyv1
  ya0 = abs ( cyv0 )
  lb = 0
  cg0 = cyv0
  cg1 = cyv1
  do k = 2, n
    cyk = 2.0D+00 * ( v0 + k - 1.0D+00 ) / z * cg1 - cg0
    if ( abs ( cyk ) <= 1.0D+290 ) then
      yak = abs ( cyk )
      ya1 = abs ( cg0 )
      if ( yak < ya0 .and. yak < ya1 ) then
        lb = k
      end if
      cby(k) = cyk
      cg0 = cg1
      cg1 = cyk
    end if
  end do

  if ( 4 < lb .and. imag ( z ) /= 0.0D+00 ) then

    do

      if ( lb == lb0 ) then
        exit
      end if

      ch2 = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )
      ch1 = cmplx ( 0.0D+00, 0.0D+00, kind = 8 )
```

```
lb0 = lb

do k = lb, 1, -1

  ch0 = 2.0D+00 * ( k + v0 ) / z * ch1 - ch2

  ch2 = ch1

  ch1 = ch0

end do

cp12 = ch0

cp22 = ch2

ch2 = cmplx ( 0.0D+00, 0.0D+00, kind = 8 )

ch1 = cmplx ( 1.0D+00, 0.0D+00, kind = 8 )

do k = lb, 1, -1

  ch0 = 2.0D+00 * ( k + v0 ) / z * ch1 - ch2

  ch2 = ch1

  ch1 = ch0

end do

cp11 = ch0

cp21 = ch2


if ( lb == n ) then

  cbj(lb+1) = 2.0D+00 * ( lb + v0 ) / z * cbj(lb) - cbj(lb-1)

end if


if ( abs ( cbj(1) ) < abs ( cbj(0) ) ) then

  cby(lb+1) = ( cbj(lb+1) * cyv0 - 2.0D+00 * cp11 / ( pi * z ) ) &

    / cbj(0)

  cby(lb) = ( cbj(lb) * cyv0 + 2.0D+00 * cp12 / ( pi * z ) ) / cbj(0)

else

  cby(lb+1) = ( cbj(lb+1) * cyv1 - 2.0D+00 * cp21 / ( pi * z ) ) &

    / cbj(1)

  cby(lb) = ( cbj(lb) * cyv1 + 2.0D+00 * cp22 / ( pi * z ) ) / cbj(1)

end if


cyl2 = cby(lb+1)

cyl1 = cby(lb)

do k = lb - 1, 0, -1

  cylk = 2.0D+00 * ( k + v0 + 1.0D+00 ) / z * cyl1 - cyl2

  cby(k) = cylk

  cyl2 = cyl1
```

```
          cyl1 = cylk
        end do

      cyl1 = cby(lb)
      cyl2 = cby(lb+1)
      do k = lb + 1, n - 1
        cylk = 2.0D+00 * ( k + v0 ) / z * cyl2 - cyl1
        cby(k+1) = cylk
        cyl1 = cyl2
        cyl2 = cylk
      end do

      do k = 2, n
        wa = abs ( cby(k) )
        if ( wa < abs ( cby(k-1) ) ) then
          lb = k
        end if
      end do

    end do

  end if

  cdy(0) = v0 / z * cby(0) - cby(1)
  do k = 1, n
    cdy(k) = cby(k-1) - ( k + v0 ) / z * cby(k)
  end do
  vm = n + v0

  return
end subroutine cjyva

!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    subroutine gamma ( x, ga )

!*****************************************************************************80
!
```

```
!! GAMMA evaluates the Gamma function.
!
!  Licensing:
!
!    The original FORTRAN77 version of this routine is copyrighted by
!    Shanjie Zhang and Jianming Jin.  However, they give permission to
!    incorporate this routine into a user program that the copyright
!    is acknowledged.
!
!  Modified:
!
!    08 September 2007
!
!  Author:
!
!    Original FORTRAN77 version by Shanjie Zhang, Jianming Jin.
!    FORTRAN90 version by John Burkardt.
!
!  Reference:
!
!    Shanjie Zhang, Jianming Jin,
!    Computation of Special Functions,
!    Wiley, 1996,
!    ISBN: 0-471-11963-6,
!    LC: QA351.C45
!
!  Parameters:
!
!    Input, real ( kind = 8 ) X, the argument.
!    X must not be 0, or any negative integer.
!
!    Output, real ( kind = 8 ) GA, the value of the Gamma function.
!
  implicit none

  real ( kind = 8 ), dimension ( 26 ) :: g = (/ &
    1.0D+00, &
    0.5772156649015329D+00, &
```

