UNIVERSITY OF ALBERTA

RADIATION TRANSPORT

BY

HUMAYON BUTT

A thesis submitted to the faculty of Graduate Studies and Research in partial fulfilment of the requirement for the degree of Master of Science.

IN

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1993

ISBN 0-315-88123-2

UNIVERSITY OF ALBERTA
RELEASE FORM

NAME OF AUTHOR:	**HUMAYON BUTT**
TITLE OF THESIS:	**RADIATION TRANSPORT FOR THE MEDUSA CODE**
DEGREE:	**MASTER OF SCIENCE**
FIELD OF STUDY:	**LASER PLASMA INTERACTION**
YEAR THIS DEGREE GRANTED:	**1993**

Humayon Butt

Apt 1909
11135 83 Ave
Edmonton, Alberta
T6G 2C6
Canada

MAy. 11 1993

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDY

The undersigned certify that they have read and recommended to the faculty of Graduate Studies and Research for acceptance, a thesis entitled RADIATION TRANSPORT FOR THE MEDUSA CODE in partial fulfilment of the requirement for the degree of Master of Science in Laser Plasma interaction.

Dr. C. E. Capjack

Dr. C. R. James

Dr. R. Rankin

Dr. R. Fedosejves

Dr. J. C. Samson

Jan. 08 1993

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. C.E. Capjack and Dr. C.R. James for their financial support. I also like to thank Dr. C.E. Capjack and Dr. R. Rankin for bringing to my attention the Diffusion Synthetic Acceleration method and for their encouragement and supervision of this project.

Mr. D. Vick provided much helpful advice and suggestions and was always available for discussion. I would like to express my thanks to him. Dr. Marvin, from Lawerence Livermore National Laboratory, USA. was very helpful in deriving boundary conditions for the diffusion equation. In his absence, I like to express my appreciation for the discussion and correspondence that he provided.

I also like to thank my brother Hamid Butt for the computer (which was used for developing the code for the model problem) and the technical support for maintaining it.

ABSTRACT

The main objective of this research is to incorporate the effects of radiation transport into MEDUSA, a code used at the University of Alberta and other research laboratories to study laser plasma interaction phenomena. The effects of radiation transport on ablation pressure and the ablation mass are investigated. When radiation transport is included, the amount of laser energy that is available for ion, kinetic and thermal energies changes. This change in energy is also investigated. The solution of the radiative transfer equation for a one dimensional slab geometry is obtained by using a neutronic transport technique known as **DIFFUSION SYNTHETIC ACCELERATION (DSA)**. The secondary objective of this research is to corroborate the validity of the simplified algorithm developed by Marchand et al at the University of Alberta. The radiation transport algorithm is developed taking into consideration its eventual extension to two dimensional geometries. However, the extension to higher dimensions is not the objective of the present research.

CHAPTER ONE
INTRODUCTION

1.1 LASER DRIVEN NUCLEAR FUSION:

In early 1970, many researchers argued that one could use lasers as a mean of igniting thermonuclear fusion[1/1]. They showed theoretically that a thin shell filled with a deuterium-tritium mixture can be compressed to many times its solid density through laser driven ablative implosion. In an ablative implosion only the exterior of the shell is heated. The shell is shocked, compressed and driven toward the origin by the reaction force produced by the material streaming off. In this way fuel inside the shell can be brought to the densities required for thermonuclear fusion.

During compression, the heated fuel pellet emits electromagnetic radiation which can significantly influence the compression efficiency. These emitted photons can preheat the uncompressed target core, thereby reducing the final target compression. This in turn degrades the pellet gain. Therefore, it is of considerable importance to know the temperature on the inside of the target shell. Such knowledge allows one to choose the proper isentrope for compression calculations as well as to check the predictions of detailed numerical simulations of laser driven implosions. If preheating is sufficiently strong, it is possible that fuel may not ignite. Therefore, it is essential to minimize the fuel preheating during the implosion phase.

When the laser intensity is in the range of 10^{16} to 10^{18} W m^{-2}, energy transport due to radiation can exceed heat conduction due to other processes. Experiments[1/2] have shown that up to 25% of the laser input energy can be lost through radiative emission. From these experiments it can be concluded that by neglecting radiation transport, one may be omitting a significant energy loss mechanism. Therefore, a computer simulation without radiation transport may not yield results that are consistent with experimental observations. Radiation transport is a very complex process due to its nonlinear nature and because of the dependence of the radiation field on space, angle, frequency and time. A treatment of the interaction of the radiation field with atoms requires a detailed knowledge of various coefficients such as the atomic population densities, the absorption coefficient and emissitivity. Despite its complexity, it is essential to include at least a qualitative analysis of radiation transport in any computer simulation.

1.2 PHYSICS OF LASER PRODUCED PLASMA:

A detailed analysis of all of the processes taking place in a laser produced plasma is beyond the scope of this thesis. In this section, a brief summary of some of the processes which take place when a laser beam is incident on matter are discussed.

When a laser impinges on a target surface, it is initially absorbed by the production of photoelectrons. This leads to electron ionization and heating of the target. The target material evaporates, the vapour is ionized by collisions and a plasma layer is formed. If the intensity of the laser beam is sufficiently high the electronic field of the incident laser beam can lead to noncollisional heating and an outward acceleration of the plasma in an explosive manner. The plasma layer expands mostly in an outward direction while heat is conducted into the solid. This inward heat conduction leads to further evaporation of the target layer. This is called **ablation** of the target material. The momentum of the ablated plasma is balanced by momentum of the solid target. The momentum transferred by the action of the ablation pressure and the effect of the pondermotive force, launches a shock wave in the solid which after a short time overtakes the front of the heat wave. Assuming classical theory of heat conduction, this time is given as[1/3]

$$t_o = b\left(\frac{M}{2k}\right)^{\frac{3}{2}}\left(\frac{3}{2}kN\right)^{-2}\phi$$

where ϕ is the flux, N is the ion density, M is the atomic mass of the target, k is the Boltzmann constant and b is 2.0 E-13 If the laser pulse is short compared to t_o the heated matter has no time to move and energy diffuses into the target by heat conduction. On the other hand, if the laser pulse lasts for a time longer than t_o then expansion is important while laser energy is being absorbed. When the laser pulse is very long compared to t_o the plasma can be divided into three phases. Phase 0 has a mass density ρ_o of the undisturbed solid. Phase 1 has mass density ρ_1 and is called the dense phase of opaque ionized matter which is heated by the shock wave. This mass penetrates into phase 0 with the speed of the shock. Phase 2 is composed of plasma subliminated from the boundary of phase 1.

Two thermodynamic quantities, pressure and energy, directly govern the hydrodynamics of the plasma. The energy deposited by the laser pulse can be used to determine the temperature of the plasma and from this one can deduce the pressure. The pressure drives the expansion of the plasma and determines the strength of the recoiled shock launched into the solid and thereby establishes the implosion velocity of the plasma.

A plasma can be considered as either nonideal or ideal. In a nonideal plasma ($Q<Z$ where Q is the ion charge and Z is the atomic number) matter is partially ionized. The coulomb interaction is strong in this case and free electrons can exist in degenerate form. The ideal plasma ($Q=Z$) is fully ionized and the electrical interaction is weak due to the screening clouds around charges. The screening energy increases as the temperature is reduced and eventually exceeds kT. When this happens a strongly coupled plasma is formed which is nonideal. An approximation for the ionization energy is given as

$$I = 13.6 \left(\frac{Q}{n}\right)^2 \ ev$$

where n is the principle quantum number of the outer most electron. The above expression shows that higher Z plasmas are more difficult to ionize. An ideal plasma emits radiation by bremsstrahlung, radiative recombination and line radiation from hydrogen like or helium like ions. In a magnetically confined plasma with high Z, the bremsstrahlung radiation can become an important mechanism for cooling. For low density plasma, radiation freely escapes and it can be used to infer parameters such as the plasma density and temperature. In a partially ionized plasma, radiation is strongly absorbed by the bound electrons and the photon population builds up toward a black body distribution.

When a laser beam is incident on a material, the energy that is absorbed by free electrons leads to an increase in their temperature. An increase in temperature increases the Debye length which increases the ion interaction. This forces ions to redistribute so as to reach equilibrium. As plasma is heated, different types of emission spectra are observed. The continuum emission spectrum reflects the energy spectrum of the free electrons. The energetic x-ray signals indicate the presence of super-thermal electrons.

In a plasma, hot regions radiate strongly and emitted photons transport energy to cooler parts of the plasma or permit energy to escape altogether. Since radiation travels rapidly, it plays an important role in plasma hydrodynamics. The radiation that is emitted and absorbed is due to many different processes which take place at a subatomic level. Some of the important processes are discussed below.

Bremsstrahlung Emission: In bremsstrahlung emission, radiation is emitted when electrons are scattered by atoms and ions. As electrons approach a nuclei they are deaccelerated and radiation is emitted.

Free-Free absorption: Absorption of a photon by a free electron is known as free-free absorption. The electron absorption cross-sectional area is obtained by using the principle of balance which states that at thermal equilibrium, the absorption rate is equal to the rate of emission and is given as[1/3]

$$\sigma^{B} = \frac{8\pi^3}{3\sqrt{3}} Z^2 a_o^5 \left(\frac{e^2}{a_o h\nu}\right)^3 \frac{e^2}{a_o \epsilon}$$

where Z is the atomic number, e is the electron charge, h is the Planckian constant, a_o is the radiation constant (=7.5607E-16), ϵ is the permeability constant of the plasma and ν is the frequency.

Radiative Recombination: Radiative recombination occurs when free electrons radiate more energy than their initial kinetic energy. At negative energies these electrons are trapped in bound states.

Photoelectric Effect: Here, a bound electron absorbs a photon and moves to a higher state.

Line Emission: Line emission occurs when a bound electron in an upper state spontaneously emits a photon of energy hν and falls into a lower state.

In a laser heated plasma, energy is deposited primarily into the electrons and the rate at which the electrons in turn transport this energy to higher density colder plasma determines both the efficiency of implosion and the plasma conditions in the region of deposition.

Mechanisms which are generally associated with the heating of the pellet are:

1. Classical electron thermal conduction.
2. Radiation transport.
3. Super thermal electron transport.
4. Shock waves.

Classical conduction is always present, and is modified by anomalous effects from ion and magnetic field fluctuations[1/4]. When excessive energy is deposited to thermal electrons during an implosion, the classical conduction wave burns through the shell, leading to preheating of the pellet. This condition can be readily avoided with proper pulse shaping. Radiative preheating arises from the reabsorption of bremsstrahlung, recombination and line radiation photons near the ablation surface where plasma electrons are typically in the range of 300 to 500 ev[1/5]. Super thermal electrons are generated when various thresholds for the absorptive instabilities are exceeded[1/6]; for example, resonant absorption readily furnishes 100 keV electrons with 10.6 μm laser light intensities of 10^{20} W m^{-2}. Shock waves[1/7] leads to large temperature gradients. They play an important role when thicker targets are used. At intermediate irradiance (10^{16} to 10^{17} W m^{-2}) the dominant heat transport mechanism can be radiation transport.

The effects of radiation transport have been experimentally investigated by many researchers[1/8,9]. In numerous experiments, different pellet materials have been used in order to observe the effects of radiation transport in low and high Z materials. Duston et al[1/2,8] used carbon as a target material, and rejected the common notion that for lower Z material radiation transport is negligible. This notion was due to the assumption that for lower Z materials the bound electrons would be easily stripped off and therefore radiation would make a negligible contribution to plasma dynamics and energy transport in these targets. Yaakobi et al[1/9] have shown that use of a low Z (plastic) coating reduces the preheating, thus leading to a higher density ablative implosion.

1.3 OBJECTIVE AND ORGANIZATION OF THE THESIS:

When a laser beam is incident on a target, radiation is emitted and then transported and reabsorbed in the target. This can significantly affect the hydrodynamics of the target and ablation parameters. In laser fusion, a precise knowledge of these parameters is crucial for target design. Even though our research does not involve laser induced nuclear fusion, inclusion of radiation transport is useful for a better understanding of radiative effects in other problems, such as use of emitted radiation as a source of x-rays[1/10] for lithography applications. Therefore, by including radiation transport in MEDUSA, the behavior of laser heated plasma can be better understood. The Diffusion Synthetic Acceleration technique is used to implement the radiative transfer equation in the MEDUSA code. The main objectives of the present research are:

i. To develop an algorithm which can be extended to solve the radiative transfer equation in two dimensions.

ii. To corroborate Marchand's algorithm.

The thesis consists of five chapters. In Chapter One, the importance of the understanding of radiation relevant to laser driven nuclear fusion is discussed, followed by a prologue to the processes involved in laser plasma interactions. In Chapter Two, the derivation and validity of the radiative transfer equation are discussed. Chapter Three includes a detailed analysis of the Diffusion Synthetic Acceleration Scheme (DSA). Chapter Four includes simulations with the radiation code when a model problem is used to test the validity of the DSA model. Chapter Five contains MEDUSA simulations using Marchand's model and the present model. The main objective is to present a comparison between Marchand's model and the Diffusion Synthetic Acceleration model.

Finally, a detailed analysis of the equations involved in the Diffusion Synthetic Acceleration model is included in Appendix A. This includes a Fourier stability analysis (where possible), the discretization of all equations involved in the algorithm, and a discussion of the boundary conditions. Marchand's model, currently used in MEDUSA to study radiative effects in laser produced plasma is reviewed in Appendix B. In Appendix C, a brief discussion is presented of the atomic physics data used in MEDUSA. A copy of the radiation transport code is included at the end of the thesis in Appendix D.

CHAPTER TWO
RADIATIVE TRANSFER EQUATION

2.1 DERIVATION OF RADIATIVE TRANSFER EQUATION:

The radiative transfer equation represents the various processes that take place in a plasma at a sub-atomic level. These processes lead to the generation and transport of radiation, resulting in a change in the number of photons in a given region. The time rate of change of the photons in a volume ΔV is given by

$$\frac{\partial[f(r,\nu,\Omega,t)\Delta V]}{\partial t} = \Delta V \frac{\partial[f(r,\nu,\Omega,t)]}{\partial t} \tag{2.1.01}$$

where $\Delta V = \Delta x \Delta y \Delta z \Delta \mu \Delta \varphi$ and f is the distribution function. Physically f is defined such that the number of photons at a given time t at a location in space r in a differential volume element dr with frequency interval $d\nu$ travelling in direction Ω in a solid angle interval $d\Omega$ is

$$dn = f \, dr \, d\nu \, d\Omega$$

This time rate of change of photons in a volume element ΔV depends on:

1. Net rate of streaming of photons out of a volume through the bounding surface.
2. Absorption of photons within the volume element.
3. Emission of photons from the volume element.
4,5. Scattering into and out of the volume element.

A detailed description of the physical interpretation of the above processes can be found elsewhere[2/1]. In this section, they are discussed briefly in order to derive the radiative transfer equation.

1. The rate of photon loss through the surface of a cube perpendicular to the x-axis is

$$streaming_x = \dot{x}f(r,\nu,\Omega,t)]_x^{x+\Delta x}\Delta y\Delta z\Delta\nu\Delta\mu\Delta\varphi = \Delta V\frac{\partial}{\partial x}[\dot{x}f(r,\nu,\Omega,t)]$$

(2.1.02)

where $\dot{x}$ denotes the x-components of the photon velocity and $\Delta y\Delta z\Delta\mu\Delta\varphi$ is the appropriate surface area. From this one can determine the net rate of streaming from the cube to be

$$streaming = \Delta V\left[\frac{\partial(\dot{x}f)}{\partial x}+\frac{\partial(\dot{y}f)}{\partial y}+\frac{\partial(\dot{z}f)}{\partial z}+\frac{\partial(\dot{\nu}f)}{\partial\nu}+\frac{\partial(\dot{\mu}f)}{\partial\mu}+\frac{\partial(\dot{\phi}f)}{\partial\varphi}\right]$$

(2.1.03)

where μ and φ derivatives represent the rate of change of photons in the directions μ and φ (the polar angle with respect to z-axis and the azimuthal angle with respect to the horizontal axis), respectively.

2. The rate of absorption is equal to the product of the number of photons in the volume element ΔV and the probability of absorption per photon per unit time

   rate of absorption $= c\sigma_a f\Delta V$ (2.1.04)

   where σ_a is the absorption coefficient.

3. The photon emission rate is equal to the spontaneous rate of emission divided by the energy of the photon

   photon emission rate $= S\Delta V/h\nu$ (2.1.05)

   where S is the rate of energy emission due to spontaneous processes.

4. The rate of out-scattering from the volume element is

$$c\Delta V\int_0^{\infty}d\nu'\int_{4\pi}d\Omega'\sigma_s(r,\nu\rightarrow\nu',\Omega.\Omega',t)\,f\,(r,\nu,\Omega,t)$$

5. The rate of in-scattering to the volume element is

$$c\Delta V\int_0^\infty d\acute{\nu}\int_{4\pi} d\acute{\Omega}\,\sigma_s(r,\acute{\nu}\rightarrow\nu,\acute{\Omega}.\Omega,t)\, f(r,\acute{\nu},\acute{\Omega},t)$$

(2.1.06;07)

In the above equations, σ_s is the scattering coefficient and c is the speed of light.

The equation of radiative transfer is obtained by summing the above equations with appropriate signs to designate a loss or gain. The resultant equation is

$$\frac{\partial f(\nu,\Omega)}{\partial t}+\frac{\partial(\dot{x}f)}{\partial x}+\frac{\partial(\dot{y}f)}{\partial y}+\frac{\partial(\dot{z}f)}{\partial z}+\frac{\partial(\dot{\nu}f)}{\partial \nu}+\frac{\partial(\dot{\mu}f)}{\partial t}+\frac{\partial(\dot{\varphi}f)}{\partial \varphi}$$
$$=\frac{S(\nu)}{h\nu}-c\sigma_a(\nu)f(\nu,\Omega)+c\int_0^\infty d\acute{\nu}\int_{4\pi} d\acute{\Omega}\,\sigma_s(\acute{\nu}\rightarrow\nu,\acute{\Omega}\bullet\Omega)f(\acute{\nu},\acute{\Omega})$$
$$-c\int_0^\infty d\acute{\nu}\int_{4\pi} d\acute{\Omega}\,\sigma_s(\nu\rightarrow\acute{\nu},\Omega\bullet\acute{\Omega})f(\nu,\Omega)$$

(2.1.08)

Equation (2.1.08) can be simplified by assuming that photons stream in straight lines. When this assumption is made, $\partial x/\partial t = c\Omega_x$, since photons travel with speed c in a given direction. Similar expressions can be obtained for the y,z axes. Since there is no change in frequency as photons stream, $\partial\nu/\partial t = 0$. Further, $\partial\mu/\partial t=\partial\varphi/\partial t=0.0$, since these angles are measured with respect to fixed axes in space. The distribution function f is related to the specific intensity as $I=ch\nu f$. Then with the above assumptions, equation (2.1.08) in terms of the specific intensity reduces to

$$\frac{1}{c}\frac{\partial I(\nu,\Omega)}{\partial t}+\Omega\bullet\nabla I(\nu,\Omega) = S(\nu)-\sigma_a(\nu)I(\nu,\Omega)$$
$$+\int_0^\infty d\acute{\nu}\int_{4\pi} d\acute{\Omega}\Big[\frac{\nu}{\acute{\nu}}\sigma_s(\acute{\nu}\rightarrow\nu,\acute{\Omega}\bullet\Omega)I(\acute{\nu},\acute{\Omega})-\sigma_s(\nu\rightarrow\acute{\nu},\Omega\bullet\acute{\Omega})I(\nu,\Omega)\Big]$$

(2.1.09)

It can be shown[2/1] that when induced processes are included, the processes of emission and scattering are enhanced by the factor

$$\left[1+\frac{c^2 I}{2h\nu^3}\right]$$

This enhancement leads to nonlinearity (in the form of I^2) in the transfer equation in the following form

$$\frac{1}{c}\frac{\partial I}{\partial t} + \Omega.\nabla I = \acute{\sigma}_a\,[B-I]$$
$$+ \int_0^\infty d\acute{\nu}\frac{\nu}{\acute{\nu}}\int_{4\pi} d\acute{\Omega}\sigma_s \acute{I}\,[1+\frac{c^2 I}{2h\nu^3}] - \int_0^\infty d\acute{\nu}\int_{4\pi} d\acute{\Omega}\sigma_s I\,[1+\frac{c^2 \acute{I}}{2h\acute{\nu}^3}]$$
$$where \qquad S=\acute{\sigma}_a B \qquad and \qquad \acute{\sigma}_a=\sigma_a[1+\frac{c^2 B}{2h\nu^3}]$$

(2.1.10)

Here, for simplicity the notations of I and σ_s are given as

$$I = I\,[r,\nu,\Omega,t] \qquad and \qquad \acute{I}= I\,[r,\acute{\nu},\acute{\Omega},t]$$

This nonlinearity severely limits one's ability to solve the transfer equation. Therefore, equation (2.1.10) is further simplified by assuming that there is no change in frequency upon scattering. With this assumption the induced in- and out-scattering contribution completely cancel each other, and the resulting equation is simply

$$\frac{1}{c}\frac{\partial I}{\partial t} + \Omega.\nabla I = \acute{\sigma}_a[B - I]$$

(2.1.11)

This equation forms the basis of the present research. The goal is to find a technique which can efficiently solve this equation. It is important to briefly discuss the validity of the radiative transfer equation, before we discuss the method used to solve the radiative transfer equation.

2.2 THE VALIDITY OF THE RADIATIVE TRANSFER EQUATION:

Many aspects of radiation transport were ignored when the radiative transfer equation was derived above. Some of them are beyond the scope of the present research and some are irrelevant to the objectives of the present research. In this section, approximations that are used in the derivation of the above transfer equation are briefly discussed.

There are two classes of approximation: those which are inherent in any radiative transfer equation and those which can be incorporated at the expense of simplicity.

The radiative transfer equation was derived using the assumption that photons are particles. However, it is well known that photons exhibit wave behavior, that is in reality a photon is a wave packet. By treating photons as particles, one intrinsically is neglecting the effects of interference, diffraction and reflection in radiation transport. In the radiative transfer equation, one deals with intensities rather than amplitudes, hence interference is nonexistent. This requires that the density of photons be sufficiently low so that overlap of the tails of the wave packets is negligibly small. Photons of sufficiently different frequencies do not interfere even when they coincide spatially; in other words, photons in the transfer equation are incoherent. It is further assumed that collision and emission processes occur instantaneously. That is, the loss or gain of photons due to these processes is characterized by σ_a, σ_s and B (absorption and scattering coefficient and the emissivity respectively) at a given instant of time rather than being dependent upon some sort of time average of these quantities over the collision or emission time. Finally, the diffraction and reflection wave behaviour of photons can not be manifested in any radiative transfer equation; since it requires that scattering centres be correlated as in a crystal and the spatial extent of wave packet be such that several scattering centres are encompassed by a photon. Therefore, in the transfer equation we treat scattering as independent and isolated events; in other words we assume scattering centres are randomly distributed or the wave packets are small compared to the distance between the scattering centres. This further implies that photons have no preferred direction.

The approximations discussed so far are inherent to all transfer equations of photons. The approximations which remain to be discussed are assumed in deriving the radiative transfer equation (2.1.08) and can be incorporated into the equation (2.1.08) at the expense of simplicity.

The two states of polarization of a photon can be added into the radiative transfer equation; this results in four transfer equations which in general are coupled. In practice photons do not move in straight lines since the refractive index is a function of position. If the refractive index is a function of time as well, then a photon changes its frequency as it streams between collisions. One can incorporate[2/1] refractive and dispersion effects into the streaming term of the transfer equation. Finally, σ_a and S are assumed to be angularly independent in (2.1.08), this implies no inherent preferred direction in the matter. However, in radiative hydrodynamic problems the material in question is normally moving. Therefore, as seen by an inertial frame observer, this motion does introduce a preferred direction, namely the direction of motion of the fluid. These angular dependence properties are not inherent properties of the material, but arise due to relative motion between the fluid and the observer. These are computed using the special theory of relativity. These angular effects are of the order of (u/c), where u is the speed of the fluid and c the speed of light; therefore, one can normally neglect the angular dependence in quantities such as the radiation intensity for nonrelativistic velocities.

CHAPTER THREE
DIFFUSION SYNTHETIC ACCELERATION

3.1 INTRODUCTION:

There are many classical methods, such as the Eddington Approximation, Asymptotic Diffusion Theory, the P-N Approximation[3/1] and the Monte Carlo Method[3/2] which have been used in the past to solve the radiative transfer equation. These methods are either too crude or are computationally inefficient. Since the late 1960's, researchers have been investigating other options for solving the radiative transfer equation. One of the most popular techniques is Synthetic Acceleration. This is sometimes used to solve neutronic transport problems, but due to its slow convergence for radiative transport problems it was not attractive. In the late 1970's and early 1980's this technique was greatly improved (by acceleration of the diffusion equation with the grey equation) and today is one of the most efficient methods of solving radiation transport problems in one dimension. The successful implementation of this method for a model problem at Lawrence Livermore National Laboratory was the main motivation for the development of the present code. Diffusion Synthetic Acceleration (**DSA**) is the main topic of this thesis, since it can have ambiguities in its formulation, a great deal of effort has been made to ensure that important aspects of the DSA technique relevant to one dimensional slab geometry are well explained.

3.2 BRIEF HISTORY:

In 1963, Kopp[3/3] reported that one can improve the spectral radius of the **source iteration** technique (also known as **power** or **lambda iteration**) by using a synthetic acceleration method. In synthetic acceleration one solves a low order form of the original equation to update the source term of the original equation for the next iteration. For example, the diffusion equation can be treated as a low order form of the transfer equation; the difficult part is to find the form of the low order equation which reduces the spectral radius and remains stable. The significant improvement and stability issues were not resolved until the late 1970's. In 1977, Alcouffe[3/4] reported the cause of instability and proved that the solution for DSA was unstable for cells with large width because the discretized form of the low order equation was not derived from the discretized form of the original equation. In 1985, Alcouffe et al[3/5] successfully reported on the implementation of the DSA method for a radiative model problem. The code presented here is similar to the work they published in 1985 with minor modifications. The DSA technique like any other method for radiative problems comes with its limitations. When it works, it is very efficient compared to the source iteration or to other known methods for one dimensional problems. Its efficiency degrades for higher dimensional and curvilinear geometry problems. These instability concerns have been eliminated to a great extent with the use of finite element discretization instead of finite difference discretization of the transfer equation in space.

3.3 DISCRETIZATION OF THE TRANSFER EQUATION:

The radiative transfer equation and the energy balance equation are solved simultaneously (this formulation allows us to test our code against a Marshak Wave Bench Mark). The equations for a one dimensional slab geometry are given by[3/5]

$$\frac{1}{c}\frac{\partial I}{\partial t}(x,\nu,\mu,t) + \mu\frac{\partial I}{\partial x}(x,\nu,\mu,t) + \sigma(x,\nu)I(x,\nu,\mu,t) = S(\nu,T) = \sigma(\nu,T)B(\nu,T)$$

$$\frac{\partial u_e(T)}{\partial t} = \beta(T)\left[\int_0^\infty \sigma(\nu,T)I_0(x,\nu,t)d\nu - c\sigma_p(T)u_e(T) + S_{ext}\right]$$

$$\textit{where} \quad u_e = \frac{c}{4\pi}\int_0^\infty B\,d\nu \quad \sigma_p = \frac{\int_0^\infty \sigma B\,d\nu}{\int_0^\infty B\,d\nu} \quad \beta = \frac{4aT^3}{c_v}$$

(3.3.01;02;03a,b,c)

Where I is the specific intensity of radiation, defined as the amount of radiative energy transferred across a surface element ds with normal **n** at point x, in frequency range of dν about ν, in an angle dμ about μ in time interval dt about t. The energy emission rate due to spontaneous processes is given by S; B is the emissivity, T is the material temperature, σ is the absorption coefficient defined such that the probability of a photon being absorbed in a distance ds is σds, c_v is the material specific heat per unit volume, a is the radiation constant, S_{ex} is the external source in the slab, u_e is the radiative energy density and I_o is the scalar intensity and is defined as

$$I_0(x,\nu,t) = \int_{-1}^{1} I(x,\mu,\nu,t)\, d\mu$$

(3.3.04)

3.3.1 DISCRETIZATION IN TIME:

The radiative transfer equation and the energy balance equation are discretized in time using a backward Euler time differencing scheme in order to ensure stability. The dependence of variables on frequency, angle, space and time is omitted for simplicity. The discretized form of (3.3.01,02,03) in time is given as

$$\frac{1}{c\Delta t^n}(I^{n+1}-I^n) + \mu\frac{\partial I^{n+1}}{\partial x} + \sigma^* I^{n+1} = \sigma^* B^{n+1}$$

$$\frac{1}{\Delta t^n}(u_e^{n+1}-u_e^n) = \beta^*\left[\int_0^\infty \sigma^* I_0^{n+1} d\nu - c\sigma_p^* u_e^{n+1} + S_{ex}\right]$$

$$u_e^{n+1} = \frac{c}{4\pi}\int_0^\infty B^{n+1}\, d\nu$$

(3.3.05;06;07)

The quantities with * superscript are computed using the temperature at time step n, since the temperature at time step (n+1) is not known. In addition, the following approximation to the function B is used[3/6]

$$B^{n+1} \approx \frac{c}{2}u_e^{n+1} b(\nu,T^*) \qquad \textit{subject to} \qquad \int_0^\infty b(\nu,T)\, d\nu = 1$$

(3.3.08)

By using (3.3.07) and (3.3.08) and assuming that the external source is zero, the function B reduces to

$$B^{n+1} = \frac{cb^*}{2} \frac{[\beta^* \Delta t^* \int_0^\infty \sigma^* I_o^{n+1} d\nu + u_e^n]}{(1 + c\Delta t^n \beta^* \sigma_p^*)}$$

$$= \frac{\chi\eta}{2}\left[\int_0^\infty \sigma^* I_0^{n+1} + \frac{u_e^n}{\beta^* \Delta t^n}\right]$$

$$where \quad \alpha = \frac{1}{\beta^* \sigma_p^*}, \quad \tau^n = \frac{1}{c\Delta t^n}, \quad \chi = \frac{\sigma^* b^*}{\sigma_p^*}, \quad \eta = \frac{1}{(1+\alpha\tau)}$$

(3.3.09;10a,b,c,d)

With these relations, equations (3.3.06;07) reduce to

$$\mu \frac{\partial I^{n+1}}{\partial x} + (\sigma^* + \tau) I^{n+1} = \frac{\chi\eta}{2}\left[\int_0^\infty \sigma^* I_0^{n+1} d\nu + \frac{u_e^n}{\beta^* \Delta t^n}\right] + \tau^n I^n$$

$$u_e^{n+1} = \frac{\eta}{2}\left[\int_0^\infty \sigma^* I_o \, d\nu + \frac{u_e^n}{\Delta \beta^*}\right]$$

(3.3.11a,b)

If the time index is suppressed in the radiative transfer equation(3.3.11a), it reduces to

$$\mu \frac{\partial I}{\partial x} + (\sigma + \tau) I = \frac{\chi\eta}{2}\int_0^\infty \sigma I_0 d\nu + Q$$

(3.3.12)

where Q contains all the terms evaluated at the previous time step and remains unchanged while the transfer equation is being solved iteratively. Q is given as

$$Q = \frac{\chi^* \eta}{2} \frac{u_e^n}{\beta^* \Delta t^n} + \tau^n I^n$$

(3.3.13)

3.3.2 DISCRETIZATION IN FREQUENCY:

We use equation (3.3.12) to discretize the radiative transfer equation in frequency. The discretized form of the equation is obtained by integrating (3.3.12) over the frequency interval dν as shown below

$$\int_{\nu_l}^{\nu_h} \mu \frac{\partial I}{\partial x}\, d\nu + \int_{\nu_l}^{\nu_h} (\sigma+\tau) I\, d\nu = \int_{\nu_l}^{\nu_h} \frac{\chi\eta}{2} \int_0^{\infty} \sigma I_0\, d\nu + \int_{\nu_l}^{\nu_h} Q\, d\nu$$

(3.3.14)

If we define

$$I_g = \int_{\nu_l}^{\nu_h} I\, d\nu \qquad Q_g = \int_{\nu_l}^{\nu_h} Q\, d\nu \qquad \sigma_g = \frac{\int_{\nu_l}^{\nu_h} \sigma I\, d\nu}{\int_{\nu_l}^{\nu_h} I\, d\nu} \qquad \chi_g = \int_{\nu_l}^{\nu_h} \chi\, d\nu$$

(3.3.15a,b,c,d)

then the multi-group transfer equation can be expressed as

$$\mu \frac{\partial I_g}{\partial x} + (\sigma_g+\tau) I_g = \frac{\chi_g \eta}{2} \Sigma_g\, \sigma_g I_{0g} + Q_g$$

(3.3.16)

3.3.3 DISCRETIZATION IN ANGULAR SPACE:

The discretized form of the radiative transfer equation (3.3.16) in angular space is obtained by integrating in angular space as defined below

$$\mu_m \frac{\partial I_{mg}}{\partial x} + (\sigma_g+\tau) I_{mg} = \frac{\chi_g \eta}{2} \Sigma_g^G\, \sigma_g I_{0g} + Q_{mg}$$

$$\textit{where} \qquad I_{0g} = \Sigma_{m=1}^{M} I_{mg}\omega_m$$

$$I_{mg} = \frac{1}{\Delta\omega_m} \int_{\omega_{m-\frac{1}{2}}}^{\omega_{m+\frac{1}{2}}} I_g\, d\omega_m$$

$$\textit{subject to} \qquad \Sigma_{m=1}^{M} \omega_m = 2$$

(3.3.17a,b,c)

Physically I_{mg} represents the specific intensity in a particular direction with weight distribution of ω_m , μ_m is the cosine angle of the specific intensity with respect to the x-axis and M is the total number of discretized angles.

3.3.4 DISCRETIZATION IN SPACE:

The spatial discretization of the radiative transfer equation (3.3.17a) is obtained by integrating in space over the cell width, as shown below

$$I_{mgi} = \frac{1}{\Delta x_i}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} I_{mg} dx$$

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mu_m \frac{\partial I_{mg}}{\partial x} dx + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (\sigma_g + \tau) I_{mg} dx = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \frac{\chi_g \eta}{2} \Sigma_g \sigma_g I_{0g} dx + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} Q_{mg} dx$$

(3.3.18)

then the transfer equation reduces to

$$\frac{\mu_m}{\Delta x_i}(I_{mgi+\frac{1}{2}} - I_{mgi-\frac{1}{2}}) + (\sigma_{gi} + \tau) I_{mgi} = \frac{\chi_{gi}\eta_i}{2} \Sigma_g \sigma_{gi} I_{0gi} + Q_{mgi}$$

(3.3.19)

3.4 SOURCE ITERATION:

Equation (3.3.19) is simply a set of (M*G) transfer equations corresponding to M angles and G frequency groups. This equation includes three unknowns I_{mgi} and $I_{mgi\pm 1/2}$, where one of $I_{mgi\pm 1/2}$ is obtained from the boundary condition of the cell. Very often one uses a relation known as Diamond Differencing (DD) to equate the number of equations with the number of unknowns. This relation is given as,

$$I_{mgi} = \frac{1}{2}(I_{mgi+\frac{1}{2}} + I_{mgi-\frac{1}{2}})$$

(3.4.01)

This simply states that the intensity at the centre of a cell is equal to the average of the intensity at the cell boundaries. The difficulty in solving the radiative transfer equation arises due to the coupling (summation over the frequency groups) of the specific intensity on the right hand side of (3.3.19). The simplest approach to solve such types of equations is by the use of the source iteration technique. This involves making an initial guess for the source term in equation (3.3.19) and solving for the intensities. The solution is then taken to be the new initial guess in the source term (the summation term on right hand side) of (3.3.19) for the next iteration and equation (3.3.19) is solved again. This is repeated until the new solution and initial guess have converged to some desired accuracy. The final form of the set of equations involved in the source iteration (SI) method is written as

$$\mu_m\left(I_{mgi+\frac{1}{2}}^{l+\frac{1}{2}} - I_{mgi-\frac{1}{2}}^{l+\frac{1}{2}}\right) + \hat{\sigma}_{gi} I_{mgi}^{l+\frac{1}{2}} = \frac{\eta \chi_{gi} \Delta x_i}{2} \Sigma_g \sigma_{gi} I_{0gi}^{l} + \Delta x_i Q_{mgi}$$

$$\textit{where} \qquad \hat{\sigma}_{gi} = (\sigma_{gi} + \tau)\Delta x_i$$

$$I_{mgi}^{l+\frac{1}{2}} = \frac{1}{2}\left[I_{mgi+\frac{1}{2}}^{l+\frac{1}{2}} + I_{mgi-\frac{1}{2}}^{l+\frac{1}{2}}\right]$$

$$I_{ogi}^{l+\frac{1}{2}} = \Sigma_m I_{mgi}^{l+\frac{1}{2}} \omega_m$$

(3.4.02a,b,c)

In the SI method one uses (3.4.02) to obtain the intensity as is shown in the appendix (section A.3). A Fourier stability analysis of the transfer equation shows that when $\tau \to 0$ or/and $\sigma \gg 1$, the spectral radius can be arbitrarily close to unity. In simple terminology, it means that the error reduction per iteration can be extremely slow or that an infinite number of iterations may be required. However, in practice when the radiative transfer equation is fully discretized and one has a finite frequency range rather than from 0 to ∞, the radius of convergence can shown to be always less than one. This is due to the fact that for a finite system all of the Fourier modes are not present. This slow convergence compels one to look for other methods which can reduce the spectral radius. The next section deals with this issue.

3.5 DIFFUSION SYNTHETIC ACCELERATION OF THE TRANSFER EQUATION:

3.5.1 INTRODUCTION TO SYNTHETIC ACCELERATION:

The idea behind synthetic acceleration can be explained as follows[3/7]. Suppose we are given an equation of the form

$$[A-B]f = S$$

where [A-B] is difficult to invert, but A can be inverted relatively easily. We try the following iteration strategy

$$f^{l+1} = A^{-1}Bf^{l} + A^{-1}S$$

where l and l+1 are iterative indices and the eigen value of $[A^{-1}B]$ determines the rate of convergence. The above iteration scheme will converge as long as the spectral radius (the largest eigenvalue) is less than one. It can be shown that one can solve above problem iteratively as

$$[A-B](f - f^{l+1}) = B(f^{l+1} - f^{l})$$

where f is the converged solution. The above equation implies that an exact solution can be obtained immediately by inverting [A-B] if the solution to f^{l+1} is known. The inversion of [A-B] is the obstacle we are trying to remove, so we are no closer to the resolution of the problem. If we assume that L is an approximation to [A-B] and if L is easily invertible, then the problem can be solved in two steps as given below

$$f^{l+1/2} = A^{-1}Bf^{l} + A^{-1}S$$

$$L(f^{l+1} - f^{l+1/2}) = B(f^{l+1/2} - f^{l})$$

This iteration strategy is called synthetic acceleration. In order to elucidate the use of synthetic acceleration, let us consider the radiative transfer equation. The radiative transfer equation and its low order form corresponding to the above set of equations is given by

$$(\mu\frac{\partial}{\partial x}+\sigma)I^{l+\frac{1}{2}}=\frac{\chi\eta}{2}\Sigma_g\Sigma_m\sigma I^{l}+Q$$

$$L(I^{l+1}-I^{l+\frac{1}{2}})=\frac{\eta\chi}{2}\Sigma_g\Sigma_m\omega_m\sigma_g(I^{l+\frac{1}{2}}-I^{l})$$

It is trivial to identify A and B, but L is not so obvious. The main task in synthetic acceleration is to find the operator L, which reduces the spectral radius and yields a stable solution. We will show by using a systematic approach that the required operator is the diffusion operator. In early 1980, Larsen[3/8] developed a four step procedure whereby one could derive beginning with virtually any differencing scheme of the transfer equation a set of linear equations[3/9]. His approach is applicable to any differencing scheme of the transfer equation without regard to geometry or mesh size. However, equations in P_1 acceleration are algebraically difficult to solve. Hence efficiency remains a serious issue for higher dimensional problems. The approach we have taken to obtain the low order equation is the one proposed by Alcouffe[3/4] and later improved by Larsen[3/10].

The main objective of the DSA scheme is to synthesize a low order form for the radiative transfer equation, which can be used to accelerate the rate of convergence of the transfer equation. In addition, we require the new method to exhibit the following characteristics:

1. It offers a solution to the exact transfer equation, rather than just an approximation, such as the diffusion equation.
2. It is computationally efficient.
3. The method should be extendable to higher dimensions and to curvilinear geometry.
4. One should be able to solve more complicated problems, such as problems involving scattering.

From our experience we are able to prove the first two characteristics of the method, proof for the later two is beyond the scope of the present research; however, it has been reported[3/11,12] that these can also be satisfied.

3.5.2 DIFFUSION SYNTHETIC ACCELERATION:

A Fourier stability analysis of the transfer equation (presented in section A.1 of the appendix) shows that the most slowly converging modes (correspond to $\lambda \approx 0$) are given as

$$D(\lambda,\nu,\mu) = \frac{\chi\eta}{2}\frac{1}{(\sigma+\tau)}\left(1-\frac{i\lambda\mu}{\sigma+\tau} + O(\lambda^2)\right) \tag{3.5.01}$$

This equation indicates that the slowly varying modes are linear function of μ. We use this fact to derive an equation which replaces (3.4.02c) and the end result is a faster convergence of (3.4.02a). Physically, the slowest converging modes correspond to near equilibrium situations. Therefore, in a sense we can guess that the low order form will resemble a diffusion equation.

We will use equation (3.3.12) to illustrate the **DSA** formulation. Its discretized version is given in Appendix A.3, since its subtle differences are not trivial. The steps that are commonly used to obtain the low order form of the transfer equation are as follows:

i. The zeroth and first moments of (3.3.12) in angular space are taken, that is equation (3.3.12) is multiplied by the weight functions 1 and ω_m followed by an integration of the resultant equations over angular space.

ii. I_1 is eliminated from the above two equations.

iii. The accelerated form of the resultant equation in (ii) is obtained by resetting the index of iteration for the zeroth and first moments of the intensity (that is $l+1/2 \rightarrow l+1$ for I_0 and I_1).

The final result is

$$-\frac{\partial}{\partial x}\frac{1}{3(\sigma+\tau)}\frac{\partial}{\partial x}I_0^{l+1} + (\sigma+\tau)I_0^{l+1} = \eta\chi\int_0^\infty \sigma I_0^l d\nu + Q_0 - \frac{\partial}{\partial x}\frac{1}{3(\sigma+\tau)}Q_1 + \frac{\partial}{\partial x}\frac{2}{3(\sigma+\tau)}\frac{\partial I_2^{l+1/2}}{\partial x} \tag{3.5.02}$$

This is simply the diffusion equation corresponding to the transfer equation when I is linear in angle, since when I is linear in angle $I_2=0$. It is shown in the appendix (section A.2) that the spectral radius for (3.3.12) with (3.5.02) goes to zero even for those modes for which (3.3.12) with (3.4.02) has spectral radius close to unity. This **apparently** seems to have **solved our problem,** but we will see this is not so. In equation (3.5.02) the right hand side contains a coupling term; therefore, we will have to solve this equation using source iteration, as was done for the radiative transfer equation.

3.6 GREY DIFFUSION EQUATION:

A Fourier stability analysis of (3.5.02) with source iteration shows that the spectral radius can be very close to unity. Therefore, to significantly improve the convergence rate of the transfer equation (3.3.12) we must improve the spectral radius of the diffusion equation (3.5.02) by another low order equation or by some other approximation. To accelerate the rate of convergence of the diffusion equation we choose to solve a low order form of diffusion equation, namely the grey diffusion equation, to obtain the new source term for the diffusion equation.

First of all we rewrite (3.5.02) as

$$-\frac{1}{3(\sigma+\tau)}\frac{\partial^2}{\partial x^2}I_0^{k+\frac{1}{2}} + (\sigma+\tau)I_0^{k+\frac{1}{2}} = \eta\chi\int_0^\infty \sigma I_0^k d\nu + S \tag{3.6.01}$$

where S includes all the terms that do not change during the iteration of the diffusion equation. In the source iteration method we take

$$I^{k+1} = I^{k+1/2} \tag{3.6.02}$$

A Fourier stability analysis of (3.6;01,02) shows that the diffusion equation can have a radius of convergence that is close to unity. If for simplicity we assume that the opacity is spatially independent in the Fourier stability analysis, then the radius of convergence and the corresponding eigen function are given by

$$\omega = \eta\int_0^\infty \frac{3\sigma(\sigma+\tau)\chi}{\lambda^2+3(\sigma+\tau)^2}\,d\nu \qquad \Psi = \eta\frac{3\sigma(\sigma+\tau)\chi}{\lambda^2+3(\sigma+\tau)^2}\,e^{i\lambda x} \tag{3.6.03a,b}$$

This also shows that the slowest converging modes correspond to modes that are close to equilibrium modes. In order to accelerate the diffusion equation, we solve it by using the synthetic approach. That is, we use a grey equation as the low order equation.

The central idea behind the grey equation is to collapse the continuous energy dependence in (3.6.01) to one energy group and design this equation such that the near equilibrium situation is treated more accurately than in the source iteration method. This new equation replaces (3.6.02). The steps involved in the derivation of the grey equation are as follows:

i. The first term in equation (3.6.01) is approximated by

$$
\begin{gathered}
-\frac{1}{3(\sigma+\tau)}\frac{\partial^2}{\partial x^2}I_o^{k+\frac{1}{2}} \rightarrow -\frac{1}{3(\sigma+\tau)}\frac{\partial^2}{\partial x^2}\varphi^{k+\frac{1}{2}}\hat{I}_o^{k+1} + \frac{1}{3(\sigma+\tau)}\frac{\partial^2}{\partial x^2}\varphi^{k+\frac{1}{2}}\hat{I}_o^{k+\frac{1}{2}} \\
-\frac{1}{3(\sigma+\tau)}\frac{\partial^2}{\partial x^2}I_o^{k+\frac{1}{2}} \\
\textit{and} \\
I_o^k \;\; or \;\; I_o^{k+\frac{1}{2}} \rightarrow \theta^{k+\frac{1}{2}}\hat{I}_o^{k+1} - \theta^{k+\frac{1}{2}}\hat{I}_o^{k+\frac{1}{2}} + I_o^{k+\frac{1}{2}} \\
\hat{I}_o^{k+\frac{1}{2}} = \int_o^\infty \sigma I^{k+\frac{1}{2}} d\nu
\end{gathered}
\tag{3.6.04}
$$

An important characteristic of this transformation is that it ensures that after convergence, the left side of (3.6.04) is the same as the right side, implying that the converged solution for the source and the accelerated equation are the same. Variables θ and φ will be explained later.

ii. Integrate (3.6.01) over the frequency spectrum.

The final result is

$$
\begin{gathered}
-\int_0^\infty \left[\frac{1}{3(\sigma+\tau)}\frac{\partial^2}{\partial x^2}\varphi^{k+\frac{1}{2}}\right] d\nu \; I_0^{k+1} + (\sigma+\tau) I_0^{k+1} + \tau\left(\int_0^\infty \theta^{k+\frac{1}{2}} d\nu\right) It_0^{k+1} \\
= \eta I_0^{k+1} + \int_0^\infty S d\nu + \int_0^\infty \left[\frac{1}{3(\sigma+\tau)}\frac{\partial^2}{\partial x^2}\left(I_0^{k+\frac{1}{2}} - \varphi^{k+\frac{1}{2}} I_0^{k+\frac{1}{2}}\right)\right] d\nu \\
\tau \int_0^\infty \left[I_0^{k+\frac{1}{2}} - \theta^{k+\frac{1}{2}} I_0^{k+\frac{1}{2}}\right] d\nu - \eta I_0^{k+\frac{1}{2}} + \eta\int_0^\infty \sigma I_0^{k+\frac{1}{2}} d\nu
\end{gathered}
\tag{3.6.05}
$$

$\theta^{k+1/2}$ and $\varphi^{k+1/2}$ are defined[3/13] such that above equation produces an exact solution without iteration when I_o corresponds to the slowest converging modes. From (3.6.03) we see that the eigenvalue and the eigenfunction corresponding to the slowest modes are

$$\omega = \eta\int_0^\infty \frac{\sigma\chi}{(\sigma+\tau)}\,d\nu \qquad \Psi = \eta\frac{\chi}{(\sigma+\tau)}\,e^{i\lambda x}$$

(3.6.06)

We take the spectral functions (in normalized form) to be

$$\theta^{k+\frac{1}{2}} = \varphi^{k+\frac{1}{2}} = \frac{\dfrac{\chi}{(\sigma+\tau)}}{\left(\displaystyle\int_0^\infty \frac{\sigma\chi}{(\sigma+\tau)}\,d\nu\right)}$$

(3.6.07)

This choice of spectral function produces the linear grey equation. It is called linear because it leads to an additive correction. The other expressions for θ and φ are given by

$$\theta^{k+\frac{1}{2}} = \frac{I_o^{k+\frac{1}{2}}}{\hat{I}_o^{k+\frac{1}{2}}} \quad ; \quad \varphi^{k+\frac{1}{2}} = \frac{\left(\dfrac{\partial I_o^{k+\frac{1}{2}}}{\partial x}\right)}{\left(\dfrac{\partial \hat{I}_o^{k+\frac{1}{2}}}{\partial x}\right)}$$

(3.6.08)

Here the correction term is multiplicative, hence it is nonlinear. The main difference between these methods lies in the approximation to the grey spectrum. In the linear case we take it to be the spectrum corresponding to equilibrium, while in the nonlinear case it is the latest spectrum averaged coefficient. This can be rephrased as follows: the linear case grey method is defined by collapsing the cell average flux as the spectral function, while in the nonlinear grey case it is the cell edge flux that is collapsed as the spectral function.

The solution to the radiative transfer equation involves solving the following set of equations (3.3.12), (3.5.02) and (3.6.05). Equation (3.3.12) is solved first, then equation (3.5.02) and (3.6.05) are solved. This form of iteration can be viewed as inner and outer iteration In the inner iteration equations (3.6.05) and (3.5.02) are repeatedly solved until (3.5.02) is converged. Then equations (3.5.02) and (3.3.12) are solved until (3.3.12) is converged to the desired accuracy. When the above equations are discretized, the consistency in the discretization of the equations and their boundary conditions is crucial for a physically acceptable and stable solution. This is addressed in Appendix (A.4).

CHAPTER FOUR
MODEL PROBLEM

The numerical aspects of the technique used to solve the radiative transfer equation are presented in Chapter Three. In this chapter, results are presented for the model problem that is used to test the present code. This chapter also includes a discussion of the issues which often arise with any code - issues such as fixups, stability, efficiency and the validity of the code. A comparison of the source iteration (SI) method and the diffusion synthetic acceleration (DSA) method is presented in this chapter. It will be shown later that when the DSA method is stable, it is always faster than the SI method and yields the same results. The extent to which DSA improves the spectral radius depends on different parameters, such as the time step, convergence criterion and the number of inner/outer iterations.

4.1 FIXUPS AND THE STABILITY OF THE CODE:

It often becomes necessary to incorporate fixups into codes that involve complicated equations and the DSA method is no exception. The main objective of these fixups is to suppress the appearance of nonphysical results. For example, the intensity is a physical quantity which can not be allowed to have a negative value. Therefore, it is important that some fixup should be implemented when negative intensity leads to instability.

The fixups are based on the trial and error approach and hence do not have any real theoretical justification, other than that they prevent nonphysical results from appearing in the simulations. Their presence in the code does not imply that they are used in every simulation. The important rule we have learned for a fixup strategy is that a fixup should be invoked only when the solution has become unstable not because it is becoming unstable. Every effort should be made to ensure that the transfer equation, the diffusion equation and the grey equation are consistent at all times. A negative value may be physically unacceptable, but mathematically a negative value is as good as a positive value. Hence a negative value should not be modified unless a physically unacceptable solution is obtained or the code has become unstable. In this section we discuss some of the fixups implemented in the code.

The first fixup that is implemented is due to the spatial differencing scheme employed in differencing the radiative transfer equation in space. The Diamond differencing scheme is used to discretize the transfer equation in space, because of the ease of its implementation. Unfortunately, it can lead to a negative specific intensity, which if not corrected can destabilize the code. When a cell edge intensity becomes negative, it is taken to be zero or is set to be the average intensity of the cell. The choice depends on a parameter which is a function of the cell width and the time step (section A.3). Another important fixup which also has been included in our theoretical analysis is the incorporation of the above fixup into the diffusion equation. The Diamond Difference relationship in the diffusion equation is redefined as

$$I_{ogi}^{l+1}=\frac{1}{2}[a_{ogi}I_{ogi-\frac{1}{2}}^{l+1}+b_{ogi}I_{ogi+\frac{1}{2}}^{l+1}]$$

When a fixup is required in the transfer equation, the coefficients a_{ogi} and b_{ogi} are given as

$$a_{ogi}=\frac{I_{ogi}^{l+\frac{1}{2}}}{I_{ogi-\frac{1}{2}}^{l+\frac{1}{2}}} \quad , \quad b_{ogi}=\frac{I_{ogi}^{l+\frac{1}{2}}}{I_{ogi+\frac{1}{2}}^{l+\frac{1}{2}}}$$

Without the fixup, these coefficients are

$$a_{ogi}=b_{ogi}=1.0$$

(4.1.01a,b,c)

The choice of these coefficients depends upon the stability of the solution rather than on some theoretical analysis. The main objective is to define these coefficients so that the diffusion matrix yields a stable solution. From equation (A.8.02), we see that the main diagonal element is not always the dominant element. This can lead to a negative solution of the diffusion equation. If a negative solution of the diffusion equation occurs too frequently, the solution for the transfer equation may become unstable or become physically unacceptable. This problem can be eliminated through the use of small time steps so that the gradient opacity is small. One can also modify the algorithm[4/1] to ensure that the main diagonal element is always dominant, but this modification leads so slower acceleration of the diffusion equation.

In simulations it was observed that when the negative flux fixup is implemented, $I_{ogi\pm1/2}$ can become zero causing a floating point error in (4.1.01b) when a_{ogi} and b_{ogi} are computed. In order to prevent a floating point error, we take $a_{ogi}=b_{ogi}=1.0$ whenever $I_{ogi\pm1/2}$ is zero.

Many switches are added into the code to ensure its stability and to prevent the code from consuming an excessive amount of time on iterations which are extremely slow to converge or start to diverge. If the diffusion equation yields a spectral radius greater than or close to 1.0, then the code automatically resorts to SI for that iteration when switch AUTACC=1. If it occurs too often, then the calculation is terminated. A similar type of switch is included to prevent the grey equation from yielding an unrealistic solution. If the solution of the diffusion or the grey equation becomes unstable too often then the calculation is again terminated. In the event that the integration is not achieved within the required precision or the resultant matrix is ill behaved, then the program is again terminated. The code has been written such that the error messages are written into a file named "analysis" for instabilities which do not cause termination. Error messages appear on the screen for those instabilities which require user input to continue or that require termination of the code. These switches improve the efficiency of the code but do not ensure DSA stability when large time step is used, as was expected in the beginning of the study.

4.2 THEORETICAL ASPECTS OF THE MODEL PROBLEM:

Often a code is tested against a bench mark problem, before it is used to simulate real problems. The problem we use to test the code is the well known Marshak Wave bench mark[4/2]. This is the same model problem used by Alcouffe et al[4/3] and Velarde et al[4/4] to test their codes. Here a strong source (acting as a laser beam) is placed at one end of the plasma. The temperature (which is changing due to the propagation of the radiation) is computed as a function of space and time. In this section the equations involved in the formulation of the model problem are presented.

We start our formulation from equations (3.3.01;02;3a) derived in Chapter Three. These equation are repeated here for convenience

$$\frac{1}{c}\frac{\partial I}{\partial t}(x,\mu,\nu,t) + \mu\frac{\partial I}{\partial x}(x,\mu,\nu,t) + \sigma(\nu,T)I(x,\mu,\nu,t) = S(\nu,T) = \sigma(\nu,T)B(\nu,T)$$

$$\frac{\partial u_e(T)}{\partial t} = \beta(T)\left[\int_0^\infty \sigma(\acute{\nu},T)I_0(x,\acute{\nu},t)d\acute{\nu} - c\sigma_p(T)u_e(T) + S_{ex}\right]$$

$$u_e(T) = \frac{c}{4\pi}\int_0^\infty B(T,\acute{\nu})\, d\acute{\nu}$$

(4.2.01;02;03)

For the model problem, we assume that a local thermodynamic equilibrium (LTE) exists in the plasma. In LTE, it is assumed that properties of the matter are dominated by atomic collisions which establish thermodynamic equilibrium locally at a position x and at time t, and that the radiation field does not affect this equilibrium even when it deviates substantially from the Planckian distribution. That is, at a given instant of time and point in space it is sufficient to specify two thermodynamic quantities, such as the temperature and density in order to compute the source term S($=\sigma$B) and the absorption coefficient σ. With the LTE assumption, B becomes the Planckian function and the transfer equation reduces to

$$\frac{1}{c}\frac{\partial I}{\partial t} + \mu\frac{\partial I}{\partial x} = \sigma(B - I)$$
$$\frac{\partial u_e}{\partial t} = \beta\left[\int_0^\infty \int_{-1}^{1} \sigma I \, d\mu d\nu - c\sigma_p u_e\right]$$
$$u_e = \frac{c}{4\pi}\int_0^\infty B \, d\nu = aT^4$$

(4.2.04;05;06)

where σ and B are defined as

$$\sigma = a_1\frac{[1-e^{-a_0\nu}]}{\nu^3} \qquad B = \frac{2h\nu^3}{c^2}[e^{a_0\nu} - 1]^{-1}$$
$$\textit{where} \quad a_0=\frac{h}{KT} \quad \textit{and} \quad a_1=3.8171E55 \; m^{-1}$$

(4.2.07)

where T is the material temperature, a (=7.560667E-16 $Jm^{-3}K^{-4}$) is the Stefan Boltzmann constant, h (=6.626E-34 Js) is the Planckian constant, k (1.3807E-23 JK^{-1}) is the Boltzmann constant. When equation (4.2.04) is discretized in frequency (Chapter Three) it reduces to

$$\mu\frac{\partial I_g}{\partial x} + (\sigma_g+\tau)I_g = \frac{\chi_g\eta}{2}\Sigma_g \, \sigma_g I_{0g} + Q_g$$

(4.2.08)

For the model problem, the initial intensity in the summation term on the right hand side of the above equation is taken to be Planckian and the absorption coefficient is computed as

$$\sigma_g = \frac{\int_{\nu_l}^{\nu_h}\sigma\omega \, d\nu}{\int_{\nu_l}^{\nu_h}\omega \, d\nu}$$

(4.2.09)

where $\omega(\upsilon)$ is the weight function corresponding to the intensity at that moment. Since this is unknown an approximation is usually made. The choice of this function is somewhat arbitrary. The weight functions which we have included in the code are known as Planckian and Rossland Mean functions. These functions are given by[4/5]

$$\omega(\nu) = B(\nu) \qquad \textit{Planckian}$$
$$\omega(\nu) = \frac{1}{\sigma(\nu)} \frac{\partial B(\nu)}{\partial T} \qquad \textit{Rossland}$$

(4.2.10a,b)

Their justification is based on the following argument[4/5]. The Planckian function B is a good approximation at thermodynamic equilibrium since the specific intensity is given by a Planckian function, and away from equilibrium the choice of the weight function is of less importance. The Rossland function is a good approximation since the specific intensity can be approximated by the equilibrium diffusion approximation. The choice of these functions becomes less significant if a large number of frequency groups are used.

The Romberg integration approach is used to integrate these functions. Lower-Upper decomposition is implemented to invert the matrices in both the diffusion and the grey equations.

4.3 SIMULATIONS OF THE MODEL PROBLEM:

The main objectives of this section are to test the code against a bench mark problem and make comparison between the two approaches used to solve the transfer equation, namely the Source Iteration (SI) and Diffusion Synthetic Acceleration (DSA) methods. The SI method always yields a stable solution irrespective of the time step involved. The DSA becomes unstable for large time steps, but when it works, it is more efficient than SI and yields the same results as the SI methods.

The model problem we used to test the code involves a slab at uniform temperature with a source at one end. The source is placed at the left boundary and it is assumed to be Planckian. This choice for a source resembles a laser beam incident on a foil. It is assumed that the specific heat of the plasma is constant during the simulation, that is it is independent of the temperature. The convergence criterion is set to be

$$[\frac{I^{n+1}}{I^{n}} - 1] \leq 10^{-2} \qquad [\frac{T^{n+1}}{T^{n}} - 1] \leq 10^{-3}$$

This simply states that the specific intensity of the transfer equation is considered to have been converged when the error between two consecutive solutions is less than 1%. A similar convergence criterion is used for the temperature but a convergence criteria of 0.1% is used instead of 1.0%. If the time step is large (such as in nano seconds) then tighter convergence is essential. Once the intensity has converged we use the so called "parametric iteration" to update the temperature. In parametric iteration we use the new temperature (obtained after solving the energy balance equation) to compute the variables (such as σ and χ) that should have been computed at the advanced time step. This is repeated until the temperature is converged to the desired accuracy. The data set used for the standard model problem is

number of cells (uniform)	20
number of frequency groups	30
number of discretized angles	08
specific heat per unit volume	7.00E+04 $JK^{-1}m^{-3}$
source temperature	1000 ev
initial slab temperature	1 ev
slab width	0.2 m
frequency range	1.00E-01-1.00E+04 ev

Additional parameters will be mentioned when appropriate.

Many runs were made to explore the performance of the DSA method. In the remainder of this section, we will discuss important aspects related to the efficiency of the code. We have observed that when very small time steps are used (e.g fraction of a picosecond) the DSA method is not very efficient compared to the SI method. When time steps are small, it is more efficient to use the Source Iteration method, because the time taken to solve the diffusion and grey equation out weighs the efficiency of the DSA method over the SI method. The efficiency of the DSA method improves as we increase the time step. In our simulations, the maximum time step allowed for a stable DSA solution is about 100.0 picoseconds for the model problem. The efficiency of DSA relative to SI depends on the convergence criteria. A tighter convergence criteria with the DSA method results in a small increase in computer time but it significantly increases with the SI method. It is observed that if the number of inner iterations (number of diffusion equations per transfer equation) is not bounded, then not only the efficiency drops but also this can destabilize the code. It is observed that if inner iterations are bounded by 10 then DSA is more efficient and stable compared to when the number of inner iterations is allowed to be unlimited. The solution of the diffusion equation without the grey equation very often increases the computer time for a given simulation (even though it improves the spectral radius) due to slow convergence rate of the SI. However, the time is substantially reduced when the diffusion equation is solved using the grey equation.

Therefore, in order to optimize the efficiency of the code, it is important that the diffusion equation is solved with the grey equation and that a bound is placed on the inner iterations. The remainder of this section includes a graphical representation and a discussion of the following topics:

i. A comparison between DSA and SI .

ii. Planckian vs Rossland weight functions.

iii. Use of exponential temperature as the initial temperature (instead of being uniform).

iv. Effect of varying the number of cells, number of frequency groups and number of discretized angles.

v. Marshak Wave Bench Mark

The source code is in a file name **"sate.f"** (a listing can be found in the appendix) and the input file is called **"sate.data"** and is given below.

satedata

ENTER 1 FOR RUNS WITH DIFFERENT TIME STEPS OTHERWISE 0	00
ENTER SCR NUM DBXL DBXU TCONT	00 02 0000 9999 01
ENTER T OR F FOR SIMULATIONS OF THE MODEL PROBLEM (SIMRUN)	F
ENTER "0" IF SOURCE IS TO BE DEPENDENT ON TIME (TEMPTD)	01
ENTER "0" IF INITIAL TEMPERATURE DISTRIBUTION IS EXP (TEMEXP)	01
ENTER "0" IF ALWAYS USE NFF (NFMUST)	01
ENTER "0 TO 20" FOR OUTPUT TO BE PRINTED (PRES)	00
ENTER BOUND FOR TRANSPORT AND DIFFUSION EQU (TB DB)	9999 999
ENTER "0" IF (INTB=0) "1" IF (INTB=PINB) "2" IF USE MTSP (NFFC)	02
ENTER IERR AND TERR AND SIER	1.00E-02 2.50E-01 1.00E-02
ENTER # OF OIT AFTER WHICH NEGATIVE FLUX IS ALWAYS USED (INSB)	999999
ENTER ABSORPTION COEFFICIENT 1E? (? = 6 TO 11) (AC)	1.00E+11
ENTER "0 TO 2" FOR CONVERGENCE CRITERION BEST IS "1" (CONVC)	01
ENTER "0" IF INITIAL INTENSITY TO BE PLANCKIAN EACH TIME (TRIN)	01
ENTER "0" IF UNIFORM SPATIAL DISTRIBUTION IS TO BE USED (USC)	00
ENTER TIME INTERVAL AT WHICH TEMPERATURE IS TO SAVED (PRTME)	1.00E-09
ENTER "0 OR 1" DIFFUSION BC (DIFBC)	00
ENTER "0 TO 2" TO OVERCOME NEGATIVE DIFFUSION INTENSITY (NDIT)	03
ENTER MAXIMUM TIME FOR SIMULATION (MAXT)	1.01E-07
ENTER M < 9 G < 61 N < 151	008 030 020
ENTER L LFL UFL TELB ITRB ITSL	
2.00E-01 1.00E+01 1.00E+03 2.00E-02 1.00E+01 1.00E-00	
ENTER "0" IF PLANCKIAN IS TO BE USED AS A WEIGHT (WFN)	00
ENTER SPECIFIC HEAT (SPH)	7.00E+04
ENTER DTIME	1.00E-12
ENTER "0" IF NEGATIVE FLUX FIX UP IS TO BE IMPLEMENTED (NF)	00
ENTER NUMBER OF TEMPERATURE ITERATIONS ALLOWED (TITA)	9999
ENTER NUMBER OF OUTER ITERATIONS ALLOWED (OITA)	999992
ENTER "0" FOR LOWER LIMITS OTHERWISE "1" (SETLIM)	00
ENTER "0" FOR SOURCE = 0 AT RIGHT "1" FOR BOTH ENDS (SETSOU)	02
ENTER "1" IF NEG. FLUX TAKEN IT'S ABSOLUTE VALUE (ABVL)	00
ENTER "1" IF CODE TO BE EXECUTED WITH MEDUSA (MEDUSA)	01
ENTER "1" IF ACCELERATION IS AUTOMATIC (AUTACL)	00 5.00E-01
ENTER "0" IF TRANSPORT EQUATION IS TO BE SOLVED BY SI (TRNACC)	00
ENTER "1" IF DIFFUSION EQUATION IS TO BE ACCELERATED (DIFACC)	02
ENTER "0" FOR LINEAR GREY MODEL OR "1" FOR NONLINEAR (LIN)	00

Figure 4.1 a, b

When finer cells are used the computational time increases significantly, while the solution for the temperature remains very much the same. An increase in the temperature at the left end implies that the closer we move to the left (by making cells more finer), a higher temperature is obtained at the left end of the slab until it reaches the maximum value at the boundary (the value of the source). The difference in temperature can be reduced if a tighter temperature convergence criterion is used in both cases. A decrease in the number of cells has a more noticeable change in the temperature profile than when the number of cells is increased. The difference in the temperature with 50 cells compared to 20 cells (Fig. 4.1a) is relatively small while the computational time is significantly increased. The use of 10 cells (Fig. 4.1b) on the other hand shows a significant difference compared to 20 cells. Therefore, use of 20 cells is sufficient to yield an accurate description of the temperature in the slab for this case.

Figure 4.2 a, b

In the standard model problem 30 frequency groups are used. From Fig. 4.2a, it can be seen that increasing the number of frequency groups has no effect on the temperature profile that is observable. However, if the number of groups (Fig. 4.2b) is decreased then a significant change in the temperature profile is observed. Therefore, 30 frequency groups are sufficient in our simulations of the model problem.

Figure 4.3 a, b

Here it is shown that when the number of discretized angles is reduced (Fig. 4.3) then the entire temperature profile is shifted upward. We did not observe any significant significant change in the temperature profile (Fig. 4.3b) when number of discretized angles is increased to 16. Therefore, we think 8 discretized angles is sufficient to take into consideration the angular dependence of the intensity.

Figure 4.4

This graph represent the Marshak Wave Bench Mark with the present code. The profiles shown here are not exactly as reported by Alcouffe or Honrubia, this is because different weight function profiles are used for the absorption coefficient. This implies that the rate of change of temperature will be different; however we expect that the equilibrium profiles should be very much the same. This is true for all of the results reported by Alcouffe and Honrubia and reported here.

Figure 4.5

Here a comparison of SI and DSA is presented. This graph shows that the accelerated and non accelerated solution are the same. We have observed this consistency for all cases. In this graph a comparison between the linear and nonlinear grey models is also presented. We observed that the linear grey model performs better than the nonlinear model. The comparison of SI and DSA is shown below

	# of transfer equ.	# of diffusion equ	# parametric iteration
SI	1432	0	63
DSA(linear)	148	578	26
DSA(nonlinear)	148	1893	27

Our present code is not as stable as was expected. There appears to be a discrepancy in our derivation which we have not been able to resolve. The boundary conditions used for the grey equation in the code are the best in a sense that they yield the same result while allowing the use of a larger time step. The cause of this instability will require further investigation in the future. From our experience we think the inconsistency exists between the grey and the diffusion equation boundary conditions.

Figure 4.6 a, b

This graph reflects the effect of the weight functions used for evaluating the absorption coefficient. In Fig. 4.6a the number of groups is 30 and in Fig. 4.6b the number of groups is 60. This shows that different choice of weight functions can lead to different temperature profiles. This graph also shows that as the number of groups is increased the choice of the weight function becomes less crucial. This conclusion is difficult to corroborate or disapprove because neither Alcouffe nor Honrubia has discussed the choice of the weight functions in their publications.

Figure 4.1 a

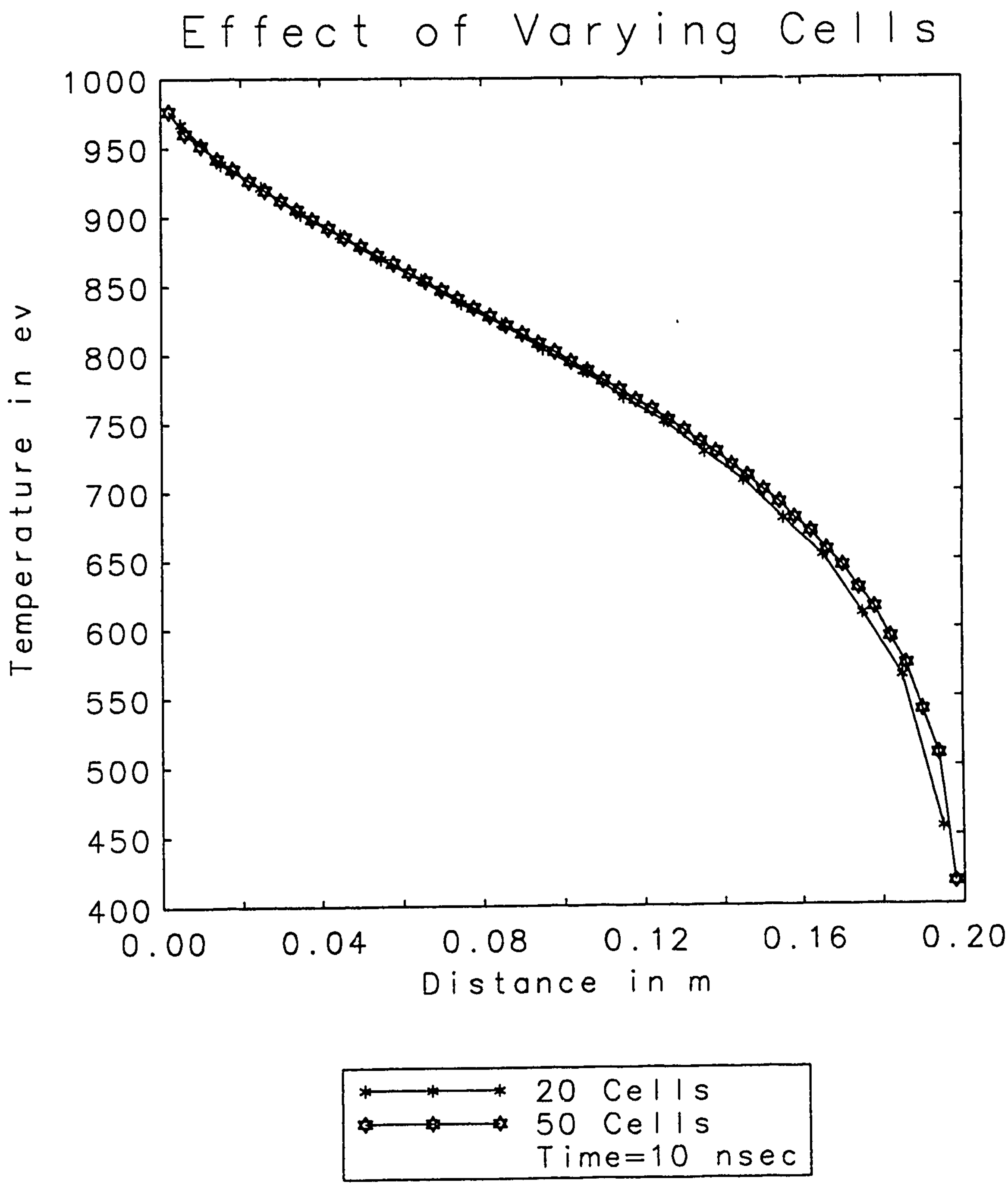

Effect of Varying Cells
1000
950
900
850
800
750
700
650
600
550
500
450
400
Temperature in ev
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
20 Cells
50 Cells
Time=10 nsec

Figure 4.1 b

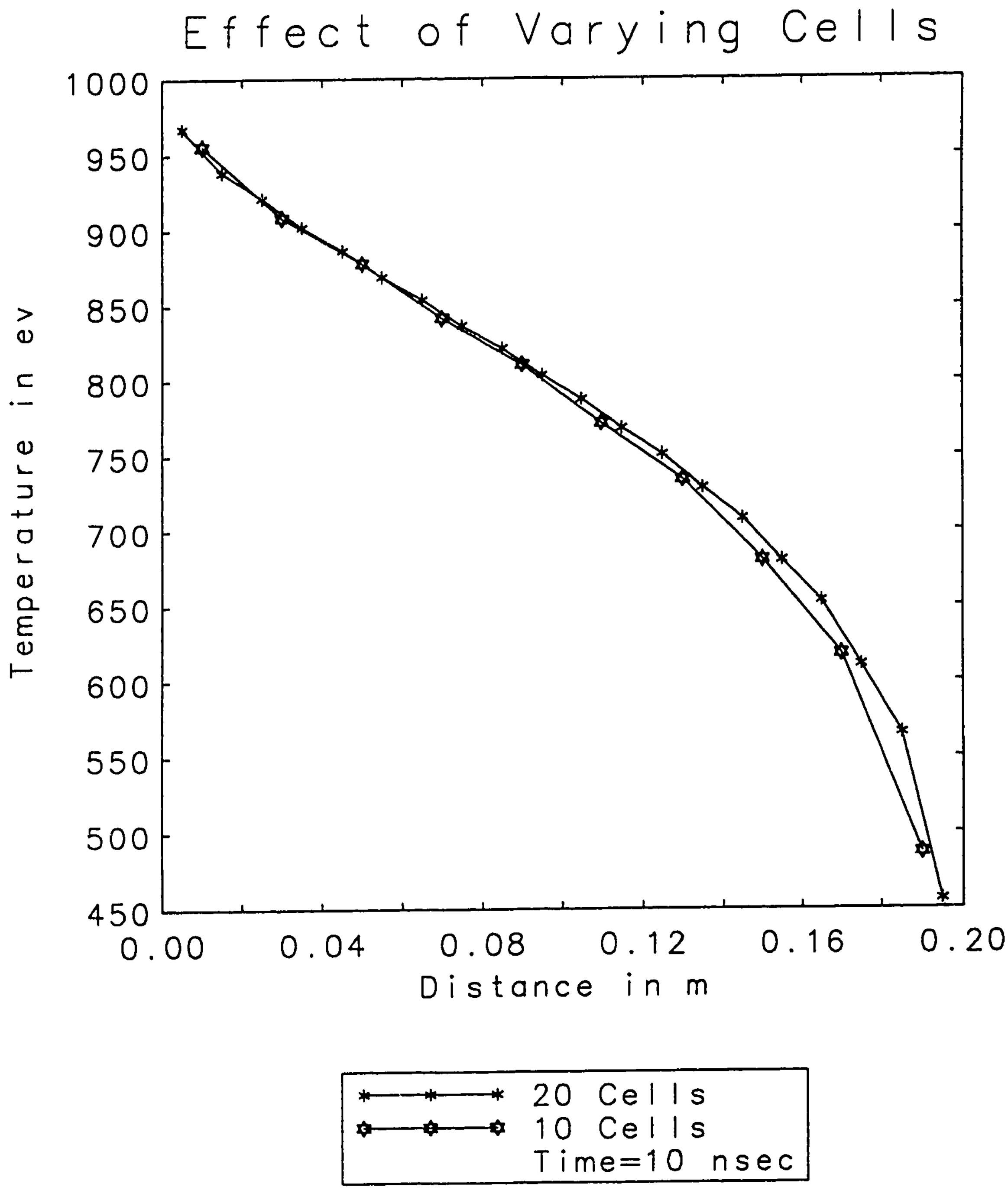

Effect of Varying Cells
Temperature in ev
1000
950
900
850
800
750
700
650
600
550
500
450
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
20 Cells
10 Cells
Time=10 nsec

Figure 4.2 a

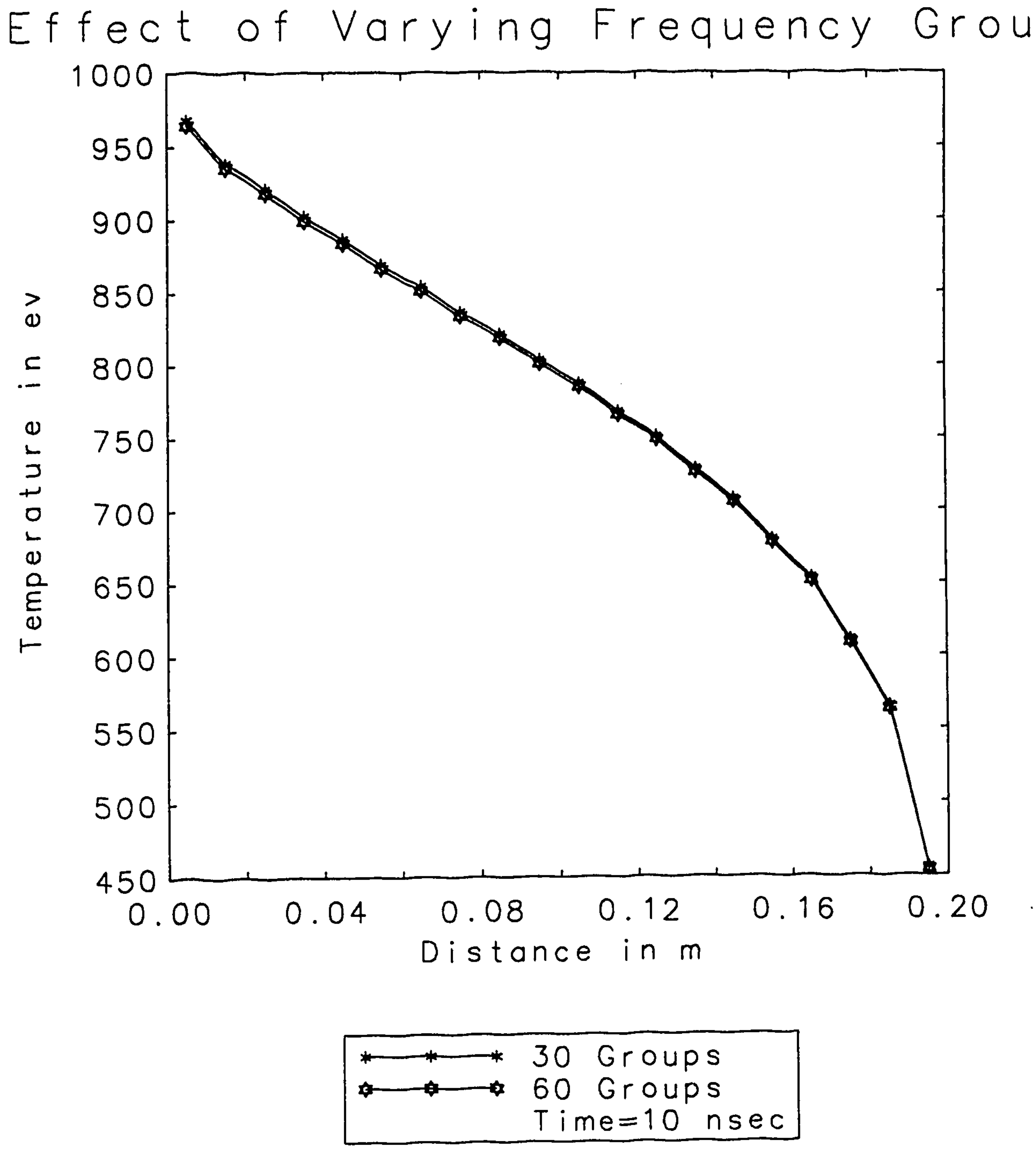

Effect of Varying Frequency Grou
1000
950
900
850
800
750
700
650
600
550
500
450
Temperature in ev
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
30 Groups
60 Groups
Time=10 nsec

Figure 4.2 b

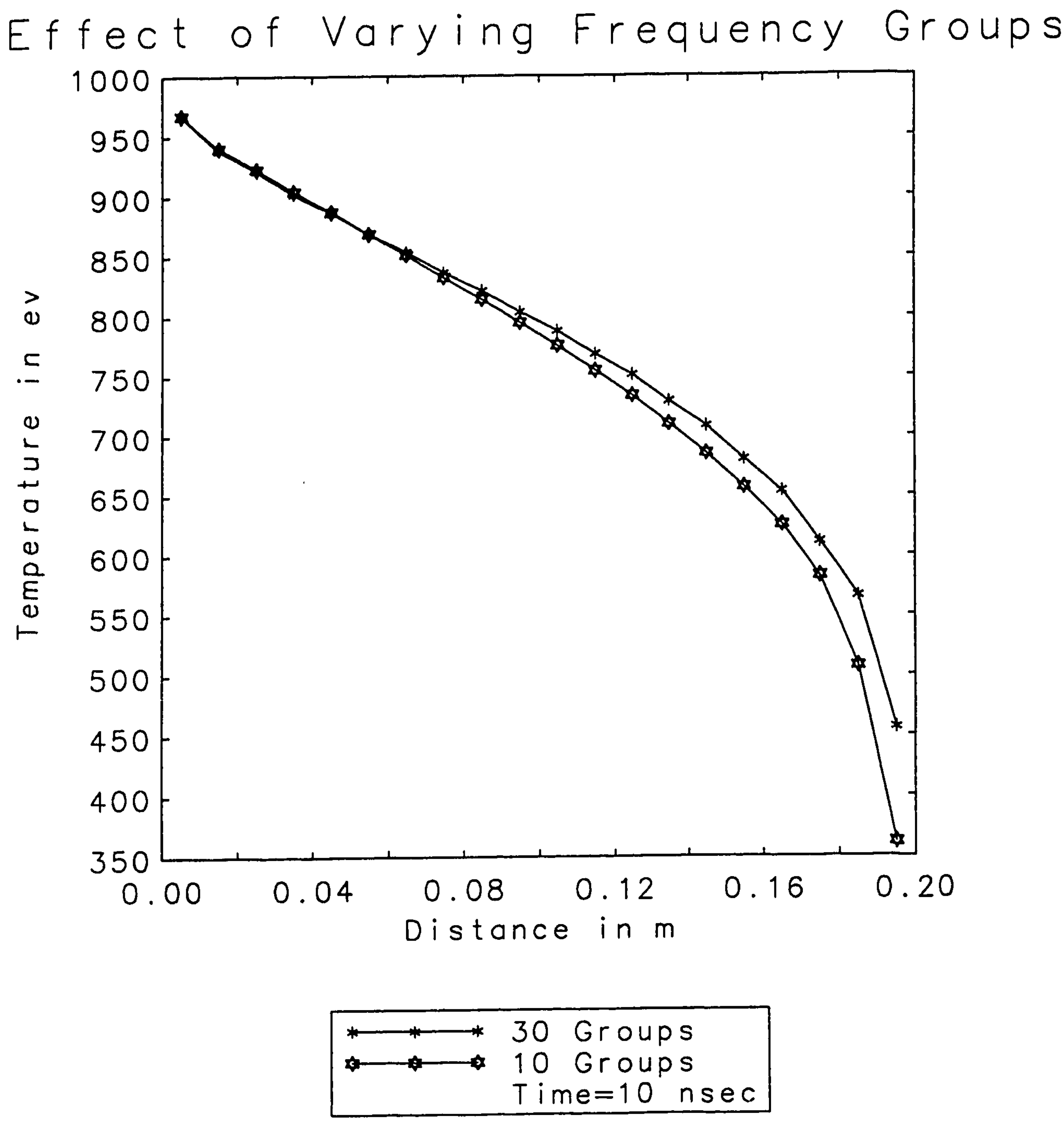

Effect of Varying Frequency Groups
1000
950
900
850
800
750
700
650
600
550
500
450
400
350
Temperature in ev
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
30 Groups
10 Groups
Time=10 nsec

Figure 4.3 a

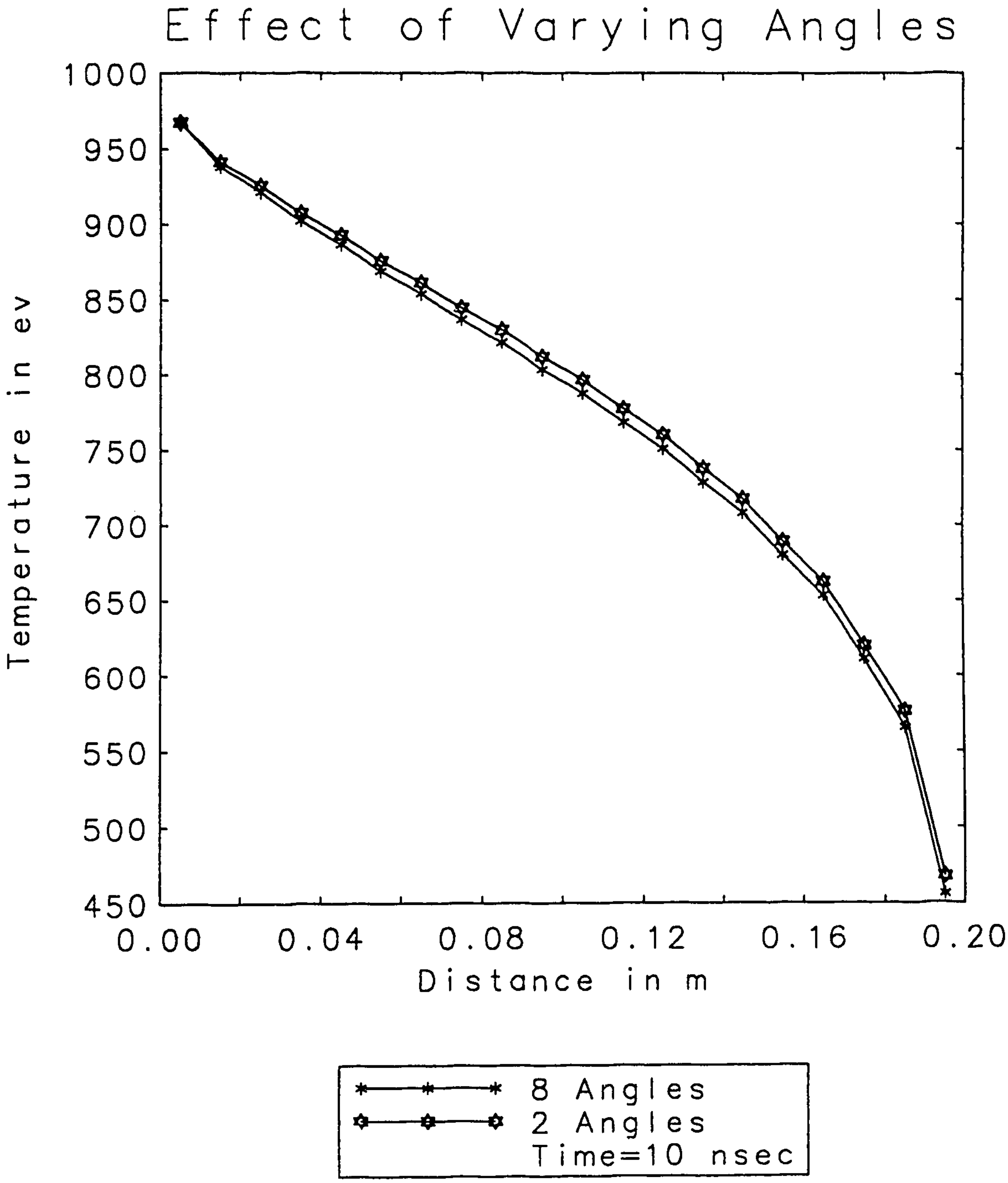

Effect of Varying Angles
1000
950
900
850
800
750
700
650
600
550
500
450
Temperature in ev
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
8 Angles
2 Angles
Time=10 nsec

Figure 4.3 b

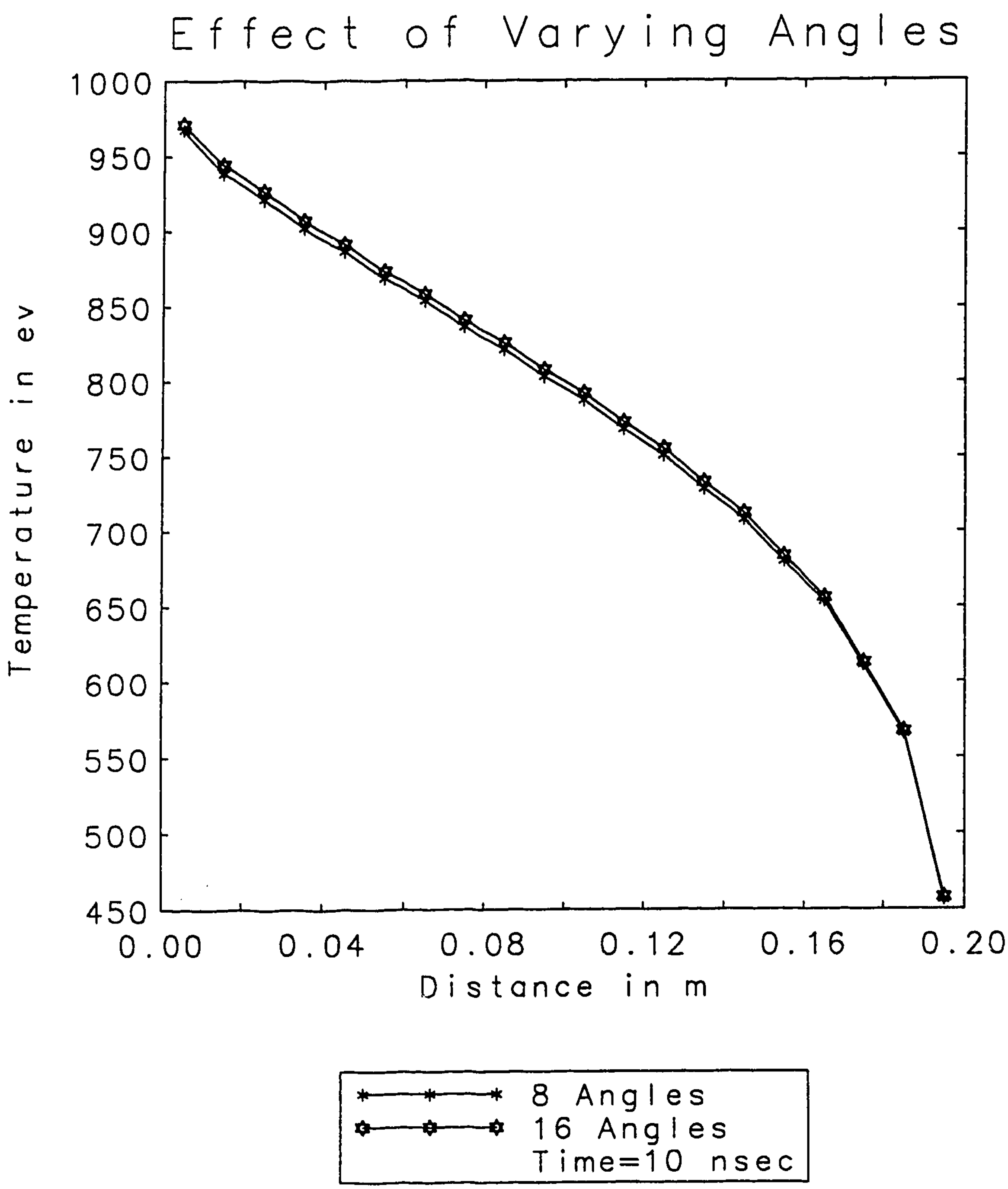

Effect of Varying Angles
1000
950
900
850
800
750
700
650
600
550
500
450
Temperature in ev
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
8 Angles
16 Angles
Time=10 nsec

Figure 4.4

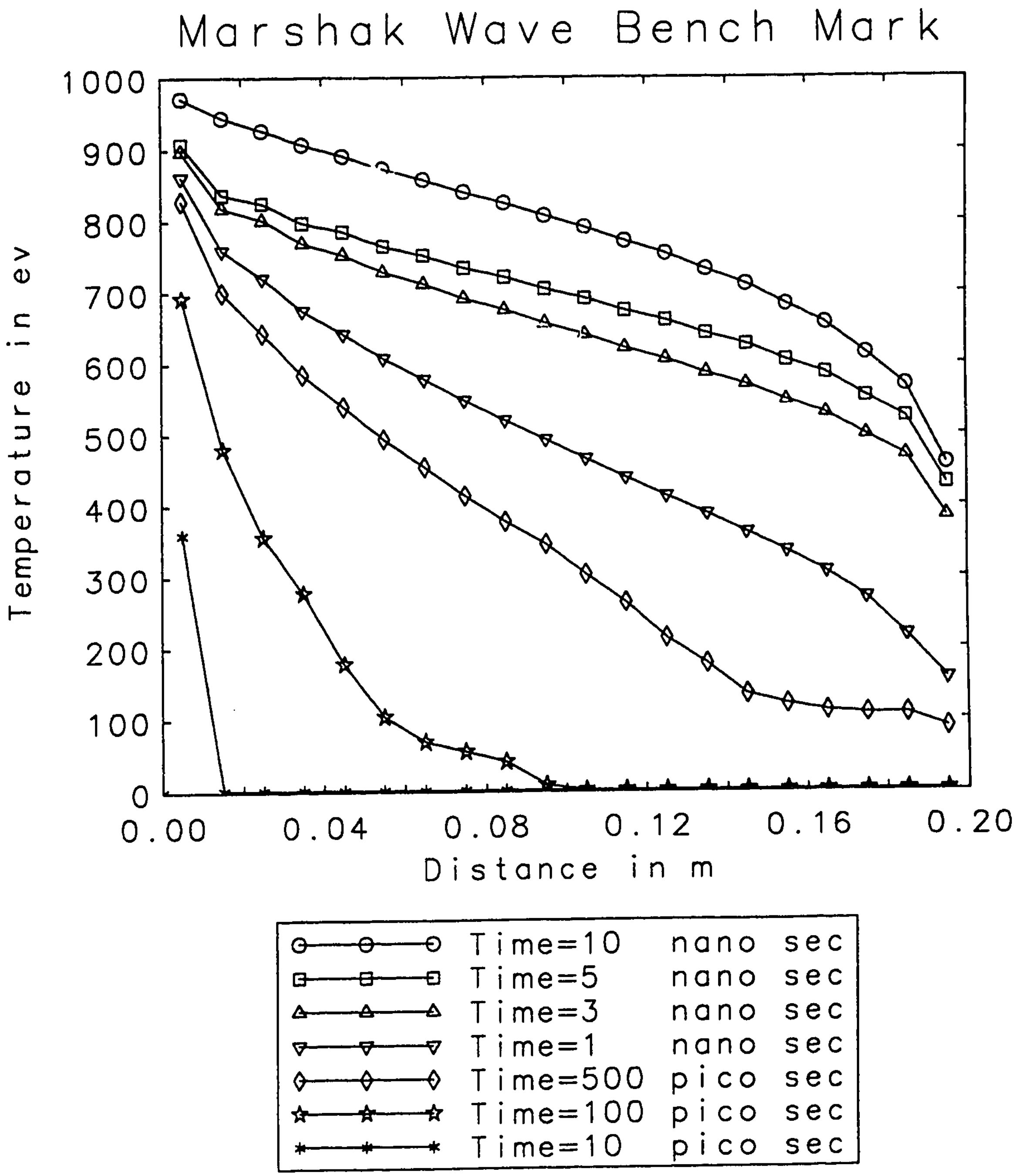

Marshak Wave Bench Mark
Temperature in ev
1000
900
800
700
600
500
400
300
200
100
0
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
Time=10 nano sec
Time=5 nano sec
Time=3 nano sec
Time=1 nano sec
Time=500 pico sec
Time=100 pico sec
Time=10 pico sec