```
 -0.6558780715202538D+00, &
 -0.420026350340952D-01, &
  0.1665386113822915D+00, &
 -0.421977345555443D-01, &
 -0.96219715278770D-02, &
  0.72189432466630D-02, &
 -0.11651675918591D-02, &
 -0.2152416741149D-03, &
  0.1280502823882D-03, &
 -0.201348547807D-04, &
 -0.12504934821D-05, &
  0.11330272320D-05, &
 -0.2056338417D-06, &
  0.61160950D-08, &
  0.50020075D-08, &
 -0.11812746D-08, &
  0.1043427D-09, &
  0.77823D-11, &
 -0.36968D-11, &
  0.51D-12, &
 -0.206D-13, &
 -0.54D-14, &
  0.14D-14, &
  0.1D-15 /)
real ( kind = 8 ), intent(out) :: ga !modification by T-MATT
real ( kind = 8 ) gr
integer ( kind = 4 ) k
integer ( kind = 4 ) m
integer ( kind = 4 ) m1
real ( kind = 8 ), parameter :: pi = 3.141592653589793D+00
real ( kind = 8 ) r
real ( kind = 8 ), intent(in) :: x !modification by T-MATT
real ( kind = 8 ) z

if ( x == aint ( x ) ) then

  if ( 0.0D+00 < x ) then
    ga = 1.0D+00
    m1 = int ( x ) - 1
    do k = 2, m1
      ga = ga * k
    end do
  else
    ga = 1.0D+300
  end if

else

  if ( 1.0D+00 < abs ( x ) ) then
    z = abs ( x )
    m = int ( z )
    r = 1.0D+00
    do k = 1, m
      r = r * ( z - real ( k, kind = 8 ) )
    end do
    z = z - real ( m, kind = 8 )
  else
    z = x
  end if

  gr = g(26)
  do k = 25, 1, -1
    gr = gr * z + g(k)
  end do

  ga = 1.0D+00 / ( gr * z )

  if ( 1.0D+00 < abs ( x ) ) then
    ga = ga * r
    if ( x < 0.0D+00 ) then
      ga = - pi / ( x* ga * sin ( pi * x ) )
    end if
  end if

end if
```

```
  return
end subroutine gamma

function msta1 ( x, mp )

!*****************************************************************************80
!
!! MSTA1 determines a backward recurrence starting point for Jn(x).
!
!  Discussion:
!
!    This procedure determines the starting point for backward
!    recurrence such that the magnitude of
!    Jn(x) at that point is about 10^(-MP).
!
!  Licensing:
!
!    This routine is copyrighted by Shanjie Zhang and Jianming Jin.  However,
!    they give permission to incorporate this routine into a user program
!    provided that the copyright is acknowledged.
!
!  Modified:
!
!    08 July 2012
!
!  Author:
!
!    Shanjie Zhang, Jianming Jin
!
!  Reference:
!
!    Shanjie Zhang, Jianming Jin,
!    Computation of Special Functions,
!    Wiley, 1996,
!    ISBN: 0-471-11963-6,
!    LC: QA351.C45.
!
!  Parameters:
```

```
!
!    Input, real ( kind = 8 ) X, the argument.
!
!    Input, integer ( kind = 4 ) MP, the negative logarithm of the
!    desired magnitude.
!
!    Output, integer ( kind = 4 ) MSTA1, the starting point.
!
  implicit none

  real ( kind = 8 ) a0
  real ( kind = 8 ) envj
  real ( kind = 8 ) f
  real ( kind = 8 ) f0
  real ( kind = 8 ) f1
  integer ( kind = 4 ) it
  integer ( kind = 4 ) mp
  integer ( kind = 4 ) msta1
  integer ( kind = 4 ) n0
  integer ( kind = 4 ) n1
  integer ( kind = 4 ) nn
  real ( kind = 8 ) x

  a0 = abs ( x )
  n0 = int ( 1.1D+00 * a0 ) + 1
  f0 = envj ( n0, a0 ) - mp
  n1 = n0 + 5
  f1 = envj ( n1, a0 ) - mp
  do it = 1, 20
    nn = n1 - ( n1 - n0 ) / ( 1.0D+00 - f0 / f1 )
    f = envj ( nn, a0 ) - mp
    if ( abs ( nn - n1 ) < 1 ) then
      exit
    end if
    n0 = n1
    f0 = f1
    n1 = nn
    f1 = f
  end do

  msta1 = nn

  return
end function msta1

function msta2 ( x, n, mp )