Figure 4.5

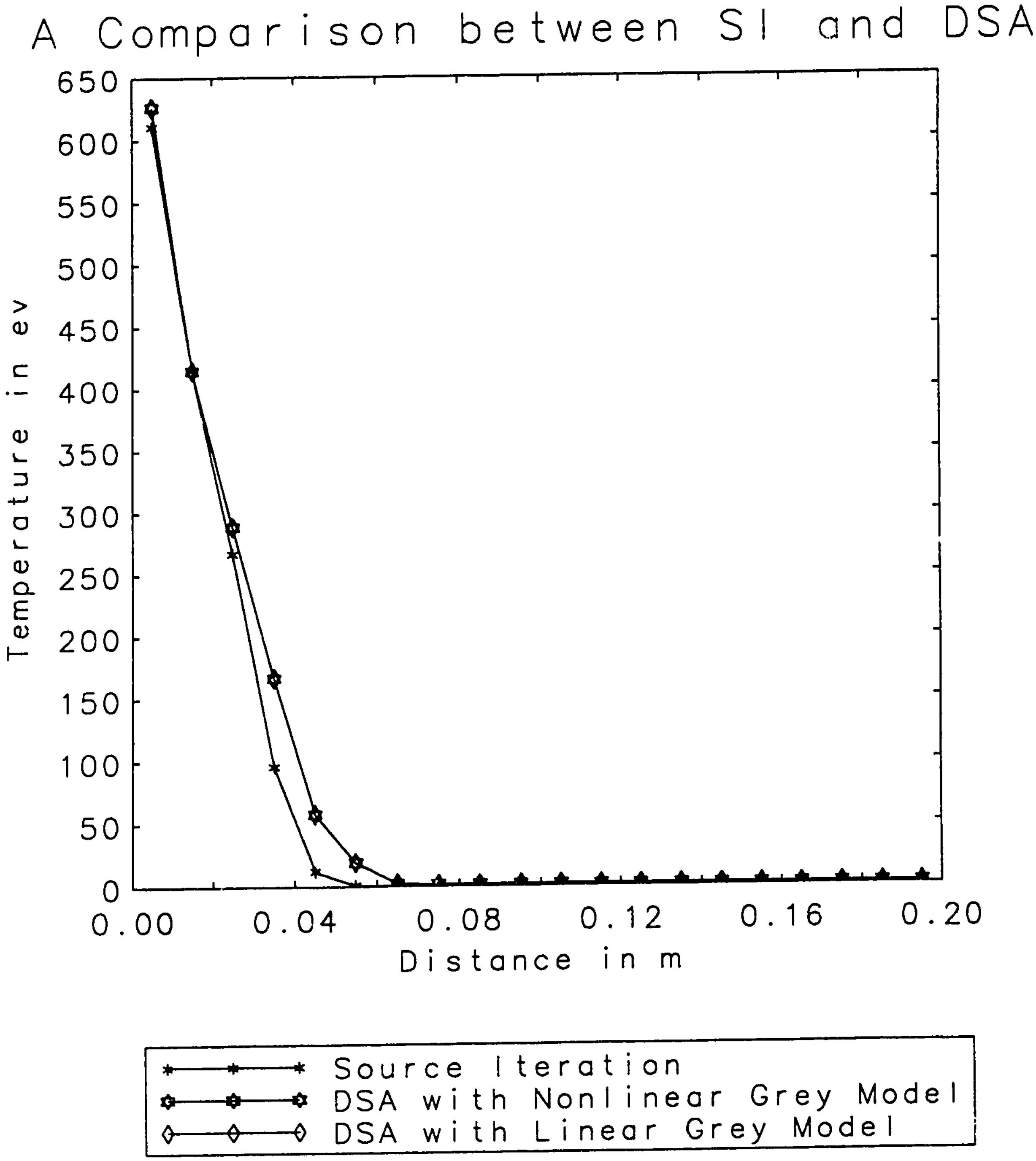

A Comparison between SI and DSA
650
600
550
500
450
400
350
300
250
200
150
100
50
0
Temperature in ev
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
Source Iteration
DSA with Nonlinear Grey Model
DSA with Linear Grey Model

Figure 4.6 a

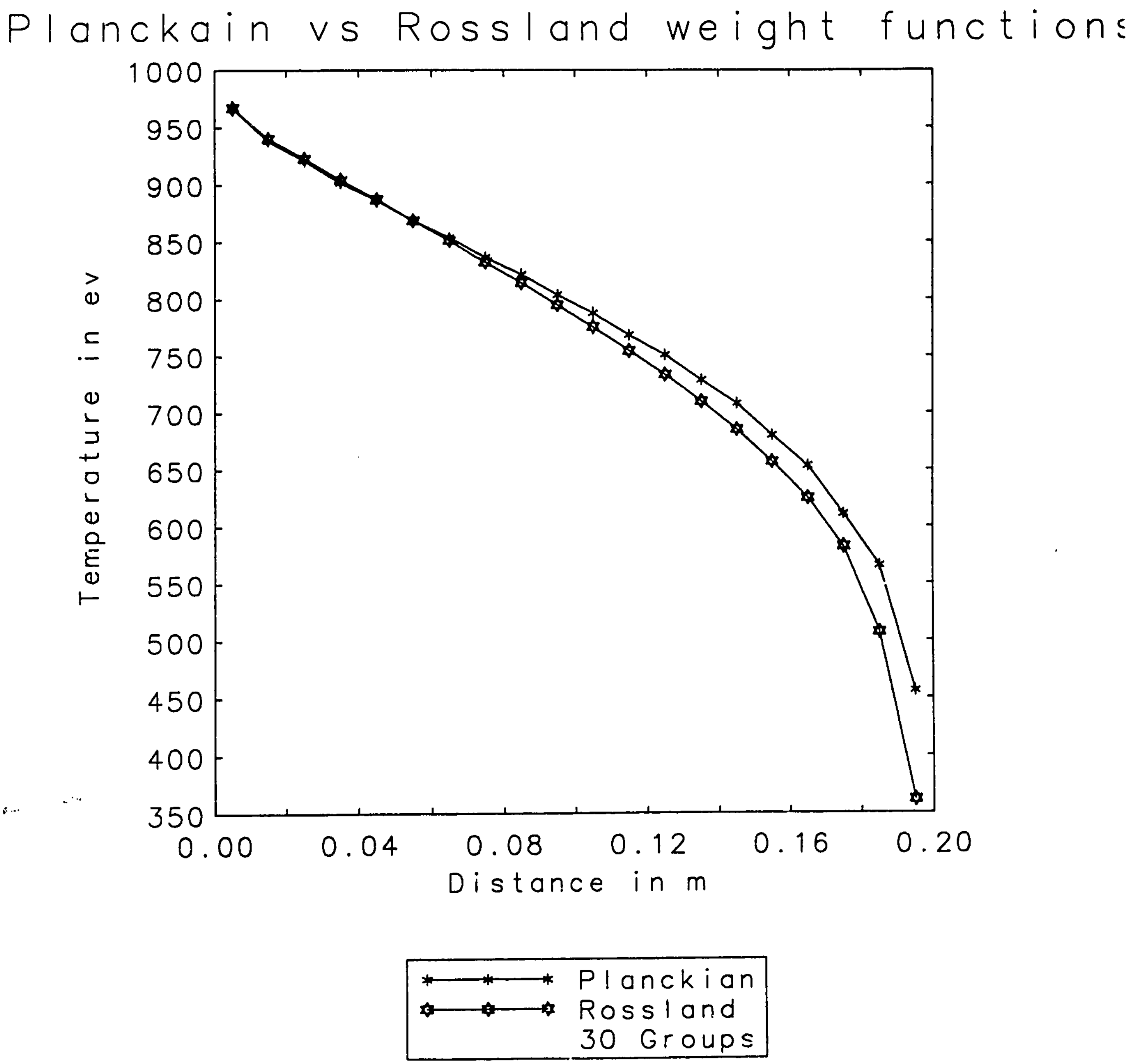
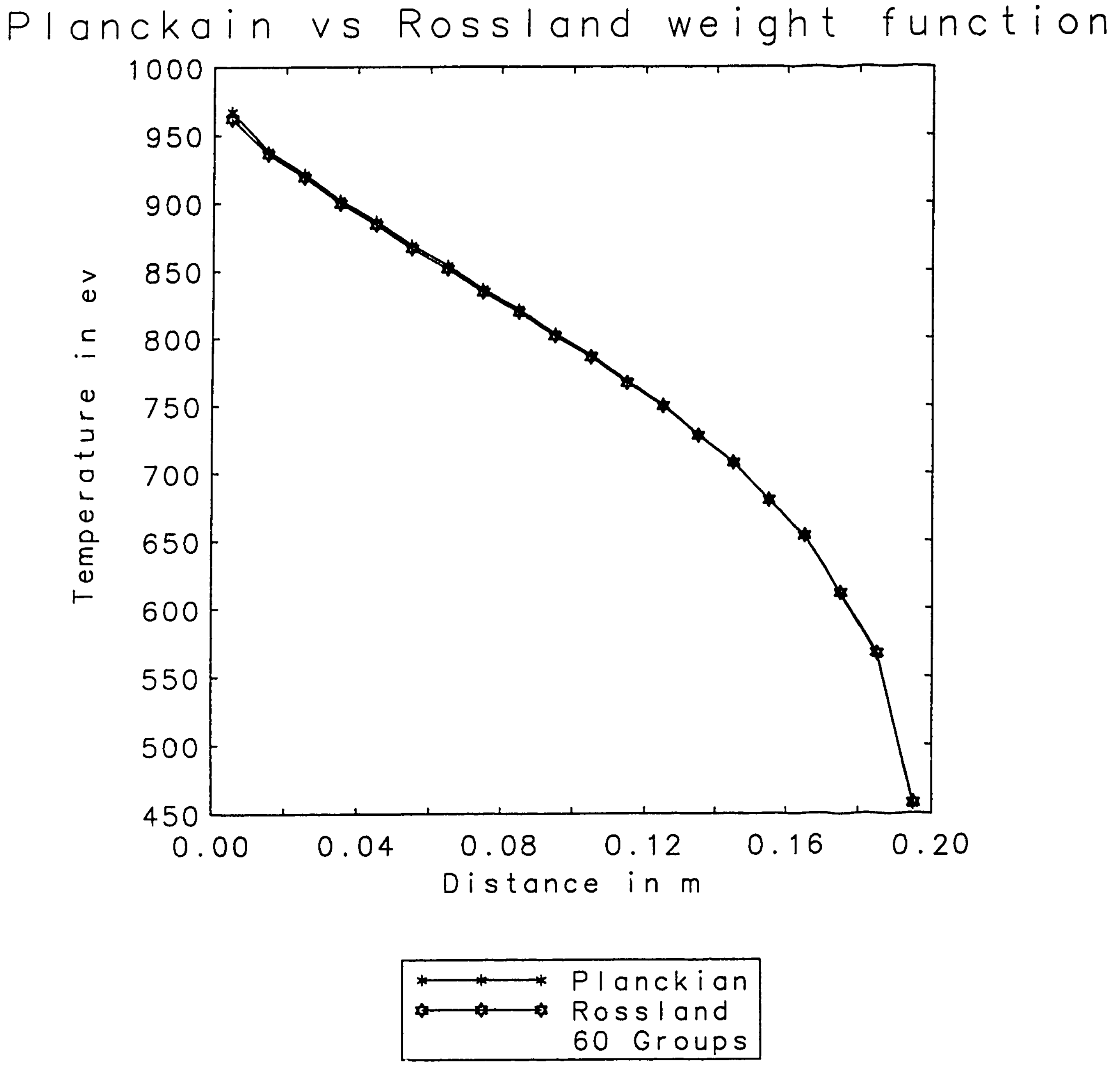

Planckain vs Rossland weight functions
Temperature in ev
1000
950
900
850
800
750
700
650
600
550
500
450
400
350
0.00
0.04
0.08
0.12
0.16
0.20
Distance in m
Planckian
Rossland
30 Groups

Figure 4.6 b


Planckain vs Rossland weight functions
1000
950
900
850
800
750
700
650
600
550
500
450
0.00
0.04
0.08
0.12
0.16
0.20
Temperature in ev
Distance in m
Planckian
Rossland
60 Groups

CHAPTER FIVE
MEDUSA SIMULATIONS

The objective of our research has been to study the nature and extent of the effects of radiation in laser produced plasma. We will show through simulations that the radiation plays an important role in the transport of heat in laser produced plasmas. The extent to which radiation affects the hydrodynamics of a plasma very much depends on the time profile and intensity of the laser beam to name few. An exclusion of radiative effects can lead to an erroneous estimate of radiation transported or loss of energy from a plasma. Although the inclusion of radiation transport into a hydrodynamic code is an arduous task, it is important because it gives a more accurate representation of the laser plasma interaction. This chapter has been divided into the following sections:

i. Introduction to MEDUSA

iii. Integration of the radiation code into MEDUSA

iii. Simulations using MEDUSA

This chapter will conclude the objective of our thesis. An extensive appendix is included at the end. The appendix includes the derivation of all of the equations used in the code **"sate.f"**.

5.1 INTRODUCTION TO MEDUSA:

In the mid 1970's a computer code named MEDUSA was written to simulate the laser fusion process. It was written for a one dimensional geometry in order to investigate the hydrodynamics and thermodynamics which take place in a small pellet. MEDUSA was refined later to accurately describe the interaction of laser light with a plasma. The MEDUSA code in its present form differs from the original due to modifications and additions made to it at University of Alberta. The present version of MEDUSA gives a more realistic simulation compared to the original one. New features included are:

i. A modified equation of state, which is valid at low temperatures and near solid density.

ii. A simultaneous solution of electron and ion diffusion and equilibration equations.

iii. A calculation of the local ionization and x-ray emission from a radiative equilibrium model.

iv. Addition of the radiation transport.

In this section a brief summary of the equations solved by MEDUSA are outlined, more details can be found elsewhere[5/1].

5.1.1 PHYSICAL MODEL:

In MEDUSA, the plasma is assumed to consist of a charge-neutral mixture of electrons and various species of ions (a collective term for atoms, ions and molecules). Thermodynamically, the electrons and ions are treated as two subsystems, each with its own internal energy, temperature and pressure. The two subsystems are coupled via a common velocity which ensures neutrality of the mixture. The exchange of energy is due to electron-ion and electron-atom collisions. Electric fields are ignored. The instantaneous local chemical composition is described by a set of fractions f_k such that

$$n_k = f_k n_i$$

where f_k is subject to $\Sigma f_k = 1$, and n_k is the ion density of the kth species. The electron density changes with time due to hydrodynamic expansion and contraction of the moving fluid. The electron and ion densities are related by

$$n_i = n_e \ / Z \quad m^{-3}$$

where Z is the atomic charge of the matter. The physical density is given as

$$\rho = n_i M m_H \quad Kg\ m^{-3}$$

where M is the atomic mass (the electron mass is ignored), m_H is the hydrogen mass. The average mass and charge is calculated from

$$M = \Sigma f_k M_k \text{ and } \quad Z = \Sigma f_k Z_k$$

5.1.2 ENERGY EQUATION:

The energy equation is given as

$$C_v \frac{dT}{dt} + B_T \frac{d\rho}{dt} + P \frac{dV}{dt} = S \qquad Jkg^{-1}$$

where S is the rate per unit mass at which energy enters each system. This is given by the following expressions for electrons and ions

$$S_e = H_e + K + J + X + Y_e$$

$$S_i = H_i - K + Q + Y_i$$

where
- H: heat flow due to thermal conduction
- K: rate of energy exchange between the electron and ion fluids.
- J: rate of Bremsstrahlung emission
- X: rate of absorption of laser light
- Q: rate of viscous shock heating
- Y: rate of nuclear energy released (it is switched off in our simulations, since it is irrelevant to our research)

and P is the pressure and C_v and B_T are given as

$$C_v = \left(\frac{\partial U}{\partial T}\right)_P \qquad B_T = \left(\frac{\partial U}{\partial \rho}\right)_T$$

5.1.3 HEAT CONDUCTION:

The heat conduction is assumed to be classical and is given as

$$H = \frac{1}{\rho} \nabla \chi \nabla T$$

where the thermal conductivity χ for electrons and ions is

$$\chi_e=\frac{1.83E-10T_e^{\frac{5}{2}}}{\log\Lambda Z\bar{Z}^2} \qquad W/m\ \ K$$

$$\chi_i=\frac{4.3E-12T_i^{\frac{5}{2}}}{\log\Lambda\sqrt{M}Z^2\bar{Z}^2} \qquad W/m\ \ K$$

$$\Lambda=\frac{1.24E06T_e^{\frac{3}{2}}n_e}{\sqrt{n_e}Z}$$

where Λ is the coulomb logarithm.

5.1.4 ENERGY EXCHANGE:

The exchange of energy between electrons and ions occurs at the rate of

$$\omega=\frac{Z^2e^4n_i\log\Lambda\sqrt{m_e}(kT_e)^{\frac{3}{2}}}{M32\sqrt{2\pi}\epsilon_o^2m_H} \qquad s^{-1}$$

the K term in the energy equation is

$$K=\frac{1}{\rho}\omega n_e k(T_i-T_e)$$

5.1.5 ENERGY EMITTED:

When radiation transport is not included, all of the radiation emitted from the plasma is taken to be via Bremsstrahlung emission by electrons and this emission is treated as a total loss. For a Maxwellian distribution it is given as

$$J=-8.5E-14\frac{n_e\sqrt{T_e}\bar{Z}^2}{M} \qquad WKg^{-1}$$

5.1.6 ENERGY ABSORBED:

The absorption is assumed to occur via inverse Bremsstrahlung at a density ρ below critical density ρ_c, where ρ_c is given as

$$\rho_c = \frac{\epsilon_o\, M\, m_H\, m_e}{Ze^2}\omega_L^2$$

At $\rho = \rho_c$ the remaining power is deposited in the next adjacent cell and reflection of the laser intensity is neglected. The absorption coefficient is given as

$$\alpha = 13.51\beta^2 \frac{(5.05 + \log\lambda T_e)Z^{\bar{2}}}{\lambda^2\sqrt{1-\beta}\,T_e^{\frac{3}{2}}} \qquad m^{-1}$$

$$\textit{where } \beta \textit{ is} \qquad \beta = \frac{\rho}{\rho_c}$$

If the initial laser intensity is P_o at $r = R_o$, then the power at r is given by

$$P = P_o e^{\alpha(R_o - r)}$$

5.1.7 MOTION OF THE PLASMA:

The motion of the plasma is governed by the Navier-Stokes equation

$$\rho\frac{du}{dt} = -\nabla P$$

where u is the velocity of the plasma and P is the total pressure, $P = P_i + P_e$.

5.1.8 EQUATION OF STATE:

The equation of state assumes that ions behave as a non-degenerate perfect gas, while electrons behave as being degenerate, non-degenerate or partially degenerate. The most recent EOS package gives a more accurate description of a metal at low temperatures and coincides with the EOS of the original MEDUSA at high temperatures. It is invoked by setting NLOM2(6)=.TRUE., NLOM2(7)=.TRUE. and STATE=-1 (for aluminum).

5.1.9 DIFFERENCING SCHEMES:

The cell edges are free to move, thereby altering the volume of the cell. The quantities at the centre are obtained by averaging the values at the cell edges. A finite differencing scheme is used to discretize the equations shown above. The cell width varies such that the mass of the cell remains constant during the simulations, since thermonuclear burning is turned off.

5.1.10 TIME DIFFERENCING AND TIME STEP CONTROL:

MEDUSA uses a five level scheme. Levels 1,3,5 correspond to the time step n-1,n,n+1 respectively and levels 2,4 correspond to the time between n-1 & n and n & n+1. If we let r, ρ, $T_{e,i}$, to be the cell coordinate, mass density and temperature (electron and ion) respectively then with known values for $(r, \rho, T_{e,i})^{n-1}$ and $u^{n-1/2}$ $(r, \rho)^n$ we determine T^n, $u^{n+1/2}$, $(r, \rho)^{n+1}$. All basic quantities are thereby advanced one step in time. Due to the non-linearity of the $T_{e,i}$ in the energy equation, this equation is solved by iteration.

Since the temperature equations are solved implicitly, the solution for the temperature is stable for all time steps. However, the time step should be chosen carefully in order to obtain results within a desirable accuracy. The maximum value of Δt is restricted by the Courant-Friedrichs-Lewy conditions and is given as

$$\Delta t^{n+\frac{1}{2}} < \frac{a_1 Min(r_{j+1}^n - r_j^n)}{c_n}$$

The time step is also constrained by the maximum variation allowed in the hydrodynamic velocity and in the ion and electron temperatures.

5.1.11 LASER PROFILE:

In our simulations we use two types of laser beam time profiles. These are gaussian and trapezoid profiles. The maximum power attributed to a given beam ranges from 10^{16} Wm^{-2} to 10^{18} Wm^{-2}. A detailed description of laser characteristics is given wherever they are used in the simulations sections.

5.2 INTEGRATION OF DSA CODE IN MEDUSA:

The atomic physics tables developed by Lee[5/2] contain the emissivity and the microscopic cross-section area at a given temperature and ion density. These quantities are in MKS units, watts/ion and m^2, respectively. For the transfer equation we need the absorption coefficient and the initial intensity. In order to find the relation between given variables and required variables, consider the radiative transfer equation and the energy balance equation

$$\mu\frac{\partial I}{\partial x} + (\sigma+\tau)I = S = \sigma B$$
$$u_e^{n+1} = \frac{\eta}{c\sigma_p^*}\left[\int_0^\infty \sigma^* I_0 d\nu + \frac{u_e^n}{\Delta t\beta^*}\right]$$
$$\textit{where} \qquad \beta = \frac{4aT^3}{c_v}$$

(5.2.01)

where I is the specific intensity in W m^{-2}, σ is the absorption coefficient in m^{-1}, S is the rate of energy emission due to spontaneous processes, c_v is the specific heat per unit volume and c is the speed of light in m/sec and B is the specific intensity.

A careful analysis shows that one can relate the emissivity and the microscopic cross-section area to the specific intensity and absorption coefficient as

$$B = \frac{N_i P\ \Delta x}{4\pi} \qquad \textit{and} \qquad \sigma = N_i A_x$$

(5.2.02)

where P: is the rate of energy emitted W/ion

A_x: is the microscopic cross-sectional area in m^2

N_i: is the ion density per unit volume

The values of the emissivity and cross-section area are obtained through interpolation at a given temperature and density by calling subroutine DINTRP. If the temperature is less than 10 ev then the emissivity is taken to be Planckian, and the cross-sectional area is obtained by calling the function named CIGMA. When CIGMA is called to obtain the absorption coefficient σ, then

$$\sigma = \frac{M \acute{A}_x}{N_A}$$

(5.2.03)

where A'_x is obtained from CIGMA, N_A is the Avogadro number, and M is the atomic mass. The specific heat is obtained from the EOS subroutine, and is given as

$$c_v = \rho_3 \frac{[c_{vi} + c_{ve}]}{10.0}$$

(5.2.04)

Where ρ_3 corresponds to mass density at time step 3 and c_{vi} and c_{ve} are the specific heats obtained from EOS at time level 3.

The emitted power is defined as the power loss per unit mass from the plasma; therefore, it is proportional to the difference in the intensity at the adjacent cell boundaries obtained after solving the transfer equation at time level 3. This can be expressed as

$$BREMS_l = \frac{2\pi}{\Delta m_l} \Sigma_g A_{xg} [\Sigma_m (I_{mgl+\frac{1}{2}} - I_{mgl-\frac{1}{2}}) \omega_m]$$

(5.2.05)

The present radiation code is invoked by setting SAHA=3.1. In all of the simulations the foil is taken to be **aluminium** and the **flux limiter** is taken to be 0.25. Variables such as temperature, pressure, velocity, average Z and mass density as a function of space are stored into file named "**out**". Quantities such as the ablation pressure, ablated mass, average Z, thermal and kinetic energy and emission to the left and right end of the plasma, as a function of time are stored into "**abl.out**". The **ablated mass** is defined as the mass contained between the outside cell and the last cell in the plasma in which an outward velocity is obtained. The **ablation pressure** is defined as the pressure of the cell embodied in the ablative surface. The option of a trapezoid time profile for the laser beam has been added. This can be invoked by setting GAUSS =-2.0, PLENTH="rise time (assumed to be same as fall time)". The input file for MEDUSA is "**mds.in**" and it is given below, this will allow another user to reproduce the results presented here without any uncertainty in the initial conditions that are used. Important variables are bolded.

```
F0000000  0  1
 ENERGY : 1.00 JOULES    FOCAL SPOT DIA: 70 MICRONS
 TARGET : ALUMINUM        2.0 NS PULSE

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
 &NEWRUN
  AK0=0.10,                AK1=0.10,                AK2=0.10,
  AK3=0.10,                AK4=0.10,                AK5=0.00,
  ANABS=1.0,               ANPULS=1.0,
  BNEUM=2.0,
  DTEMAX=0.1,              DTIMAX=0.1,              DUMAX=0.1,
  DTPRNT=2.5D-10,          DELTAT=1.0D-12,          DRGLAS=0.0,
  DRPLAS=0.0,              DEUTER=0.0
  FHOT=0.0,                FTHOT=0.0,               FLIMIT=1.00,
  FNE=1.0,
  GAUSS= -2.0,             GAMMAE=1.66666667,       GAMMAI=1.66666667,
  HELIU3=0.0,              HELIU4=0.0,              HYDROG=0.0,
  LAMDA1=2.48D-06,
  MESH=40,                 MXDUMP=10,
  NITMAX=50,               NCASE=1,                 NRUN=99999,
  NLABS=.TRUE.,            NLMOVE=.TRIE.,           NLBURN=.FALSE.,
  NLCRI1=.TRUE.,           NLBRMS=.TRUE.,           NLITE=.TRUE.,
  NLDEPO=.FALSE.,          NLDUMP=.FALSE.,          NLFUSE=.FALSE.,
  NREP=0,                  NTRLMS=0.0,              NSLEDG=9999,
  NDUMP=10,                NETRAL=0.0,              NFILM=1,
  NGROUP=5,                NHDCPY=100,              NIN=5,
  NLCHED=.FALSE.,          NLEDGE=10,               NLEMP=.TRUE.,
  NLFILM=.TRUE.,           NLHCPY=.FALSE.,          NLPRNT=.TRUE.,
  NLREPT=.FALSE.,          NONLIN=1,                NSHELL=150,
  NP1=1,                   NP2=40,                  NP3=1,
  NPRNT = 50000,           NGEOM=1,                 NREP=0,
  NLECON=.TRUE.,           NLICON=.TRUE.,           NLOMT2(11)=.TRUE.,
  NLPFE=.FALSE.,           NLPFI=.TRUE.,            NLX=.TRUE.,
  NLOMT2(6)=.TRUE.,        NLOMT2(7)=.TRUE.,
  OUTAMI=1.0,
  PIQ(55)=2.0,             PONDF=-1.0,              PLENTH=1.0D-10,
  PMULT=2.0,               PMAX=3.16D+17,
  QSHELL(1)=0.97,
  RINI=1.0D-04,            ROGLAS=0.0,              ROPLAS=0.0,
  RHOGAS=2700.0,           RHOINI=2700.0,           RSHELL=1.0D-04,
  RHOT=0.0,
  SAHA=3.1,                STATE=-1.0,              SIMULT=1.0,
  SCR=1.0,                 SCRHO=1.0,               SCTE=1.0,
  SCTI=1.0,                SCTIME=1.0,              SCP=1.0,
  TEINI=3.0D02,            TIINI=3.0D02,            TINUCL=1.0D+07,
  TRITIU=0.0,              TON=0.0,                 TOFF=3.9D-09,
  TSTOP=5.0D-09,
  XMASS=26.9815,           XTRA=1.0,                XZ=13.0,
  ZGLAS=0.0,               ZPLAS=0.0,
 &END
  HNUMIN=8000.0,           HNUMAX=9000.0,           NLEMRA=.FALSE.
```

5.3 MEDUSA SIMULATIONS:

This section includes the MEDUSA simulations. The main objective is to validate Marchand's Model with the present code. This is done by comparing the MEDUSA simulations which use Marchand's model and the present code. It will be shown that there are cases where the two algorithms yield different results. These results will be presented followed by a discussion of which algorithm appears to be more accurate. Since Marchand's model was successfully tested against experimental observation, we believe that the present code must also confirm the same experimental observations.

The graphs to be presented in this section reflect the consistency and the differences of the two models. It is observed that Marchand's model works well for a trapezoid profile beam, but appears to give erroneous results when a gaussian profile beam is used. The effects of radiation transport are the subject of this section. The input file for MEDUSA is given on the previous page.

Figure 5.1 a, b, c

This graph represents the electron temperature and normalized mass density of the plasma as a function of space using the DSA model and Marchand's 2-cell simplified model. The laser beam has a trapezoid temporal profile (with RISE TIME=1.0E-10 sec, TOFF=4.0E-09 sec and PMAX=3.16E17 Wm^{-2}, λ=2.48E-07). These graphs correspond to different time steps and are presented here to show the consistency between the present model and Marchand's model at various time steps.

Figure 5.2 a, b, c

This graph represents the pressure and normalized mass density of the plasma as a function of space with the DSA model and Marchand's model. The laser beam is the same as the one used in Fig. 5.1 (trapezoid). Each graph can be viewed as a snap shot of the pressure and the mass density at different times. Again the consistency between the two models is evident.

Figure 5.3

This graph represents the ablation mass as a function of time with the DSA model and Marchand's model. The laser profile is the same as in Fig. 5.1.

Figure 5.4

This graph represents the ablation pressure as a function of time with the DSA model and Marchand's model. The laser profile is the same as Fig. 5.1.

Figure 5.5 a,b,c

These graphs represent the electron temperature and mass density as a function of space, when a gaussian laser profile (with FWHM=1.7E-09 sec, PMAX=5.0E16 Wm^{-2}, λ=0.35E-06) is used. This figure shows that when a gaussian profile is used the two models do not give as consistent electron temperature as they did for trapezoid laser profile.

Figure 5.6 a,b,c

Here the pressure and mass density are shown as a function of space, when a gaussian profile for the laser beam is used (same as in Fig. 5.5). Even though the electron temperature was the same for both models, the pressure is not the same. The inconsistency in pressure is quiet conspicuous, the dip in pressure about the ablation surface in Fig. 5.6 b of Marchand's model appears to be nonphysical.

Figure 5.7

This graph represents the ablation mass of the plasma when a laser beam with a gaussian profile is used. Here the inconsistency is so conspicuous that one needs to reject either the DSA model or Marchand's model. It appears that the DSA model is more accurate than Marchand's model, because in Figure 5.3 the intensity is higher than in Fig. 5.7, which implies that the ablation mass in Fig. 5.7 should be less than the ablation mass in Fig. 5.3. This is not true in Marchand's model.

Figure 5.8

This graph represents the ablation pressure of the plasma when a laser beam with a gaussian profile is used. Even though the ablation mass from the two models is quiet different, the ablation pressure is not very different. This reflects the complexity of the radiation transport, that is, consistency in one result does not necessary imply that other parameters are consistent too.

Figure 5.9 a,b,c,d,e

These graphs show the effects of radiation transport on the ablated mass, ablation pressure, kinetic energy, thermal energy and ion energy. The laser profile is a trapezoid (with PLENTH=1.0E-10, PMAX=3.16E17). In these graphs results of MEDUSA with radiation transport (by invoking DSA model or Marchand model) and with no radiation transport (it implies that all radiation emitted is lost rather than being transported) are presented.

Fig. 5.9 a,b show the effect of radiation transport on the ablation mass and ablation pressure. Mora[5/34] has shown that the ablation mass and ablation pressure are proportional to intensity and to the critical density of the plasma and are given as

$$m_{abl} \propto I_o^{\frac{1}{2}} n_c^{\frac{1}{4}} \qquad P_{abl} \propto I_o^{\frac{3}{4}} n_c^{\frac{1}{8}}$$

These relations indicate that ablation pressure is less sensitive to radiation transport compared to ablation mass. The weak dependence of pressure on radiation is due to the fact that with radiation transport the thermal energy is converted into x-rays which can escape the plasma, this conversion reduces the ablation pressure. Radiation transport on the other hand transports energy to higher density plasma which in turn contributes to an increase in ablation pressure. The graphs in Fig. 5.9 a,b qualitatively confirm such relations.

When radiation transport is not included, all of the energy emitted due to bremsstrahlung is taken to be a loss. With radiation transport some of the energy is transported to a colder dense plasma. Therefore, it is expected that when radiation transport is included the kinetic energy, ion energy and thermal energy should be relatively large compared to the case when no radiation is included. This conclusion is qualitatively confirmed from Fig. 5.9 c,d,e.

Figure 5.1 a

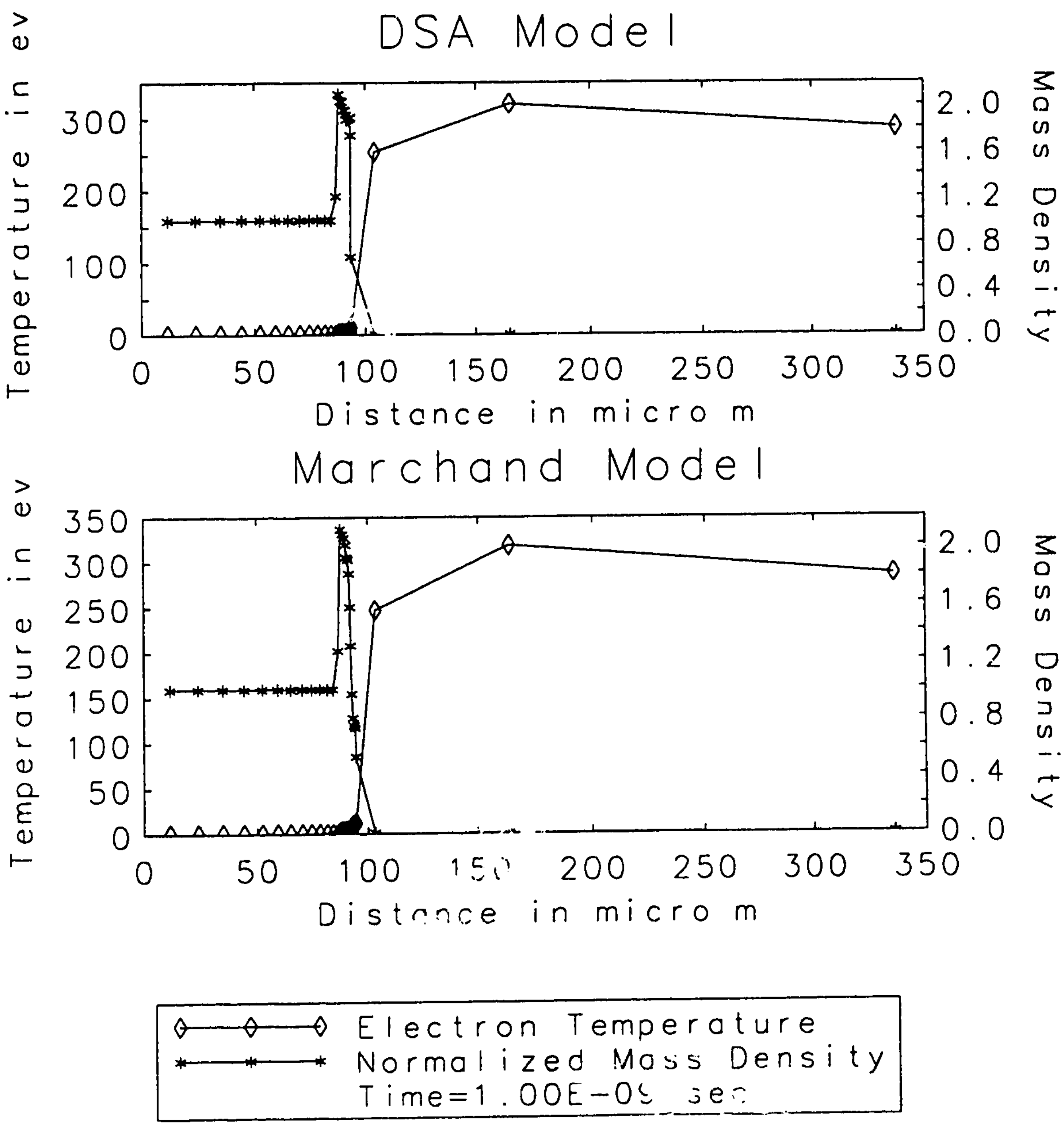

DSA Model
Temperature in ev
Mass Density
0
100
200
300
0.0
0.4
0.8
1.2
1.6
2.0
0 50 100 150 200 250 300 350
Distance in micro m
Marchand Model
Temperature in ev
Mass Density
0
50
100
150
200
250
300
350
0.0
0.4
0.8
1.2
1.6
2.0
0 50 100 150 200 250 300 350
Distance in micro m
Electron Temperature
Normalized Mass Density
Time=1.00E-09 sec

Figure 5.1 b

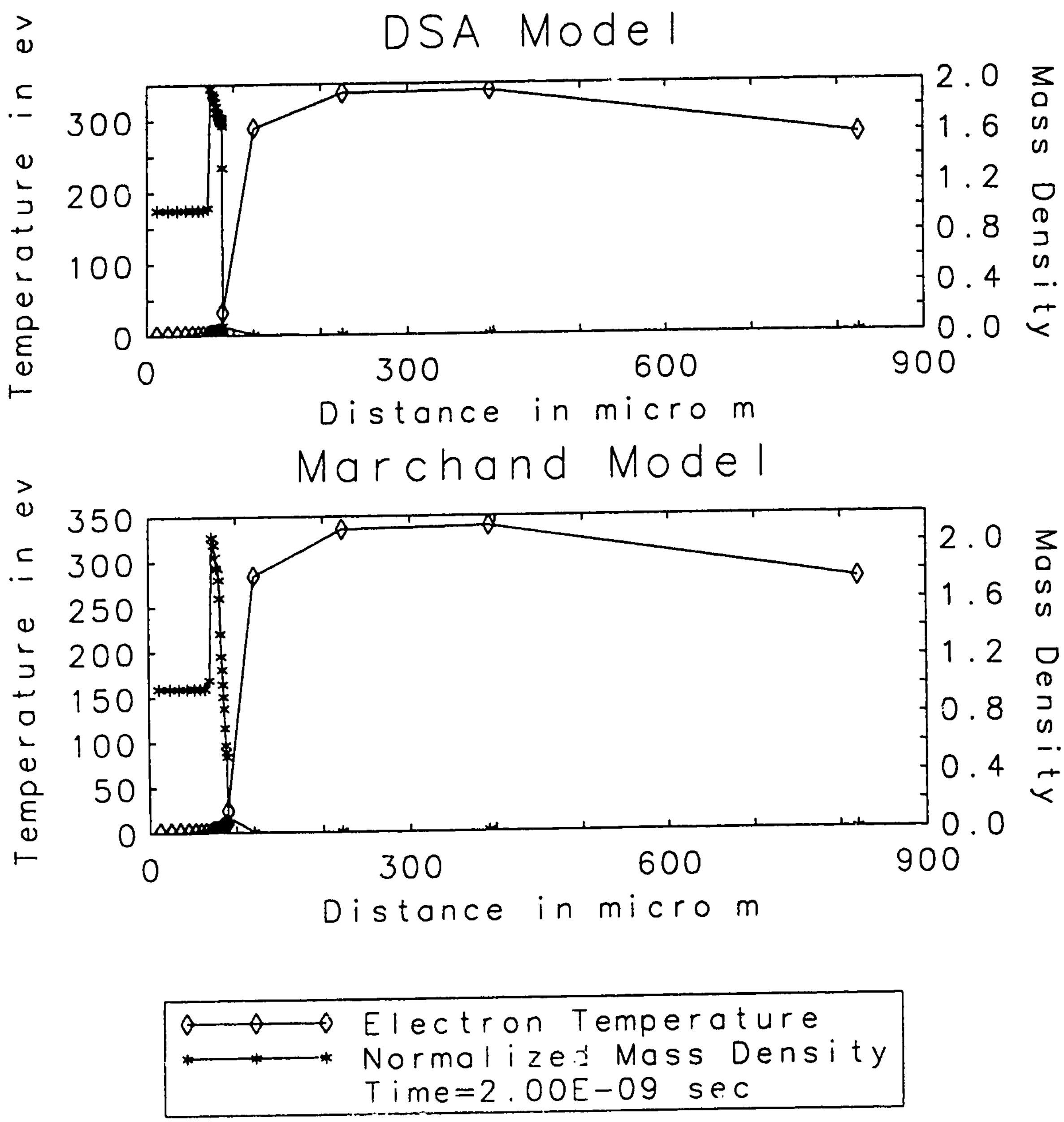

DSA Model
Temperature in ev
0
100
200
300
Mass Density
0.0
0.4
0.8
1.2
1.6
2.0
0
300
600
900
Distance in micro m
Marchand Model
Temperature in ev
0
50
100
150
200
250
300
350
Mass Density
0.0
0.4
0.8
1.2
1.6
2.0
0
300
600
900
Distance in micro m
Electron Temperature
Normalized Mass Density
Time=2.00E-09 sec

Figure 5.1 c

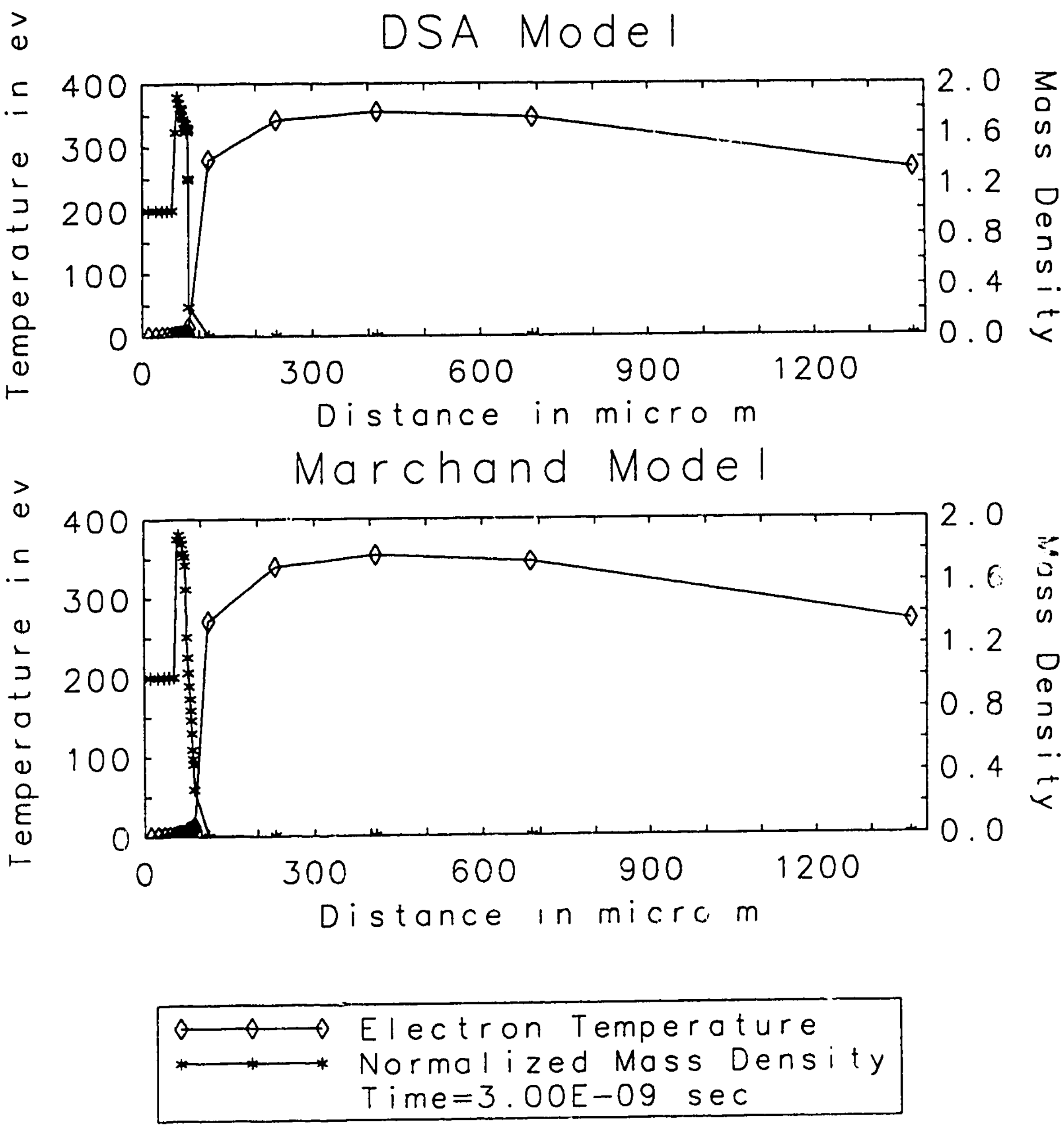

DSA Model
Temperature in ev
Mass Density
400
300
200
100
0
2.0
1.6
1.2
0.8
0.4
0.0
0
300
600
900
1200
Distance in micro m
Marchand Model
Temperature in ev
Mass Density
400
300
200
100
0
2.0
1.6
1.2
0.8
0.4
0.0
0
300
600
900
1200
Distance in micro m
Electron Temperature
Normalized Mass Density
Time=3.00E-09 sec

Figure 5.2 a

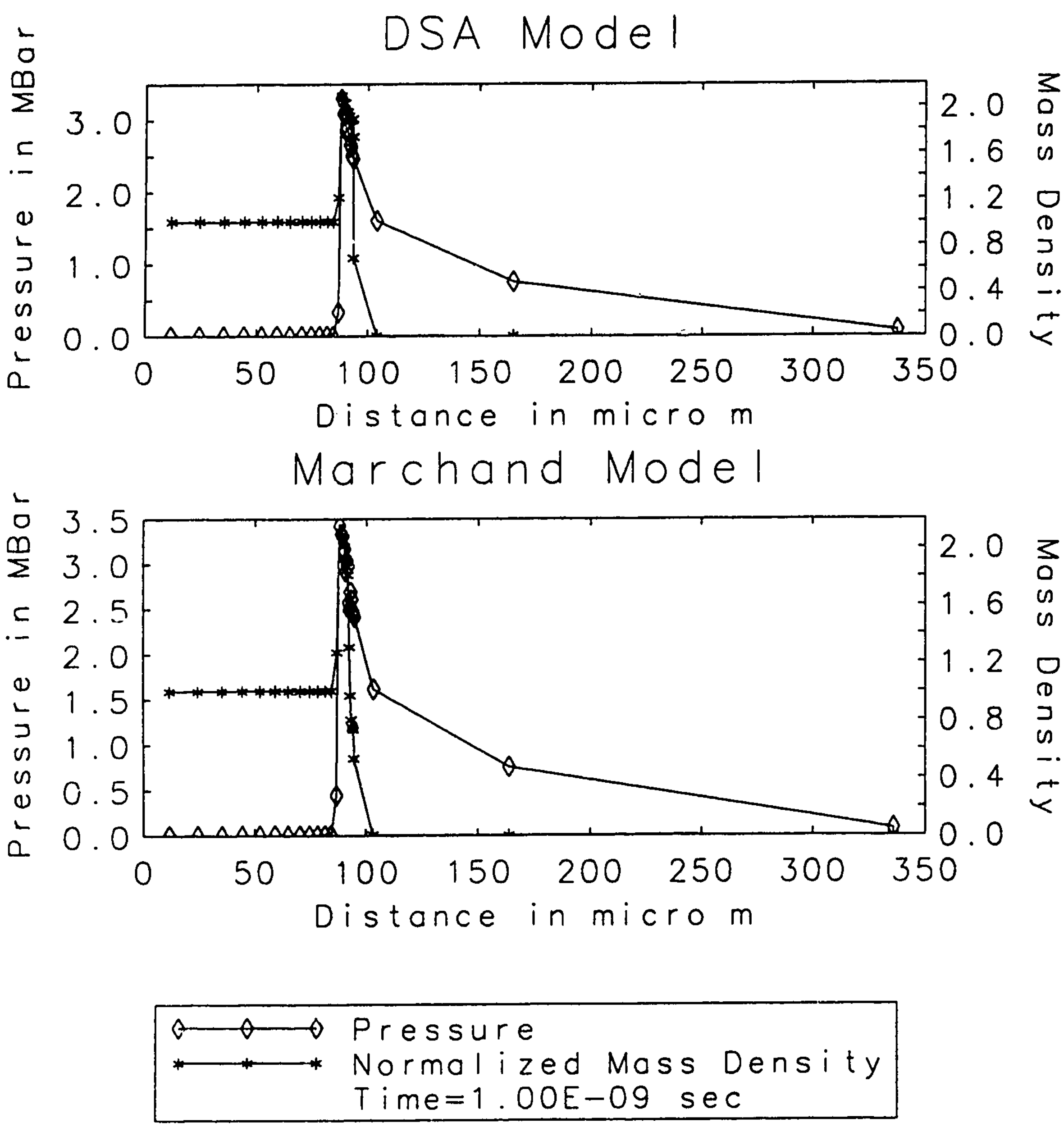

DSA Model
Pressure in MBar
Mass Density
Distance in micro m
Marchand Model
Pressure in MBar
Mass Density
Distance in micro m
Pressure
Normalized Mass Density
Time=1.00E-09 sec

Figure 5.2 b

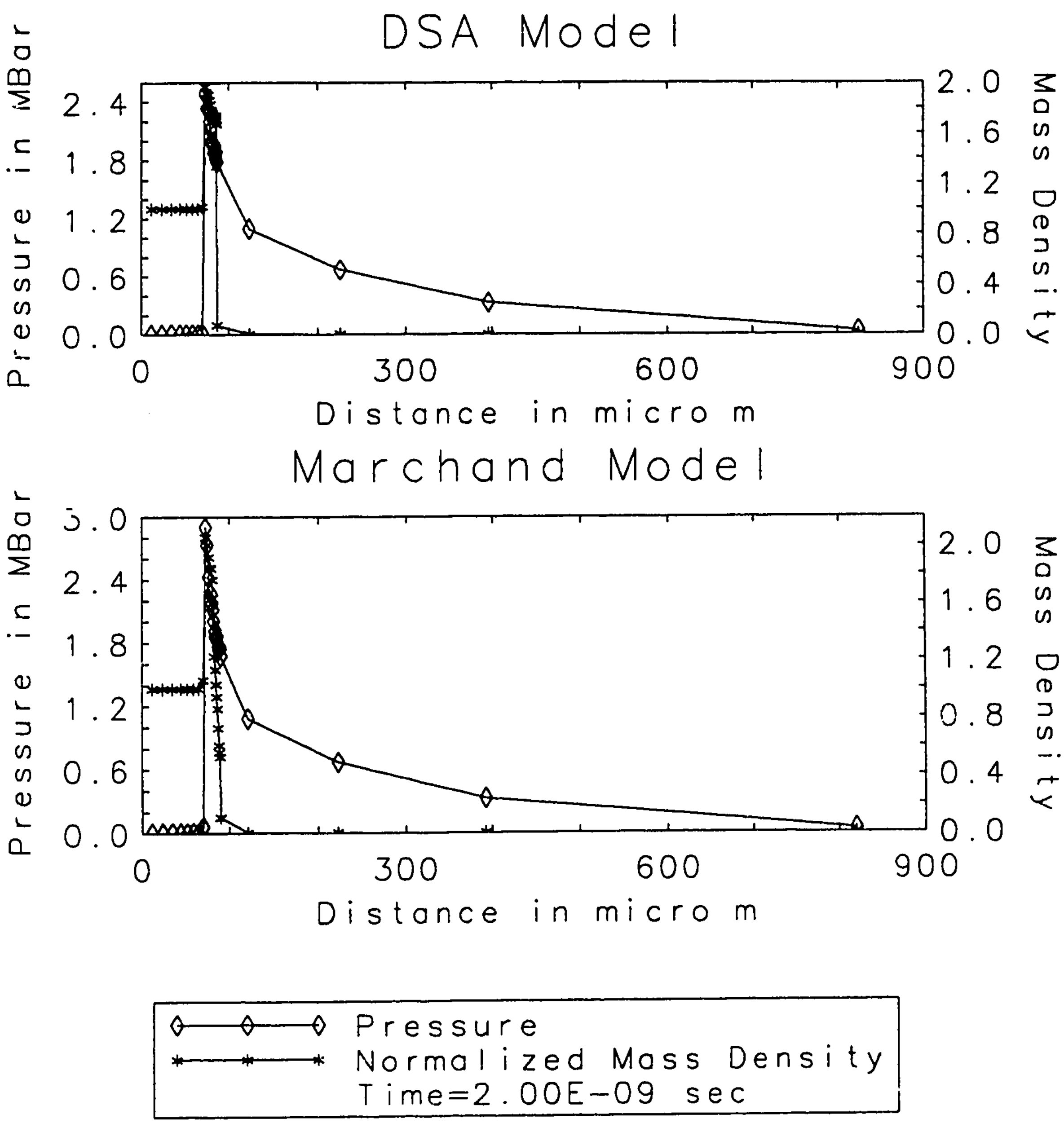

DSA Model
Pressure in MBar
2.4
1.8
1.2
0.6
0.0
2.0
1.6
1.2
0.8
0.4
0.0
Mass Density
0
300
600
900
Distance in micro m
Marchand Model
Pressure in MBar
3.0
2.4
1.8
1.2
0.6
0.0
2.0
1.6
1.2
0.8
0.4
0.0
Mass Density
0
300
600
900
Distance in micro m
Pressure
Normalized Mass Density
Time=2.00E-09 sec

Figure 5.2 c

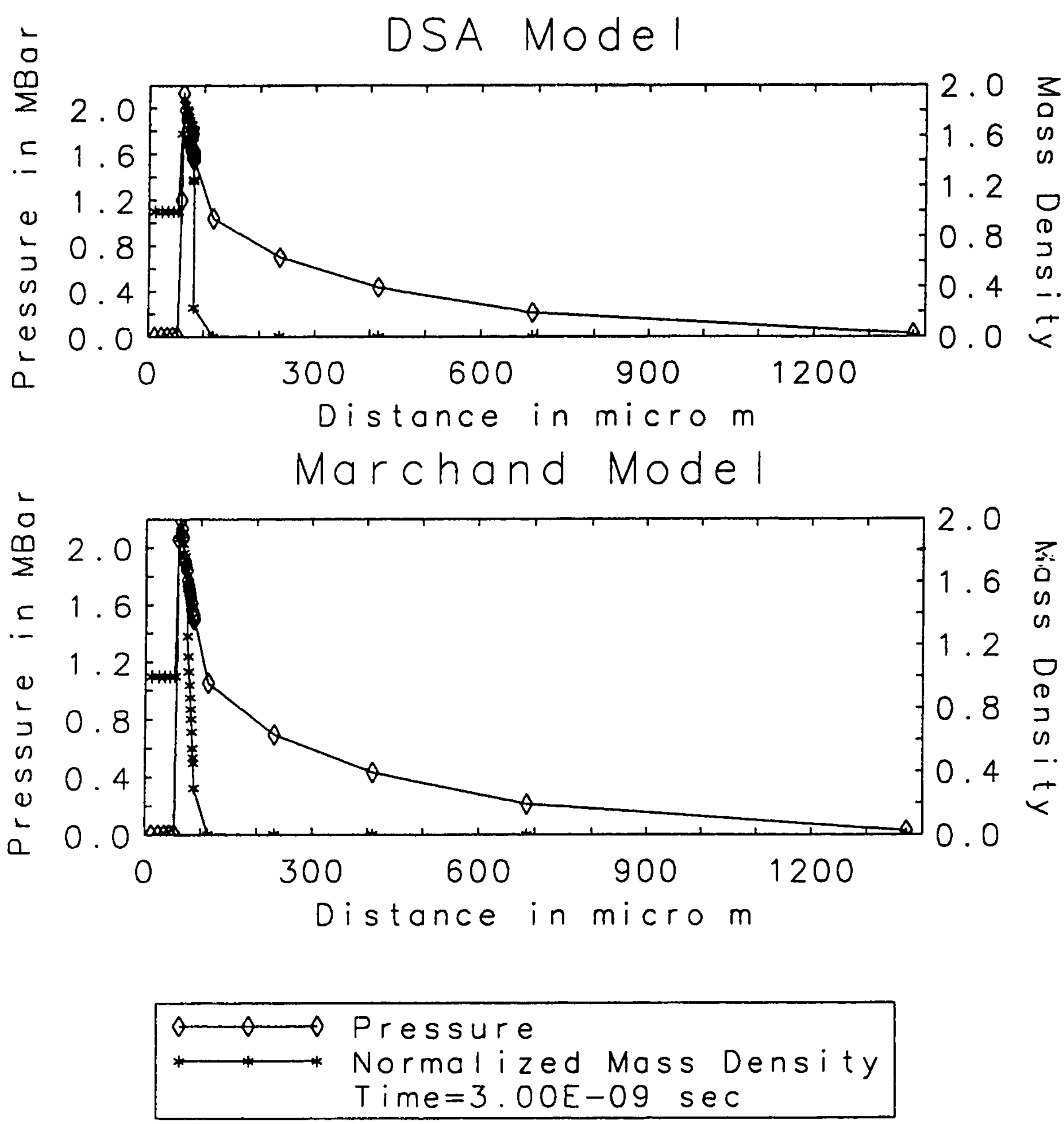

DSA Model
Pressure in MBar
Mass Density
Distance in micro m
0
300
600
900
1200
0.0
0.4
0.8
1.2
1.6
2.0
Marchand Model
Pressure in MBar
Mass Density
Distance in micro m
0
300
600
900
1200
0.0
0.4
0.8
1.2
1.6
2.0
Pressure
Normalized Mass Density
Time=3.00E-09 sec

Figure 5.3

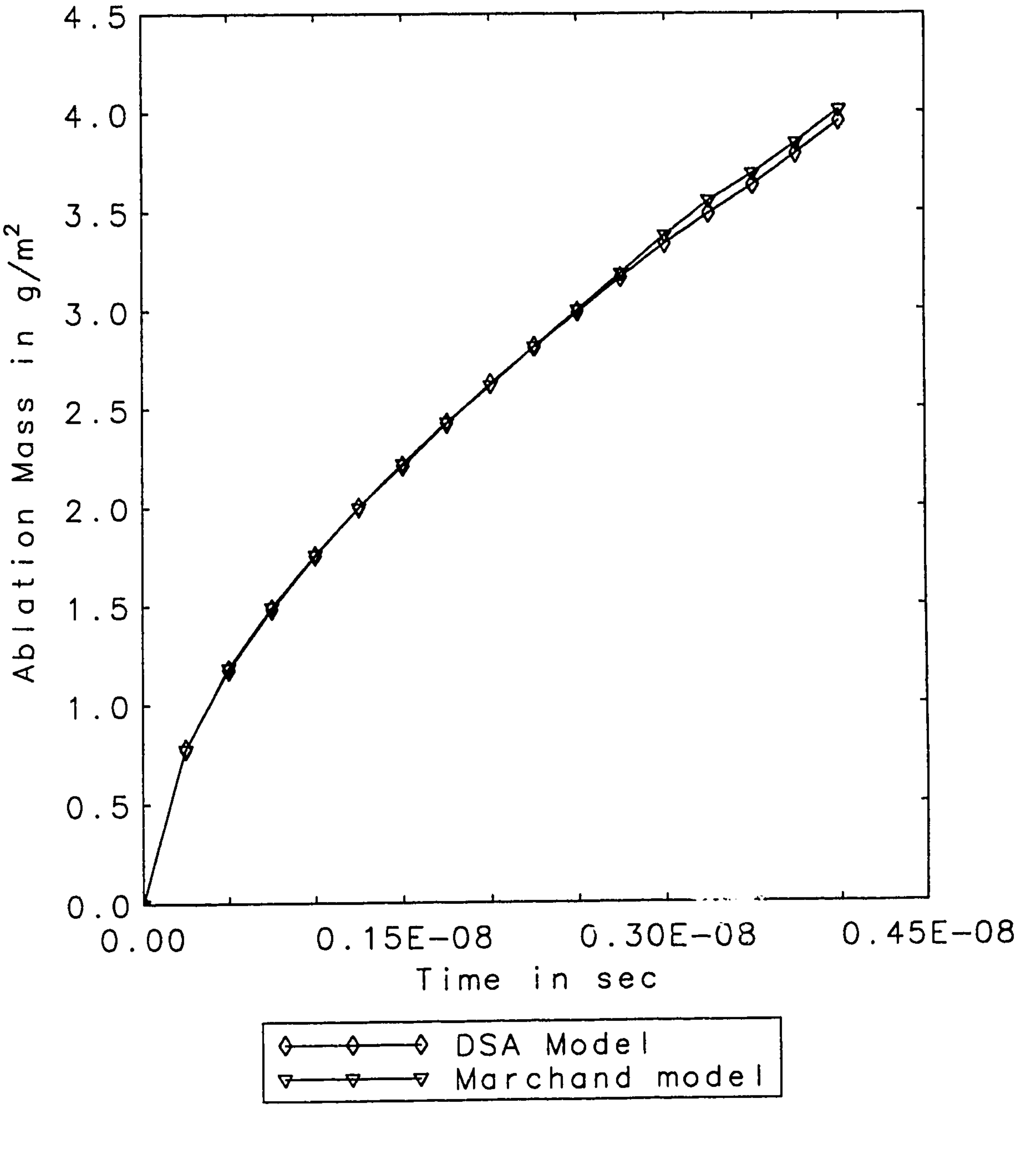

4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Ablation Mass in g/m²
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec
DSA Model
Marchand model

Figure 5.4

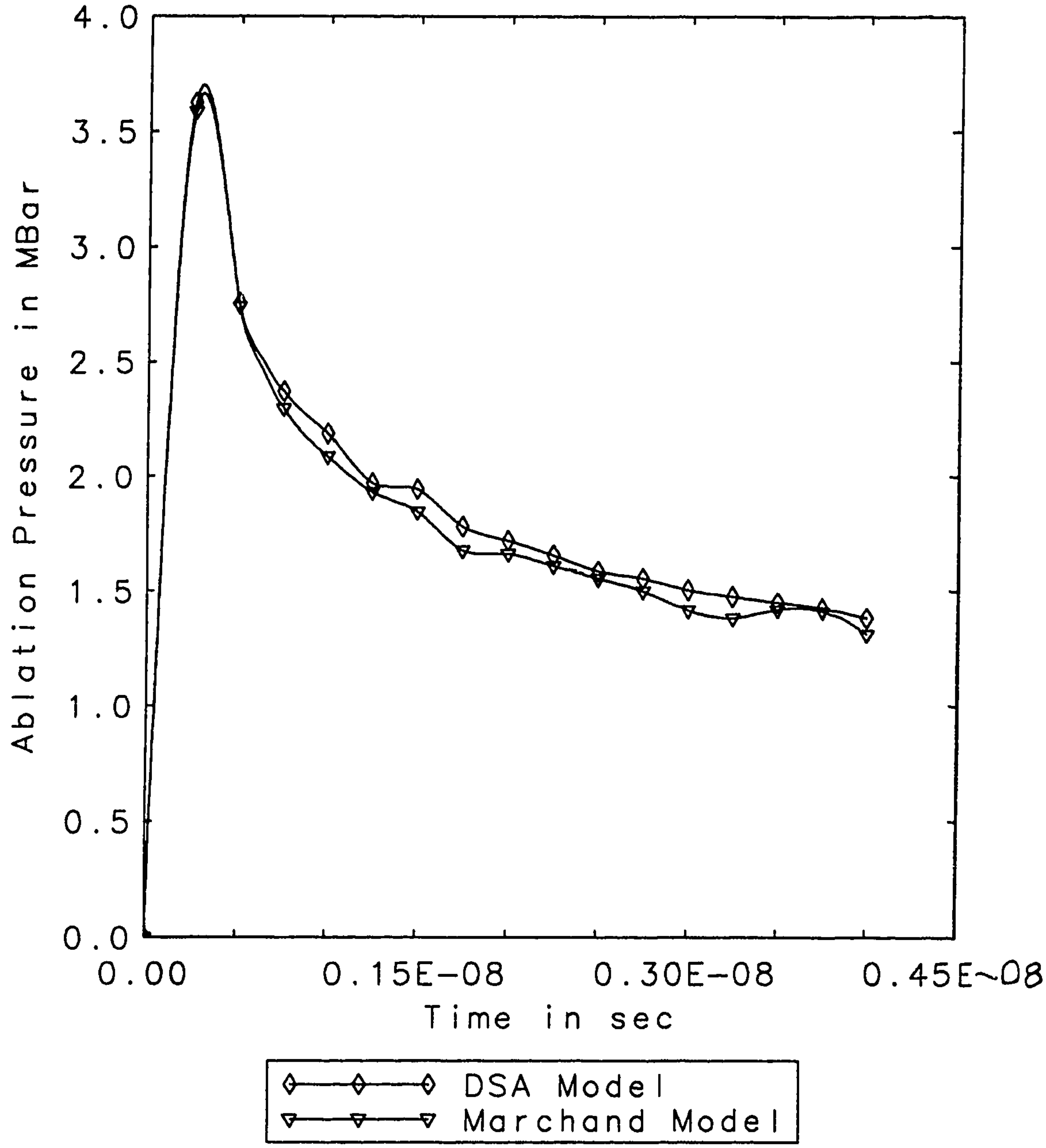

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Ablation Pressure in MBar
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec
DSA Model
Marchand Model

Figure 5.5 a

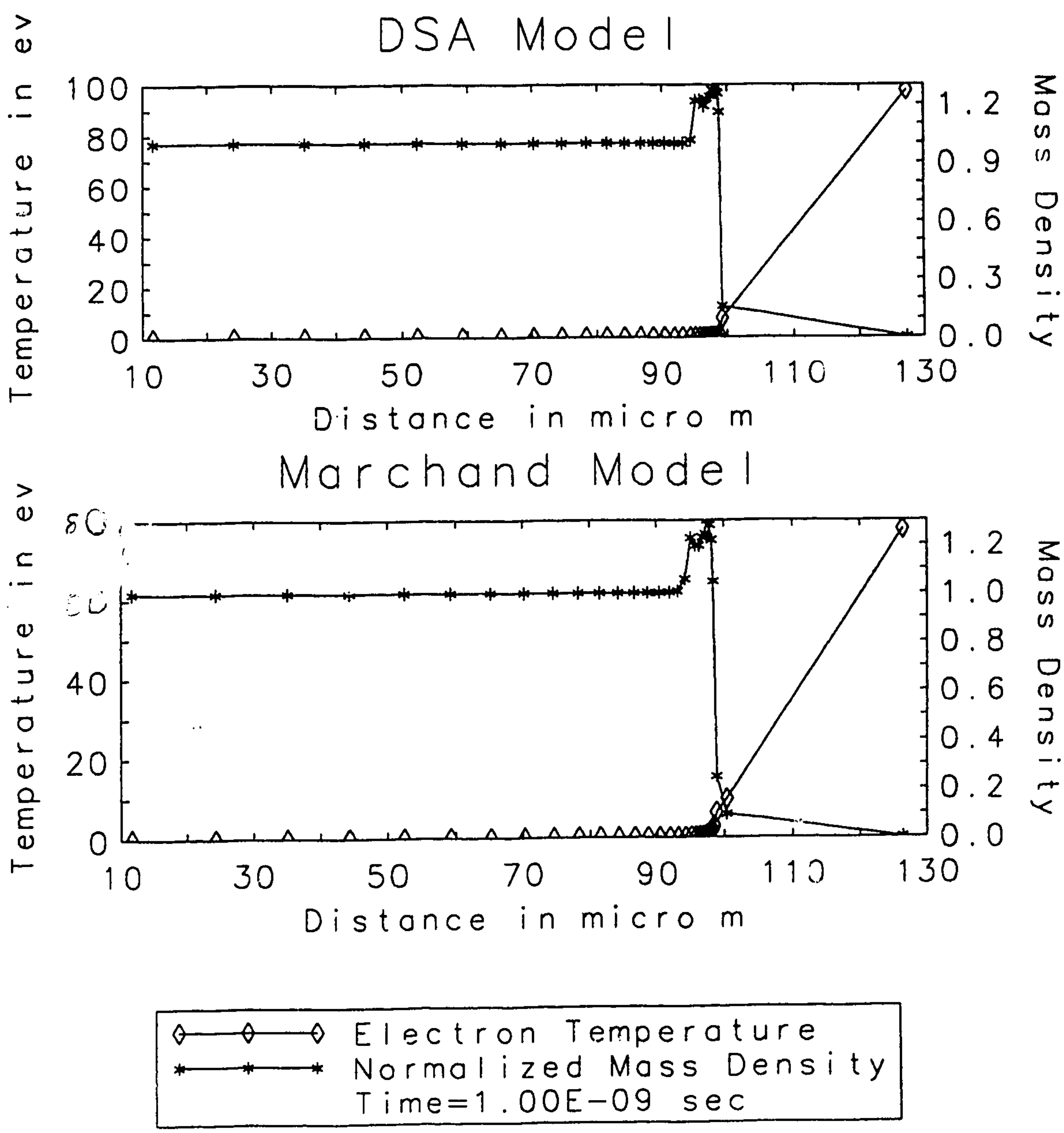

DSA Model
Temperature in ev
100
80
60
40
20
0
10
30
50
70
90
110
130
Distance in micro m
Mass Density
1.2
0.9
0.6
0.3
0.0
Marchand Model
Temperature in ev
40
20
0
10
30
50
70
90
130
Distance in micro m
Mass Density
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Electron Temperature
Normalized Mass Density
Time=1.00E-09 sec

Figure 5.5 b

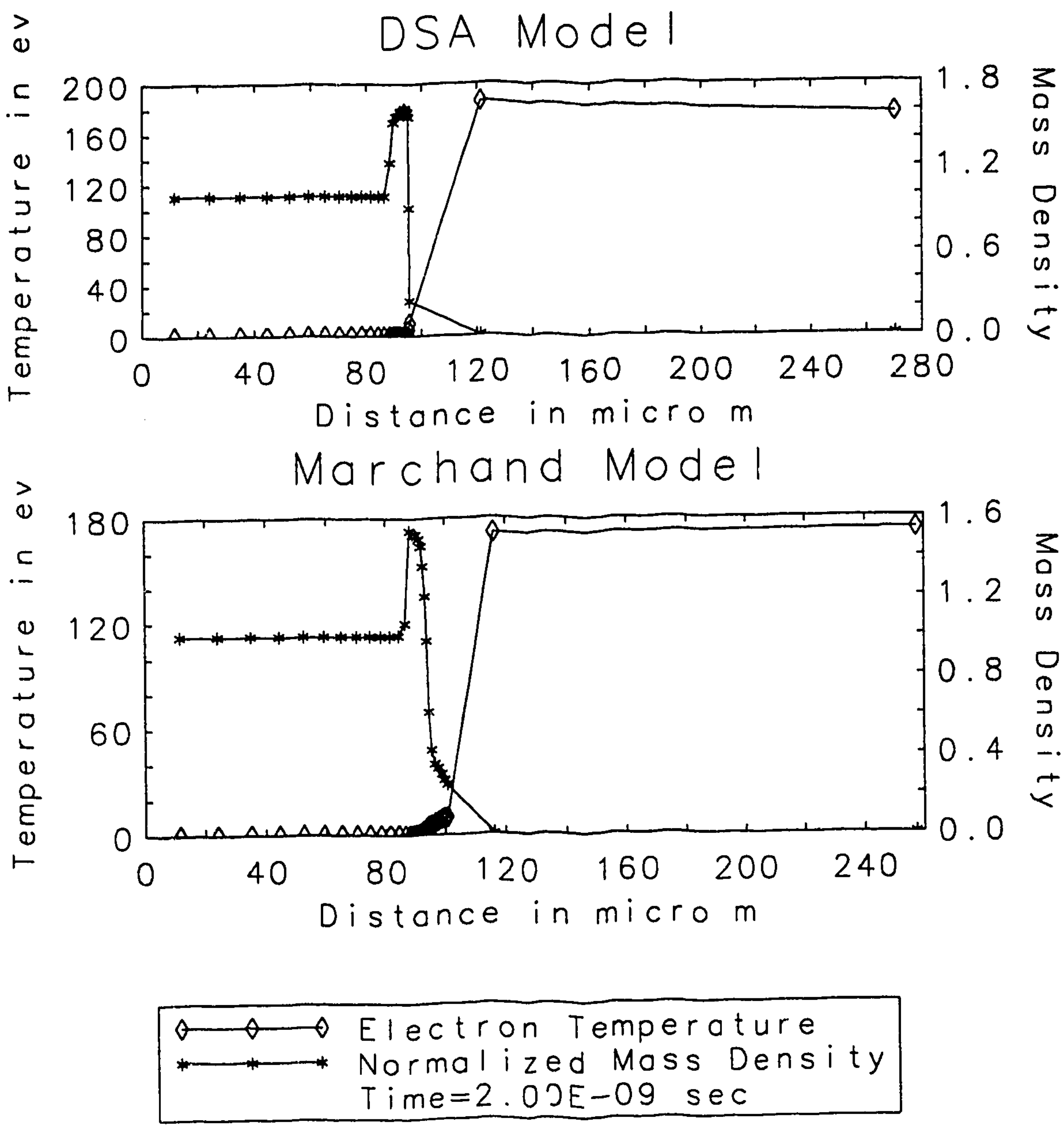

DSA Model
Temperature in ev
200
160
120
80
40
0
1.8
1.2
0.6
0.0
Mass Density
0
40
80
120
160
200
240
280
Distance in micro m
Marchand Model
Temperature in ev
180
120
60
0
1.6
1.2
0.8
0.4
0.0
Mass Density
0
40
80
120
160
200
240
Distance in micro m
Electron Temperature
Normalized Mass Density
Time=2.00E-09 sec

Figure 5.5 c

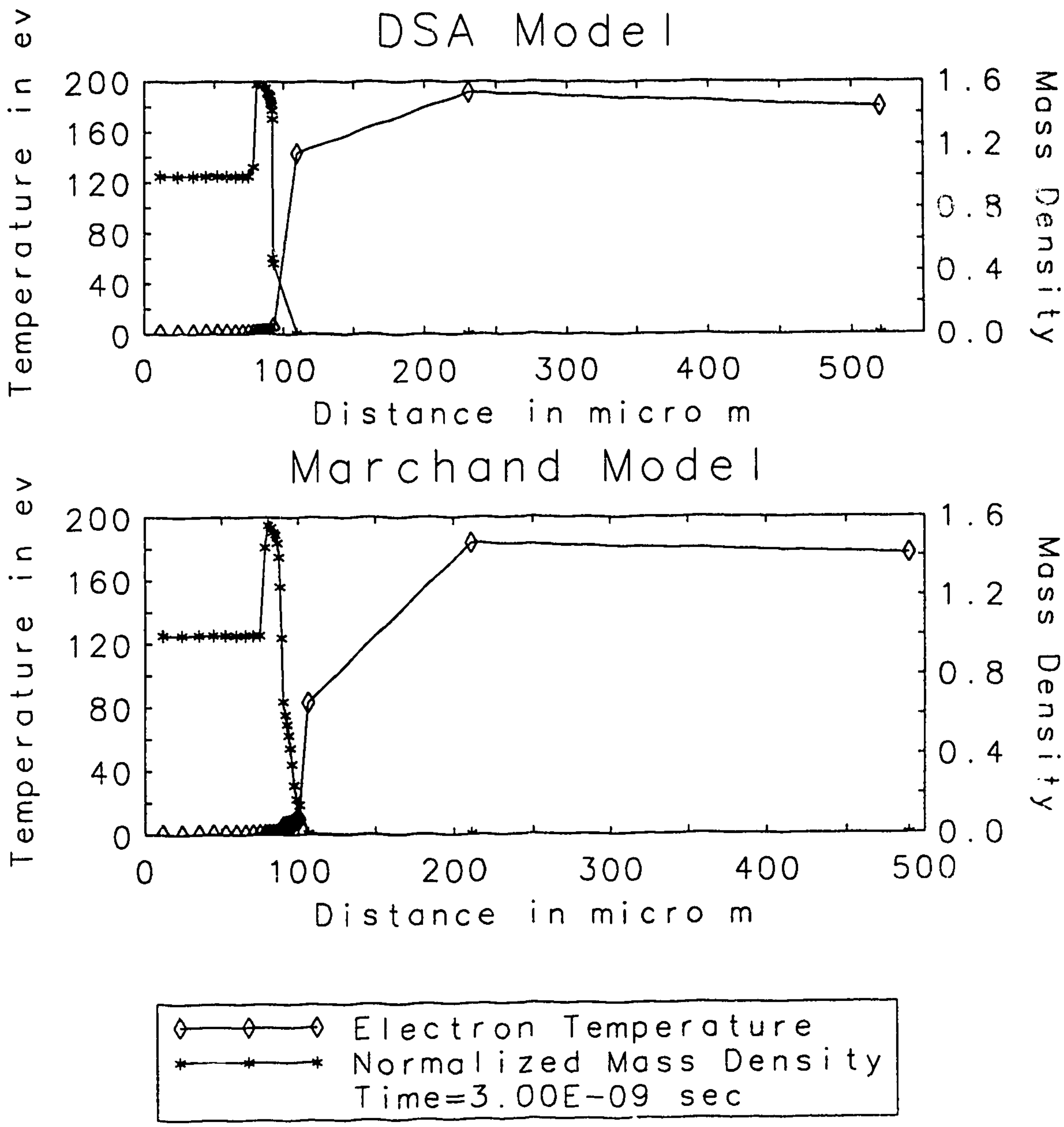

DSA Model
Temperature in ev
200
160
120
80
40
0
Mass Density
1.6
1.2
0.8
0.4
0.0
0
100
200
300
400
500
Distance in micro m
Marchand Model
Temperature in ev
200
160
120
80
40
0
Mass Density
1.6
1.2
0.8
0.4
0.0
0
100
200
300
400
500
Distance in micro m
Electron Temperature
Normalized Mass Density
Time=3.00E-09 sec

Figure 5.6 a

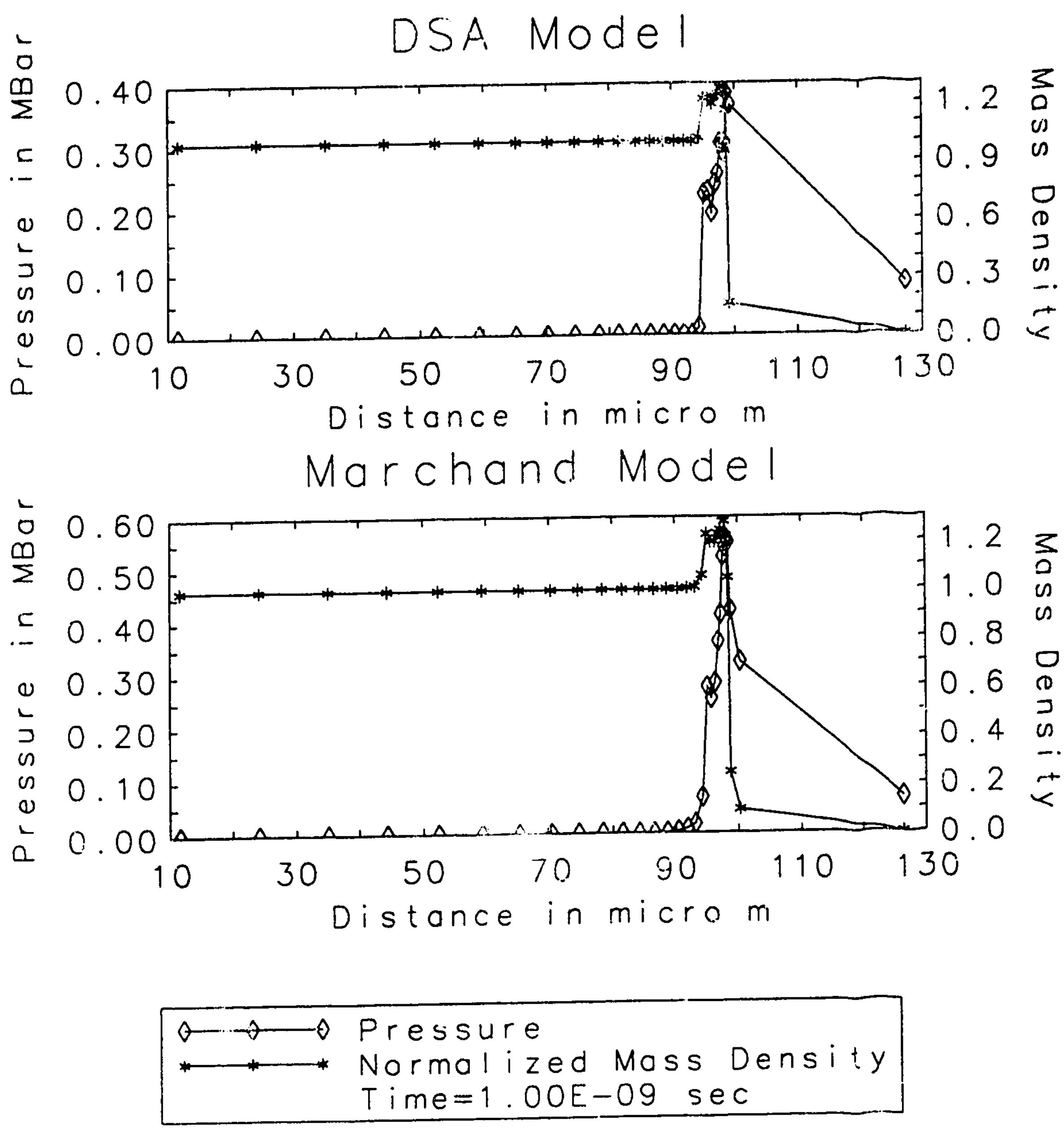

DSA Model
Pressure in MBar
0.40
0.30
0.20
0.10
0.00
10
30
50
70
90
110
130
Distance in micro m
Mass Density
1.2
0.9
0.6
0.3
0.0
Marchand Model
Pressure in MBar
0.60
0.50
0.40
0.30
0.20
0.10
0.00
Mass Density
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Pressure
Normalized Mass Density
Time=1.00E-09 sec

Figure 5.6 b

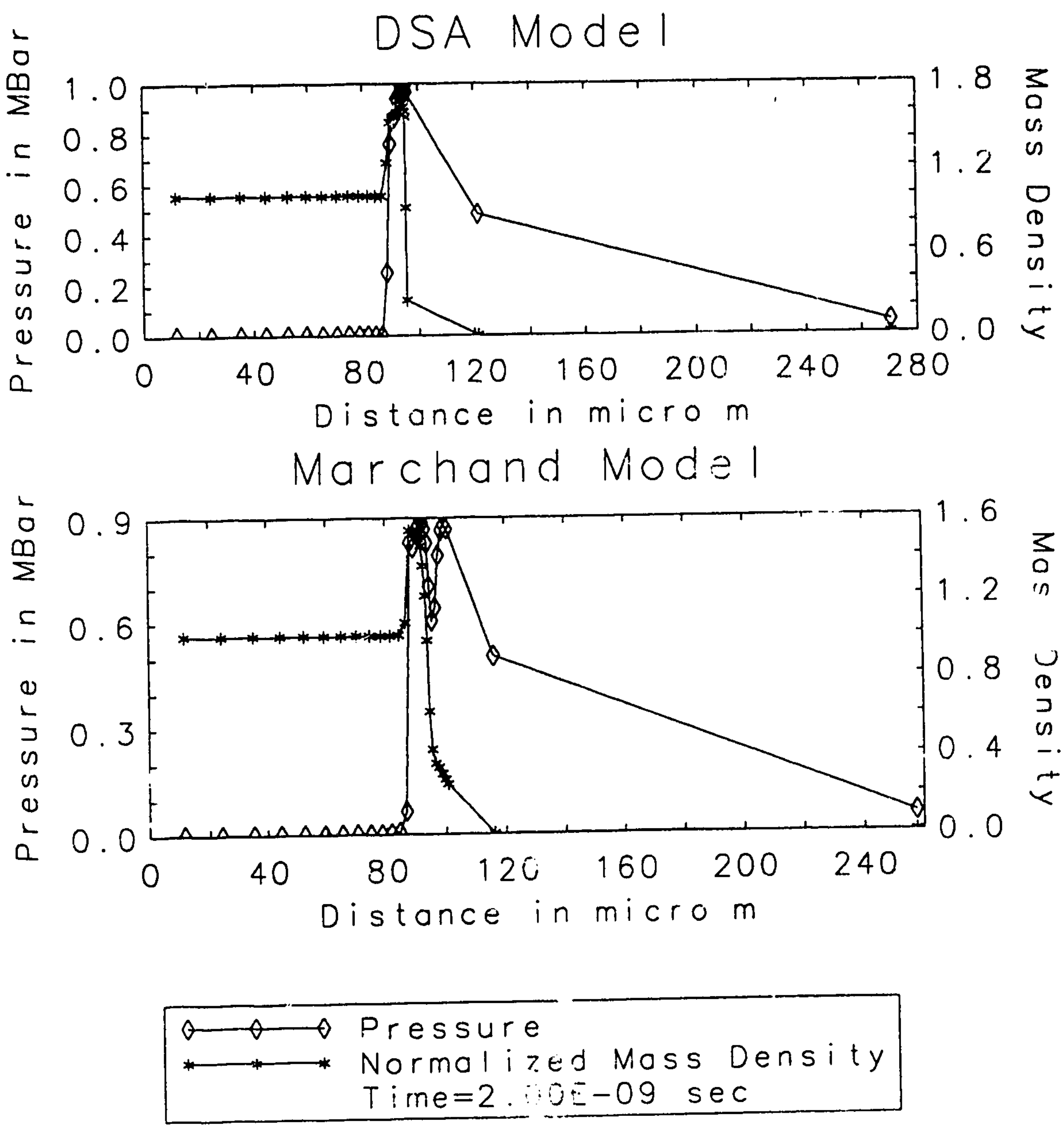

DSA Model
Pressure in MBar
1.0
0.8
0.6
0.4
0.2
0.0
1.8
1.2
0.6
0.0
Mass Density
0
40
80
120
160
200
240
280
Distance in micro m
Marchand Model
Pressure in MBar
0.9
0.6
0.3
0.0
1.6
1.2
0.8
0.4
0.0
Mass Density
0
40
80
120
160
200
240
Distance in micro m
Pressure
Normalized Mass Density
Time=2.00E-09 sec

Figure 5.6 c

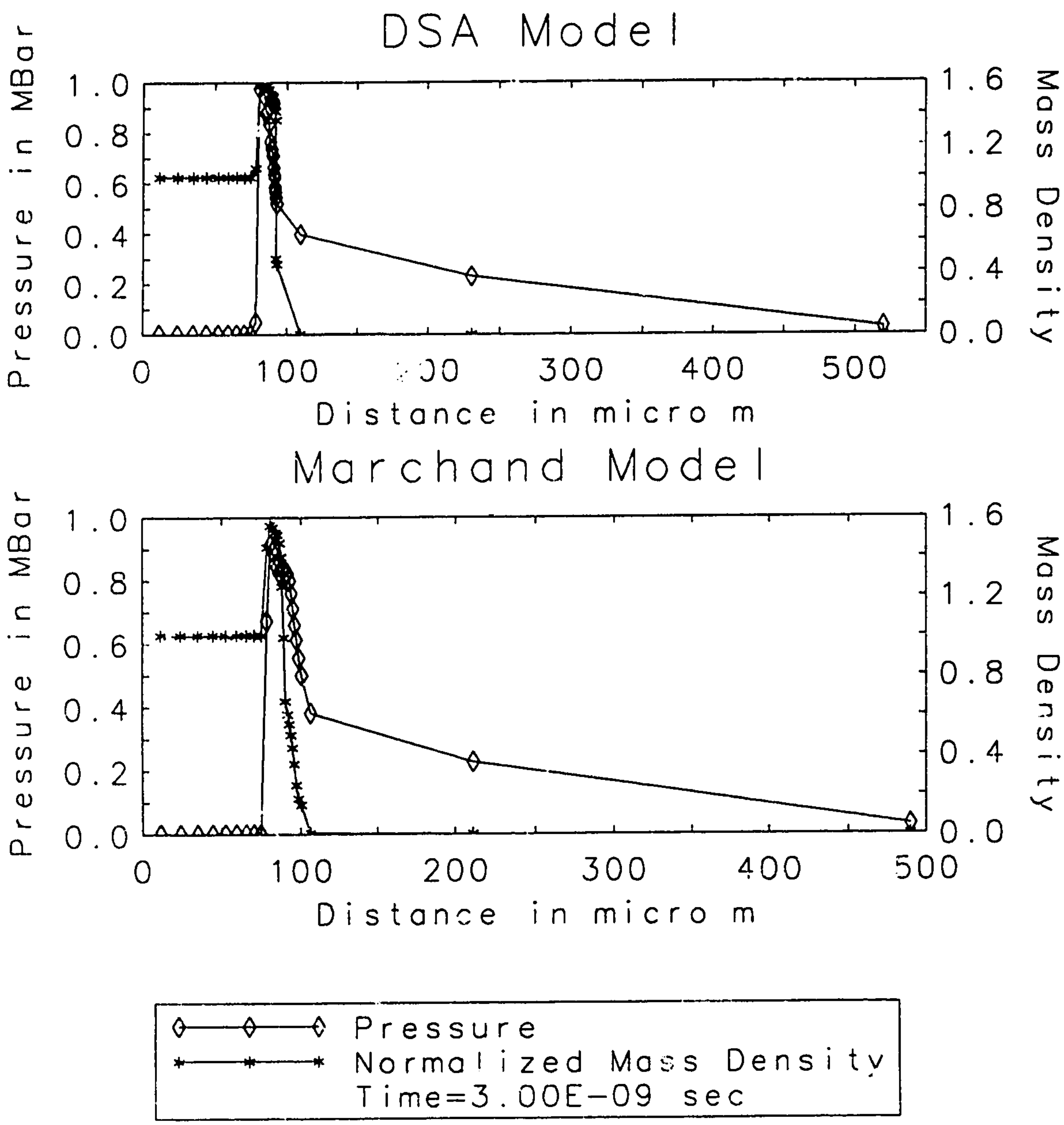
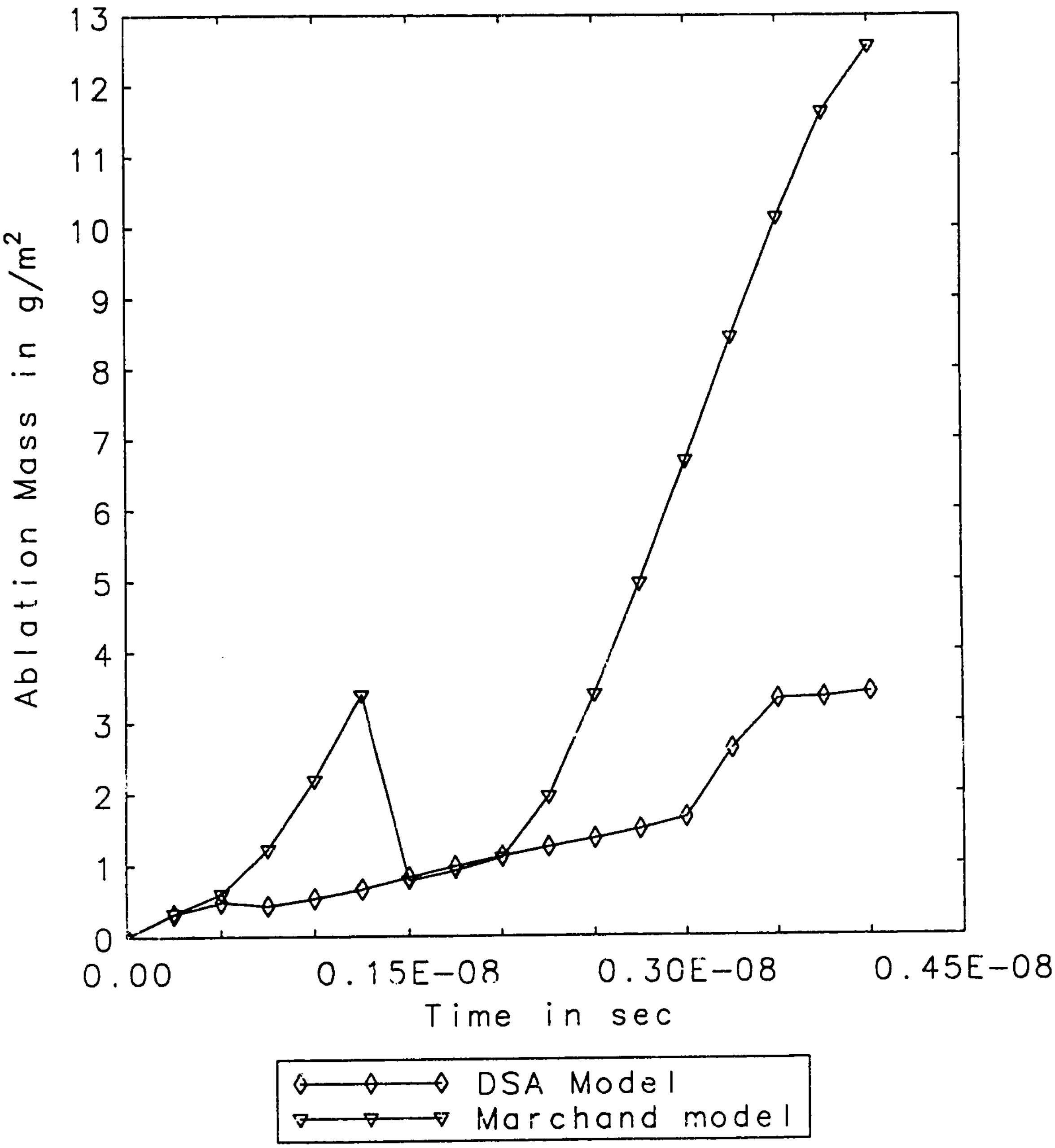

DSA Model
Pressure in MBar
Mass Density
1.0
0.8
0.6
0.4
0.2
0.0
1.6
1.2
0.8
0.4
0.0
0
100
300
400
500
Distance in micro m
Marchand Model
Pressure in MBar
Mass Density
1.0
0.8
0.6
0.4
0.2
0.0
1.6
1.2
0.8
0.4
0.0
0
100
200
300
400
500
Distance in micro m
Pressure
Normalized Mass Density
Time=3.00E-09 sec

Figure 5.7



13
12
11
10
9
8
7
6
5
4
3
2
1
0
Ablation Mass in g/m²
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec
DSA Model
Marchand model

Figure 5.8

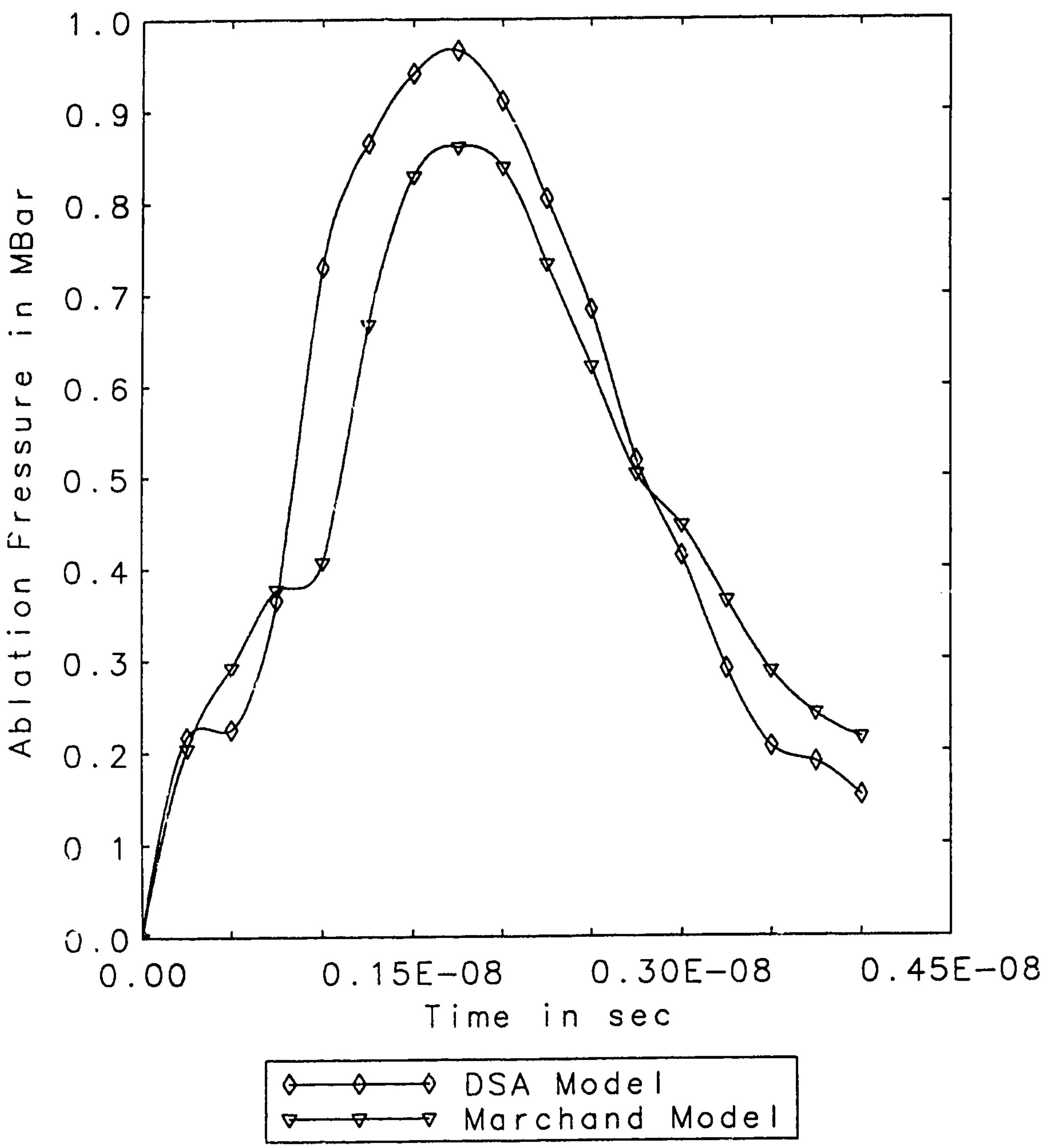
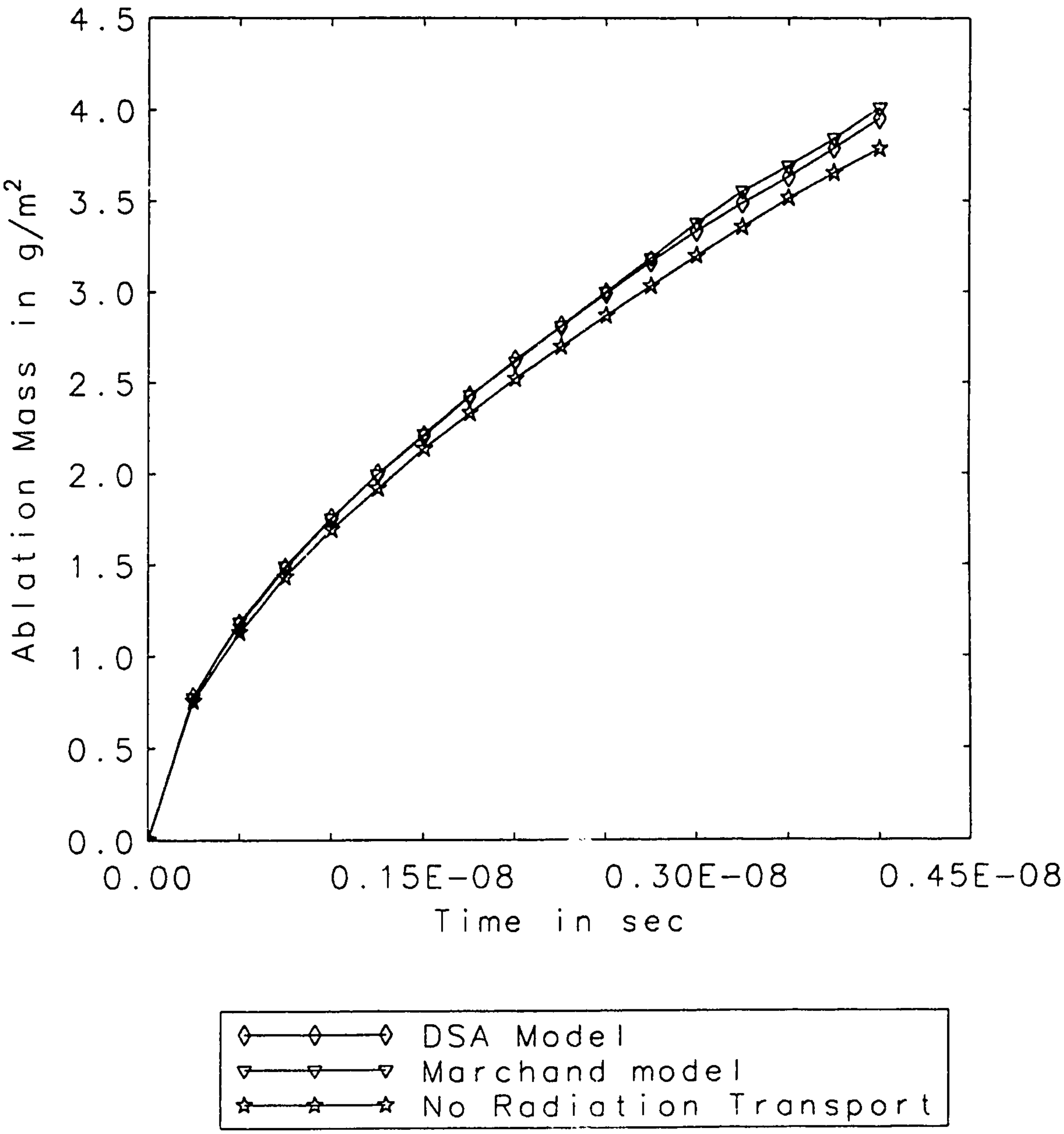