!*****************************************************************************80
!
!! MSTA2 determines a backward recurrence starting point for Jn(x).
!
!  Discussion:
!
!    This procedure determines the starting point for a backward
!    recurrence such that all Jn(x) has MP significant digits.
!
!  Licensing:
!
!    This routine is copyrighted by Shanjie Zhang and Jianming Jin.  However,
!    they give permission to incorporate this routine into a user program
!    provided that the copyright is acknowledged.
!
!  Modified:
!
!    08 July 2012
!
!  Author:
!
!    Shanjie Zhang, Jianming Jin
!
!  Reference:
!
!    Shanjie Zhang, Jianming Jin,
!    Computation of Special Functions,
!    Wiley, 1996,
!    ISBN: 0-471-11963-6,
!    LC: QA351.C45.
!
!  Parameters:
!
!    Input, real ( kind = 8 ) X, the argument of Jn(x).
!
!    Input, integer ( kind = 4 ) N, the order of Jn(x).
!
!    Input, integer ( kind = 4 ) MP, the number of significant digits.
!
!    Output, integer ( kind = 4 ) MSTA2, the starting point.
!
  implicit none

  real ( kind = 8 ) a0
  real ( kind = 8 ) ejn
  real ( kind = 8 ) envj
  real ( kind = 8 ) f
  real ( kind = 8 ) f0
  real ( kind = 8 ) f1
  real ( kind = 8 ) hmp
  integer ( kind = 4 ) it
  integer ( kind = 4 ) mp
  integer ( kind = 4 ) msta2
  integer ( kind = 4 ) n
  integer ( kind = 4 ) n0
  integer ( kind = 4 ) n1
  integer ( kind = 4 ) nn
  real ( kind = 8 ) obj
  real ( kind = 8 ) x

  a0 = abs ( x )
  hmp = 0.5D+00 * mp
  ejn = envj ( n, a0 )

  if ( ejn <= hmp ) then
    obj = mp
    n0 = int ( 1.1D+00 * a0 )
  else
    obj = hmp + ejn
    n0 = n
  end if

  f0 = envj ( n0, a0 ) - obj
  n1 = n0 + 5
  f1 = envj ( n1, a0 ) - obj

  do it = 1, 20
    nn = n1 - ( n1 - n0 ) / ( 1.0D+00 - f0 / f1 )
    f = envj ( nn, a0 ) - obj
    if ( abs ( nn - n1 ) < 1 ) then
      exit
    end if
    n0 = n1
    f0 = f1
    n1 = nn
    f1 = f
  end do

  msta2 = nn + 10

  return
end function msta2

function envj ( n, x )

!*****************************************************************************80
!
!! ENVJ is a utility function used by MSTA1 and MSTA2.
!
!  Licensing:
!
!    This routine is copyrighted by Shanjie Zhang and Jianming Jin.  However,
!    they give permission to incorporate this routine into a user program
!    provided that the copyright is acknowledged.
!
```

```
!  Modified:
!
!    14 March 2012
!
!  Author:
!
!    Shanjie Zhang, Jianming Jin
!
!  Reference:
!
!    Shanjie Zhang, Jianming Jin,
!    Computation of Special Functions,
!    Wiley, 1996,
!    ISBN: 0-471-11963-6,
!    LC: QA351.C45.
!
!  Parameters:
!
!    Input, integer ( kind = 4 ) N, ?
!
!    Input, real ( kind = 8 ) X, ?
!
!    Output, real ( kind = 8 ) ENVJ, ?
!
  implicit none

  real ( kind = 8 ) envj
  integer ( kind = 4 ) n
  real ( kind = 8 ) x

  envj = 0.5D+00 * log10 ( 6.28D+00 * n ) - n * log10 ( 1.36D+00 * x / n )

  return
end function envj

subroutine lpmns ( m, n, x, pm, pd )

!*****************************************************************************80
```

```
!
!! LPMNS computes associated Legendre functions Pmn(X) and derivatives P'mn(x).
!
!  Licensing:
!
!    This routine is copyrighted by Shanjie Zhang and Jianming Jin.  However,
!    they give permission to incorporate this routine into a user program
!    provided that the copyright is acknowledged.
!
!  Modified:
!
!    18 July 2012
!
!  Author:
!
!    Shanjie Zhang, Jianming Jin
!
!  Reference:
!
!    Shanjie Zhang, Jianming Jin,
!    Computation of Special Functions,
!    Wiley, 1996,
!    ISBN: 0-471-11963-6,
!    LC: QA351.C45.
!
!  Parameters:
!
!    Input, integer ( kind = 4 ) M, the order of Pmn(x).
!
!    Input, integer ( kind = 4 ) N, the degree of Pmn(x).
!
!    Input, real ( kind = 8 ) X, the argument.
!
!    Output, real ( kind = 8 ) PM(0:N), PD(0:N), the values and derivatives
!    of the function from degree 0 to N.
!
  implicit none
```