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec
Ablation Pressure in MBar
DSA Model
Marchand Model

Figure 5.9 a



4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Ablation Mass in g/m²
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec
DSA Model
Marchand model
No Radiation Transport

Figure 5.9 b

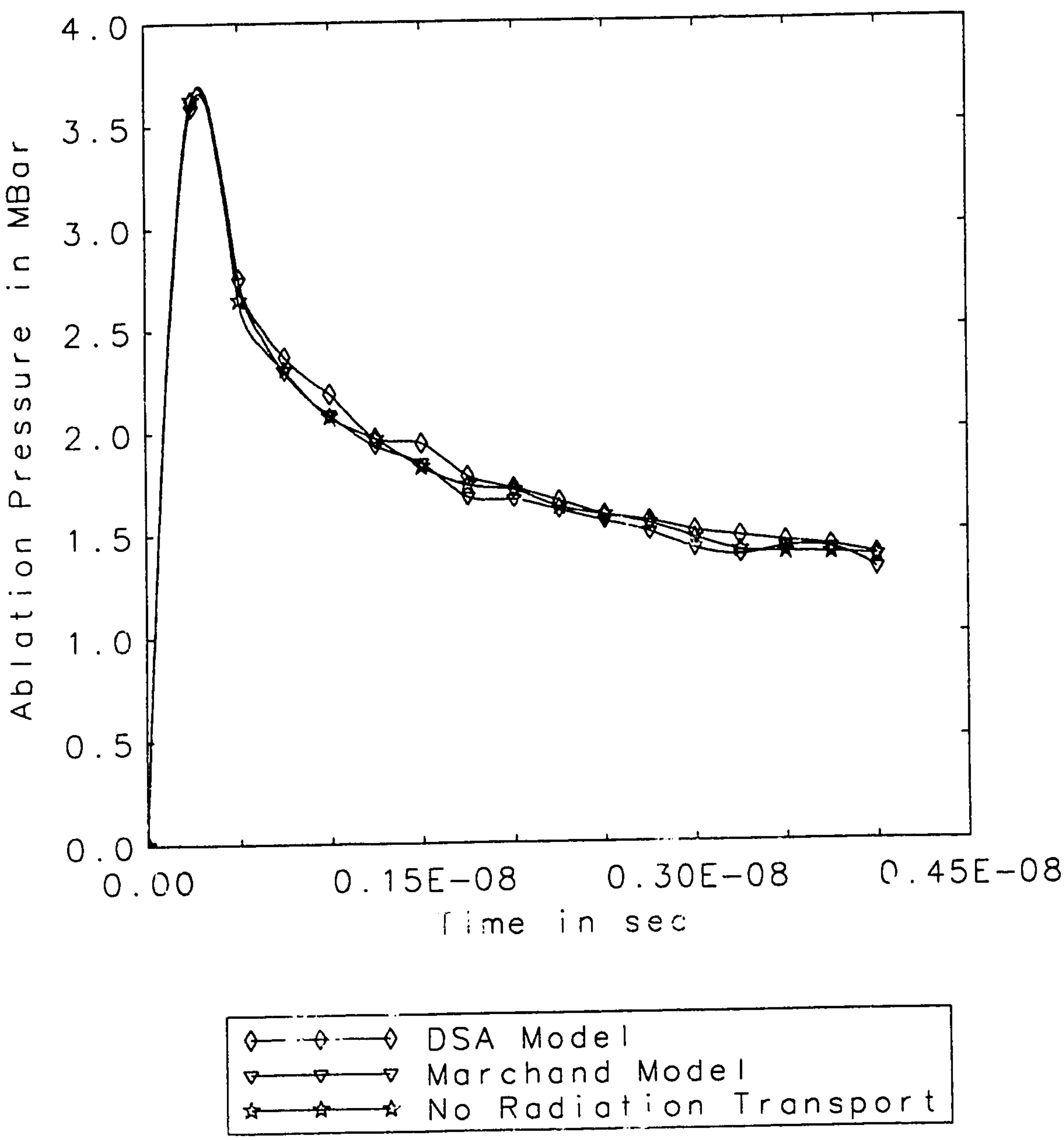

Ablation Pressure in MBar
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec
DSA Model
Marchand Model
No Radiation Transport

Figure 5.9 c

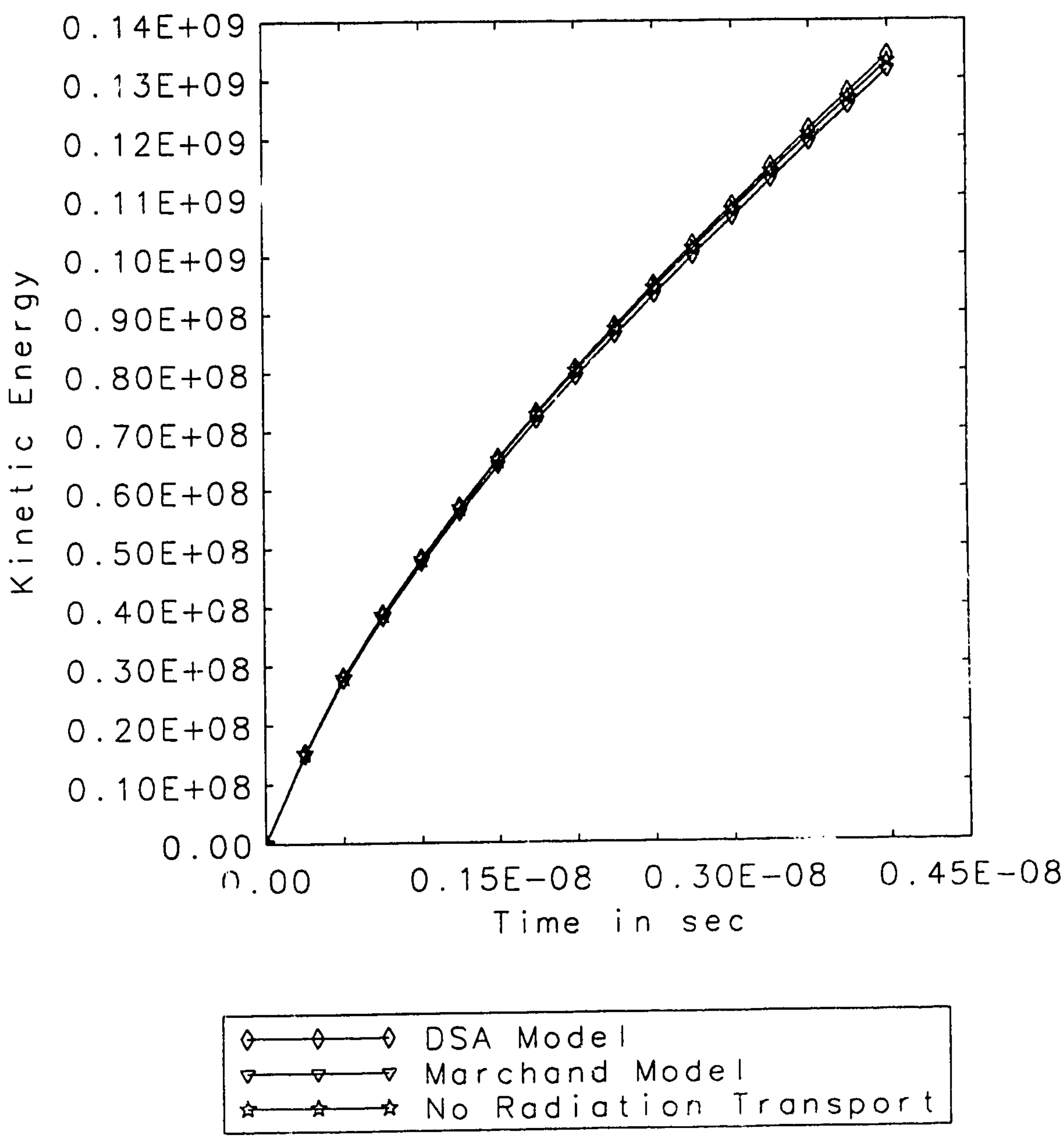

0.14E+09
0.13E+09
0.12E+09
0.11E+09
0.10E+09
0.90E+08
0.80E+08
0.70E+08
0.60E+08
0.50E+08
0.40E+08
0.30E+08
0.20E+08
0.10E+08
0.00
Kinetic Energy
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec
DSA Model
Marchand Model
No Radiation Transport

Figure 5.9 d

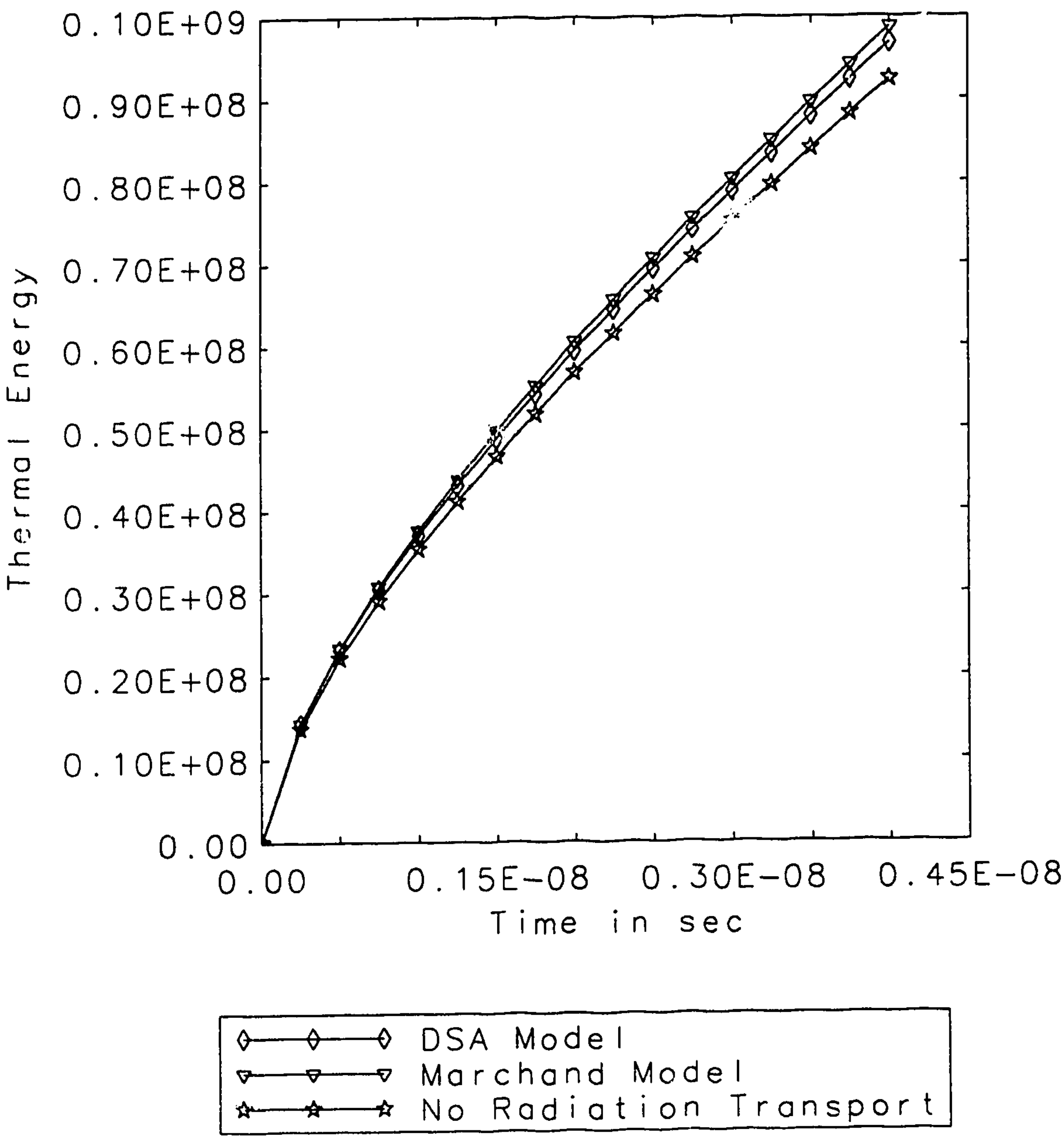
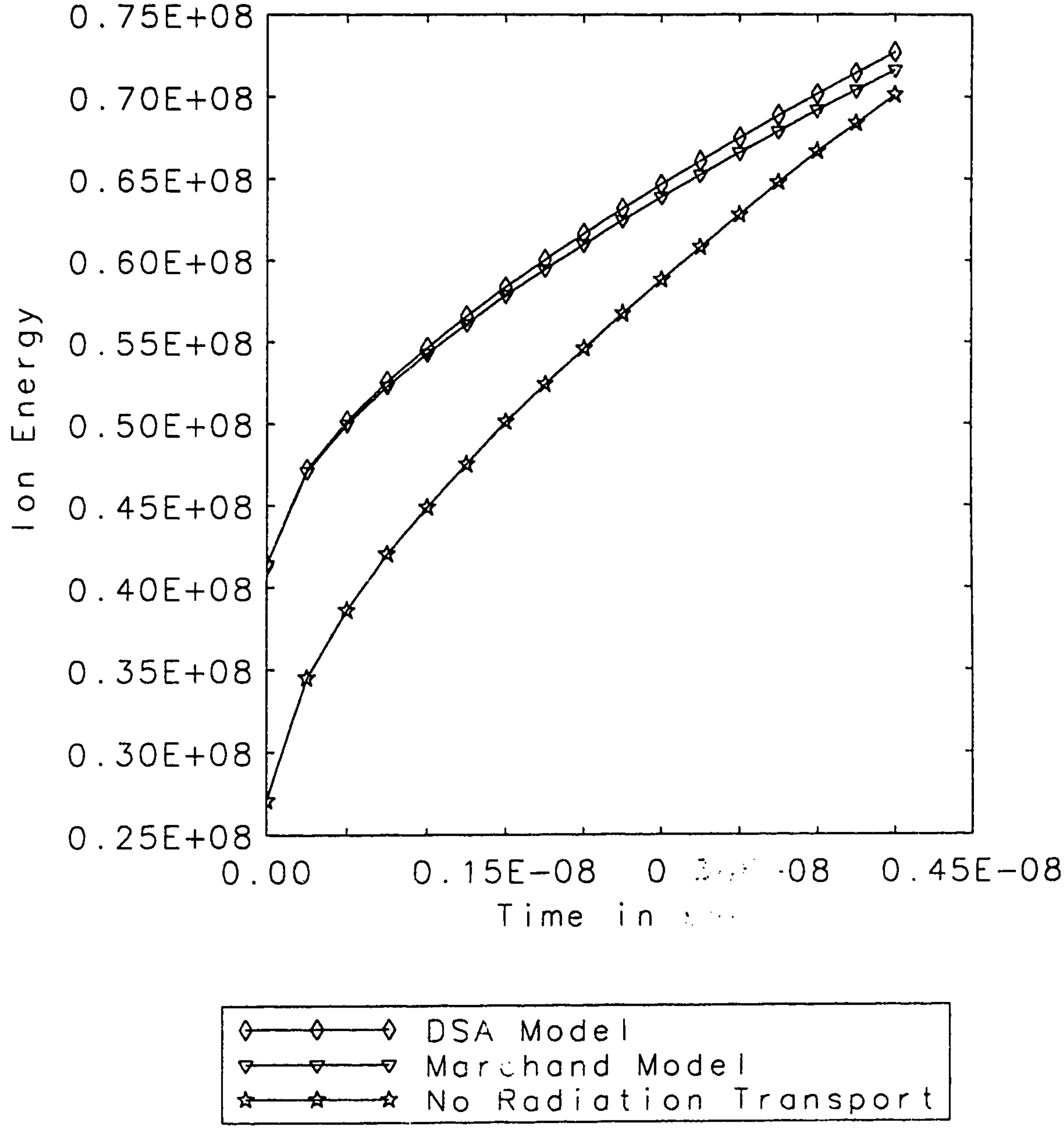
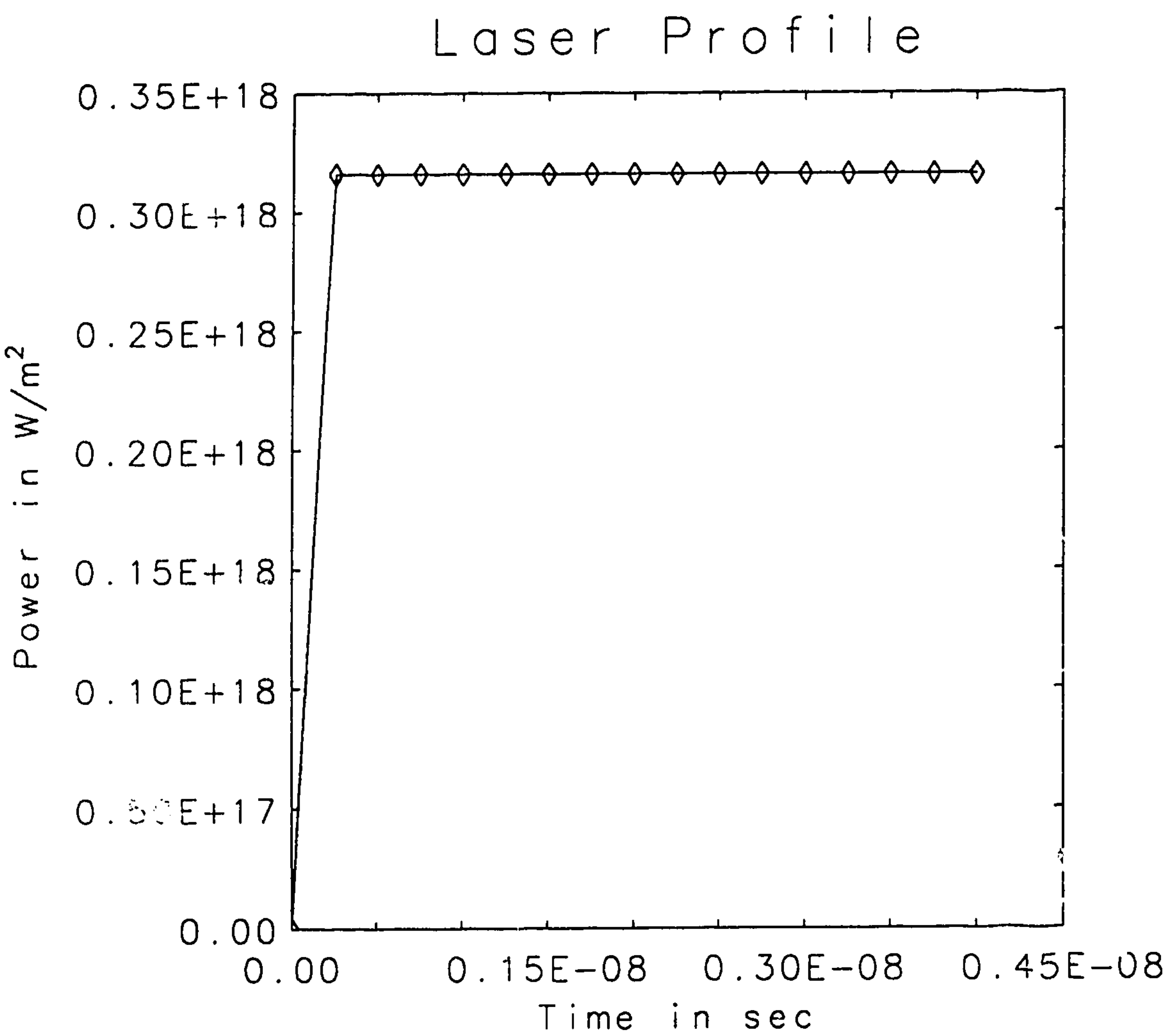
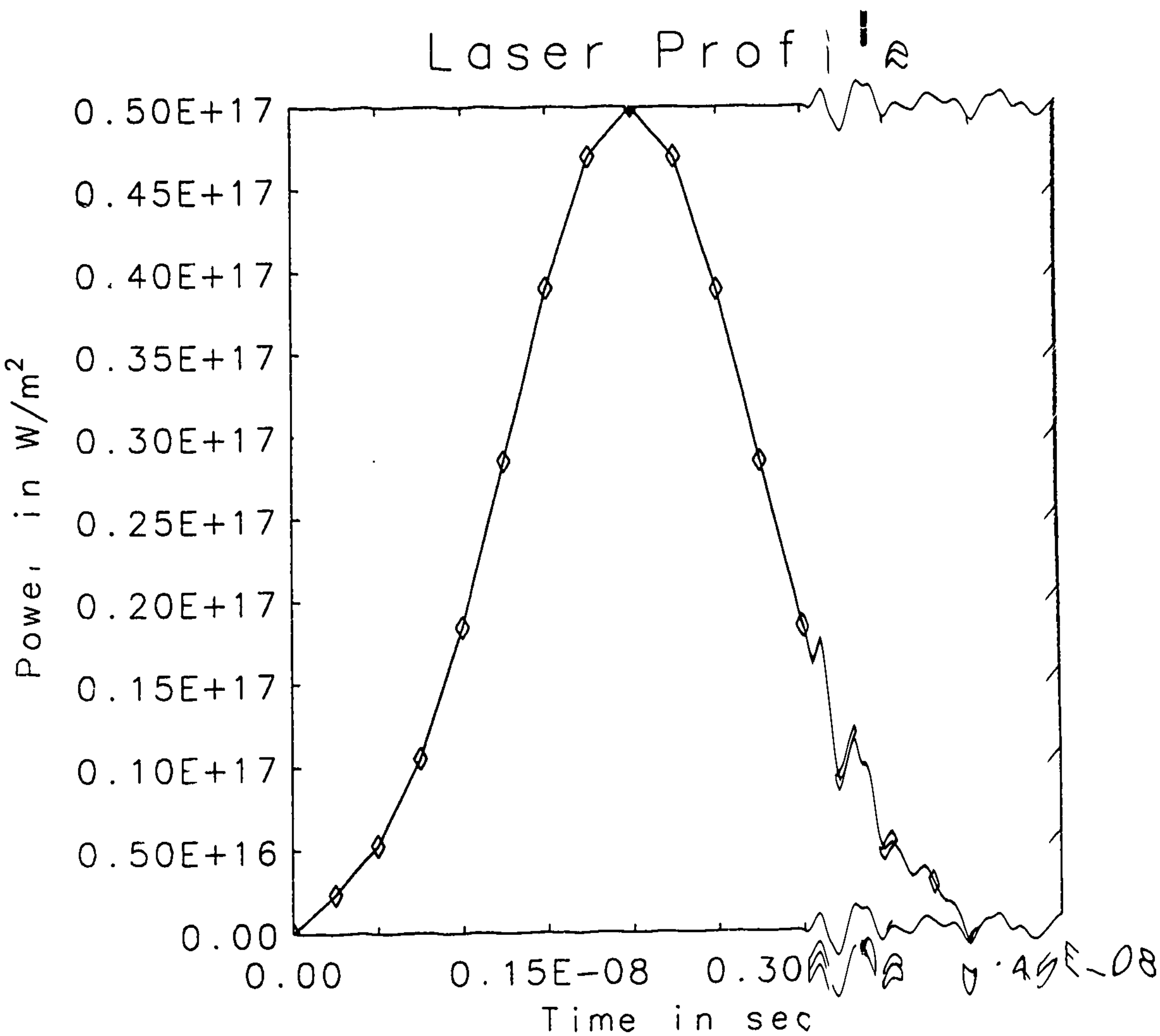

0.10E+09
0.90E+08
0.80E+08
0.70E+08
0.60E+08
0.50E+08
0.40E+08
0.30E+08
0.20E+08
0.10E+08
0.00
Thermal Energy
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec
DSA Model
Marchand Model
No Radiation Transport

Figure 5.9 e


0.75E+08
0.70E+08
0.65E+08
0.60E+08
0.55E+08
0.50E+08
0.45E+08
0.40E+08
0.35E+08
0.30E+08
0.25E+08
Ion Energy
0.00
0.15E-08
0.45E-08
Time in
DSA Model
Marchand Model
No Radiation Transport



Laser Profile
0.35E+18
0.30E+18
0.25E+18
0.20E+18
0.15E+18
0.10E+18
0.00
Power in W/m2
0.00
0.15E-08
0.30E-08
0.45E-08
Time in sec


Laser Profile
Power, in W/m²
0.50E+17
0.45E+17
0.40E+17
0.35E+17
0.30E+17
0.25E+17
0.20E+17
0.15E+17
0.10E+17
0.50E+16
0.00
0.00
0.15E−08
Time in sec

CONCLUDING REMARKS:

The objective of the research was to solve the radiative transfer equation in one dimension by using a method that can be extended to higher dimensions or curvilinear geometries. Another objective of the research was to corroborate Marchand's simplified 2-cell model.

The Diffusion Synthetic Acceleration (DSA) method is chosen to solve the radiative transfer equation. This method has been successfully used to solve neutronic transport problems in higher dimensions, hence it is believed that similar extension should be possible for radiative problems. The Marshak Wave bench mark is used to test the validity of the code. It is observed that Diffusion synthetic Acceleration is very efficient compared to other methods such as SI; however, it becomes unstable at large time steps.

A comparison of DSA and Marchand's model shows that these two models give similar results when a trapezoid laser pulses are used. However, the results are not comparable when a gaussian laser beams are used. Analysis of the results shows that Marchand's model appears to give erroneous results, the ablated mass obtained with Marchand's model is unrealistic, too much mass is being ablated. Therefore, Marchand's model requires further investigation if the above error in the ablation mass is to be rectified. Marchand's model on the other hand is computationally more efficient than DSA, but its disadvantage is its applicability in only one dimensional slab geometry problems. Further, in Marchand's model the angular dependence of the intensity is ignored, which is more crucial in higher dimensions. Therefore, Marchand's model is good for nongaussian laser beams in one dimensional slab geometry, while the DSA model can be extended to higher dimensions including curvilinear geometries, though the equations involved become more complicated.

It is observed that the effects of radiation transport is more significant on the ablation mass as compared to the ablation pressure, as predicted by Mora. Inclusion of radiation transport results in an increase in thermal, kinetic and ion energy. This is due to the fact that when radiation transport is included energy is transported as well rather than being just lost. From the simulations it appears that the increase in energy is about 7 to 10 % for kinetic, thermal and ion energies. More simulations are required to estimate the x-ray conversion efficiency and the effects of radiation transport when different wavelengths or higher intensities are used.

BIBLIOGRAPHY

1/1 A. Brueckner and Sieba Jorna; Review of Modern Physics Vol. 46, P.325, Apr 1974.
1/2 D. Duston, R.W. Clark and J. Davis; Physical Review A Vol. 31, P. 3220, May 1985.
1/3 H. Motz, Physics of Laser Fusion
1/4 R.C. Malone, R.L. McCrory and R.L. Morse; Physical Review Letter, Vol. 34, P. 721, 1975.
1/5 G.S. Fraley and R.J. Mason; Physical Review Letter, Vol. 35, P. 520, 1975.
1/6 D.W. Forslund et al; Physical Review A Vol. 11, P. 679, 1975.
1/7 E.A. McLean et al; Physical Review Letters, Vol. 45, P. 1246, 1980.
1/8 D. Duston, R.W. Clark, J. Davis and J.P. Apruzese; Physical Review Vol. A27, P. 1441, 1983.
1/9 B. Yaakobi, J. Delettrez, L.M. Goldman, R.L. McCroy, W. Seka and J.M. Soures; Optics communications Vol. 41, P. 355, May 1982.
1/10 R. Sigel et al; Physics Fluids, Vol. B2, P. 199, 1989.

2/1 G. C. Pomraning; Equations of Radiation Hydrodynamics.

3/1 Same as 2/1.
3/2 J.A. Fleck and J.D. Cummings; Journal of Computational Physics, Vol. 08, P. 313, 1971.
3/3 H.J. Kopp; Nuclear Science and Engineering, Vol. 17, P. 65, 1964.
3/4 R.E. Alcouffe; Nuclear Sience and Engineering, Vol. 64, P. 344, 1987.
3/5 Multiple Time Scales, Ed. J.U. Brackbill and B.I. Cohen P.74.
3/6 E.W. Larsen; Nuclear Science and Engineering, Vol. 100, P. 255, 1988.
3/7 M.L. Adams and W.R. Martin; Nuclear Science and Engineering, Vol. 100, P. 177, 1988.
3/8 E.D. Larsen, Personal correspondence.
3/9 Same as 2/1.
3/10 E.W. Larsen, Nuclear Science and Engineering, Vol. 82, P. 47, 1981.
3/11 Transport and Reactor Report, Ed. R.D. O' Dell and R.E. Alcouffe, P. 29, Aug 1982.
3/12 J.E. Morel, Nuclear Science and Engineering, Vol. 82, P. 34, 1982.
3/13 J.E. Morel, E.W. Larsen and M.K. Matzen; Journal of Quan. Spectrosc. Radiat. Transfer. Vol. 34, P. 243, 1985.

4/1 B.A. Clark; Nuclear Science and Engineering, Vol. 90, P. 171, 1984.
4/2 R.E. Marshak; Physics of Fluids, Vol. 01, P.24, 1958.
4/3 Same as 3/5.
4/4 G. Velarde et al; Laser and Particle Beams, Vol. 7, Part 02, 305, 1989.
4/5 Same as 2/1.

5/1 Notes on MEDUSA from Dr. C.E. Capjack.
5/2 Notes on atomic tables from Dr. C.E. Capjack.
5/3 P. Mora; Physics Fluids, Vol. 25, P. 1051, 1982.
5/4 R. Marchand, R. Fedosejevs, C.E. Capjack and Y. T. Lee; Laser and Partoicle Beams, Vol. 6, Part 2, P.183, 1988.
A/1 same as 3/4.
A/2 Personal correspodance with M.L. Adams.

B/1 K. Audenaerde et al; Journal of Computational Physics Vol. 34, P. 510, 1987.
B/2 R. Marchand, A. Birnboim, C.E. Capjack and D. Salzmann; Canadian Journal of Physics, Vol. 67, P.155, 1989.
B/3 Same as 5/4.

C/1 Same as 5/2.

APPENDIX

A. NUMERICAL ASPECTS OF DSA

A.1 FOURIER ANALYSIS OF THE RADIATIVE TRANSFER EQUATION:

We start our Fourier stability analysis of the radiative transfer equation when it is discretized in time. As shown earlier in Chapter Three (3.3.12), this is given as

$$\mu\frac{\partial I}{\partial x} + (\sigma+\tau)I = \frac{\chi\eta}{2}\int_0^\infty \sigma I_0\, d\nu + Q$$
$$where \quad I_0 = \int_{-1}^{1} I d\mu$$

(A.1.01)

Due to the coupling in frequency on the right hand side, we solve the above equation iteratively. The iteration strategy that is used to solve the radiative transfer equation is known as source iteration. Source iteration (SI) of (A.1.01) is expressed as

$$\mu\frac{\partial I^{l+\frac{1}{2}}}{\partial x} + (\sigma+\tau)I^{l+\frac{1}{2}} = \frac{\chi\eta}{2}\int_0^\infty \sigma I_o^l\, d\nu + Q$$
$$I_0^{l+1} = I_0^{l+\frac{1}{2}} = \int_{-1}^{1} I^{l+\frac{1}{2}}\, d\mu$$

(A.1.02a,b)

where the . . 1/2 index corresponds to the most recent intensity obtained by solving (A.1.02a) with an initial intensity of I_o^l. For the next iteration, the source term (A.1.02a) is updated by using (A.1.02b). To study the convergence properties of (A.1.02), we define

$$D^{l+\frac{1}{2}} = I^{l+\frac{1}{2}} - I^{l-\frac{1}{2}} = \omega^l D\, e^{i\lambda x}$$
$$D_0^l = I_0^l - I_0^{l-1} = \omega^l D_0\, e^{i\lambda x}$$

where ω is the eigen-value and D_o is subject to the condition

$$\int_0^\infty \sigma\, D_o\, d\nu = 1$$

(A.1.03a,b,c)

We take σ and χ to be spatially independent for simplicity in the Fourier stability analysis. The following steps are taken to obtain the radius of convergence:

i. Write (A.1.02a) for l-1 iteration.

ii. Subtract the above equation from equations (A.1.02a) and substitute (A.1.03a,b) wherever possible.

iii. Simplify the above result using equation (A.1.03c).

iv. Repeat the first two steps for equation (A.1.02b).

Since the steps are relatively simple, only the final results are given.

$$(iii) \rightarrow (i\mu\lambda + \sigma + \tau)\, D = \frac{\chi\eta}{2}$$
$$(iv) \rightarrow \omega D_0 = \int_{-1}^{1} D\, d\mu$$

(A.1.04a,b)

The radius of convergence is obtained by manipulating (A.1.04a,b) and using (A.1.03c). The steps that are performed are as follows:

i. Multiply (A.1.04b) by σ and integrate over entire frequency spectrum

ii. Substitute (A.1.04a) into the above equation

iii. Simplify the final result by using A.1.03c

The final result after angular integration is

$$\omega = \frac{1}{2}\int_0^\infty \sigma\left[\int_{-1}^{1} \chi\eta\, \frac{d\mu}{(i\mu\lambda+\sigma+\tau)}\right] d\nu = \int_0^\infty \chi\eta \frac{\sigma}{\lambda} \tan^{-1}\frac{\lambda}{(\sigma+\tau)}\, d\nu$$

(A.1.05)

The spectral radius can be viewed as the rate at which both D and D_0 go to zero, which is just the error reduction per iteration. In practice it is taken to be the largest eigen-value. In the source iteration method this occurs when $\lambda = 0$ and $\tau = 0$. When this is true the spectral radius is given by

$$spr = \int_0^\infty \chi d\nu = \eta$$

where $\eta \rightarrow 1$ when $\alpha\tau \rightarrow 0$. This is true when $\tau \rightarrow 0$ or $\sigma > 1$.

This result shows that the SI method can require an arbitrarily large number of iterations before a converged solution is obtained.

A.2 FOURIER ANALYSIS OF THE RADIATIVE TRANSFER EQUATION USING THE DIFFUSION EQUATION:

A Fourier stability analysis of the transfer equation when it is accelerated using the diffusion equation involves equations (3.3.12;3.3.04;3.5.02) presented earlier

$$\mu \frac{\partial I^{l+\frac{1}{2}}}{\partial x} + (\sigma+\tau) I^{l+\frac{1}{2}} = \frac{\chi\eta}{2} \int_0^\infty \sigma I_0^l \, d\nu + Q$$

$$I_0^{l+\frac{1}{2}} = \int_{-1}^{1} I^{l+\frac{1}{2}}$$

$$-\frac{1}{3(\sigma+\tau)} \frac{\partial^2 F_0^{l+1}}{\partial x^2} + (\sigma+\tau) F_0^{l+1} = \chi\eta \int_0^\infty \sigma F_0^{l+\frac{1}{2}} d\nu + \chi\eta \int_o^\infty \sigma (I_o^{l+\frac{1}{2}} - I_o^l) d\nu$$

(A.2.01;02;03)

Let us define

$$D^{l+\frac{1}{2}} = I^{l+\frac{1}{2}} - I^{l+\frac{1}{2}} = \omega^l D e^{\iota x \lambda}$$

$$D_o^l = I_o^l - I_o^{l-1} = \omega^l D_o e^{\iota x \lambda}$$

$$f_0^{l+1} = F_0^{1+1} - F_0^l = \omega^l f_0 \, e^{\iota \lambda x}$$

$$\text{where} \quad F_0^{l+1} = I_0^{l+1} - I_0^{l+1/2}$$

(A.2.04a,b,c,d)

A Fourier analysis of the transfer equation when accelerated with the diffusion equation involves the following steps:

i. Rewrite (A.2.01;02;03) for steps l-1, and subtract them from (A.2.01;02;03) respectively. After substituting the first equality of (A.2.04), we obtain

$$\mu \frac{\partial D^{l+\frac{1}{2}}}{\partial x} + (\sigma+\tau) D^{l+\frac{1}{2}} = \frac{\chi\eta}{2} \int_0^\infty \sigma D_0^l \, d\nu$$

$$D_0^{l+\frac{1}{2}} = \int_{-1}^{1} D^{l+\frac{1}{2}} \, d\mu$$

$$-\frac{1}{3(\sigma+\tau)} \frac{\partial^2 f_0^{l+1}}{\partial x^2} + (\sigma+\tau) f_0^{l+1} = \chi\eta \int_0^\infty \sigma f_0^{l+1}$$

$$+ \chi\eta \int_0^\infty \left[\sigma \left(\int_{-1}^{l} D^{l+\frac{1}{2}} d\mu \right) - D_0^l \right] d\nu$$

(A.2.06a,b,c)

ii. Substitute the second equality of (A.2.04) (Fourier decomposition) into (A.2.06a), to obtain

$$i\lambda\mu\omega^1 D e^{i\lambda x} + (\sigma+\tau)\omega^1 D e^{i\lambda x} = \frac{\chi\eta}{2} \int_0^\infty \omega^1 D_0 e^{i\lambda x} \, d\nu$$

From this we have that

$$D = \frac{\chi\eta}{2} \frac{\int_0^\infty D_0 d\nu}{((\sigma+\tau)+i\lambda\mu)} = \frac{((\sigma+\tau)-i\lambda\mu)\chi\eta}{2} \frac{\int_0^\infty D_0 d\nu}{((\sigma+\tau)^2+\lambda^2\mu^2)}$$

$$D_0^{l+\frac{1}{2}} = \int_{-1}^{1} \omega^l D e^{i\lambda x} \, d\mu$$

$$\frac{\lambda^2}{3(\sigma+\tau)} \omega^l f_0 \, e^{i\lambda x} + (\sigma+\tau) \, \omega^l f_0 e^{i\lambda x} = \chi\eta \int_0^\infty \sigma \omega^l f_0 \, e^{i\lambda x} \, d\nu$$

$$+ \chi\eta \int_0^\infty \left[\sigma \int_{-1}^{1} \omega^l D e^{i\lambda x} \, d\mu - \omega^1 D_0 e^{i\lambda x} \right] d\nu$$

After simplification the above results yields

$$\left(\frac{\lambda^2 + 3(\sigma+\tau)^2}{3(\sigma+\tau)}\right) f_o = \chi\eta\int_o^\infty \sigma f_o d\nu + \chi\eta\int_0^\infty \left[\sigma(\int_{-1}^{1} D d\mu) - D_o)d\nu\right]$$

(A.2.07a,b,c,d)

iii. From (A.2.04d) we have

$$f_o^{l+1} = D_o^{l+1} - D_o^{l+\frac{1}{2}}$$
$$\omega^l f_o e^{i\lambda x} = \omega^{l+1} D_o e^{i\lambda x} - \int_{-1}^{1} D \ e^{i\lambda x}\omega^l d\mu$$
$$f_o = \omega D_o - \int_{-1}^{1} D d\mu$$

(A.2.08a,b,c)

iv. Substitute f_o in (A.2.07c) to get

$$\left(\frac{\lambda^2 + 3(\sigma+\tau)^2}{3(\sigma+\tau)}\right) \left(\omega D_0 - \int_{-1}^{1} D \, d\mu\right) = \chi\eta\int_0^\infty \left[\sigma(\omega D_0 - \int_{-1}^{1} D \, d\mu) \, d\nu\right]$$
$$+ \chi\eta\int_0^\infty \sigma\int_{-1}^{1} D d\mu d\nu - \chi\eta\int_0^\infty \sigma D_0 d\nu$$

v. Multiply the above equation by σ and divide by

$$\left(\frac{\lambda^2+3(\sigma+\tau)^2}{3(\sigma+\tau)}\right)$$

Then integrate over frequency spectrum, to obtain

$$\omega\left[\int_0^\infty \sigma D_0 d\nu - \int_0^\infty \frac{\chi\eta\sigma}{\frac{\lambda^2+3(\sigma+\tau)^2}{3(\sigma+\tau)}} d\nu\right] = \int_0^\infty \sigma\int_{-1}^{1} D d\mu d\nu - \int_0^\infty \left[\frac{\chi\eta\sigma}{\frac{\lambda^2+3(\sigma+\tau)^2}{3(\sigma+\tau)}}\right] d\nu$$

(A.2.09)

vi. Substitute D from (A.2.07) and use the normalization relation

$$\omega = \frac{\int_0^\infty \sigma \int_{-1}^{1} \chi\eta \frac{((\sigma+\tau)-i\lambda\mu)}{2((\sigma+\tau)^2+\lambda^2\mu^2)} d\mu \, d\nu - \int_0^\infty \frac{3\chi\eta\sigma(\sigma+\tau)}{(\lambda^2+3(\sigma+\tau)^2)} d\nu}{\left(1-\int_0^\infty \frac{3\chi\eta\sigma(\sigma+\tau)d\nu}{(3(\sigma+\tau)^2+\lambda^2)}\right)}$$

(A.2.10)

vii. Simplify the above equation (imaginary parts disappears after $\int d\mu$)

$$\omega = \frac{\int_0^\infty \frac{\sigma\chi\eta}{2} (\sigma+\tau)d\nu \int_{-1}^{1}\left(\frac{1}{((\sigma+\tau)^2+\lambda^2\mu^2)} - \frac{3}{(\lambda^2+3(\sigma+\tau)^2)}\right)d\mu}{\left(1-\int_0^\infty \frac{3\chi\eta\sigma(\sigma+\tau)d\nu}{\lambda^2+3(\sigma+\tau)^2}\right)}$$

viii. After further simplification, the final result is

$$\omega = \frac{\frac{1}{2}\int_0^\infty \left[\frac{\sigma\chi(\sigma+\tau)}{(\lambda^2+3(\sigma+\tau)^2)}\int_{-1}^{1}\left(\frac{\lambda^2(1-3\mu^2)}{((\sigma+\tau)^2+\lambda^2\mu^2)}\right)d\mu\right]d\nu}{\left(1-\int_0^\infty \frac{3\chi\eta\sigma(\sigma+\tau)}{(\lambda^2+3(\sigma+\tau)^2)}d\nu\right)}$$

(A.2.11)

This shows that in the limit of $\lambda \to 0$ the spectral radius goes to zero for all values of $\tau \geq 0$. Therefore, with the solution of the diffusion equation, the solution of the transfer equation will converge rapidly even for slowly converging modes.

As stated earlier, the diffusion equation itself faces the same problem as the transfer equation, namely slow convergence. It can be shown, by using a Fourier stability analysis similar to that used for the transfer equation with SI (A.1), that the spectral radius for the diffusion equation with the source iteration technique is given by

$$\omega = \eta \int_0^\infty \frac{3\sigma(\sigma+\tau)\chi}{(\lambda^2+3(\sigma+\tau)^2)} d\nu$$

$$spr = \eta\int_0^\infty d\nu \frac{\sigma\chi}{(\sigma+\tau)} \to 1 \quad when \quad \tau\to 0 \ and/or \ \sigma\to\infty$$

(A.2.12)

This slow convergence can reduce the effectiveness of the diffusion equation in accelerating the transfer equation. Therefore, another equation is derived which is designed such that for slow converging modes a more rapidly converging solution is obtained. Fourier analysis is not presented here due to the complexity in its derivation, but a similar approach can be used to obtain the spectral radius of the transfer equation when it is accelerated using the diffusion equation which in turn has been accelerated by the linear grey equation.

A.3 SOLUTION OF THE RADIATIVE TRANSFER EQUATION:

We rewrite (3.4.02) here for completeness

$$\mu_m\left(I^{l+\frac{1}{2}}_{mgi+\frac{1}{2}} - I^{l+\frac{1}{2}}_{mgi-\frac{1}{2}}\right) + \sigma_{gi} I^{l+\frac{1}{2}}_{mgi} = \frac{\chi_{gi}\Delta x_i}{2}\Sigma_g \sigma_{gi} I^l_{0gi} + \Delta x_i Q_{mgi}$$

$$I^{l+\frac{1}{2}}_{mgi} = \frac{1}{2}\left(I^{l+\frac{1}{2}}_{mgi+\frac{1}{2}} + I^{l+\frac{1}{2}}_{mgi-\frac{1}{2}}\right)$$

$$I^{l+1}_{ogi} = \Sigma_m I^{l+\frac{1}{2}}_{mgi}\,\omega_m$$

(A.3.01a,b,c)

The solution to the above equation is obtained by a so called "sweep from left to right" and then "sweep from right to left" method. The intensity incident from the left is computed independent of the intensity going to the right and vice versa.

CASE I. $\mu_m > 0$

i. Substitute (A.3.01b) for $I^{l+1/2}_{mgi+1/2}$ into (A.3.01a) and solve for $I^{l+1/2}_{mgi}$

ii. Substitute the new value for $I^{l+1/2}_{mgi}$ in (A.3.01b) to obtain $I^{l+1/2}_{mgi+1/2}$ which will be used as $I^{l+1/2}_{mgi-1/2}$ for the next cell, and so on.

iii. If $I^{l+1/2}_{mgi+1/2}$ is less than zero, let $I^{l+1/2}_{mgi+1/2}=0$ or $I^{l+1/2}_{mgi+1/2}=I^{l+1/2}_{mgi}$ in (A.3.01) and solve again for $I^{l+1/2}_{mgi}$

CASE II. $\mu_m < 0$

The procedure is the same as for $\mu_m > 0$ except that $I^{l+1/2}_{mgi\pm1/2}$ is replaced by

$I^{l+1/2}_{mgi\pm1/2}$.

For the next iteration, $I^{l+1/2}_{mgi}$ obtained from (A.3.01b) is used to get the new source term for (A.3.01a). The final set of equations for $\mu_m > 0$ is given as

$$I^{l+\frac{1}{2}}_{mgi} = \frac{\left[\frac{\chi_{gi}\eta_i\Delta x_i}{2}\,\Sigma_g\,\sigma_{gi}I^{l}_{ogi} + \Delta x_i\,Q_{mgi} + 2\mu_m I^{l+\frac{1}{2}}_{mgi-\frac{1}{2}}\right]}{(\hat{\sigma}_{gi} + 2\mu_m)}$$

$$I^{l+\frac{1}{2}}_{mgi+\frac{1}{2}} = 2\,I^{l+\frac{1}{2}}_{mgi} - I^{l+\frac{1}{2}}_{mgi-\frac{1}{2}}$$

If $I^{l+1/2}_{mgi}$ is negative then, according to a parameter ξ ($=c\Delta t/\Delta x$)

$$\xi < 1 \qquad I^{l+\frac{1}{2}}_{mgi} = \frac{\left[\frac{\chi_{gi}\eta_i\Delta x_i}{2}\,\Sigma_g\sigma_{gi}I^{1}_{ogi} + \Delta x_i Q_{mgi} + \mu_m I^{1+\frac{1}{2}}_{mgi-\frac{1}{2}}\right]}{\hat{\sigma}_{gi}}$$

$$I^{l+\frac{1}{2}}_{mgi+\frac{1}{2}} = 0.0$$

$$\xi \geq 1 \qquad I^{1+\frac{1}{2}}_{mgi} = \frac{\left[\frac{\chi_{gi}\eta_i\Delta x_i}{2}\,\Sigma_g\sigma_{gi}I^{1}_{ogi} + \Delta x_i Q_{mgi} + \mu_m I^{1+\frac{1}{2}}_{mgi-\frac{1}{2}}\right]}{(\hat{\sigma}_{gi} + \mu_m)}$$

$$I^{l+\frac{1}{2}}_{mgi+\frac{1}{2}} = I^{l+\frac{1}{2}}_{mgi}$$

(A.3.02a,b;03a,b;04a,b)

If $\mu_m < 0$,

$$I^{l+\frac{1}{2}}_{mgi} = \frac{\left[\frac{\chi_{gi}\eta_i\Delta x_i}{2}\,\Sigma_g\,\sigma_{gi}I^{l}_{ogi} + \Delta\chi_i\,Q_{mgi} - 2\mu_m I^{l+\frac{1}{2}}_{mgi+\frac{1}{2}}\right]}{(\hat{\sigma}_{gi} - 2\mu_m)}$$

$$I^{l+\frac{1}{2}}_{mgi-\frac{1}{2}} = 2\,I^{l+\frac{1}{2}}_{mgi} - I^{l+\frac{1}{2}}_{mgi+\frac{1}{2}}$$

If $I^{l+1/2}_{mgi}$ is negative then,

$$\xi < 1 \qquad I_{mgi}^{l+\frac{1}{2}} = \frac{\left[\frac{\chi_{gi}\eta_i\Delta x_i}{2}\Sigma_g\sigma_{gi}I_{ogi}^{l} + \Delta x_i Q_{mgi} - \mu_m I_{mgi+\frac{1}{2}}^{l+\frac{1}{2}}\right]}{\hat{\sigma}_{gi}}$$

$$I_{mgi-\frac{1}{2}}^{l+\frac{1}{2}} = 0.0$$

$$\xi \geq 1 \qquad I_{mgi}^{l+\frac{1}{2}} = \frac{\left[\frac{\chi_{gi}\eta_i\Delta x_i}{2}\Sigma_g\sigma_{gi}I_{ogi}^{l} - \Delta x_i Q_{mgi} + \mu_m I_{mgi-\frac{1}{2}}^{l+\frac{1}{2}}\right]}{(\hat{\sigma}_{gi} - \mu_m)}$$

$$I_{mgi-\frac{1}{2}}^{l+\frac{1}{2}} = I_{mgi}^{l+\frac{1}{2}}$$

(A.3.05a,b;06a,b;07a,b)

The new source term is obtained by summing the new specific intensity from the transfer equation as

$$I_{ogi}^{l+1} = I_{ogi}^{l+\frac{1}{2}} = \Sigma_m I_{mgi}^{l+\frac{1}{2}} \omega_m$$

(A.3.08)

Therefore, one complete solution of the transfer equation with source iteration can involve equations (A.3.03;04;05;06;07;08).

A.4 DERIVATION OF DISCRETIZED FORM OF THE DIFFUSION EQUATION:

In order to obtain a diffusion equation which is consistent with the transfer equation just solved, one must start with (3.5.02) instead of (3.3.12). Alcouffe showed[A/1] that the diffusion obtain equation from (3.7.27) leads to unstable solutions. The following steps are taken in order to obtain the discretized diffusion equation from the transfer equation.

i. Multiply equation (A.3.01) by ω_m and $\omega_m\mu_m$ and sum over the angular space to obtain the zeroth and first order moment in angular space. This yields

$$(I^{l+\frac{1}{2}}_{1gi+\frac{1}{2}} - I^{l+\frac{1}{2}}_{1gi-\frac{1}{2}}) + \hat{\sigma}_{gi}\, I^{l+\frac{1}{2}}_{ogi} = \chi_{gi}\eta_i \Delta x_i \Sigma_g \sigma_{gi} I^{l}_{ogi} + \Delta x_i Q_{ogi}$$

$$\frac{2}{3}\,(I^{l+\frac{1}{2}}_{2gi+\frac{1}{2}} - I^{l+\frac{1}{2}}_{2gi-\frac{1}{2}}) + \frac{1}{3}\,(I^{l+\frac{1}{2}}_{ogi+\frac{1}{2}} - I^{l+\frac{1}{2}}_{ogi-\frac{1}{2}}) + \hat{\sigma}_{gi} I^{l+\frac{1}{2}}_{1gi} = Q_{1gi}\Delta x_i$$

(A.4.01;02)

Here we used the facts that

$$I^{l+\frac{1}{2}}_{2gi} = \Sigma_m\, I^{l+\frac{1}{2}}_{mgi}\ \omega_m\ (\frac{3\mu_m^{\ 2}-1}{2}) = \frac{3}{2}\ \Sigma\ \mu_m^{\ 2}\ \omega_m\ I^{l+\frac{1}{2}}_{mgi} - \frac{1}{2}\ I^{l+\frac{1}{2}}_{ogi}$$

$$\Sigma_m\ \omega_m \mu_m = 0$$

$$Q_{ogi} = \frac{\chi_{gi}\ \eta_i\ u_g^{\ n}}{\beta_i \Delta t^{\ n}} + \tau^n\ I^n_{ogi}$$

$$Q_{1gi} = \tau^n I_1 gi$$

(A.4.04a,b,c)

ii. The accelerated form of the diffusion equation is obtained by accelerating only the first two moments, as given below

$$(I^{l+1}_{1gi+\frac{1}{2}} - I^{l+1}_{1gi-\frac{1}{2}}) + \hat{\sigma}_{gi}\ I^{l+1}_{ogi} = \chi_{gi}\ \eta_i\ \Delta x_i\ \Sigma_g\ \sigma_{gi}\ I^{l+1}_{ogi} + \Delta x_i\ Q_{ogi}$$

$$\frac{2}{3}\,(I^{l+\frac{1}{2}}_{2gi+\frac{1}{2}} - I^{l+\frac{1}{2}}_{2gi-\frac{1}{2}}) + \frac{1}{3}\,(I^{l+1}_{ogi+\frac{1}{2}} - I^{l+1}_{ogi-\frac{1}{2}}) + \hat{\sigma}_{gi}\ I^{l+1}_{1gi} = Q_{1gi}\Delta x_i$$

(A.4.05;06)

The DD relationship is redefined in order to eliminate the instability due to the negative flux fix up in the transfer equation, as

$$I^{l+1}_{ngi} = \frac{1}{2}\left[a_{ogi}\, I^{l+1}_{ogi+\frac{1}{2}} + b_{ogi}\, I^{l+1}_{ogi-\frac{1}{2}}\right] \qquad n = 0,1$$

(A.4.07)

where the choice of a_{og} and b_{og} is somewhat based on trial and error approach.

To obtain the diffusion equation in the form of a tridiagonal matrix, the following manipulative steps are taken

i. Equate (A.4.02) and (A.4.06).

ii. Substitute (A.4.07) in it for n=1 with $a_{1gi} = b_{1gi} = 1$.

iii. Write the resultant for i+1.

iv. Divide (ii) and (iii) by σ_{gi} and σ_{gi+1} respectively.

v. Eliminate (A.4.05) for $I^{l+1}_{1gi-1/2}$.

vi. Rewrite (A.4.05) for i=i+1 and eliminate for $I^{l+1/2}_{1gi+3/2}$.

vii. Substitute $I^{l+1}_{1gi-1/2}$ and $I^{l+1}_{1gi+3/2}$ from v and vi into step iv.

viii. Substitute (A.4.07) for n=0 in (vii) .

ix. Subtract the resulting two equation from the two equations obtained in step vii.

The final result is

$$\begin{aligned}
&- \frac{1}{3\hat{\sigma}_{gi+1}}\left[I^{l+1}_{ogi+\frac{3}{2}} - I^{l+1}_{ogi+\frac{1}{2}}\right] + \frac{1}{3\hat{\sigma}_{gi}}\left[I^{l+1}_{ogi+\frac{1}{2}} - I^{l+1}_{ogi-\frac{1}{2}}\right] \\
&+ \frac{\hat{\sigma}_{gi+1}}{4}\left[a_{ogi+1}I^{l+1}_{ogi+\frac{3}{2}} + b_{ogi+1}I^{l+1}_{ogi+\frac{1}{2}}\right] + \frac{\hat{\sigma}_{gi}}{4}\left[a_{ogi}I^{l+1}_{ogi+\frac{1}{2}} + b_{ogi}I^{l+1}_{ogi-\frac{1}{2}}\right] \\
&= \frac{1}{4}\left[\chi_{gi+1}\eta_{i+1}\Delta x_{i+1}\Sigma_{g}\ \sigma_{gi+1}(a_{0gi+1}I^{l+1}_{0gi+\frac{3}{2}} + b_{0gi+1}I^{l+1}_{0gi+\frac{1}{2}})\right] \\
&+ \frac{1}{4}\left[\chi_{gi}\eta_{i}\Delta x_{i}\Sigma_{g}\ \sigma_{gi}(a_{0gi}I^{l+1}_{0gi+\frac{1}{2}} + b_{0gi}I^{l+1}_{0gi-\frac{1}{2}})\right] + S_{gi}^{l+\frac{1}{2}}
\end{aligned}$$

(A.4.08)

Where $S_{gi}^{l+1/2}$ consists of variables which do not change during iteration of the diffusion equation, and is given as

$$S_{gl}^{l+\frac{1}{2}} = \frac{\Delta x_l \, Q_{0gl}}{2} + \frac{\Delta x_{l+1} \, Q_{0gl+1}}{2} - \frac{1}{3\dot{\sigma}_{gl+1}}\left[I_{ogl+\frac{3}{2}}^{l+\frac{1}{2}} - I_{ogl+\frac{1}{2}}^{l+\frac{1}{2}} \right] + \frac{1}{3\dot{\sigma}_{gl}}\left[I_{ogl+\frac{1}{2}}^{l+\frac{1}{2}} - I_{ogl-\frac{1}{2}}^{l+\frac{1}{2}} \right] - \frac{1}{2}\left[I_{1gl+\frac{3}{2}}^{l+\frac{1}{2}} - I_{1gl-\frac{1}{2}}^{l+\frac{1}{2}} \right]$$

(A.4.09)

A.5 DERIVATION OF THE DISCRETIZED FORM OF THE GREY EQUATION:

A discretized form of the diffusion equation was obtained from the discretized transfer equation to ensure stability. The grey diffusion equation is derived from the discretized form of the diffusion equation, namely equation (A.4.08).

The accelerated form of the diffusion equation is obtained by summing the frequency groups with a particular weight function. The following substitution is made in (A.4.08) before summing over the frequency groups in order to obtain the grey equation.

Consider the following transformation to be used in the diffusion equation

$$f_{gl}^{k^*} = (a_{ogl} I_{ogl+\frac{1}{2}}^{k^*} + b_{ogl} I_{ogl-\frac{1}{2}}^{k^*}) - \theta_{gl}^{k+\frac{1}{2}} (a_{ol} I_{ol+\frac{1}{2}}^{k+1} + b_{ol} I_{ol-\frac{1}{2}}^{k+1}) + f_{gl}^{k+\frac{1}{2}} - \theta_{gl}^{k+\frac{1}{2}} (a_{ol} I_{ol+\frac{1}{2}}^{k+\frac{1}{2}} + b_{ol} I_{ol-\frac{1}{2}}^{k+\frac{1}{2}})$$

$$g_{gl}^{k+\frac{1}{2}} = (I_{ogl+\frac{1}{2}}^{k+\frac{1}{2}} - I_{ogl-\frac{1}{2}}^{k+\frac{1}{2}}) - \varphi_{gl}^{k+\frac{1}{2}} (I_{ol+\frac{1}{2}}^{k+1} - I_{ol-\frac{1}{2}}^{k+1}) + g_{gl}^{k+\frac{1}{2}} - \varphi_{gl}^{k+\frac{1}{2}} (I_{ol+\frac{1}{2}}^{k+\frac{1}{2}} - I_{ol-\frac{1}{2}}^{k+\frac{1}{2}})$$

(A.5.01a,b)

$$a_{\alpha l} = \frac{\Sigma_g \left(a_{\alpha gl}\, I^{k+\frac{1}{2}}_{\alpha gl+\frac{1}{2}}\right)}{I^{k+\frac{1}{2}}_{\alpha l+\frac{1}{2}}}, \quad b_{\alpha l} = \frac{\Sigma_g \left(b_{\alpha gl}\, I^{k+\frac{1}{2}}_{\alpha gl-\frac{1}{2}}\right)}{I^{k+\frac{1}{2}}_{\alpha l-\frac{1}{2}}}, \quad I^{k+\frac{1}{2}}_{\alpha l+\frac{1}{2}} = \Sigma_g\, I^{k+\frac{1}{2}}_{\alpha gl+\frac{1}{2}}$$

(A.5.02a,b,c)

k' can be k or k+1/2 corresponding to values before and after acceleration. The resultant equation is

$$
\begin{aligned}
&- \Sigma_g \frac{1}{3\hat{\sigma}_{gl+1}}\left[\varphi^{k+1/2}_{gl+1}\left(I^{k+1}_{\alpha l+\frac{3}{2}} - I^{k+1}_{\alpha l+\frac{1}{2}}\right)^{\#1} + \left(I^{k+\frac{1}{2}}_{\alpha gl+\frac{3}{2}} - I^{k+\frac{1}{2}}_{\alpha gl+\frac{1}{2}}\right)^{\#2} - \varphi^{k+1/2}_{gl+1}\left(I^{k+\frac{1}{2}}_{\alpha l+\frac{3}{2}} - I^{k+\frac{1}{2}}_{\alpha l+\frac{1}{2}}\right)^{\#3}\right] \\
&+ \Sigma_g \frac{1}{3\hat{\sigma}_{gl}}\left[\varphi^{k+1}_{gl}\left(I^{k+1}_{\alpha l+\frac{1}{2}} - I^{k+1}_{\alpha l-\frac{1}{2}}\right)^{\#4} + \left(I^{k+\frac{1}{2}}_{\alpha gl+\frac{1}{2}} - I^{k+\frac{1}{2}}_{\alpha l-\frac{1}{2}}\right)^{\#5} - \varphi^{k+\frac{1}{2}}_{gl}\left(I^{k+\frac{1}{2}}_{\alpha l+\frac{1}{2}} - I^{k+\frac{1}{2}}_{\alpha l-\frac{1}{2}}\right)^{\#6}\right] \\
&+ \Sigma_g \frac{\hat{\sigma}_{gl+1}}{4}\left[\theta^{k+1/2}_{gl+1}\left(a_{\alpha l+1} I^{k+1}_{\alpha l+\frac{3}{2}} + b_{\alpha l+1} I^{k+1}_{\alpha l+\frac{1}{2}}\right)^{\#7} + f^{k+\frac{1}{2}\,\#8}_{gl+1} - \theta^{k+\frac{1}{2}}_{gl+1}\left(a_{\alpha l+1} I^{k+\frac{1}{2}}_{\alpha l+\frac{3}{2}} + b_{\alpha l+1} I^{k+\frac{1}{2}}_{\alpha l+\frac{1}{2}}\right)^{\#9}\right] \\
&+ \Sigma_g \frac{\hat{\sigma}_{gl}}{4}\left[\theta^{k+\frac{1}{2}}_{gl}\left(a_{\alpha l} I^{k+1}_{\alpha gl+\frac{1}{2}} + b_{\alpha l} I^{k+1}_{\alpha gl-\frac{1}{2}}\right)^{\#10} + f^{k+\frac{1}{2}\,\#11}_{gl} - \theta^{k+\frac{1}{2}}_{gl}\left(a_{\alpha l} I^{k+\frac{1}{2}}_{\alpha l+\frac{1}{2}} + b_{l} I^{k+\frac{1}{2}}_{\alpha l-\frac{1}{2}}\right)^{\#12}\right] \\
&= \Sigma_g \frac{1}{4}\chi_{gl+1}\eta_{i+1}\Delta x_l \Sigma_g \sigma_{gl+1}\left[\theta^{k+1/2}_{gl+1}\left(a_{\alpha l+1} I^{k+1}_{\alpha l+\frac{3}{2}} + b_{\alpha l+1} I^{k+1}_{\alpha l+\frac{1}{2}}\right)^{\#13} + f^{k+1/2\,\#14}_{gl+1} - \theta^{k+\frac{1}{2}}_{gl+1}\left(a_{\alpha l+1} I^{k+\frac{1}{2}}_{\alpha l+\frac{3}{2}} + b_{\alpha l+1} I^{k+1/2}_{\alpha l+\frac{1}{2}}\right)^{\#15}\right] \\
&+ \Sigma_g \frac{1}{4}\chi_{gl}\eta_l \Delta x_l \Sigma_g \sigma_{\alpha gl}\left[\theta^{k+\frac{1}{2}}_{\alpha gl}\left(a_{\alpha l} I^{k+\frac{1}{2}}_{\alpha l+\frac{1}{2}} + b_{\alpha l} I^{k+\frac{1}{2}}_{\alpha l-\frac{1}{2}}\right)^{\#16} + f^{k+\frac{1}{2}\,\#17}_{gl} - \theta^{k+\frac{1}{2}}_{gl}\left(a_{\alpha l} I^{k+\frac{1}{2}}_{l+\frac{1}{2}} + b_{\alpha l} I^{k+\frac{1}{2}}_{\alpha l-\frac{1}{2}}\right)^{\#18}\right] \\
&+ \Sigma_g S^{l+\frac{1}{2}\,\#19}_{gl}
\end{aligned}
$$

(A.5.04)

These relations simply reflect that a correction term is added to both sides of (A.4.08) in order that the accelerated form of the new equation yields the solution more rapidly by projecting out the slow converging modes. Before the final form of the Grey equation is presented, consider the following manipulative steps taken to obtain the final form of the grey equation.

Suppose we have an arbitrary function ψ_g which is a function of frequency and space. Consider the two terms (#7 & #13) from (A.5.04)

$$\Sigma_g \left[\hat{\sigma}_{gi}\ \psi_{gi} - \Delta x_i\ \chi_{gi}\ \eta_i\ \Sigma_{\dot{g}}\ \sigma_{\dot{g}}\psi_{\dot{g}i}\right]$$

where other variables are as previously defined

$$= \Sigma_g \left[\ (\sigma_{gi} + \tau)\Delta x_i\ \psi_{gi} - \Delta x_i\ \chi_{gi}\ \eta_i\ \Sigma_{\dot{g}}\ \sigma_{\dot{g}i}\ \psi_{\dot{g}i}\right]$$

$$= \Sigma_g \left[\ \sigma_{gi}\ \Delta x_i + \tau\Delta x_i - \frac{\Delta x_i\ \chi_{gi}}{(1+\alpha_i\tau)}\ \Sigma_g\ \sigma_{gi}\right]\psi_{gi}$$

$$= \Sigma_g \left[\ \sigma_{gi}\ \Delta x_i + \tau\Delta x_i - \frac{\Delta x_i\sigma_{gi}}{(1+\alpha_i\tau)}\ \Sigma_g\ \chi_{gi}\right]\psi_{gi}$$

$$= \Sigma_g \left[\ \hat{\sigma}_{gi}\ \psi_{gi} - \Delta x_i\chi_{gi}\eta_i\Sigma_{\dot{g}}\sigma_{\dot{g}i}\psi_{\dot{g}i}\right]$$

$$= \Sigma_g \left[\tau\Delta x_i + \frac{\alpha_i\sigma_{gi}\tau\Delta x_i}{(1+\alpha_i\tau)}\right]\psi_{gi}$$

$$= \Sigma_g \left[1 + \alpha_i\sigma_{gi}\eta_i\right]\psi_{gi}\tau\Delta x_i$$

$$let \qquad \xi_{gi} = (1+\alpha_i\sigma_{gi}\eta_i)\tau\Delta x_i$$

(A.5.05)

If ψ_g is taken to be $\theta^{k+1/2}$,then above result can be written as

$$\xi_i = \Sigma_g\ \xi_{gi}\ \theta_{gi}^{k+\frac{1}{2}}$$

With the above manipulation and keeping only k+1 terms on left side of equation (A.5.04), it reduces to

$$
\begin{aligned}
&- \sum_g \frac{\varphi_{gl+1}^{k+\frac{1}{2}}}{3\hat{\sigma}_{gl+1}}\left(I_{\alpha l+\frac{3}{2}}^{k+1} - I_{\alpha l+\frac{1}{2}}^{k+1}\right) + \sum_g \frac{\varphi_{gl}^{k+\frac{1}{2}}}{3\hat{\sigma}_{gl}}\left(I_{\alpha l+\frac{1}{2}}^{k+1} - I_{\alpha l-\frac{1}{2}}^{k+1}\right) \\
&+ \frac{\xi_{l+1}}{4}\left(a_{\alpha l+1} I_{\alpha l+\frac{3}{2}}^{k+1} + b_{\alpha l+1} I_{\alpha l+\frac{1}{2}}^{k+1}\right) + \frac{\xi_l}{4}\left(a_{\alpha l} I_{\alpha l+\frac{1}{2}}^{k+1} + b_{\alpha l} I_{\alpha l-\frac{1}{2}}^{k+1}\right) \\
&= \sum_g \left[\frac{1}{3\hat{\sigma}_{gl+1}}\left(I_{\sigma gl+\frac{3}{2}}^{k+\frac{1}{2}} - I_{\sigma gl+\frac{1}{2}}^{k+\frac{1}{2}}\right) - \frac{1}{3\hat{\sigma}_{gl}}\left(I_{\sigma gl+\frac{1}{2}}^{k+\frac{1}{2}} - I_{\sigma gl-\frac{1}{2}}^{k+\frac{1}{2}}\right)\right] \\
&- \sum_g \left[\frac{\varphi_{gl+1}^{k+\frac{1}{2}}}{3\hat{\sigma}_{gl+1}}\left(I_{\alpha l+\frac{3}{2}}^{k+\frac{1}{2}} - I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}}\right) - \frac{\varphi_{gl}^{k+\frac{1}{2}}}{3\hat{\sigma}_{gl}}\left(I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} - I_{\alpha l-\frac{1}{2}}^{k+\frac{1}{2}}\right)\right] \\
&- \frac{1}{4}\sum_g \left(\xi_{gl+1} f_{gl+1}^{k+\frac{1}{2}} + \xi_{gl} f_{gl}^{k+\frac{1}{2}}\right) \\
&+ \frac{1}{4}\left[\xi_{l+1}\left(a_{\alpha l+1} I_{\alpha l+\frac{3}{2}}^{k+\frac{1}{2}} + b_{\alpha l+1} I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}}\right) + \xi_l\left(a_{\alpha l} I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} + b_{\alpha l} I_{\alpha l-\frac{1}{2}}^{k+\frac{1}{2}}\right)\right] \\
&+ \sum_g S_{gl}^{k+\frac{1}{2}}
\end{aligned}
$$

(A.5.06)

The solution of the grey equation depends on the choice of the spectral functions θ and φ. One expression for θ and φ corresponds to the spectral function causing slowest convergence. In this case θ and φ are taken to be same and are given as

$$\theta_{gl}^{k+\frac{1}{2}} = \varphi_{gl}^{k+\frac{1}{2}} = \frac{\chi_{gl}/(\sigma_{gl}+\tau)}{\Sigma_g[\chi_{gl}/(\sigma_{gl}+\tau)]}$$

(A.5.07)

This is known as the linear case. The other expression for θ and φ is known as the non linear case and is given as

$$\theta_{gl}^{k+\frac{1}{2}} = \frac{f_{gl}^{k+\frac{1}{2}}}{\left(a_{\alpha l}\, I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} + b_{\alpha l}\, I_{\alpha l-\frac{1}{2}}^{k+\frac{1}{2}}\right)}, \quad \varphi_{gl}^{k+\frac{1}{2}} = \frac{g_{gl}^{k+\frac{1}{2}}}{\left(I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} - I_{\alpha l-\frac{1}{2}}^{k+\frac{1}{2}}\right)}$$

(A.5.08a,b)

When the nonlinear expression is substituted in (A.5.06), it reduces to

$$- \Sigma_g \frac{1}{3\hat{\sigma}_{gl+1}} \left[\varphi_{gl+1}^{k+\frac{1}{2}} \left(I_{\alpha l+\frac{3}{2}}^{k+1} - I_{\alpha l+\frac{1}{2}}^{k+1} \right) \right] + \Sigma_g \frac{1}{3\hat{\sigma}_{gl}} \left[\varphi_{gl}^{k+\frac{1}{2}} \left(I_{\alpha l+\frac{1}{2}}^{k+1} - I_{gl-\frac{1}{2}}^{k+1} \right) \right]$$
$$+ \frac{1}{4} \left[\xi_{l+1} \left(a_{\alpha l+1} I_{\alpha l+\frac{3}{2}}^{k+1} + b_{\alpha l+1} I_{0gl+\frac{1}{2}}^{k+1} \right) \right] + \frac{1}{4} \left[\xi_l \left(a_{\alpha l} I_{\alpha l+\frac{1}{2}}^{k+1} + b_{\alpha l} I_{\alpha l-\frac{1}{2}}^{k+1} \right) \right] = \Sigma_g S_{gl}^{l+\frac{1}{2}}$$

(A.5.09)

The following steps are utilized to obtain above equation. Consider the following expression from right side of (A.5.06)

$$\Sigma_g \frac{\varphi_{gl}^{k+\frac{1}{2}}}{\hat{\sigma}_{gl}} \left(I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} - I_{\alpha l-\frac{1}{2}}^{k+\frac{1}{2}}\right) + \xi_l \left(a_{\alpha l}\, I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} - b_{\alpha l}\, I_{\alpha l-\frac{1}{2}}^{k+\frac{1}{2}}\right)$$

(A.5.10)

If we substitute the nonlinear expression (A.5.08a,b) in (A.5.10)

$$\sum_g \frac{1}{\hat{\sigma}_{gl}} \frac{\left(I_{0gl+\frac{1}{2}}^{k+\frac{1}{2}} - I_{0gl-\frac{1}{2}}^{k+\frac{1}{2}}\right)\left(I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} - I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}}\right)}{\left(I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} - I_{\alpha l-\frac{1}{2}}^{k-\frac{1}{2}}\right)}$$
$$+ \sum_g \left[\frac{\xi_g f_{gl}^{k+\frac{1}{2}} \left(a_{\alpha l} I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} + b_{\alpha l} I_{\alpha l-\frac{1}{2}}^{k+\frac{1}{2}}\right)}{\left(a_{\alpha l} I_{\alpha l+\frac{1}{2}}^{k+\frac{1}{2}} + b_{\alpha l} I_{\alpha l-\frac{1}{2}}^{k+\frac{1}{2}}\right)} \right]$$
$$= \sum_g \frac{1}{\hat{\sigma}_{gl}} \left(I_{0gl+\frac{1}{2}}^{k+\frac{1}{2}} - I_{0gl-\frac{1}{2}}^{k+\frac{1}{2}}\right) + \sum_g \xi_g f_{gl}^{k+\frac{1}{2}}$$

(A.5.11)

Here we have used (A.5.01) and (A.5.05).

Now it is trivial to see how all terms except S_{gi} on the right hand side of (A.5.06) cancel each other.

A.6 BOUNDARY CONDITIONS OF THE DIFFUSION EQUATION:

The boundary conditions[A/2] are obtained by adding a term into the flux (scalar intensity) at cell boundaries in order to make it consistent with the scalar intensity inside the slab. Conventionally, a quantity known as the one-way flux must be conserved between the accelerated and non-accelerated equations.

The one way flux at the left and right boundaries is given as,

$$J_{\frac{1}{2}}^{l+\frac{1}{2}} = \Sigma_{\mu_m>0}\ \omega_m \mu_m I_{mg\frac{1}{2}}^{l+\frac{1}{2}}$$
$$J_{I+\frac{1}{2}}^{l+\frac{1}{2}} = \Sigma_{\mu_m<0}\ \omega_m \mu_m I_{mgI+\frac{1}{2}}^{l+\frac{1}{2}}$$

(A.6.01)

The diffusion equation derived here is based on the fact that if the specific intensity is linear in angular space, then the exact solution of the diffusion equation is obtained without iteration (**assuming** no negative flux fixup is implemented). The specific intensity at the boundaries is defined as

$$I_{mg}^{l+\frac{1}{2}} = \frac{1}{2} I_{0g}^{l+\frac{1}{2}} + \frac{3}{2}\mu_m I_{1g}^{l+\frac{1}{2}}$$

(A.6.02)

The one way flux becomes

$$J_{\frac{1}{2}}^{l+\frac{1}{2}} = \Sigma_{\mu_m>0}\ \omega_m \mu_m [\frac{1}{2} I_{0g\frac{1}{2}}^{l+\frac{1}{2}} + \frac{3}{2}\mu_m I_{1g\frac{1}{2}}^{l+\frac{1}{2}}]$$
$$J_{I+\frac{1}{2}}^{l+\frac{1}{2}} = \Sigma_{\mu_m<0}\ \omega_m \mu_m [\frac{1}{2} I_{0gI+\frac{1}{2}}^{l+\frac{1}{2}} + \frac{3}{2}\mu_m I_{1gI+\frac{1}{2}}^{l+\frac{1}{2}}]$$

(A.6.03a,b)

It's accelerated form is given as

$$J^{l+1}_{\frac{1}{2}} = \Sigma_{\mu_m>0}\ \omega_m\mu_m[\frac{1}{2}I^{l+1}_{0g\frac{1}{2}} + \frac{3}{2}\mu_m I^{l+1}_{1g\frac{1}{2}}]$$
$$J^{l+1}_{I+\frac{1}{2}} = \Sigma_{\mu_m<0}\ \omega_m\mu_m[\frac{1}{2}I^{l+1}_{0gI+\frac{1}{2}} + \frac{3}{2}\mu_m I^{l+1}_{1gI+\frac{1}{2}}]$$

(A.6.04a,b)

By subtracting (A.6.03) from (A.6.04), and using the condition that $J^{l+1/2} = J^{l+1}$, we obtain the following accelerated diffusion intensity at the boundaries after summing over all angles

$$I^{l+1}_{0g\frac{1}{2}}\Sigma_{\mu_m>0}\ \omega_m\mu_m = I^{l+\frac{1}{2}}_{0g\frac{1}{2}}\Sigma_{\mu_m>0}\ \omega_m\ \mu_m + I^{l+\frac{1}{2}}_{1g\frac{1}{2}} - I^{l+1}_{1g\frac{1}{2}}$$
$$I^{l+1}_{0gI+\frac{1}{2}}\Sigma_{\mu_m<0}\omega_m\mu_m = I^{l+\frac{1}{2}}_{0gI+\frac{1}{2}}\Sigma_{\mu_m<0}\ \omega_m\mu_m + I^{l+\frac{1}{2}}_{1gI+\frac{1}{2}} - I^{l+1}_{1gI+\frac{1}{2}}$$
$$since\quad \Sigma_m\ \omega_m\mu_m^2 = \frac{1}{3}$$

(A.6.05a,b)

However $I^{l+1}_{1g1/2}$ and $I^{l+1}_{1gI+1/2}$ are unknown. To find expressions for these variables in terms of $I^{l+1}_{0gi\pm1/2}$ quantities we go back to the original equation used to derive the diffusion equation within the slab.

$I^{l+1}_{1g1/2}$ and $I^{l+1}_{1gI+1/2}$ are computed as follows

A. $I^{l+1}_{1g1/2}$

i. Divide (A.4.06) by σ_{gi}

ii. Substitute (A.4.07) in above for n=1

iii. Multiply (A.4.05) by (-1/2) and add the resultant into above

iv. Substitute (A.4.06) for n=0 into iv

v. Solve for $I^{l+1}_{1g1/2}$ by taking i = 1

B. $I^{l+1}_{1gI+1/2}$

A similar procedure is used, except that we now multiply by 1/2 (instead of -1/2) and solve for $I^{k-1}_{1gI+1/2}$ by taking i = I (instead of i=1), The final result is

$$I^{l+1}_{1g\frac{1}{2}} = \frac{\hat{\sigma}_{gl}}{4}(a_{0gl}I^{l+1}_{0g\frac{3}{2}} + b_{0gl}I^{l+1}_{0g\frac{1}{2}}) - \frac{\chi_{gl}\eta_1\Delta x_1}{4}\Sigma_{g'}\ \sigma_{g1}(a_{0g1}I^{l+1}_{0g\frac{3}{2}} + b_{0g1}I^{l+1}_{0g\frac{1}{2}})$$
$$-\frac{\Delta x_1 Q_{0gl}}{2} + \frac{Q_{1gl}\Delta x_1}{\hat{\sigma}_{gl}} - \frac{1}{3\hat{\sigma}_{gl}}(I^{l+1}_{0g\frac{3}{2}} - I^{l+1}_{0g\frac{1}{2}}) - \frac{2}{3\hat{\sigma}_{gl}}(I^{l+\frac{1}{2}}_{2g\frac{3}{2}} - I^{l+\frac{1}{2}}_{2g\frac{1}{2}})$$

$$I^{l+1}_{1gI+\frac{1}{2}} = -\frac{\hat{\sigma}_{gI}}{4}(a_{0gI}I^{l+1}_{0gI+\frac{1}{2}} + b_{0gI}I^{l+1}_{0gI-\frac{1}{2}}) + \frac{\chi_{gI}\eta_I\Delta x_I}{4}\Sigma_{g'}\ \sigma_{gI}(a_{0gI}I^{l+1}_{0gI+\frac{1}{2}} + b_{0gI}I^{l+1}_{0gI-\frac{1}{2}})$$
$$+\frac{\Delta x_I Q_{0gI}}{2} + \frac{Q_{1gI}\Delta x_I}{\hat{\sigma}_{gI}} - \frac{1}{3\sigma_{gI}}(I^{l+1}_{0gI+\frac{1}{2}} - I^{l+1}_{0gI-\frac{1}{2}}) - \frac{2}{3\hat{\sigma}_{gI}}(I^{l+\frac{1}{2}}_{2gI+\frac{1}{2}} - I^{l+\frac{1}{2}}_{2gI-\frac{1}{2}})$$

(A.6.06a,b)

After substituting (A.6.06) in (A.6.05), we rewrite these equations in similar form as the diffusion equation. The final form of the accelerated diffusion equation at the boundaries is given as

$$(\Sigma_{\mu_m>0}\ \omega_m\mu_m + \frac{\hat{\sigma}_{gl}}{4}b_{0gl} + \frac{1}{3\hat{\sigma}_{gl}})I^{l+1}_{0g\frac{1}{2}} + (\frac{\hat{\sigma}_{gl}}{4}a_{0gl} - \frac{1}{3\hat{\sigma}_{0gl}})I^{l+1}_{0g\frac{3}{2}}$$
$$= I^{l+\frac{1}{2}}_{0g\frac{1}{2}}\Sigma_{\mu_m>0}\ \omega_m\mu_m + I^{l+\frac{1}{2}}_{1g\frac{1}{2}} + \frac{\chi_{gl}\eta_1\Delta x_1}{4}\Sigma_{g'}\ \sigma_{g1}(a_{0g1}I^{l+1}_{0g\frac{3}{2}} + b_{0g1}I^{l+1}_{0g\frac{1}{2}})$$
$$+\frac{\Delta x_1 Q_{0gl}}{2} - \frac{Q_{1gl}\Delta x_1}{\hat{\sigma}_{gl}} + \frac{2}{3\hat{\sigma}_{gl}}(I^{l+\frac{1}{2}}_{2g\frac{3}{2}} - I^{l+\frac{1}{2}}_{2g\frac{1}{2}})$$

$$(-\ \Sigma_{\mu_m<0}\ \omega_m\mu_m + \frac{\hat{\sigma}_{gI}}{4}a_{0gI} + \frac{1}{3\hat{\sigma}_{gI}})I^{l+1}_{0gI+\frac{1}{2}} + (\frac{\hat{\sigma}_{gI}}{4}b_{0gI} - \frac{1}{3\hat{\sigma}_{gI}})I^{l+1}_{0gI-\frac{1}{2}}$$
$$= -\ I^{l+\frac{1}{2}}_{0gI+\frac{1}{2}}\Sigma_{\mu_m<0}\omega_m\mu_m - I^{l+\frac{1}{2}}_{1gI+\frac{1}{2}} + \frac{\chi_{gI}\eta_I\Delta x_I}{4}\Sigma_{g'}\ \sigma_{gI}(a_{0gI}I^{l+1}_{0gI+\frac{1}{2}} + b_{0gI}I^{l+1}_{0gI-\frac{1}{2}})$$
$$+\frac{\Delta x_I Q_{0gI}}{2} + \frac{Q_{1gI}\Delta x_I}{\hat{\sigma}_{gI}} - \frac{2}{3\hat{\sigma}_{gI}}I_{0g}(I^{l+\frac{1}{2}}_{2gI+\frac{1}{2}} - I^{l+\frac{1}{2}}_{2gI-\frac{1}{2}})$$

(A.6.07a,b)

A.7 BOUNDARY CONDITIONS OF THE GREY EQUATION:

The boundary conditions for the grey equation are obtained from the diffusion boundary conditions using the transformation (A.5.02) that was used to obtain the grey equation from the diffusion equation inside the slab. The following steps are used:

i. Substitute (A.5.01) in (A.6.06).

ii. Sum over frequency groups.

iii. Use similar simplification as was done to obtain (A.5.06) from (A.5.04).

The final form of the grey equation at the boundaries can then be expressed as

$$\begin{aligned}
&\left(\Sigma_{\mu_m>0}\, \omega_m\mu_m + \Sigma_g \frac{\varphi_{gl}}{3\hat{\sigma}_{gl}} + \frac{1}{4}\xi_1 b_{01}\right) I^{k+1}_{0\frac{1}{2}} + \left(\frac{1}{4}\xi_1 a_{01} - \Sigma_g \frac{1}{3\hat{\sigma}_{gl}}\varphi_{gl}\right) I^{k+1}_{0\frac{3}{2}} \\
&= I_{0\frac{1}{2}}^{l+\frac{1}{2}} \Sigma_{\mu_m>0}\omega_m\mu_m + I_{1\frac{1}{2}}^{l+\frac{1}{2}} - \frac{1}{4}\Sigma_g \xi_g f_{gl}^{k+\frac{1}{2}} + \frac{\xi_1}{4}\left(a_{01} I_{0\frac{3}{2}}^{k+\frac{1}{2}} + b_{01} I_{0\frac{1}{2}}^{k+\frac{1}{2}}\right) \\
&\quad + \frac{\Delta x_1}{2}\Sigma_g Q_{0gl} - \Sigma_g \frac{Q_{1gl}\Delta x_1}{\hat{\sigma}_{gl}} + \Sigma_g \frac{2}{3\hat{\sigma}_{gl}}\left(I_{2g\frac{3}{2}}^{l+\frac{1}{2}} - I_{2g\frac{1}{2}}^{l+\frac{1}{2}}\right) \\
&\quad - \frac{1}{3}\Sigma \frac{\varphi_{gl}^{k+\frac{1}{2}}}{\hat{\sigma}_{gl}}\left(I_{0\frac{3}{2}}^{k+\frac{1}{2}} - I_{0\frac{1}{2}}^{k+\frac{1}{2}}\right) + \frac{1}{3}\Sigma\left(\frac{1}{\hat{\sigma}_{gl}}\left(I_{0g\frac{3}{2}}^{k+\frac{1}{2}} - I_{0g\frac{1}{2}}^{k+\frac{1}{2}}\right)\right.
\end{aligned}$$

(A.7.01;02)

A.8 SOLUTION OF THE DIFFUSION AND GREY EQUATIONS:

To obtain the solution for the diffusion equation, we rewrite (A.4.08) and (A.6.07) in a tridiagonal form, and invert the matrix. In matrix form (A.4.08) is written as

$$a_{gi} I^{l+1}_{0gi-\frac{1}{2}} + b_{gi} I^{l+1}_{0gi+\frac{1}{2}} + c_{gi} I^{l+1}_{0gi+\frac{3}{2}} = R^{l+1}_{gi}$$

(A.8.01)

where coefficients from (A.4.08) for i=2..I are given as

$$a_{gi} = -\frac{1}{3\hat{\sigma}_{gi-1}} + \frac{\hat{\sigma}_{gi-1}}{4} b_{0gi-1}$$

$$b_{gi} = \frac{1}{3\hat{\sigma}_{gi}} + \frac{1}{3\hat{\sigma}_{gi-1}} + \frac{\hat{\sigma}_{gi} b_{0gi}}{4} + \frac{\hat{\sigma}_{gi-1} a_{0gi-1}}{4}$$

$$c_{gi} = -\frac{1}{3\hat{\sigma}_{gi}} + \frac{\hat{\sigma}_{gi} a_{0gi}}{4}$$

$$R_{gi}^{l+1} = \frac{1}{4}\Sigma_i^{*} \left[\chi_{gi-1}\eta_{i-1}\Delta x_{i-1}\Sigma_{g} \; \sigma_{gi-1}(a_{0gi-1} I^{l+1}_{0gi-\frac{1}{2}} + b_{0gi} I^{l+1}_{gi-\frac{3}{2}})\right] + S^{l+\frac{1}{2}}_{gi-1}$$

(A.8.02a,b,c,d)

The expression for left and right boundaries from (A.6.07) are

$$a_{gl} = 0.0$$

$$b_{gl} = \Sigma_{\mu_m>0}\omega_m\mu_m + \frac{1}{3\hat{\sigma}_{gl}} + \frac{1}{4}\hat{\sigma}_{gl} b_{0gl}$$

$$c_{gl} = -\frac{1}{3\hat{\sigma}_{gl}} + \frac{1}{4}\hat{\sigma}_{gl} a_{0gl}$$

$$R_{gl}^{l} = I^{l+\frac{1}{2}}_{0g\frac{1}{2}}\Sigma_{\mu_m>0}\omega_m\mu_m + I^{l+\frac{1}{2}}_{1g\frac{1}{2}} + \frac{\Delta x_1 Q_{0gl}}{2} - \frac{\Delta x_1 Q_{1gl}}{\hat{\sigma}_{gl}}$$
$$+ \frac{2}{3\hat{\sigma}_{gl}}(I^{l+\frac{1}{2}}_{2g\frac{3}{2}} - I^{l+\frac{1}{2}}_{2g\frac{1}{2}}) + \frac{1}{4}\; \chi_{gl}\eta_1\Delta x_1\Sigma_{g} \; \sigma_{g1}(a_{0g1} I^{l+1}_{0g\frac{3}{2}} + b_{0g1} I^{l+1}_{0g\frac{1}{2}})$$

(A.8.03a,b,c,d)

$$a_{gI+1} = \frac{1}{4}\hat{\sigma}_{gI} b_{0gI-1} - \frac{1}{3\hat{\sigma}_{gI-1}}$$

$$b_{gI+1} = -\Sigma_{\mu_m<0}\omega_m\mu_m + \frac{1}{3\hat{\sigma}_{gI-1}} + \frac{1}{4}\hat{\sigma}_{gI-1} a_{0gI-1}$$

$$c_{gI+1} = 0.0$$

$$R_{gI+1} = -I^{l+\frac{1}{2}}_{0gI+\frac{1}{2}}\Sigma_{\mu_m<0}\omega_m\mu_m - I^{l+\frac{1}{2}}_{1gI+\frac{1}{2}} + \frac{\Delta x_I Q_{0gI}}{2} + \frac{\Delta x_I Q_{1gI}}{\hat{\sigma}_{gI}}$$
$$- \frac{2}{3\hat{\sigma}_{gI}}(I^{l+\frac{1}{2}}_{2gI+\frac{1}{2}} - I^{l+\frac{1}{2}}_{2gI-\frac{1}{2}}) + \frac{1}{4}\; \chi_{gI}\eta_I\Delta x_I\Sigma_{g} \; \sigma_{gI}(a_{0gI} I^{l+1}_{0gI+\frac{1}{2}} + b_{0gI} I^{l+1}_{gI-\frac{1}{2}})$$

(A.8.04a,b,c,d)

In the source iteration method a new solution is substituted in (A.8.02d;3d) and the matrix is inverted again for the new solution. This is repeated until some desired convergence is achieved. To accelerate the diffusion equation we solve the grey equation instead of substituting the new solution on the right side of the diffusion equation. The Grey equation is solved in like manner to the diffusion equation. The set of equations is given as

$$a_i I_{\alpha i-\frac{1}{2}}^{k+\frac{1}{2}} + b_i I_{\alpha i+\frac{1}{2}}^{k+\frac{1}{2}} + c_i I_{\alpha i+\frac{3}{2}}^{k+\frac{1}{2}} = R_i^k$$

(A.8.05)

where coefficients from (A.5.04) for i=2..N are given as,

$$a_i = -\frac{1}{3}\Sigma_g \frac{\varphi_{gi-1}}{\hat{\sigma}_{gi-1}} + \frac{1}{4}\xi_{i-1} b_{\alpha i-1}$$

$$b_i = \frac{1}{3}\Sigma_g \left[\frac{\varphi_{gi}}{\hat{\sigma}_{gi}} + \frac{\varphi_{gi-1}}{\hat{\sigma}_{gi-1}}\right] + \frac{1}{4}\left[\xi_i b_{\alpha i} + \xi_{i-1} a_{\alpha i-1}\right]$$

$$c_i = -\frac{1}{3}\Sigma_g \frac{\varphi_{gi}}{\hat{\sigma}_{gi}} + \frac{1}{4}\xi_i a_{\alpha i}$$

For linear case

$$R_i^{k+\frac{1}{2}} = \Sigma_g \frac{1}{3\hat{\sigma}_{gi+1}}\left(I_{0gi+\frac{3}{2}}^{k+\frac{1}{2}} - I_{0gi+\frac{1}{2}}^{k+\frac{1}{2}}\right) - \Sigma_g \frac{1}{3\hat{\sigma} gi}\left(I_{0gi+\frac{1}{2}}^{k+\frac{1}{2}} - I_{0gi-\frac{1}{2}}^{k+\frac{1}{2}}\right)$$
$$- \frac{1}{3}\Sigma_g \frac{\varphi_{gi+1}}{\hat{\sigma}_{gi+1}}\left(I_{\alpha i+\frac{3}{2}}^{k+\frac{1}{2}} - I_{\alpha i+\frac{1}{2}}^{k+\frac{1}{2}}\right) + \frac{1}{3}\Sigma_g \frac{\varphi_{gi}}{\hat{\sigma}_{gi}}\left(I_{\alpha i+\frac{1}{2}}^{k+\frac{1}{2}} - I_{\alpha i-\frac{1}{2}}^{k+\frac{1}{2}}\right)$$
$$-\frac{1}{4}\Sigma_g\left[\xi_{gi+1} f_{gi+1}^{k+\frac{1}{2}} + \xi_g f_{gi}^{k+\frac{1}{2}}\right] + \frac{1}{4}\xi_i\left[a_{\alpha i} I_{\alpha i+\frac{1}{2}}^{k+\frac{1}{2}} + b_{\alpha i} I_{\alpha i-\frac{1}{2}}^{k+\frac{1}{2}}\right]$$
$$+ \frac{1}{4}\xi_{i+1}\left[a_{\alpha i+1} I_{\alpha i+\frac{3}{2}}^{k+\frac{1}{2}} + b_{\alpha i+1} I_{\alpha i+\frac{1}{2}}^{k+\frac{1}{2}}\right] + \Sigma_g S_{gi}^{k+\frac{1}{2}}$$

For non linear case

$$R_i^{k+\frac{1}{2}} = \Sigma_g S_{gi}^{k+\frac{1}{2}}$$

(A.8.06a,b,c,d)

At the left and right boundaries from (A.7.01;02) we have

$$a_1 = 0.0$$

$$b_1 = \frac{1}{3}\Sigma_g \frac{\varphi_{gl}}{\hat{\sigma}_{gl}} + \frac{1}{4}\xi_1 b_{01} + \Sigma_{\mu_m>0}\omega_m\mu_m$$

$$c_1 = -\frac{1}{3}\Sigma_g \frac{\varphi_{gl}}{\hat{\sigma}_{gl}} + \frac{1}{4}\xi_1 a_{01}$$

For linear case

$$\begin{aligned} R_1^k = {} & I_{0\frac{1}{2}}^{l+\frac{1}{2}}\Sigma_{\mu_m>0}\omega_m\mu_m + I_{1\frac{1}{2}}^{l+\frac{1}{2}} + \Sigma_g \frac{1}{3\hat{\sigma}_{gl}}\left(I_{0g\frac{3}{2}}^{k+\frac{1}{2}} - I_{0g\frac{1}{2}}^{k+\frac{1}{2}}\right) \\ & - \frac{1}{3}\Sigma_g \frac{\varphi_{gl}}{\hat{\sigma}_{gl}}\left(I_{0\frac{3}{2}}^{k+\frac{1}{2}} - I_{0\frac{1}{2}}^{k+\frac{1}{2}}\right) + \Sigma_g \frac{2}{3\hat{\sigma}_{gl}}\left(I_{2g\frac{3}{2}}^{l+\frac{1}{2}} - I_{2g\frac{1}{2}}^{l+\frac{1}{2}}\right) + \frac{\Delta x_1}{2}\Sigma Q_{0gl} \\ & - \Delta x_1 \Sigma_g \frac{Q_{1gl}}{\hat{\sigma}_{gl}} - \frac{1}{4}\Sigma_g \xi_g f_{gl}^{k+\frac{1}{2}} + \frac{1}{4}\xi_1\left(a_{01} I_{0\frac{3}{2}}^{k+\frac{1}{2}} + b_{01} I_{0\frac{1}{2}}^{k+\frac{1}{2}}\right) \end{aligned}$$

For non linear case

$$\begin{aligned} R_1^k = {} & I_{0\frac{1}{2}}^{l+\frac{1}{2}}\Sigma_{\mu_m>0}\omega_m\mu_m + I_{1\frac{1}{2}}^{l+\frac{1}{2}} + \Sigma_g \frac{2}{3\hat{\sigma}_{gl}}\left(I_{2g\frac{3}{2}}^{l+\frac{1}{2}} - I_{2g\frac{1}{2}}^{l+\frac{1}{2}}\right) \\ & + \frac{\Delta x_1}{2}\Sigma_g Q_{0gl} - \Delta x_1 \Sigma_g \frac{Q_{1gl}}{\hat{\sigma}_{gl}} \end{aligned}$$

$$a_{I+1} = -\frac{1}{3}\Sigma_g \frac{\varphi_{gI}}{\hat{\sigma}_{gI}} + \frac{1}{4}\xi_I b_{0I}$$

$$b_{I+1} = \frac{1}{3}\Sigma_g \frac{\varphi_{gI}}{\hat{\sigma}_{gI}} + \frac{1}{4}\xi_I a_{0I} - \Sigma_{\mu_m<0}\omega_m\mu_m$$

$$c_{I+1} = 0.0$$

For linear case

$$\begin{aligned} R_{I+1}^{k+\frac{1}{2}} = {} & -I_{0I+\frac{1}{2}}^{l+\frac{1}{2}}\Sigma_{\mu_m<0}\omega_m\mu_m - I_{1I\frac{1}{2}}^{l+\frac{1}{2}} - \Sigma_g \frac{1}{3\hat{\sigma}_{gI}}\left(I_{0gI+\frac{1}{2}}^{k+\frac{1}{2}} - I_{0gI-\frac{1}{2}}^{k+\frac{1}{2}}\right) \\ & + \Sigma_g \frac{\varphi_{gI}}{3\hat{\sigma}_{gI}}\left(I_{0I+\frac{1}{2}}^{k+\frac{1}{2}} - I_{0I-\frac{1}{2}}^{k+\frac{1}{2}}\right) - \Sigma_g \frac{2}{3\hat{\sigma}_{gI}}\left(I_{2gI+\frac{1}{2}}^{l+\frac{1}{2}} - I_{2gI-\frac{1}{2}}^{l+\frac{1}{2}}\right) + \frac{\Delta x_I}{2}\Sigma Q_{0gI} \\ & + \Delta x_I \Sigma_g \frac{Q_{1gI}}{\hat{\sigma}_{gI}} - \frac{1}{4}\Sigma_g \xi_g f_{gI}^{k+\frac{1}{2}} + \frac{1}{4}\xi_I\left(a_{0I} I_{0I+\frac{1}{2}}^{k+\frac{1}{2}} + b_{0I} I_{0I-\frac{1}{2}}^{k+\frac{1}{2}}\right) \end{aligned}$$

For non linear case

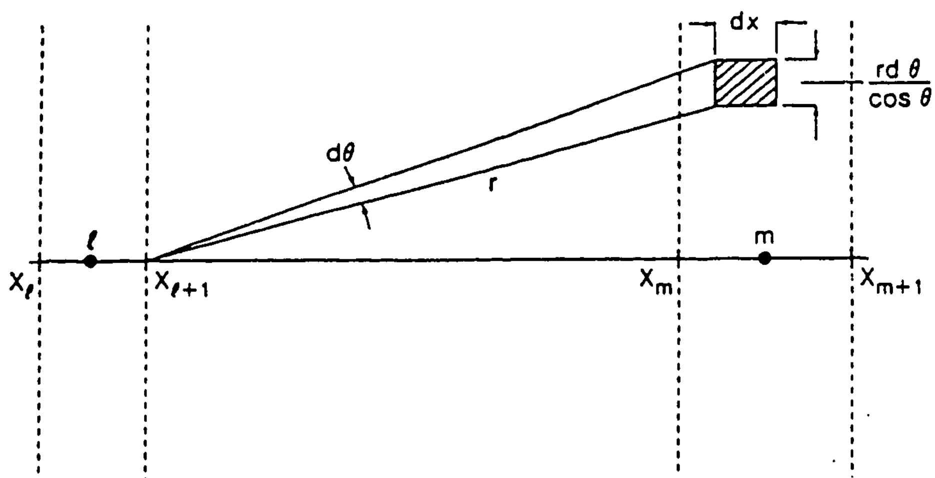
$$\begin{aligned} R_{I+1}^{k+\frac{1}{2}} = {} & -I_{0I+\frac{1}{2}}^{l+\frac{1}{2}}\Sigma_{\mu_m<0}\omega_m\mu_m - I_{1I+\frac{1}{2}}^{l+\frac{1}{2}} - \Sigma_g \frac{2}{3\hat{\sigma}_{gI}}\left(I_{2gI+\frac{1}{2}}^{l+\frac{1}{2}} - I_{2gI-\frac{1}{2}}^{l+\frac{1}{2}}\right) \\ & + \frac{\Delta x_I}{2}\Sigma_g Q_{0gI} + \Delta x_I \Sigma_g \frac{Q_{1gI}}{\hat{\sigma}_{gI}} \end{aligned}$$

(A.8.07a,b,c;A.8.08a,b,c,d)

B. MARCHAND'S MODEL

Conceptually Marchand's approach is quite similar to the approach used by Audenaerde et al[B/1] to solve the neutron particle transport problem. The details of the mathematics involved can be found elsewhere[B/2]. The following discussion emphasises the conceptual understanding of Marchand's model in it's exact form and then the two-cell simplified model.