```
integer ( kind = 4 ) n

integer ( kind = 4 ) k
integer ( kind = 4 ) m
real ( kind = 8 ) pm(0:n)
real ( kind = 8 ) pm0
real ( kind = 8 ) pm1
real ( kind = 8 ) pm2
real ( kind = 8 ) pmk
real ( kind = 8 ) pd(0:n)
real ( kind = 8 ) x
real ( kind = 8 ) x0

do k = 0, n
  pm(k) = 0.0D+00
  pd(k) = 0.0D+00
end do

if ( abs ( x ) == 1.0D+00 ) then

  do k = 0, n
    if ( m == 0 ) then
      pm(k) = 1.0D+00
      pd(k) = 0.5D+00 * k * ( k + 1.0D+00 )
      if ( x < 0.0D+00 ) then
        pm(k) = ( -1.0D+00 ) ** k * pm(k)
        pd(k) = ( -1.0D+00 ) ** ( k + 1 ) * pd(k)
      end if
    else if ( m == 1 ) then
      pd(k) = 1.0D+300
    else if ( m == 2 ) then
      pd(k) = -0.25D+00 * ( k + 2.0D+00 ) * ( k + 1.0D+00 ) &
        * k * ( k - 1.0D+00 )
      if ( x < 0.0D+00 ) then
        pd(k) = ( -1.0D+00 ) ** ( k + 1 ) * pd(k)
      end if
    end if
  end do
```

```
    return
  end if

  x0 = abs ( 1.0D+00 - x * x )
  pm0 = 1.0D+00
  pmk = pm0
  do k = 1, m
    pmk = ( 2.0D+00 * k - 1.0D+00 ) * sqrt ( x0 ) * pm0
    pm0 = pmk
  end do
  pm1 = ( 2.0D+00 * m + 1.0D+00 ) * x * pm0
  pm(m) = pmk
  pm(m+1) = pm1
  do k = m + 2, n
    pm2 = ( ( 2.0D+00 * k - 1.0D+00 ) * x * pm1 &
      - ( k + m - 1.0D+00 ) * pmk ) / ( k - m )
    pm(k) = pm2
    pmk = pm1
    pm1 = pm2
  end do

  pd(0) = ( ( 1.0D+00 - m ) * pm(1) - x * pm(0) ) &
    / ( x * x - 1.0D+00 )
  do k = 1, n
    pd(k) = ( k * x * pm(k) - ( k + m ) * pm(k-1) ) &
      / ( x * x - 1.0D+00 )
  end do

  return
end subroutine lpmns

subroutine lpn ( n, x, pn, pd )

!*****************************************************************************80
!
!! LPN computes Legendre polynomials Pn(x) and derivatives Pn'(x).
!
!  Licensing:
```

```
!
!    This routine is copyrighted by Shanjie Zhang and Jianming Jin.  However,
!    they give permission to incorporate this routine into a user program
!    provided that the copyright is acknowledged.
!
!  Modified:
!
!    07 July 2012
!
!  Author:
!
!    Shanjie Zhang, Jianming Jin
!
!  Reference:
!
!    Shanjie Zhang, Jianming Jin,
!    Computation of Special Functions,
!    Wiley, 1996,
!    ISBN: 0-471-11963-6,
!    LC: QA351.C45.
!
!  Parameters:
!
!    Input, integer ( kind = 4 ) N, the maximum degree.
!
!    Input, real ( kind = 8 ) X, the argument.
!
!    Output, real ( kind = 8 ) PN(0:N), PD(0:N), the values and derivatives
!    of the polyomials of degrees 0 to N at X.
!
  implicit none

  integer ( kind = 4 ) n

  integer ( kind = 4 ) k
  real ( kind = 8 ) p0
  real ( kind = 8 ) p1
  real ( kind = 8 ) pd(0:n)
  real ( kind = 8 ) pf
  real ( kind = 8 ) pn(0:n)
  real ( kind = 8 ) x

  pn(0) = 1.0D+00
  pn(1) = x
  pd(0) = 0.0D+00
  pd(1) = 1.0D+00
  p0 = 1.0D+00
  p1 = x

  do k = 2, n

    pf = ( 2.0D+00 * k - 1.0D+00 ) / k * x * p1 &
      - ( k - 1.0D+00 ) / k * p0
    pn(k) = pf

    if ( abs ( x ) == 1.0D+00 ) then
      pd(k) = 0.5D+00 * x ** ( k + 1 ) * k * ( k + 1.0D+00 )
    else
      pd(k) = k * ( p1 - x * pf ) / ( 1.0D+00 - x * x )
    end if

    p0 = p1
    p1 = pf

  end do

  return
end subroutine lpn
```