In order to elucidate Marchand's algorithm, consider the geometry shown in the figure below.



Using the analogy of the Andemaerde analysis, the radiation power intensity that originates from a ring of radius R of thickness dx in cell m and is incident on the right boundary of lth cell is

$$dP_{l+1}^{m}(\mu,\nu)d\mu d\nu = (2\pi R dR dx)d\nu \; n_m J_m \frac{\mu}{r^2} e^{-\tau(x_{l+1},x)/\mu}$$

(B.01)

where $\mu = \cos\theta$ and τ is the optical depth between x_{l+1} and x, and can be expressed as

$$\tau(x_{l+1},x)=\tau(x_{l+1},x_m)+\tau(x_m,x) = \tau_{lm}+\frac{\tau(x-x_m)}{\Delta x_m}$$

(B.02)

where $\tau_m = n_m\sigma_m\Delta x_m$, n_m is the ion density, and I_m is the power radiated per atom, per unit frequency interval $d\nu$ per unit solid angle in cell m.

The total radiation specific intensity from cell m incident on interface x_{l+1} is

$$P_{l+1}^m(\mu,\nu)=\int_0^{\Delta x_m} dP_{l+1}^m(\mu,\nu)dx$$

(B.03)

By using equation (B.01;02) in (B.03), (B.03) reduces to

$$P_{l+1}^m\ d\mu d\nu=\frac{2\pi I_m}{\sigma_m}\mu d\mu d\nu\ \{e^{-\frac{\tau_{lm}}{\mu}}\}\ \{1-e^{-\frac{\tau_m}{\mu}}\}$$

(B.04)

We can represent the contribution due to all cells left of cell l as

$$P_{l+1}^m\ d\mu=\sum_{m>l}\frac{2\pi I_m}{\sigma_m}\mu d\mu\ \{e^{-\frac{\tau_{lm}}{\mu}}\}\{1-e^{-\frac{\tau_m}{\mu}}\}$$

(B.05)

To compute the total intensity reaching position x_{l+1}(the right side of the cell) from all directions, we integrate over μ

$$Q_{l+1}^R=\int_0^1 d\mu P_{l+1}^R$$

$$Q_{l+1}^R=\sum_{m>l}\frac{2\pi I_m}{\sigma_m}\mu[E_3(\tau_{lm})-E_3(\tau_{lm}+\tau_m)]$$

$$\textit{where} \qquad E_3(\tau) = \int_1^\infty \frac{dz}{z^3}e^{-\tau z}$$

(B.06)

where E_3 is the exponential integral of order three. Similarly, the contribution from all directions at the left boundary of the cell is

$$Q_{l+1}^{L}=\sum_{m\leq l}\frac{2\pi I_m}{\sigma_m}\mu[E_3(\tau_{l+1,m})-E_3(\tau_{l+1,m}+\tau_m)]$$

(B.07)

Then using the fact that energy is conserved, the net gain/loss of energy per unit volume and per unit energy in lth cell is

$$W_l=\frac{[Q_{l+1}^{R}-Q_l^{R}+Q_l^{L}-Q_{l+1}^{L}]}{\Delta x}$$

$$W_l=-\{\frac{4\pi I_l}{\Delta x\sigma_l}[\frac{1}{2}-E_3(\tau_l)]\}+\sum_{m\neq l}\frac{2\pi I_m}{\sigma_l\Delta x}\{E_3(\tau_{lm})-E_3(\tau_{lm}+\tau_l)$$
$$-E_3(\tau_{lm}+\tau_m)+E_3(\tau_{lm}+\tau_l+\tau_m)\}$$

(B.08)

Here the first term represents the power emitted out of cell l and the second term represents the power (emitted from all other cells and) absorbed by the lth cell. Unfortunately, due to coupling of each zone with other neighbouring zones, the simulation run times are significantly larger than those in which radiation is neglected.

In order to reduce the radiation time, Marchand developed an intuitive approach to solve the above problem. He claimed that one can approximate the plasma as being either optically thin or optically thick. In the limit where the plasma is optically thin, the intensity coming from the right (or left) of a given cell is the same as that produced by a single cell of luminosity equal to the sum of intensities emitted from all cells (since nothing is absorbed), that is

$$P_{l+1}=2\pi Z \qquad \textit{where} \qquad Z=\Sigma_m I_m\Delta x_m$$

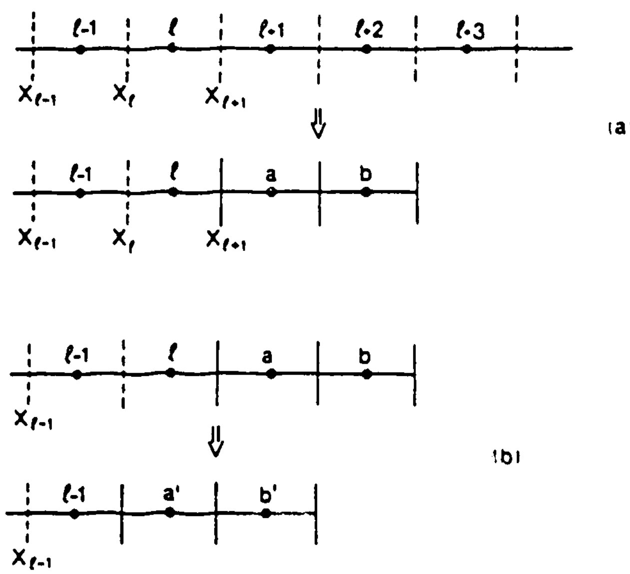
(B.09)

For the optically thick case, one treats it as if it is comprised of two cells, one optically thick (emission is negligible) and the other optically thin (here absorption is negligible), then the intensity emitted out of the original cell is

$$P(\mu)=2\pi Z_b e^{-\frac{\tau_a}{\mu}} \qquad \textit{where} \qquad Z_b = I_b \Delta x_b$$

(B.10)

where cell "a" is non-radiating with optical thickness τ_a and cell "b" is optically thin and radiates with luminosity Z_b. Marchand called this approach of reducing n cells into two effective cells the **recursive approach**. The figure below illustrate what is meant by recursive approach.



In the recursive approach one assumes that the radiation intensity everywhere in the plasma can be approximated by the intensity generated by a combination of two cells, one optically thick and one optically thin. Following the preceding analysis, the intensity incident on the right boundary of the lth cell from the right is

$$P_l^R = 2\pi Z_{b'} e^{-\frac{\tau_{a'}}{\mu}} = 2\pi\mu\frac{I_l}{\sigma_l}(1-e^{-\frac{\tau_l}{\mu}}) + Z_b e^{-\frac{(\tau_a+\tau_l)}{\mu}}$$

(B.11)

here the unknowns $Z_{b'}$ and $\tau_{a'}$ are chosen so that the dominant angular dependence of the specific intensity is around $|\mu| = 1$; because in the cold region the specific intensity is strongly peaked around $|\mu| = 1$. With this simplification the number of computational operations is proportional to the number of cells N rather than N^2 as it is for the exact model.

To physically justify the applicability of his algorithm Marchand made some assumptions[B/3]. He assumed that for problems in consideration, the time scale which characterizes macroscopic changes of the medium is long compared to the typical photon transit time; hence to a good approximation radiation transport can be considered as a stationary problem. The range of energies is restricted to 30 ev $< h\nu <$ 10 Kev for which photon scattering mean free paths are typically much longer than the absorption mean free path, implying scattering is negligible. In this limit, it is possible to describe the radiation transport in terms of photons being produced isotropically and propagating in straight lines until they are absorbed. The absorption probability is prescribed by the local absorption cross-section. Therefore, Marchand's model is applicable for problems in which time dependence or scattering can be ignored. The angular dependence of the intensity is ignored. Although our present radiation transport code also ignores scattering, the technique that is used to solve the radiative transfer equation can be made applicable ,with appropriate modification, to problems involving scattering.

C. LEE'S RADIATIVE TABLES

Spectral-line intensities from laser produced plasmas are useful as temperature and density diagnostics. These emission spectra allow one to test the accuracy of the theoretical rate coefficients. However, before one can predict the emission spectra, one needs to know the plasma's charge state distribution and ion-level population as a function of it's electron temperature and density.

The tables of emission spectra used in MEDUSA are produced by Y.T. Lee et al at Lawrence Livermore National Laboratory. A detailed description of the model used for Lee's table can be found elsewhere[C/1]. Here the formulation and regions of validity of these tables is briefly discussed.

Since the laser produced plasmas of interest are produced with electron density ranging from 10^{17} to 10^{21} cm^{-3}, neither the coronal nor the Saha-Boltzmann equilibrium models are applicable. The coronal model gives ionization balance for ions in their ground states, hence it's valid for plasmas with electron densities less than 10^{16} cm^{-3}. On the other hand, Saha-Boltzmann equation is only valid for higher density plasmas. The coronal model assumes that all ions in the plasma are in their ground state. The local thermodynamic equilibrium (LTE) model significantly over estimates the average ionization state at all temperature and the coronal model underestimates the ionization state at low temperature by neglecting excited sate ionization. Lee uses a kinetic model injunction with coronal and Saha-Boltzmann model to calculate the time dependent ionization balance and ion-level populations of the non-LTE plasma in the electron density range of interest. This model reduces to coronal equilibrium at low density and Saha-Boltzmann equilibrium at high density.

The ground-state and excited-state energy levels for each ionization stage are generated from screening constant calculated by W. Lokke et al and R. More. There is one energy level per principal quantum number, and quantum lowering is obtained from a modified Stewart and Pyatt formula to account for dense plasma effects. The following atomic process are included in the model

i. excitation and de-excitation collisions

ii. Radiative spontaneous emission

iii. Ionization due to collision

iv. Radiative recombination

v. Three body recombination

vi. Dielectronic recombination

The model assumes that each energy level couples with every level of the same ionization stage, but only to the ground state of the next ionization stage. The contribution of electron-impact ionization from excited levels is also taken into account.

The results of ionization balance model are used to normalized the relative populations that are determined by the balance of collisions and spontaneous radiative transitions to calculate spectral line intensity (of L shell transition) as a function of electron temperature and density. In the calculation, it is assumed that the effects of ionization and radiative recombination on the excited-state population are negligible, since electron collision rates are very large (of the order of 2) compare to both the ionization and radiative recombination rates at the temperatures and densities under consideration.

The model assumes that the plasma has small temperature and density gradients, and the effect of radiation transport is neglected. In situations where radiation transport and gradients are not negligible, the model is still useful in setting an upper bound for experimental data.

```
C2345678101234567820123456783012345678401234567850123456786012345678 7012
C    ****************************************************************
C  *    RADIATIVE TRANSPORT CODE BY HUMAYON BUTT SEPT 92     *
C  *  TECHNIQUE USED IS CALLED DIFFUSION SYNTHETIC ACCELERATION  *
C  *         SEE M.Sc THESIS FOR DETAILS              *
C  * DOCUMENTATION IS SHOWN IN LOWER CASE BUT ALL VARIABLES ARE  *
C  * MEANT TO BE IN UPPER CASE, f.e. (FOR EXAMPLE) IF YOU WANT TO  *
C  * SCAN FOR "nrun" SCAN "NRUN"                      *
C    ****************************************************************
C
      SUBROUTINE SATE

      PARAMETER (NANG=24,NFG=64,NCL=151)
      PARAMETER (RK=7.560667E-16,PK=6.6261E-34)
      PARAMETER (BK=1.3807E-23,SP=3.0E08,EVE=1.6022E-19)
      PARAMETER (TCF=EVE/BK,PI=3.141593,A0=PK/(BK*TCF))

      INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      INTEGER DRT, DIFBC, DIFACC, IIT, TITA
      INTEGER INSB, IRUN, LIN, MEDUSA
      INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      INTEGER NRN, NCODE, NCOUNT, PRNT
      INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      INTEGER USC, WFN, CONV, DBXL, DBXU, NUM
      INTEGER NCP1, NCM1, NAD2
      INTEGER SMTST1, SMTST2, SMTST3, SMTST4, SMTST5
      INTEGER IT
      INTEGER MDT, NITT

      COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
      COMMON /I3/ INSB, IRUN, LIN, MEDUSA
      COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM
      COMMON /I8/ NCP1, NCM1, NAD2
      COMMON /SMT/ SMTST1, SMTST2, SMTST3, SMTST4, SMTST5
      COMMON /IMD/ MDT, NITT, RTEON

      REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFG,NCL), ATEM(NCL)
      REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
      REAL*8 BETA(NCL), BO(NFG,NCL)
      REAL*8 CV(NCL), CW, CHI(NFG,NCL)
      REAL*8 DL(NCL), DINT(NFG,NCL + 1), PRNTM
      REAL*8 DIST(NCL), DTIME, DSPR
      REAL*8 ETA(NCL)
      REAL*8 FINT(NFG,NCL + 1), FINL(NFG), FINR(NFG)
      REAL*8 FR(NFG + 1), FQ(NFG,NCL)
      REAL*8 HILIM
      REAL*8 INLB(NFG), INRB(NFG), ITSL, IERED(NCL), IERR
      REAL*8 INTB(NANG,NFG,NCL + 1), IINT(NANG,NFG,NCL), IZIN(NFG,NCL)
      REAL*8 ITRB, ITER(NFG,NCL)
      REAL*8 L, LFL, LOWLIM
      REAL*8 MAXT
      REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
      REAL*8 NFFT(NANG,NFG,NCL), NDIN(NFG,NCL + 1), NZIN(NFG,NCL)
      REAL*8 PMAC(NCL), PINT(NANG,NFG,NCL), PRTIME
      REAL*8 PGIN(NCL + 1), PFGI(NCL + 1), PLANK(NFG,NCL)
      REAL*8 PDIN(NFG,NCL + 1), PITE(NFG,NCL), PZIN(NFG,NCL)
      REAL*8 PAB(NCL)
      REAL*8 Q(NANG,NFG,NCL)
      REAL*8 SPH, SIRC, SPR
      REAL*8 SIGM(NFG,NCL), GRSI(NCL + 1)
      REAL*8 SOLU(NCL + 1), SPRB
      REAL*8 S(NFG,NCL), SUM1, SUM2, GRPQ(NCL), GRZQ(NCL)
      REAL*8 SINT(NFG,NCL + 1)
      REAL*8 TTIME, TOU, TERR, TELB
      REAL*8 TRNFVR(NCL), TEMP(NCL)
      REAL*8 UFL
      REAL*8 WF(NANG)
      REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER

      COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
      COMMON /R02/ ABC, AB, AO
      COMMON /R03/ BETA, BO
      COMMON /R04/ CV, CW, CHI
      COMMON /R05/ DL, DINT, PRNTM
      COMMON /R06/ DIST, DTIME, DSPR
      COMMON /R07/ ETA
      COMMON /R071/ FINT, FINL, FINR
      COMMON /R08/ FR, FQ
      COMMON /R09/ HILIM
      COMMON /R10/ INLB, INRB, ITSL, IERED, IERR
      COMMON /R11/ INTB, IINT, IZIN
      COMMON /R12/ ITRB, ITER
      COMMON /R13/ L, LFL, LOWLIM
      COMMON /R14/ MAXT
      COMMON /R15/ NATE, NINT
      COMMON /R16/ NFFT, NDIN, NZIN
      COMMON /R17/ PMAC, PINT, PRTIME
      COMMON /R18/ PGIN, PFGI, PLANK
      COMMON /R181/ PDIN, PITE, PZIN
      COMMON /R182/ PAB
      COMMON /R19/ Q
      COMMON /R20/ SPH, SIRC, SPR
      COMMON /R21/ SIGM, GRSI
      COMMON /R22/ SOLU, SPRB
      COMMON /R23/ S, SUM1, SUM2, GRPQ, GRZQ
      COMMON /R231/ SINT
      COMMON /R24/ TTIME, TOU, TERR, TELB
      COMMON /R25/ TRNFVR, TEMP
      COMMON /R26/ UFL
      COMMON /R27/ WF
      COMMON /R28/ ZLP, ZINT, ZQ, DIER

      LOGICAL RTEON, TRN, DIFF, LOR, NLOR, SIMRUN, SMTST
      COMMON /L1/ TRN, DIFF, LOR, NLOR, SIMRUN, SMTST
      COMMON /RDC/ NRN, NCODE, NCOUNT, PRNT

C*** fsig & fplnk are analytical expressions of sigma and
C  planckian function [eqn (4.2.09 & 4.2.07)]
      EXTERNAL FSIG, FPLNK, F2, F3

      IF (NCODE .EQ. 0) THEN
C**** ncode is set to 0 only once to read the input file "satedata" when
C   this code is used to impelment radiative effects in medusa, when
C   this code returns medusa ncode is set to 1.
C   "open" files related to this radiation code
C   "pcrdd" input data file for the radiation code
C   "analysis" error messages when this code becomes unstable
C   "tmp" contains the temperature for the model problem
C   "result" contains values of the variables according to pres value
C**** parameter.
      OPEN (UNIT=4,FILE='satedata',STATUS='OLD')
      OPEN (UNIT=99,FILE='analysis',STATUS='OLD')
      OPEN (UNIT=8,FILE='result',STATUS='OLD')
      OPEN (UNIT=7,FILE='hflux',STATUS='OLD')
      OPEN (UNIT=11,FILE='energy',STATUS='OLD')

C**** nrun=1 different runs with different time steps keeping everything
C   else same, only work for the model problem. Turned off for now.
      NRUN = 0
C**** scr=1 output if any to the files result, otherwise on the screen
C   num a debug varible - used to check consistency in SI & DSA
C   dbxl & dbxu used to print a variable(s) after dbxl # of iteration
C   until dbxu # of iteration, # of variable depends on the value of
C**** pres. tcont=1 means no parameter iteration.
      WRITE (6,*)'enter SCR NUM DBXL DBXU TCONT'
      READ (4,10) SCR, NUM, DBXL, DBXU, TCONT
   10 FORMAT (50X, I2, 1X, I2, 1X, I4, 1X, I4, 1X, I2)
      READ (4,20) SIMRUN
   20 FORMAT (66X, L2)
      WRITE (8,30) SCR, NUM, DBXL, DBXU, TCONT
   30 FORMAT (1X, 'SRC NUM DBXL DBXU TCONT =', 5(1X,I4))
      WRITE (8,40) SIMRUN
   40 FORMAT (1X, L2)
      IF (SCR .NE. 0) THEN
C**** place output in "satscr" instead of on the screen
       OPEN (UNIT=6,FILE='satscr',STATUS='OLD')
      END IF
C**** read initial data for the radiation code
C*******************************
      CALL DATAIN
C*******************************
C??     PRNT = PRES
      NCP1 = NC + 1
      NCM1 = NC - 1
      NAD2 = NA / 2
      SMTST = .FALSE.
      PRNTM = 1.00E-12
C**** read legendre polynomial's zeros and weight function
C*******************************
      CALL LPL
C*******************************
C**** initialize the iteration counters
      NCNT = 0
      TIT = 0
      OIT = 0
      IIT = 0
      TOIT = 0
      TIIT = 0
      COUNT = 0
      DCOUNT = 0
      NDICNT = 0
     END IF
     IT = 0
   50 TTIME = 0.0
     IF (SIMRUN) THEN
C**** this will simulate all possible combinations of sl, dsa without
C   grey and with (linear and nonlinear) grey with different bounds
C   (2,5,20,50,100,500) at different time steps (1,2,5,10,20,50,100,
C   200,500,1000) pico seconds. note this switch over rides any
C**** initial mode of acceleration.
      TRNACC = 0
      DIFACC = 0
      LIN = 0
      TRN = .FALSE.
      DIFF = .FALSE.
      LOR = .FALSE.
      NLOR = .FALSE.
C*******************************
      CALL RUNTST(IT)
C*******************************
C**** sets above simulation paprameters
     END IF
     CONT = 0
   60 CONTINUE
   70 IF (MEDUSA .EQ. 1) THEN
C**** get data necessary for medusa simulation
C   MDT acts as a switch to execute a part of tables subroutine
C   this subroutine reads the radiative tables and evaluates the
C   variables for the RTE, it's function is same as the model
C**** subroutine for the model problem.
      MDT = 0
C*******************************
      CALL TABLES
C*******************************
      MDT = 1
      IF (NITT .NE. 1 .OR. ( .NOT. RTEON)) RETURN
     ELSE
```

```
C**** define the model problem
C*****************************
      CALL MODEL
C*****************************
     END IF

C**** RT used as reset parameter for computation of spectral radius of
C     the transfer equation
C     smtst# used to prevent the code from falling apart, these switches
C     stop further computation of an equation when it appears to be
C**** failing to converge or starts to diverge.
     RT = 0
  80 SMTST1 = 0
     SMTST2 = 0
     SMTST3 = 0
     SMTST4 = 0
     SMTST5 = 0

     TOIT = TOIT + 1
     IF (TOIT .GT. OITA) THEN
C**** compute new temperature
C*****************************
      CALL TEMPER
      CALL CPUTM
C*****************************
      WRITE (8,*)'program stoped because oita exceeded', TOIT, OITA
      STOP
     END IF
     IIT = 0
C**** DRT used as reset parameter for computation of spectral radius
C**** of the diffusion equation
     DRT = 0
C*****************************
C**** solve radiative transfer equation
     CALL TRNPRT
C*****************************
     OIT = OIT + 1
C**** check if radiative transfer equation is converged

     CALL TRCONV
C*****************************
     DO 110 J = 1, NG
      DO 100 K = 1, NC
       DO 90 I = 1, NA
        PINT(I,J,K) = NINT(I,J,K)
  90   CONTINUE
 100  CONTINUE
 110 CONTINUE
     IF (CONV .EQ. 0 .OR. OIT .GE. TB .OR. (SMTST)) THEN
C**** transport intensity is converged or transport bound is exceeded or
C**** smtst is true
      IF (SMTST) THEN
       WRITE (99,*)'outer iteration terminated due to spr .ge. 1.00 t
    1o often'
       SMTST = .FALSE.
      END IF
      GO TO 290
     END IF
     IF (TRNACC .EQ. 0) THEN
C*****************************
      CALL TRNSI
C**** solve transfer equation using source iteration
C*****************************
      GO TO 80
     ELSE IF (TRNACC .EQ. 1) THEN
      IIT = 0
 120  TIIT = TIIT + 1
      DCOUNT = DCOUNT + NDICNT
      NDICNT = 0
C**** solve multifrequency diffusion equation
C*****************************
      CALL MFDE
C*****************************
C**** check if multifrequency diffusion equation is converged
C*****************************
      CALL DFCONV
C*****************************
      IIT = IIT + 1
      IF (IIT .GE. DB .OR. CONV .EQ. 0 .OR. (SMTST)) THEN
C**** diffusion equation is converged or diffusion bound is exceeded or
C**** smtst is true
       IF (NUM .EQ. 1) THEN
C**** used to compare converged accelerated and unaccelerated diffusion
C     intensity, i.i. (that is) the solution of diffusion equation with
C**** SI and with grey diffusion equation
        WRITE (8,*)'toit tiit oit iit tit', TOIT, TIIT, OIT, IIT,
    1        TIT
        WRITE (98,130) ((J,K,NDIN(J,K),K=1,NCP1),J=1,NG)
 130    FORMAT (3(1X,I2,1X,I2,1X,1PE10.3,1X))
        WRITE (*,*)'enter 1 to continue'
        READ (5,*) NUM
        IF (NUM .NE. 1) THEN
         STOP
        END IF
       END IF
       IF (SMTST) THEN
        SMTST = .FALSE.
        IF (SMTST2 .GE. NG*NC*0.05) THEN
         WRITE (99,*)'inner iteration terminated due to too often i
    1ll solution of'
         SMTST2 = 0
        ELSE IF (SMTST3 .GE. NG*NC*0.01) THEN
         WRITE (99,*)'inner iteration terminated due to dspr .ge. 0
    1.99 too often'
         SMTST3 = 0
        END IF
         WRITE (99,140) TOIT, TIIT, OIT, TIT
 140     FORMAT (1X, 'toit=', I8, 1X, 'tiit=', I4, 1X, 'oit=', I4,
    1        1X, 'tit=', I4)
        END IF
        NDZI = 0
        DO 170 J = 1, NG
         DO 160 K = 1, NC
          ZINT(J,K) = (AO(J,K)*NDIN(J,K + 1) + BO(J,K)*NDIN(J,K)) /
    1      2
          IF (ZINT(J,K) .LE. 0.0) THEN
           NDZI = NDZI + 1
           WRITE (99,150) J, K, ZINT(J,K), TOIT, TIT
 150       FORMAT (1X, 'zint from diff is neg. j k zint toit tit',
    1          2(I2,1X), 1PE10.3, 2(1X,I2))
          END IF
 160     CONTINUE
 170    CONTINUE
C****     WRITE (8,*)'zint neg. from diff this often',ndzi
       DO 190 K = 1, NC
        AB(K) = 0.0
        DO 180 J = 1, NG
         AB(K) = AB(K) + ABC(J,K) * ZINT(J,K)
 180    CONTINUE
 190   CONTINUE
       IF (ABVL .EQ. 1) THEN
C**** take absolute value of converged ndin when it is negative
        DO 210 K = 1, NC
         DO 200 J = 1, NG
          IF (NDIN(J,K) .LE. 0.0) THEN
           NDIN(J,K) = ABS(NDIN(J,K))
          END IF
 200     CONTINUE
 210    CONTINUE
       END IF
       DO 230 K = 1, NC
        IF (AB(K) .LE. 0.0) THEN
         WRITE (99,220) K, AB(K), TOIT, TIIT, OIT, TIT
 220     FORMAT (1X, 'ab(k) from diff is neg. may cause instability
    1', I2, 1X, 1PE11.3, 4(1X,I8))
        END IF
 230   CONTINUE
       GO TO 80
      END IF
 240  IF (DIFACC .LE. 1) THEN
C****    solve diffusion equation using source iteration
C*****************************
       CALL DIFFSI
C*****************************
      ELSE IF (DIFACC .EQ. 2) THEN
C**** solve diffusion equation using grey equation
C*****************************
       CALL GRDE
C*****************************
      END IF
      NEGD = 0
      DO 260 K = 1, NCP1
       DO 250 J = 1, NG
        IF (NDIN(J,K) .LE. 0) THEN
         NEGD = NEGD + 1
        END IF
 250   CONTINUE
 260  CONTINUE
C****      write(8,*)'ndin neg this often',negd
      DO 280 J = 1, NG
       DO 270 K = 1, NCP1
        DINT(J,K) = NDIN(J,K)
 270   CONTINUE
 280  CONTINUE
      GO TO 120
     END IF
C****   compute new temperature
 290 TIT = TIT + 1
     OIT = 0
     DO 320 K = 1, NC
      DO 310 J = 1, NG
       ZINT(J,K) = 0.0
       DO 300 I = 1, NA
        ZINT(J,K) = ZINT(J,K) + NINT(I,J,K) * WF(I)
 300   CONTINUE
 310  CONTINUE
 320 CONTINUE
     IF (NUM .EQ. 0) THEN
C****   used to compare converged accelerated and unaccelerated zint
C****   i.i. solution of the RTE with SI and with diffusion equation
      WRITE (8,*)'j k zint', TOIT, TIIT, OIT, IIT, TIT
      WRITE (98,330) ((J,K,ZINT(J,K),K=1,NC),J=1,NG)
 330  FORMAT (3(1X,I2,1X,I2,1X,1PE10.3,1X))
C*****************************
      CALL CPUTM
C*****************************
      WRITE (*,*)'enter 0 to continue'
      READ (5,*) NUM
      IF (NUM .NE. 0) THEN
       STOP
      END IF
     END IF
     DO 350 K = 1, NC
      AB(K) = 0.0
      DO 340 J = 1, NG
       AB(K) = AB(K) + ABC(J,K) * ZINT(J,K)
 340  CONTINUE
 350 CONTINUE
     DO 370 K = 1, NC
      IF (AB(K) .LE. 0) THEN
       WRITE (8,360) K, TOIT, OIT, TIT, AB(K)
 360   FORMAT (1X, 'temp may become neg. k toit oit tit ab(k)',
```

```
      1       4(1X,I8), 1X, 11PE10.3)
        END IF
  370 CONTINUE
C****    WRITE (*,*)'test tiit tit', TOIT, TIIT, TIT
      IF (MEDUSA .EQ. 0) THEN
C****    compute new temperature
C*****************************
        CALL TEMPER
C*****************************
C**** check if new temperature is converged
C*****************************
        CALL TEMCON
C*****************************
        IF (SIMRUN) THEN
         IF (CONT .EQ. 0) THEN
C*****************************
          CALL CPUTM
C*****************************
          CALL RUNTST(IT)
C*****************************
          GO TO 60
         ELSE
          GO TO 70
         END IF
        END IF
        IF (CONT .EQ. 0 .OR. TIT .GE. TITA) THEN
         IF (TIT .GE. TITA) THEN
          WRITE (8,380) TTIME
  380     FORMAT (1X, 'unconverged temperature at time:', 1PE11.4)
         ELSE
          WRITE (8,390) TTIME
  390     FORMAT (1X, 'converged temperature at time:', 1PE11.4)
         END IF
C*****************************
         CALL CPUTM
C*****************************
         IF (NRUN .EQ. 0) THEN
          STOP
         END IF
         NCNT = NCNT + 1
         IF (NCNT .EQ. 1) THEN
          DTIME = 1.00E-11
         ELSE IF (NCNT .EQ. 2) THEN
          DTIME = 1.00E-10
         ELSE IF (NCNT .EQ. 3) THEN
          DTIME = 2.00E-10
         ELSE IF (NCNT .EQ. 4) THEN
          DTIME = 4.00E-10
         ELSE IF (NCNT .EQ. 5) THEN
          DTIME = 6.00E-10
         ELSE IF (NCNT .EQ. 6) THEN
          DTIME = 1.00E-09
         ELSE IF (NCNT .EQ. 7) THEN
          DTIME = 1.60E-09
         ELSE IF (NCNT .EQ. 8) THEN
          DTIME = 2.00E-09
         ELSE IF (NCNT .EQ. 9) THEN
          DTIME = 3.00E-09
         ELSE IF (NCNT .EQ. 10) THEN
          DTIME = 5.00E-09
         ELSE IF (NCNT .EQ. 11) THEN
          DTIME = 1.00E-08
         ELSE IF (NCNT .EQ. 12) THEN
          DTIME = 1.00E-07
         ELSE
          STOP
         END IF
         TIT = 0
         OIT = 0
         IIT = 0
         TOIT = 0
         TIIT = 0
         COUNT = 0
         DCOUNT = 0
         NDICNT = 0
         GO TO 50
        ELSE
C**** solve the transfer equation again
         GO TO 70
        END IF
      ELSE
C*****************************
        CALL TABLES
C*****************************
      END IF
      END
C
C ******************************************************************
C *    read the input                                              *
C ******************************************************************
C
      SUBROUTINE DATAIN

      PARAMETER (NANG=24,NFQ=64,NCL=151)
      PARAMETER (RK=7.560667E-16,PK=6.6261E-34)
      PARAMETER (BK=1.3807E-23,SP=3.0E08,EVE=1.6022E-19)
      PARAMETER (TCP=EVE/BK,PI=3.141593,A0=PK/(BK*TCP))

      INTEGER SETLIM

      INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      INTEGER DRT, DIFBC, DIFACC, IIT, TITA
      INTEGER INSB, IRUN, LIN, MEDUSA
      INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      INTEGER USC, WFN, CONV, DBXL, DBXU, NUM

      COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
      COMMON /I3/ INSB, IRUN, LIN, MEDUSA
      COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM

      REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFQ,NCL), ATEM(NCL)
      REAL*8 DIST(NCL), DTIME, DSPR
      REAL*8 HILIM
      REAL*8 INLB(NFQ), INRB(NFQ), ITSL, IERED(NCL), IERR
      REAL*8 INTB(NANG,NFQ,NCL + 1), IINT(NANG,NFQ,NCL), IZIN(NFQ,NCL)
      REAL*8 ITRB, ITER(NFQ,NCL)
      REAL*8 L, LFL, LOWLIM
      REAL*8 MAXT
      REAL*8 PMAC(NCL), PINT(NANG,NFQ,NCL), PRTIME
      REAL*8 SPH, SIRC, SPR
      REAL*8 SOLU(NCL + 1), SPRB
      REAL*8 TTIME, TOU, TERR, TELB
      REAL*8 UFL
      REAL*8 ZLP(NANG), ZINT(NFQ,NCL), ZQ(NFQ,NCL), DIER

      COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
      COMMON /R06/ DIST, DTIME, DSPR
      COMMON /R09/ HILIM
      COMMON /R10/ INLB, INRB, ITSL, IERED, IERR
      COMMON /R11/ INTB, IINT, IZIN
      COMMON /R12/ ITRB, ITER
      COMMON /R13/ L, LFL, LOWLIM
      COMMON /R14/ MAXT
      COMMON /R17/ PMAC, PINT, PRTIME
      COMMON /R20/ SPH, SIRC, SPR
      COMMON /R22/ SOLU, SPRB
      COMMON /R24/ TTIME, TOU, TERR, TELB
      COMMON /R26/ UFL
      COMMON /R28/ ZLP, ZINT, ZQ, DIER

      WRITE (6,*)'ENTER "0" IF SOUCRE IS TO BE DEPENDENT ON TIME'
      READ (4,10) TEMPTD
   10 FORMAT (66X, I2)
      WRITE (8,20) TEMPTD
   20 FORMAT (1X, 'TEMPTD = ', I2)
      WRITE (6,*)'ENTER "0" IF INITIAL TEMPERATURE DIRIBUTION IS EXPON
     1ANTIAL'
      READ (4,30) TEMEXP
   30 FORMAT (66X, I2)
      WRITE (8,40) TEMEXP
   40 FORMAT (1X, 'TEMEXP = ', I2)
      READ (4,50) NFMUST
   50 FORMAT (66X, I2)
      WRITE (8,60) NFMUST
   60 FORMAT (1X, 'NFMUST = ', I2)
      WRITE (6,*)'ENTER "0 TO 20" FOR VARIABLE PRINTOUT'
      READ (4,70) PRES
   70 FORMAT (66X, I2)
      WRITE (8,80) PRES
   80 FORMAT (1X, 'PRES   = ', I2)
      WRITE (6,*)'ENTER BOUND FOR TRANSPORT AND DIFFUSION EQU'
      READ (4,90) TB, DB
   90 FORMAT (60X, I4, 1X, I3)
      WRITE (8,100) TB, DB
  100 FORMAT (1X, 'TB  DB = ', 1X, 2(I5))
      WRITE (6,*)'ENTER "0"IF IN NFF INTENSITY AT CELL BOUNDARY IS TO
     1BE'
      WRITE (6,*)'ZERO OR "1" IF IT IS TO BE CELL AVERAGE INTENSITY OR
     1'
      WRITE (6,*)'"2" IF MTS PAPER CRIETRION IS TO BE USED'
      READ (4,110) NFFC
  110 FORMAT (66X, I2)
      WRITE (8,120) NFFC
  120 FORMAT (1X, 'NFFC   = ', I2)
      WRITE (6,*)'ENTER IERR AND TERR AND DIER'
      READ (4,130) IERR, TERR, DIER
  130 FORMAT (41X, 3(1X,1PE8.2))
      WRITE (8,140) IERR, TERR, DIER
  140 FORMAT (1X, 'IERR   = ', 1PE10.3, 5X, 'TERR = ', 1PE10.3, 5X,
     1      'DIER = ', 1PE10.3)
      WRITE (6,*)'ENTER #OF OIT AFTER WHICH NEGATIVE FLUX IS ALWAYS US
     1ED'
      READ (4,150) INSB
  150 FORMAT (62X, I5)
      WRITE (8,160) INSB
  160 FORMAT (1X, 'INSB   = ', I5)
      WRITE (6,*)'ENTER ABSORPTION COEFICIENT 1E? (? = 6 TO 11)'
      READ (4,170) AC
  170 FORMAT (60X, 1PE8.2)
      WRITE (8,180) AC
  180 FORMAT (1X, 'AC = ', 1PE10.3)
      WRITE (6,*)'ENTER 0 IF NDIF IS TO BE USED FOR CONVERGENCE OR "1"'
      WRITE (6,*)'IF (NINT-PINT) IS TO BE USED AND "2" IF IT IS TO BE
     1NORMALED'
      READ (4,190) CONVC
  190 FORMAT (66X, I2)
      WRITE (8,200) CONVC
  200 FORMAT (1X, 'CONVC  = ', I2)
      READ (4,210) TRIN
  210 FORMAT (66X, I2)
      WRITE (8,220) TRIN
  220 FORMAT (1X, 'TRIN  = ', I2)
      WRITE (6,*)'ENTER "0" IF UNIFORM SPATIAL DIRIBUTION IS TO BE USED'

      READ (4,230) USC
  230 FORMAT (66X, I2)
```

```
      WRITE (8,240) USC
  240 FORMAT (1X, 'USC    = ', I2)
      READ (4,250) PRTIME
  250 FORMAT (60X, 1PE8.2)
      WRITE (8,260) PRTIME
  260 FORMAT (1X, 'PRTIME  = ', 1PE11.2)
      WRITE (6,*)'ENTER "0 TO 2" FOR DIFFUSION BC'
      READ (4,270) DIFBC
  270 FORMAT (66X, I2)
      WRITE (8,280) DIFBC
  280 FORMAT (1X, 'DIFBC  = ', I2)
      WRITE (6,*)'ENTER"0" IF NEW DIFFUSION INTENSITY IS NEGATIVE IT I
     1S'
      WRITE (6,*)'TAKEN TO BE ZERO"1" IF IT IS TAKEN TO BE THE PREVIOU
     1S'
      WRITE (6,*)'INTENSITY FOR THAT GROUP & CELL'
      READ (4,290) NDIT
  290 FORMAT (66X, I2)
      WRITE (8,300) NDIT
  300 FORMAT (1X, 'NDIT   = ', I2)
      WRITE (6,*)'ENTER MAXIMUM TIME FOR SIMULATION OF MODEL PROBLEM'
      READ (4,310) MAXT
  310 FORMAT (60X, 1PE8.2)
      WRITE (8,320) MAXT
  320 FORMAT (1X, 'MAXT   = ', 1PE11.2)
      WRITE (6,*)'ENTER NA < 24  NG < 64  NC < 151 '
      READ (4,330) NA, NG, NC
  330 FORMAT (57X, I3, 1X, I3, 1X, I3)
      WRITE (8,340) NA, NG, NC
  340 FORMAT (1X, 'NA = ', I3, 5X, 'NG = ', I3, 5X, 'NC = ', I3)
      WRITE (6,*)'ENTER  L   LFL   UFL    TELB   ITRB   ITSL'
      READ (4,350) L, LFL, UFL, TELB, ITRB, ITSL
  350 FORMAT (/4X, 6(1X,1PE8.2))
      WRITE (8,360)
  360 FORMAT (1X, 4X, 'L', 8X, 'LFL', 6X, 'UFL', 5X, 'TELB', 5X, 'ITRB',
     1       5X, 'ITSL')
      WRITE (8,370) L, LFL, UFL, TELB, ITRB, ITSL
  370 FORMAT (1X, 6(1X,1PE8.2))
      WRITE (6,*)'ENTER "0",IF PLANCKIAN IS TO BE USED AS A WEIGHT'
      WRITE (6,*)'FUNCTION OR "1" IF ROSSLAND FUNCTIONS IS TO BE USED'
      READ (4,380) WFN
  380 FORMAT (66X, I2)
      WRITE (8,390) WFN
  390 FORMAT (1X, 'WFN    = ', I2)
      WRITE (6,*)'ENTER SPECIFIC HEAT'
      READ (4,400) SPH
  400 FORMAT (60X, 1PE8.2)
      WRITE (8,410) SPH
  410 FORMAT (1X, 'SPECIFIC HEAT = ', 1PE10.3)
      WRITE (6,*)'ENTER DTIME'
      READ (4,420) DTIME
  420 FORMAT (60X, 1PE8.2)
      WRITE (8,430) DTIME
  430 FORMAT (1X, 'DTIME  = ', 1PE10.3)
      NF = 0
      WRITE (6,440) NF
  440 FORMAT (1X, 'NF     = ', I2)
      WRITE (6,*)'ENTER NUMER OF TEMPERATURE ITERATIONS ALLOWED'
      READ (4,450) TITA
  450 FORMAT (/64X, I4)
      WRITE (8,460) TITA
  460 FORMAT (1X, 'TITA   = ', I4)
      WRITE (6,*)'ENTER NUMER OF OUTER ITERATIONS ALLOWED'
      READ (4,470) OITA
  470 FORMAT (62X, I6)
      WRITE (8,480) OITA
  480 FORMAT (1X, 'OITA   = ', I6)
      READ (4,490) SETLIM
  490 FORMAT (66X, I2)
      WRITE (8,500) SETLIM
  500 FORMAT (1X, 'SETLIM = ', I2)
      SETBOU = 2
      WRITE (6,*)'ENTER 0 IF FLUX AT RIGHT BOUNDAR IS 0, 1 IF BOTH END S
     1'
      READ (4,510) SETBOU
  510 FORMAT (66X, I2)
      WRITE (8,520) SETBOU
  520 FORMAT (1X, 'SETBOU = ', I2)
      WRITE (6,*)'ENTER "1" FOR A & B TO BE 1 AT BOUNDARIES'
      READ (4,530) ABVL
  530 FORMAT (66X, I2)
      WRITE (8,540) ABVL
  540 FORMAT (1X, 'ABVL = ', I2)
      WRITE (6,*)'ENTER "1" IF CODE TO BE RUN WITH MEDUSA'
      READ (4,550) MEDUSA
  550 FORMAT (/66X, I2)
      WRITE (8,560) MEDUSA
  560 FORMAT (1X, 'MEDUSA = ', I2)
      WRITE (6,*)'ENTER 1 IF CRITERIAN IS AUTOMATICALLY DECIDED DEPENDIN
     1G ON THE SPR OF'
      WRITE (6,*)'TRANSPORT EQs, ALSO ENTER THE VALEU FOR SPR FOR AUTOMA
     1TIC ACCELERATION (<.5)'
      READ (4,570) AUTACL, SPRB
  570 FORMAT (56X, I2, 2X, 1PE8.2)
      WRITE (6,*)'ENTER "0" IF TRANSPORT EQUATION IS TO BE SOLVED BY SI'
      WRITE (6,*)'OR "1" IF BY DSA'
      READ (4,580) TRNACC
  580 FORMAT (66X, I2)
      WRITE (8,590) TRNACC
  590 FORMAT (1X, 'TRNACC = ', I2)
      WRITE (6,*)'ENTER"1"IF DIFFUSION EQUATION IS TO BE ACCELERATED'
      WRITE (6,*)'USING GREY METHOD'
      READ (4,600) DIFACC
  600 FORMAT (66X, I2)
      WRITE (8,610) DIFACC
  610 FORMAT (1X, 'DIFACC = ', I2)
  620 WRITE (6,*)'ENTER "0" FOR LINEAR GREY MODEL OR "1" FOR NONLINEAR G
     1REY MODEL'
      READ (4,630) LIN
  630 FORMAT (66X, I2)
      WRITE (8,640) LIN
  640 FORMAT (1X, 'LIN    = ', I2)
      IF (SETLIM .EQ. 0) THEN
        LOWLIM = 60.0
        HILIM = 70.0
      ELSE IF (SETLIM .EQ. 1) THEN
        LOWLIM = 90.0
        HILIM = 100.0
      ELSE IF (SETLIM .EQ. 2) THEN
        LOWLIM = 140.0
        HILIM = 150.0
      ELSE IF (SETLIM .EQ. 3) THEN
        LOWLIM = 190.0
        HILIM = 200.0
      ELSE IF (SETLIM .EQ. 4) THEN
        LOWLIM = 290.0
        HILIM = 300.0
      ELSE IF (SETLIM .EQ. 5) THEN
        LOWLIM = 250.0
        HILIM = 300.0
      ELSE IF (SETLIM .EQ. 6) THEN
        LOWLIM = 690.0
        HILIM = 700.0
      END IF
      WRITE (8,650) LOWLIM, HILIM
  650 FORMAT (1X, 'lowlim=', 1PE10.3, 1X, 'hilim=', 1PE10.3)
      RETURN
      END
C
C    *****************************************************************
C    * this subroutine defines the modal problem used to test the   *
C    * code. another user can modify the model by changing the      *
C    * variables in this subroutine, any change in other subroutine *
C    * will result in alteration of the method itself.              *
C    *****************************************************************
C
      SUBROUTINE MODEL

      PARAMETER (NANG=24,NFG=64,NCL=151)
      PARAMETER (RK=7.560667E-16,PK=6.6261E-34)
      PARAMETER (BK=1.3807E-23,SP=3.0E08,EVE=1.6022E-19)
      PARAMETER (TCF=EVE/BK,GK=8.3144,PI=3.141593,A0=PK/(BK*TCF))
      PARAMETER (A1=3.8171E+55)

      REAL*8 DIR, D, E
      REAL*8 FP(NFG + 1,NCL), ITLB
      REAL*8 LLFL, LFIN, LFR(NFG + 1), LITLB, LUFL
      REAL*8 RES2, RES3
      REAL*8 TRNFV

      INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      INTEGER DRT, DIFBC, DIFACC, IIT, TITA
      INTEGER INSB, IRUN, LIN, MEDUSA
      INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      INTEGER USC, WFN, CONV, DBXL, DBXU, NUM
      INTEGER NCP1, NCM1, NAD2
      INTEGER SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

      LOGICAL TRN, DIFF, LGR, NLGR, SIMRUN, SMTST
      COMMON /L1/ TRN, DIFF, LGR, NLGR, SIMRUN, SMTST

      COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
      COMMON /I3/ INSB, IRUN, LIN, MEDUSA
      COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM
      COMMON /I8/ NCP1, NCM1, NAD2
      COMMON /SMT/ SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

      REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFG,NCL), ATEM(NCL)
      REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
      REAL*8 BETA(NCL), BO(NFG,NCL)
      REAL*8 CV(NCL), CW, CHI(NFG,NCL)
      REAL*8 DL(NCL), DINT(NFG,NCL + 1), PRNTM
      REAL*8 DIST(NCL), DTIME, DSPR
      REAL*8 ETA(NCL)
      REAL*8 FR(NFG + 1), FQ(NFG,NCL)
      REAL*8 HILIM
      REAL*8 INLB(NFG), INRB(NFG), ITSL, IERED(NCL), IERR
      REAL*8 INTB(NANG,NFG,NCL + 1), IINT(NANG,NFG,NCL), IZIN(NFG,NCL)
      REAL*8 ITRB, ITER(NFG,NCL)
      REAL*8 L, LFL, LOWLIM
      REAL*8 MAXT
      REAL*8 NATB(NCL), NINT(NANG,NFG,NCL)
      REAL*8 NFFT(NANG,NFG,NCL), NDIN(NFG,NCL + 1), NZIN(NFG,NCL)
      REAL*8 PMAC(NCL), PINT(NANG,NFG,NCL), PRTIME
      REAL*8 PGIN(NCL + 1), PFGI(NCL + 1), PLANK(NFG,NCL)
      REAL*8 Q(NANG,NFG,NCL)
      REAL*8 SPH, SIRC, SPR
      REAL*8 SIGM(NFG,NCL), GRSI(NCL + 1)
      REAL*8 SOLU(NCL + 1), SPRB
      REAL*8 S(NFG,NCL), SUM1, SUM2, GRFQ(NCL), GRZQ(NCL)
      REAL*8 TTIME, TOU, TERR, TELB
      REAL*8 TRNFVR(NCL), TEMP(NCL)
      REAL*8 UFL
      REAL*8 WF(NANG)
      REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER
```

```
      COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
      COMMON /R02/ ABC, AB, AO
      COMMON /R03/ BETA, BO
      COMMON /R04/ CV, CW, CHI
      COMMON /R05/ DL, DINT, PRNTM
      COMMON /R06/ DIST, DTIME, DSPR
      COMMON /R07/ ETA
      COMMON /R08/ FR, FQ
      COMMON /R09/ HILIM
      COMMON /R10/ INLB, INRB, ITSL, IERED, IERR
      COMMON /R11/ INTB, IINT, IZIN
      COMMON /R12/ ITRB, ITER
      COMMON /R13/ L, LFL, LOWLIM
      COMMON /R14/ MAXT
      COMMON /R15/ NATE, NINT
      COMMON /R16/ NFFT, NDIN, NZIN
      COMMON /R17/ PMAC, PINT, PRTIME
      COMMON /R18/ PGIN, PFGR, PLANK
      COMMON /R19/ Q
      COMMON /R20/ SPH, SIRC, SPR
      COMMON /R21/ SIGM, GRSI
      COMMON /R22/ SOLU, SPRB
      COMMON /R23/ S, SUM1, SUM2, GRFQ, GRZQ
      COMMON /R24/ TTIME, TOU, TERR, TELB
      COMMON /R25/ TRNFVR, TEMP
      COMMON /R26/ UFL
      COMMON /R27/ WF
      COMMON /R28/ ZLP, ZINT, ZQ, DIER

      REAL*8 SPLNK(NCL), ABB(NCL), DIFRD(NCL)
      REAL*8 SUMCHI(NCL), CHI(NCL), RESSIG, RESPLN
      REAL*8 PLNINT(NFG,NCL), SIGINT(NFG,NCL), SGPLIN(NFG,NCL)
      EXTERNAL FPLNK, F2, F3, FSIG
      INTRINSIC ALOG10, EXP
C
CL       C2.8 NON-LTE DATA FOR AVRG-Z,RADIATIVE COOLING ETC...
C   VERSION JUNE 1986 Y.T. LEE
      COMMON /LEEDRY/ HNUFDG(64), DTEMP(19), DDENS(22), ENION(19,22),
     1   TBLZ(19,22), TBLZSQ(19,22), POWER(19,22,64),
     2   SIGMA(19,22,64), HNUMIN, HNUMAX, ZATOM, NT, NR, NGHIGH,
     3   NGLOW, NGRMAX, NTMAX
C
      IF (TOIT .EQ. 0) THEN
C**** define the frequency groups and the cell width ,initial temp-
C**** erature, specific heat for the model problem - only once
       DIR = 0.0
       IF (USC .EQ. 0) THEN
C**** take uniformly spaced spatial cells
        CW = L / NC
        DIST(1) = CW / 2
        DO 10 K = 2, NC
         DIST(K) = DIST(K - 1) + CW
   10   CONTINUE
        DO 20 K = 1, NC
         DL(K) = CW
   20   CONTINUE
       END IF
C**** frequency spectrum is logarithmically grouped
       LLFL = LOG(LFL)
       LUFL = LOG(UFL)
       LFIN = (LUFL - LLFL) / NG
       LFR(1) = LLFL
       DO 30 J = 2, NG + 1
        LFR(J) = LFR(J - 1) + LFIN
   30  CONTINUE
       DO 40 J = 1, NG + 1
        FR(J) = EXP(LFR(J))
   40  CONTINUE
       DO 50 K = 1, NC
C**** specific heat is taken to be uniform in the model
        CV(K) = SPH
   50  CONTINUE
C**** define initial temperature distribution
       IF (TEMEXP .EQ. 0) THEN
C**** let initial temperature to be exponential from left to right from
C**** 1 Kev to 1 ev
        LITLB = LOG(TELB)
        DO 60 K = 1, NC
           TEMP(K) = TELB * EXP((-LITLB*(DIST(K) - DIST(K)/2))/(L - (
     1    DIST(K)/2)))
   60   CONTINUE
       ELSE
C**** take initial temperature to be uniform
        DO 70 K = 1, NC
         TEMP(K) = ITSL
   70   CONTINUE
       END IF
      END IF
      IF (CONT .EQ. 0) THEN
       IF ( .NOT. SIMRUN) THEN
        TTIME = TTIME + DTIME
        IF (TTIME .GT. MAXT .AND. TOIT .NE. 0) THEN
         WRITE (*,*)'time simulate', TTIME - DTIME, SIMRUN
         WRITE (6,*)'program terminated, maxt reached'
         STOP
        END IF
       END IF
       TOU = 1 / (SP*DTIME)
       IF (NFFC .EQ. 0) THEN
C**** take intensity at cell boundary to be zero when it is negative
C   TRNFVR determines the type of modification to be used for
C**** the negative flux fix up
        DO 80 K = 1, NC
         TRNFVR(K) = 0.5
   80   CONTINUE
        TRNFV = (SP*DTIME) / DL(1)
       ELSE IF (NFFC .EQ. 1) THEN
C**** take intensity at cell boundary to be cell average intensity
C**** when it is negative
        DO 90 K = 1, NC
         TRNFVR(K) = 2.0
   90   CONTINUE
        TRNFV = (SP*DTIME) / DL(1)
       ELSE IF (NFFC .EQ. 2) THEN
C**** implement negative flux fixup as shown in the thesis
        DO 100 K = 1, NC
         TRNFVR(K) = (SP*DTIME) / DL(K)
  100   CONTINUE
       END IF
      END IF
       IF (CONT .EQ. 0) THEN
       IF (ITT .GE. 1) THEN
        DO 110 K = 1, NC
         TEMP(K) = NATE(K)
  110   CONTINUE
       END IF
       DO 130 K = 1, NC
        DO 120 J = 1, NG + 1
         FF(J,K) = FR(J) / TEMP(K)
  120   CONTINUE
  130  CONTINUE
       IF (TRIN .EQ. 0 .OR. TOIT .EQ. 0) THEN
        IF (WFN .EQ. 0) THEN
C**** calculate initial intensity assuming planckian distribution
         DO 160 K = 1, NC
          DO 150 J = 1, NG
           D = FF(J,K)
           E = FF(J + 1,K)
           IF (D .GT. LOWLIM) THEN
            D = LOWLIM
           END IF
           IF (E .GT. HILIM) THEN
            E = HILIM
           END IF
           CALL ROMB(FPLNK, D, E, RESPLN)
C****      WRITE (6,*)'res1 ', RESPLN
           PLANK(J,K) = (2/(SP**2)) * (TEMP(K)**4) * (BK*TCF) *
     1      RESPLN / (A0**3)
           DO 140 I = 1, NA
            IINT(I,J,K) = (2/(SP**2)) * (TEMP(K)**4) * (BK*TCF) *
     1       RESPLN / (A0**3)
            PINT(I,J,K) = IINT(I,J,K)
  140      CONTINUE
  150     CONTINUE
  160    CONTINUE
        ELSE
C**** calculate initial intensity assuming Rosseland distribution
         DO 190 K = 1, NC
          DO 180 J = 1, NG
           D = FF(J,K)
           E = FF(J + 1,K)
           IF (D .GT. LOWLIM) THEN
            D = LOWLIM
           END IF
           IF (E .GT. HILIM) THEN
            E = HILIM
           END IF
C>>>        CALL ROMB(F2, D, E, RES2)
           CALL ROMB(F3, D, E, RES3)
C****       CALL ROMB(FPLNK, D, E, RESPLN)
           PLANK(J,K) = 1.60652E-20 * AC * RES3 * (TEMP(K)**6)
C****       WRITE (6,*)'res2 and res3 ', RES2, RES3 , PLANK(J,K)
           DO 170 I = 1, NA
            IINT(I,J,K) = PLANK(J,K)
            PINT(I,J,K) = IINT(I,J,K)
  170      CONTINUE
  180     CONTINUE
  190    CONTINUE
        END IF
       ELSE
C**** take initial intensity to be the intensity from transfer equation
        DO 220 K = 1, NC
         DO 210 J = 1, NG
          DO 200 I = 1, NA
           IINT(I,J,K) = NINT(I,J,K)
           PINT(I,J,K) = IINT(I,J,K)
  200     CONTINUE
  210    CONTINUE
  220   CONTINUE
       END IF
       DO 250 J = 1, NG
        DO 240 K = 1, NC
         ZINT(J,K) = 0.0
         DO 230 I = 1, NA
          ZINT(J,K) = ZINT(J,K) + IINT(I,J,K) * WF(I)
  230    CONTINUE
         IZIN(J,K) = ZINT(J,K)
  240   CONTINUE
  250  CONTINUE
       DO 270 K = 1, NC
        SPLNK(K) = 0.0
        DO 260 J = 1, NG
         D = FF(J,K)
         E = FF(J + 1,K)
         IF (D .GT. LOWLIM) THEN
          D = LOWLIM
         END IF
         IF (E .GT. HILIM) THEN
          E = HILIM
         END IF
         CALL ROMB(FPLNK, D, E, RESPLN)
         SPLNK(K) = SPLNK(K) + RESPLN
```

```
 260   CONTINUE
 270  CONTINUE
       DO 280 K = 1, NC
        IERED(K) = 4 * PI * 5.03343E+07 * SPLNK(K) / SP
 280  CONTINUE
       IF (PRES .GE. 5 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
       WRITE (8,290) (K,DL(K),CV(K),TEMP(K)/11604,IERED(K),K=1,NC)
 290    FORMAT (2('k',I2,1X,'dl=',1PE10.3,1X,'cv=',1PE10.3,1X,'temp=',
     1      1PE10.3,1X,'Ue=',1PE10.3,1X))
       WRITE (8,*)'freq group limits model'
       WRITE (8,300) (J,FR(J),J=1,NG + 1)
 300    FORMAT (8('g',I2,1X,F11.4,1X))
       WRITE (8,*)'i j k iint m model'
       WRITE (8,310) (((I,J,K,IINT(I,J,K),K=1,NC),J=1,NG),I=1,NA)
 310    FORMAT (6('i',I2,1X,'j',I2,1X,'k',I2,1X,1PE10.3,1X))
       WRITE (8,*)'j k zint model'
       WRITE (8,320) ((J,K,ZINT(J,K),K=1,NC),J=1,NG)
 320    FORMAT (8(I2,1X,I2,1X,1PE10.3,1X))
       WRITE (8,*)'j k integration limits model'
       WRITE (8,330) ((J,K,FF(J,K),K=1,NC),J=1,NG + 1)
 330    FORMAT (8(I2,1X,I2,1X,1PE10.3))
        END IF
       END IF
       IF (CONT .EQ. 1) THEN
C**** solve opacity at new temp.
        GO TO 400
       END IF
       IF (TTIME .GE. 1.0005E-10 .AND. TOIT .NE. 0) THEN
C**** no need to calculate intensity at the boundary, since temperature
C**** remains constant after 0.1 ns for time dependent case.
        GO TO 400
       END IF
       IF (TEMPTD .EQ. 0) THEN
C**** express temperature on left end as linear function of time
C**** untill it reaches its maximum value (1000 ev)
        IF (TTIME .LT. 1.00005E-10) THEN
         ITLB = ((TELB - 1)*TTIME/1.0E-10 + 1)
        ELSE
         ITLB = TELB
        END IF
       ELSE
        ITLB = TELB
       END IF
       WRITE (6,*)'itlb itrb', ITLB, ITRB
       DO 340 J = 1, NG
        D = FR(J) / ITLB
        E = FR(J + 1) / ITLB
        IF (D .GT. LOWLIM) THEN
         D = LOWLIM
        END IF
        IF (E .GT. HILIM) THEN
         E = HILIM
        END IF
        CALL ROMB(FPLNK, D, E, RESPLN)
C       WRITE (6,280) J, RESPLN
C 280  FORMAT (1X, 'J = ', I2, 2X, 'INTEGRAL INLB = ', 1PE10.3)
        INLB(J) = (2/(SP**2)) * (ITLB**4) * (BK*TCF) * RESPLN / (A0**3)
 340 CONTINUE
       DO 350 J = 1, NG
        D = FR(J) / ITRB
        E = FR(J + 1) / ITRB
        IF (D .GT. LOWLIM) THEN
         D = LOWLIM
        END IF
        IF (E .GT. HILIM) THEN
         E = HILIM
        END IF
        CALL ROMB(FPLNK, D, E, RESPLN)
C       WRITE (6,300) J, RESPLN
C 300  FORMAT (1X, 'J = ', I2, 2X, 'INTEGRAL INRB = ', 1PE10.3)
        INRB(J) = (2/(SP**2)) * (ITRB**4) * (BK*TCF) * RESPLN / (A0**3)
 350 CONTINUE
       IF (SETBOU .EQ. 0) THEN
C****   no source on the right boundary
        DO 360 J = 1, NG
         INRB(J) = 0.0
 360    CONTINUE
       ELSE IF (SETBOU .EQ. 1) THEN
C****   no source on the left or right boundary
        DO 370 J = 1, NG
         INLB(J) = 0.0
         INRB(J) = 0.0
 370    CONTINUE
       END IF
       IF (PRES .GE. 18. AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
       WRITE (8,380) ITLB, ITRB
 380   FORMAT (1X, 'ITLB = ', 1PE10.3, 1X, 'ITRB = ', 1PE10.3)
       WRITE (8,390) (J,INLB(J),INRB(J),J=1,NG)
 390   FORMAT (4('J',I2,1X,2(1PE10.3,1X)))
       END IF
 400 IF (ITT .LT. 1) THEN
       DO 410 K = 1, NC
        ATEM(K) = TEMP(K)
 410   CONTINUE
      END IF
      DO 430 K = 1, NC
       DO 420 J = 1, NG + 1
        FF(J,K) = FR(J) / ATEM(K)
 420   CONTINUE
 430 CONTINUE
      IF (WFN .EQ. 0) THEN
       DO 450 K = 1, NC
        SPLNK(K) = 0.0
        ABB(K) = 0.0
        DO 440 J = 1, NG
         D = FF(J,K)
         E = FF(J + 1,K)
         IF (D .GT. LOWLIM) THEN
          D = LOWLIM
         END IF
         IF (E .GT. HILIM) THEN
          E = HILIM
         END IF
         CALL ROMB(FPLNK, D, E, RESPLN)
         CALL ROMB(FSIG, D, E, RESSIG)
         PLNINT(J,K) = RESPLN
         SIGINT(J,K) = RESSIG
         SPLNK(K) = SPLNK(K) + PLNINT(J,K)
         ABB(K) = ABB(K) + PLNINT(J,K) * SIGINT(J,K)
         SGPLIN(J,K) = (EXP(-D) - EXP(-E))
 440    CONTINUE
 450  CONTINUE
       DO 470 K = 1, NC
        DO 460 J = 1, NG
         CHI(J,K) = SIGINT(J,K) * PLNINT(J,K) / ABB(K)
         ABC(J,K) = 27.0 * AC * SGPLIN(J,K) / ((ATEM(K)**3)*PLNINT(J,
     1   K))
 460    CONTINUE
 470  CONTINUE
      ELSE
       DO 490 K = 1, NC
        SPLNK(K) = 0.0
        ABB(K) = 0.0
        DO 480 J = 1, NG
         D = FF(J,K)
         E = FF(J + 1,K)
         IF (D .GT. LOWLIM) THEN
          D = LOWLIM
         END IF
         IF (E .GT. HILIM) THEN
          E = HILIM
         END IF
         CHI(J,K) = (EXP(-D) - EXP(-E))
         CALL ROMB(F2, D, E, RES2)
         CALL ROMB(F3, D, E, RES3)
         CALL ROMB(FPLNK, D, E, RESPLN)
         CALL ROMB(FSIG, D, E, RESSIG)
         PLNINT(J,K) = RESPLN
         SIGINT(J,K) = RESSIG
         SPLNK(K) = SPLNK(K) + PLNINT(J,K)
         ABB(K) = ABB(K) + PLNINT(J,K) * SIGINT(J,K)
         SGPLIN(J,K) = (EXP(-D) - EXP(-E))
C        WRITE (6,*)'res2 and res3 ', RES2, RES3
 480    CONTINUE
 490  CONTINUE
       DO 510 K = 1, NC
        DO 500 J = 1, NG
         CHI(J,K) = SIGINT(J,K) * PLNINT(J,K) / ABB(K)
         ABC(J,K) = 27.0 * (AC) * RES2 / ((ATEM(K)**3)*RES3)
 500    CONTINUE
 510  CONTINUE
      END IF
      DO 520 K = 1, NC
       PMAC(K) = 27.0 * AC * ABB(K) / ((ATEM(K)**3)*SPLNK(K))
       BETA(K) = 4 * RK * ((ATEM(K)*TCF)**3) / CV(K)
       ALFA(K) = 1 / (PMAC(K)*BETA(K))
 520 CONTINUE
      DO 530 K = 1, NC
       ETA(K) = 1 / (1 + ALFA(K)*TOU)
       DEFRD(K) = SPLNK(K) * 4 * PI / SP
 530 CONTINUE
      SIRC = 0.0
      DO 540 K = 1, NC
       SIRC = SIRC + ETA(K)
 540 CONTINUE
      SIRC = SIRC / NC
      IF (PRES .GE. 2 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1     THEN
       WRITE (8,*)'beta pmac alfa in model'
       WRITE (8,550) (K,BETA(K),PMAC(K),ALFA(K),K=1,NC)
 550  FORMAT (3(I2,1X,3(1PE10.3,1X)))
      END IF
      DO 580 J = 1, NG
       DO 570 K = 1, NC
        FQ(J,K) = 0.0
        DO 560 I = 1, NA
         Q(I,J,K) = CHI(J,K) * IERED(K) * ETA(K) / (2*DTIME*BETA(K))
     1    + TOU * IINT(I,J,K)
         FQ(J,K) = FQ(J,K) + IINT(I,J,K) * WF(I) * ZLP(I)
 560    CONTINUE
        SIGM(J,K) = (ABC(J,K) + TOU) * DL(K)
        ZQ(J,K) = CHI(J,K) * IERED(K) * ETA(K) / (DTIME*BETA(K)) +
     1  TOU * IZIN(J,K)
 570  CONTINUE
 580 CONTINUE
      DO 600 K = 1, NC
       GRZQ(K) = 0.0
       GRFQ(K) = 0.0
       SUMCHI(K) = 0.0
       DO 590 J = 1, NG
        GRZQ(K) = GRZQ(K) + ZQ(J,K)
        GRFQ(K) = GRFQ(K) + FQ(J,K) / SIGM(J,K)
        SUMCHI(K) = SUMCHI(K) + CHI(J,K)
 590  CONTINUE
 600 CONTINUE
      IF (PRES .GE. 2 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1     THEN
      WRITE (8,*)'sum of chis and abbchi model'
      WRITE (8,610) (K,SUMCHI(K),ABB(K),K=1,NC)
 610  FORMAT (5(I2,1X,2(1PE11.4,1X)))
```

```fortran
      WRITE (8,*)'j k chi abc plank at model'
      WRITE (8,620) ((J,K,CHI(J,K),ABC(J,K),PLANK(J,K),J=1,NG),K=1,NC)
620   FORMAT (3(I2,1X,I2,1X,3(1PE10.3,1X)))
      WRITE (8,*)' i j k q model'
      WRITE (8,630) (((I,J,K,Q(I,J,K),K=1,NC),J=1,NG),I=1,NA)
630   FORMAT (6(I2,1X,I2,1X,I2,1X,1PE10.3,1X))
      PAUSE
     END IF
     IF (TIT .GE. 1) THEN
      DO 660 J = 1, NG
       DO 650 K = 1, NC
        ZINT(J,K) = 0.0
        DO 640 I = 1, NA
         ZINT(J,K) = ZINT(J,K) + NINT(I,J,K) * WF(I)
640     CONTINUE
650    CONTINUE
660   CONTINUE
     END IF
     DO 680 K = 1, NC
      AB(K) = 0.0
      DO 670 J = 1, NG
       AB(K) = AB(K) + ABC(J,K) * ZINT(J,K)
670   CONTINUE
680  CONTINUE
     IF (PRES .GE. 2 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
    1     THEN
      WRITE (8,*)'j k zint zq model'
      WRITE (8,690) ((J,K,ZINT(J,K),ZQ(J,K),J=1,NG),K=1,NC)
690   FORMAT (5(I2,1X,I2,1X,1PE10.3,1X,1PE10.3))
     END IF
     RETURN
     END
C
C    ******************************************************************
C    *  this subroutine solves the RTE equation and evaluates the    *
C    * the varibale needed for solving the diffusion equation        *
C    ******************************************************************
C
     SUBROUTINE TRNPRT

     PARAMETER (NANG=24,NFG=64,NCL=151)
     PARAMETER (SP=3.0E08)

     INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
     INTEGER DIRT, DIFBC, DIFACC, IIT, TITA
     INTEGER INSB, IRUN, LIN, MEDUSA
     INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
     INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
     INTEGER TB, TEMEXP, TBT, TOIT, TEMPTD, TIT, TRIN
     INTEGER USC, WFN, CONV, DBXL, DBXU, NUM
     INTEGER NCP1, NCM1, NAD2

     COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
     COMMON /I2/ DIRT, DIFBC, DIFACC, IIT, TITA
     COMMON /I3/ INSB, IRUN, LIN, MEDUSA
     COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
     COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
     COMMON /I6/ TB, TEMEXP, TBT, TOIT, TEMPTD, TIT, TRIN
     COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM
     COMMON /I8/ NCP1, NCM1, NAD2

     REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFG,NCL), ATEM(NCL)
     REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
     REAL*8 BETA(NCL), BO(NFG,NCL)
     REAL*8 CV(NCL), CW, CHI(NFG,NCL)
     REAL*8 DL(NCL), DINT(NFG,NCL + 1), PRNTM
     REAL*8 DIST(NCL), DTIME, DSPR
     REAL*8 ETA(NCL)
     REAL*8 FINT(NFG,NCL + 1), FINL(NFG), FINR(NFG)
     REAL*8 FR(NFG + 1), FQ(NFG,NCL)
     REAL*8 INLB(NFG), INRB(NFG), ITEL, IERED(NCL), IERR
     REAL*8 INTB(NANG,NFG,NCL + 1), IINT(NANG,NFG,NCL), IZIN(NFG,NCL)
     REAL*8 ITRB, ITER(NFG,NCL)
     REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
     REAL*8 NFFT(NANG,NFG,NCL), NDIN(NFG,NCL + 1), NZIN(NFG,NCL)
     REAL*8 PMAC(NCL), PINT(NANG,NFG,NCL), PRTIME
     REAL*8 PGIN(NCL + 1), PFGI(NCL + 1), PLANK(NFG,NCL)
     REAL*8 PDIN(NFG,NCL + 1), PITE(NFG,NCL), PZIN(NFG,NCL)
     REAL*8 PAB(NCL)
     REAL*8 Q(NANG,NFG,NCL)
     REAL*8 SIGM(NFG,NCL), GRSI(NCL + 1)
     REAL*8 S(NFG,NCL), SUM1, SUM2, GRFQ(NCL), GRZQ(NCL)
     REAL*8 SINT(NFG,NCL + 1)
     REAL*8 TTIME, TOU, TERR, TELB
     REAL*8 TRNFVR(NCL), TEMP(NCL)
     REAL*8 WF(NANG)
     REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER

     COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
     COMMON /R02/ ABC, AB, AO
     COMMON /R03/ BETA, BO
     COMMON /R04/ CV, CW, CHI
     COMMON /R05/ DL, DINT, PRNTM
     COMMON /R06/ DIST, DTIME, DSPR
     COMMON /R07/ ETA
     COMMON /R071/ FINT, FINL, FINR
     COMMON /R08/ FR, FQ
     COMMON /R10/ INLB, INRB, ITEL, IERED, IERR
     COMMON /R11/ INTB, IINT, IZIN
     COMMON /R12/ ITRB, ITER
     COMMON /R15/ NATE, NINT
     COMMON /R16/ NFFT, NDIN, NZIN
     COMMON /R17/ PMAC, PINT, PRTIME
     COMMON /R18/ PGIN, PFGI, PLANK
     COMMON /R181/ PDIN, PITE, PZIN
     COMMON /R182/ PAB
```



```fortran
     COMMON /R19/ Q
     COMMON /R21/ SIGM, GRSI
     COMMON /R23/ S, SUM1, SUM2, GRFQ, GRZQ
     COMMON /R231/ SINT
     COMMON /R24/ TTIME, TOU, TERR, TELB
     COMMON /R25/ TRNFVR, TEMP
     COMMON /R27/ WF
     COMMON /R28/ ZLP, ZINT, ZQ, DIER

     COMMON /IMT/ J, N

     DO 60 I = 1, NAD2
      DO 50 J = 1, NG
       DO 40 K = 1, NC
        NFFT(I,J,K) = 0.0
        INTB(I,J,1) = INLB(J)
        NINT(I,J,K) = ((CHI(J,K)*ETA(K)*AB(K)/2 + Q(I,J,K))*DL(K) +
    1    2*ZLP(I)*INTB(I,J,K)) / (SIGM(J,K) + 2*ZLP(I))
        INTB(I,J,K + 1) = 2 * NINT(I,J,K) - INTB(I,J,K)
        IF (PRES .GE. 15 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
    1       THEN
         WRITE (8,10) I, J, K, PINT(I,J,K), NINT(I,J,K),
    1     NINT(I,J,K) / PINT(I,J,K), INTB(I,J,K + 1)
10        FORMAT (1X,'I J K PI NI NI/P INTB=', 3(I2,1X),
    1        4(1PE10.3,1X))
        END IF
C**** negative flux fix up criterion is selected (63), according
C**** to TRNFVR value
        IF (NF .EQ. 0) THEN
         IF (INTB(I,J,K + 1) .LT. 0) THEN
          NFFT(I,J,K) = 1.0
          COUNT = COUNT + 1
          IF (TRNFVR(K) .LT. 1) THEN
           NINT(I,J,K) = ((CHI(J,K)*ETA(K)*AB(K)/2 + Q(I,J,K))*
    1       DL(K) + ZLP(I)*INTB(I,J,K)) / SIGM(J,K)
           INTB(I,J,K + 1) = 0.0
           IF (PRES .GE. 15 .AND. TOIT .GE. DBXL .AND. TOIT .LE.
    1          DBXU) THEN
            WRITE (8,20) I, J, K, PINT(I,J,K), NINT(I,J,K),
    1        NINT(I,J,K) / PINT(I,J,K), INTB(I,J,K + 1)
20           FORMAT (1X,'I J K PI NI NI/P INTB=', 3(I2,1X),
    1           4(1PE10.3,1X))
           END IF
          ELSE
           NINT(I,J,K) = ((CHI(J,K)*ETA(K)*AB(K)/2 + Q(I,J,K))*
    1       DL(K) + ZLP(I)*INTB(I,J,K)) / (SIGM(J,K) + ZLP(I))
           INTB(I,J,K + 1) = NINT(I,J,K)
           IF (PRES .GE. 15 .AND. TOIT .GE. DBXL .AND. TOIT .LE.
    1          DBXU) THEN
            WRITE (8,30) I, J, K, PINT(I,J,K), NINT(I,J,K),
    1        NINT(I,J,K) / PINT(I,J,K), INTB(I,J,K + 1)
30           FORMAT (1X,'I J K PI NI NI/P INTB=', 3(I2,1X),
    1           4(1PE10.3,1X))
           END IF
          END IF
         END IF
        END IF
40     CONTINUE
50    CONTINUE
60   CONTINUE
      DO 120 I = NAD2 + 1, NA
       DO 110 J = 1, NG
        DO 100 K = NC, 1, -1
         INTB(I,J,NCP1) = INRB(J)
         NFFT(I,J,K) = 0.0
         NINT(I,J,K) = ((CHI(J,K)*ETA(K)*AB(K)/2 + Q(I,J,K))*DL(K) -
    1     2*ZLP(I)*INTB(I,J,K + 1)) / (SIGM(J,K) - 2*ZLP(I))
         INTB(I,J,K) = 2 * NINT(I,J,K) - INTB(I,J,K + 1)
         IF (PRES .GE. 15 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
    1       THEN
          WRITE (8,70) I, J, K, PINT(I,J,K), NINT(I,J,K),
    1      NINT(I,J,K) / PINT(I,J,K), INTB(I,J,K)
70         FORMAT (1X,'I J K PI NI NI/P INTB=', 3(I2,1X),
    1         4(1PE10.3,1X))
         END IF
C**** negative flux fix up criterion is selected, according to
C**** TRNFVR value.
         IF (NF .EQ. 0) THEN
          IF (INTB(I,J,K) .LT. 0) THEN
           NFFT(I,J,K) = 1.0
           COUNT = COUNT + 1
           IF (TRNFVR(K) .LT. 1) THEN
            NINT(I,J,K) = ((CHI(J,K)*ETA(K)*AB(K)/2 + Q(I,J,K))*
    1        DL(K) - ZLP(I)*INTB(I,J,K + 1)) / SIGM(J,K)
            INTB(I,J,K) = 0.0
            IF (PRES .GE. 15 .AND. TOIT .GE. DBXL .AND. TOIT .LE.
    1           DBXU) THEN
             WRITE (8,80) I, J, K, PINT(I,J,K), NINT(I,J,K),
    1         NINT(I,J,K) / PINT(I,J,K), INTB(I,J,K)
80            FORMAT (1X,'I J K PI NI NI/P INTB=', 3(I2,1X),
    1            4(1PE10.3,1X))
            END IF
           ELSE
            NINT(I,J,K) = ((CHI(J,K)*ETA(K)*AB(K)/2 + Q(I,J,K))*
    1        DL(K) - ZLP(I)*INTB(I,J,K + 1)) / (SIGM(J,K) - ZLP(I))
            INTB(I,J,K) = NINT(I,J,K)
            IF (PRES .GE. 15 .AND. TOIT .GE. DBXL .AND. TOIT .LE.
    1           DBXU) THEN
             WRITE (8,90) I, J, K, PINT(I,J,K), NINT(I,J,K),
    1         NINT(I,J,K) / PINT(I,J,K), INTB(I,J,K)
90            FORMAT (1X,'I J K PI NI NI/P INTB=', 3(I2,1X),
    1            4(1PE10.3,1X))
            END IF
           END IF
          END IF
         END IF
```

```
      100   CONTINUE
      110  CONTINUE
      120 CONTINUE
          DO 150 J = 1, NG
          DO 140 K = 1, NCP1
            DINT(J,K) = 0.0
            DO 130 I = 1, NA
              DINT(J,K) = DINT(J,K) + INTB(I,J,K) * WF(I)
      130   CONTINUE
      140  CONTINUE
      150 CONTINUE
         IF (PRES .GE. 2) THEN
          WRITE (8,160) (((I,J,K,PINT(I,J,K),NINT(I,J,K),I=1,NA),J=1,NG),
        1  K=1,NC)
      160  FORMAT (3(3(I2,1X),2(1PE10.3,1X)))
         END IF
         IF (TRNACC .EQ. 1) THEN
          DO 190 J = 1, NG
           DO 180 K = 1, NCP1
             DINT(J,K) = 0.0
             FINT(J,K) = 0.0
             SINT(J,K) = 0.0
             DO 170 I = 1, NA
              DINT(J,K) = DINT(J,K) + INTB(I,J,K) * WF(I)
              FINT(J,K) = FINT(J,K) + INTB(I,J,K) * ZLP(I) * WF(I)
              SINT(J,K) = SINT(J,K) + INTB(I,J,K) * WF(I) * (3*ZLP(I)**
        1       2 - 1) / 2
      170     CONTINUE
             PDIN(J,K) = DINT(J,K)
      180   CONTINUE
      190  CONTINUE
          DO 220 K = 1, NC
           DO 210 J = 1, NG
             ZINT(J,K) = 0.0
             PZIN(J,K) = 0.0
             DO 200 I = 1, NA
              ZINT(J,K) = ZINT(J,K) + NINT(I,J,K) * WF(I)
              PZIN(J,K) = PZIN(J,K) + PINT(I,J,K) * WF(I)
      200     CONTINUE
      210   CONTINUE
      220  CONTINUE
          DO 240 K = 1, NC
           PAB(K) = 0.0
           DO 230 J = 1, NG
            PAB(K) = PAB(K) + ABC(J,K) * PZIN(J,K)
      230   CONTINUE
      240  CONTINUE
          DO 260 K = 1, NCP1
           PFGR(K) = 0.0
           PGIN(K) = 0.0
           DO 250 J = 1, NG
            PFGR(K) = PFGR(K) + FINT(J,K)
            PGIN(K) = PGIN(K) + DINT(J,K)
      250   CONTINUE
      260  CONTINUE
          DO 280 K = 1, NC
           GRSI(K) = 0.0
           DO 270 J = 1, NG
            GRSI(K) = GRSI(K) + (SINT(J,K + 1) - SINT(J,K)) / SIGM(J,K)
          270   CONTINUE
      280  CONTINUE
         IF (PRES .GE. 5 .AND. TOIT .GT. DBXL .AND. TOIT .LE. DBXU)
        1     THEN
          WRITE (8,*)'k pfgi pgin'
          WRITE (8,290) (K,PFGR(K),PGIN(K),K=1,NCP1)
      290   FORMAT (1X, 3('k',I2,1X,2(1PE10.3),1X))
          WRITE (8,*)'k grsi'
          WRITE (8,300) (K,GRSI(K),K=1,NC)
      300   FORMAT (1X, 5('k',I2,1X,1PE10.3,1X))
         END IF
C**** negative flux fix up cristerion is selected
         IF (NF .EQ. 0) THEN
          DO 330 J = 1, NG
           DO 320 K = 1, NC
            DO 310 I = 1, NA
             IF (NFFT(I,J,K) .GT. 0.5) THEN
              IF (DINT(J,K + 1) .EQ. 0.0) THEN
               AO(J,K) = 1.0
              ELSE
               AO(J,K) = ZINT(J,K) / DINT(J,K + 1)
              END IF
              IF (DINT(J,K) .EQ. 0.0) THEN
               BO(J,K) = 1.0
              ELSE
               BO(J,K) = ZINT(J,K) / DINT(J,K)
              END IF
              GO TO 320
             ELSE
              AO(J,K) = 1.0
              BO(J,K) = 1.0
             END IF
      310     CONTINUE
      320    CONTINUE
      330   CONTINUE
         END IF
         DO 350 K = 1, NC
          AB(K) = 0.0
          DO 340 J = 1, NG
           AB(K) = AB(K) + ABC(J,K) * (AO(J,K)*DINT(J,K + 1) + BO(J,K)*
        1   DINT(J,K))
      340   CONTINUE
          PAB(K) = 2 * PAB(K)
      350  CONTINUE
         IF (PRES .GE. 5 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
        1     THEN
          WRITE (8,*)'k ab for diff'
          WRITE (8,360) (K,AB(K),K=1,NC)
      360   FORMAT (4(1X,'k',I2,1X,1PE10.3))
         END IF
         DO 380 J = 1, NG
          DO 370 K = 1, NCM1
           S(J,K) = (DL(K)*ZQ(J,K) + DL(K + 1)*ZQ(J,K + 1)) / 2 - (
        1   FINT(J,K + 2) - FINT(J,K)) / 2 - (DINT(J,K + 2) - DINT(J,K +
        2   1)) / (3*SIGM(J,K + 1)) + (DINT(J,K + 1) - DINT(J,K)) / (3*
        3   SIGM(J,K))
      370   CONTINUE
      380  CONTINUE
         DO 400 K = 1, NC
          DO 390 J = 1, NG
           ITER(J,K) = DL(K) * CHI(J,K) * ETA(K) * AB(K) / 4
           PITE(J,K) = DL(K) * CHI(J,K) * ETA(K) * PAB(K) / 4
      390   CONTINUE
      400  CONTINUE
         IF (PRES .GE. 7 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
        1     THEN
          WRITE (8,*)'toit dxxt', TOIT
          WRITE (8,*)'j k ao bo fq'
          WRITE (8,410) ((J,K,AO(J,K),BO(J,K),FQ(J,K),K=1,NC),J=1,NG)
      410   FORMAT (1X, 2(I2,1X,I2,1X,3(1PE10.3,1X)))
          WRITE (8,*)'j k s(j,k)'
          WRITE (8,420) ((J,K,S(J,K),K=1,NCM1),J=1,NG)
      420   FORMAT (1X, 4(I2,1X,I2,1X,1PE10.3,1X))
          WRITE (8,*)'k j zq sigm'
          WRITE (8,430) ((J,K,ZQ(J,K),SIGM(J,K),K=1,NC),J=1,NG)
      430   FORMAT (2(I2,1X,I2,2(1PE10.3,1X)))
          WRITE (8,*)'k j fint pdin sint'
          WRITE (8,440) ((J,K,FINT(J,K),PDIN(J,K),SINT(J,K),K=1,NCP1),J=
        1  1,NG)
      440   FORMAT (1X, I2, 1X, I2, 1X, 1PE10.3, 1X, 1PE10.3, 1X, 1PE10.3)
          END IF
         END IF
         RETURN
         END
C
C **********************************************************************
C * this subroutine computes the spectral radius and determines if *
C * the intensity is converged.                                    *
C **********************************************************************
C
      SUBROUTINE TRCONV

      PARAMETER (NANG=24,NFG=64,NCL=151)

      REAL*8 DIFFF
      REAL*8 NDIF
      REAL*8 PDIF
      REAL*8 SNIN(NCL), SPIN(NCL)
      REAL*8 TINT

      INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      INTEGER DRT, DIFBC, DIFACC, IIT, TITA
      INTEGER INSB, IRUN, LIN, MEDUSA
      INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      INTEGER UBC, WFN, CONV, DBXL, DBXU, NUM
      INTEGER NCP1, NCM1, NAD2
      INTEGER SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

      LOGICAL TRN, DIFF, LGR, NLGR, SIMRUN, SMTST
      COMMON /L1/ TRN, DIFF, LGR, NLGR, SIMRUN, SMTST

      COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
      COMMON /I3/ INSB, IRUN, LIN, MEDUSA
      COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      COMMON /I7/ UBC, WFN, CONV, DBXL, DBXU, NUM
      COMMON /I8/ NCP1, NCM1, NAD2
      COMMON /SMT/ SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

      REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
      REAL*8 DL(NCL), DINT(NFG,NCL + 1), PRNTM
      REAL*8 DIST(NCL), DTIME, DEPR
      REAL*8 FR(NFG + 1), FQ(NFG,NCL)
      REAL*8 INLB(NFG), INRB(NFG), ITSL, IERED(NCL), IERR
      REAL*8 HILIM
      REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
      REAL*8 PMAC(NCL), PINT(NANG,NFG,NCL), PRTIME
      REAL*8 SOLU(NCL + 1), SPRB
      REAL*8 SPH, SIRC, SPR
      REAL*8 WF(NANG)

      COMMON /R02/ ABC, AB, AO
      COMMON /R05/ DL, DINT, PRNTM
      COMMON /R06/ DIST, DTIME, DEPR
      COMMON /R08/ FR, FQ
      COMMON /R09/ HILIM
      COMMON /R10/ INLB, INRB, ITSL, IERED, IERR
      COMMON /R15/ NATE, NINT
      COMMON /R17/ PMAC, PINT, PRTIME
      COMMON /R22/ SOLU, SPRB
      COMMON /R20/ SPH, SIRC, SPR
      COMMON /R27/ WF

      IF (RT .LT. 1) THEN
       DO 30 K = 1, NC
        SNIN(K) = 0.0
        SPIN(K) = 0.0
        DO 20 J = 1, NG
         DO 10 I = 1, NA
```

```
           SPIN(K) = SPIN(K) + (ABC(J,K)*PINT(I,J,K)*WF(I)*(FR(J + 1)
     1      -FR(J)))
           SNIN(K) = SNIN(K) + (ABC(J,K)*NINT(I,J,K)*WF(I)*(FR(J + 1)
     1      -FR(J)))
   10     CONTINUE
   20    CONTINUE
   30   CONTINUE
        DIFFF = 0.0
        DO 40 K = 1, NC
         DIFFF = DIFFF + (((SNIN(K) - SPIN(K))/2)**2) * DL(K)
   40   CONTINUE
        PDIF = SQRT(DIFFF/4)
        NDIF = PDIF
        RT = 2
        GO TO 110
       ELSE
        DO 70 K = 1, NC
         SNIN(K) = 0.0
         DO 60 J = 1, NG
          DO 50 I = 1, NA
           SNIN(K) = SNIN(K) + (ABC(J,K)*NINT(I,J,K)*WF(I)*(FR(J + 1)
     1      -FR(J)))
   50     CONTINUE
   60    CONTINUE
   70   CONTINUE
        DIFFF = 0.0
        DO 80 K = 1, NC
         DIFFF = DIFFF + (((SNIN(K) - SPIN(K))/2)**2) * DL(K)
   80   CONTINUE
        NDIF = SQRT(DIFFF/4)
       END IF
       SPR = NDIF / PDIF
       PDIF = NDIF
       IF (PRES .GE. 2) THEN
          WRITE (6,90) TOIT, TIIT, TIT
   90   FORMAT (1X, 'TOIT=', I8, 3X, 'TIIT=', I8, 3X, 'TIT=', I8)
        WRITE (6,100) NDIF, SPR, SIRC, DSPR
  100   FORMAT (1X, 'INTE ERR=', 1PE10.3, 3X, 'TRAN SPR=', 1PE10.3, 3X,
     1      'SIRC=', 1PE10.3, 3X, 'DSPR=', 1PE10.3)
       END IF
       IF (SPR .GE. 1.00) THEN
        SMTST1 = SMTST1 + 1
        IF (SMTST1 .GE. 5) THEN
         WRITE (8,*)'transport terminated due to large spr'
         SMTST = .TRUE.
        END IF
       END IF
  110 DO 120 K = 1, NC
        SPIN(K) = SNIN(K)
  120 CONTINUE
       IF (CONVC .EQ. 0) THEN
        IF (NDIF .GE. IERR) GO TO 210
        CONV = 1
       ELSE IF (CONVC .EQ. 1) THEN
        DO 150 J = 1, NG
         DO 140 K = 1, NC
          DO 130 I = 1, NA
           IF (ABS((NINT(I,J,K) - PINT(I,J,K))/PINT(I,J,K)) .GE.
     1      IERR) GO TO 210
  130     CONTINUE
  140    CONTINUE
  150   CONTINUE
       ELSE IF (CONVC .EQ. 2) THEN
        DO 200 K = 1, NC
         TINT = 0.0
         DO 170 J = 1, NG
          DO 160 I = 1, NA
           TINT = TINT + PINT(I,J,K)
  160     CONTINUE
  170    CONTINUE
         DO 190 J = 1, NG
          DO 180 I = 1, NA
           IF (ABS((NINT(I,J,K) - PINT(I,J,K))/TINT) .GE. IERR)
     1      GO TO 210
  180     CONTINUE
  190    CONTINUE
  200   CONTINUE
       END IF
       CONV = 0
       RETURN
  210 CONV = 1
       IF (AUTACL .EQ. 1) THEN
        IF (SPR .GT. SPRB .AND. DSPR .LT. 0.80) THEN
         TRNACC = 1
         DIFACC = 2
        ELSE
         TRNACC = 0
        END IF
       END IF
       RETURN
       END
C
C    ****************************************************************
C    * following subroutine computes the iterative terms used to    *
C    * solve the transfer equation using SI.                        *
C    ****************************************************************
C
       SUBROUTINE TRNSI

       PARAMETER (NANG=24,NFG=64,NCL=151)

       INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
       INTEGER DRT, DIFBC, DIFACC, IIT, TITA
       INTEGER INSB, IRUN, LIN, MEDUSA
       INTEGER NA, NG, NC, NF, NDIT, NFPC, NDICNT, NFMUST
       INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
       INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
       INTEGER USC, WFN, CONV, DBXL, DBXU, NUM
       INTEGER NCP1, NCM1, NAD2

       COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
       COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
       COMMON /I3/ INSB, IRUN, LIN, MEDUSA
       COMMON /I4/ NA, NG, NC, NF, NDIT, NFPC, NDICNT, NFMUST
       COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
       COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
       COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM
       COMMON /I8/ NCP1, NCM1, NAD2

       REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
       REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
       REAL*8 WF(NANG)
       REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER

       COMMON /R02/ ABC, AB, AO
       COMMON /R15/ NATE, NINT
       COMMON /R27/ WF
       COMMON /R28/ ZLP, ZINT, ZQ, DIER

       DO 30 J = 1, NG
        DO 20 K = 1, NC
         ZINT(J,K) = 0.0
         DO 10 I = 1, NA
          ZINT(J,K) = ZINT(J,K) + NINT(I,J,K) * WF(I)
   10    CONTINUE
   20   CONTINUE
   30 CONTINUE
       DO 50 K = 1, NC
        AB(K) = 0.0
        DO 40 J = 1, NG
         AB(K) = AB(K) + ABC(J,K) * ZINT(J,K)
   40   CONTINUE
   50 CONTINUE
       RETURN
       END
C
C    ****************************************************************
C    * this subroutine solves the diffusion equation                *
C    ****************************************************************
C

       SUBROUTINE MFDE

       PARAMETER (NANG=24,NFG=64,NCL=151)
       PARAMETER (PI=3.141593)

       REAL*8 A(NCL + 1)
       REAL*8 B(NCL + 1)
       REAL*8 C(NCL + 1)
       REAL*8 R(NCL + 1)

       COMMON /IMT/ J, N
       COMMON /RMT/ A, B, C, R

       LOGICAL TRN, DIFF, LOR, NLOR, SIMRUN, SMTST
       COMMON /L1/ TRN, DIFF, LOR, NLOR, SIMRUN, SMTST

       INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
       INTEGER DRT, DIFBC, DIFACC, IIT, TITA
       INTEGER INSB, IRUN, LIN, MEDUSA
       INTEGER NA, NG, NC, NF, NDIT, NFPC, NDICNT, NFMUST
       INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
       INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
       INTEGER USC, WFN, CONV, DBXL, DBXU, NUM
       INTEGER NCP1, NCM1, NAD2
       INTEGER SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

       COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
       COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
       COMMON /I3/ INSB, IRUN, LIN, MEDUSA
       COMMON /I4/ NA, NG, NC, NF, NDIT, NFPC, NDICNT, NFMUST
       COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
       COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
       COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM
       COMMON /I8/ NCP1, NCM1, NAD2
       COMMON /SMT/ SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

       REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFG,NCL), ATEM(NCL)
       REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
       REAL*8 BETA(NCL), BO(NFG,NCL)
       REAL*8 CV(NCL), CW, CHI(NFG,NCL)
       REAL*8 DL(NCL), DINT(NFG,NCL + 1), PRNTM
       REAL*8 DIST(NCL), DTIME, DSPR
       REAL*8 ETA(NCL)
       REAL*8 FINT(NFG,NCL + 1), FINL(NFG), FINR(NFG)
       REAL*8 FR(NFG + 1), FQ(NFG,NCL)
       REAL*8 INTB(NANG,NFG,NCL + 1), INT(NANG,NFG,NCL), IZIN(NFG,NCL)
       REAL*8 ITRB, ITER(NFG,NCL)
       REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
       REAL*8 NFFT(NANG,NFG,NCL), NDIN(NFG,NCL + 1), NZIN(NFG,NCL)
       REAL*8 PMAC(NCL), PINT(NANG,NFG,NCL), PRTIME
       REAL*8 PGIN(NCL + 1), PFGI(NCL + 1), PLANK(NFG,NCL)
       REAL*8 PDIN(NFG,NCL + 1), PITE(NFG,NCL), PZIN(NFG,NCL)
       REAL*8 SIGM(NFG,NCL), ORSI(NCL + 1)
       REAL*8 SOLU(NCL + 1), SPRB
       REAL*8 S(NFG,NCL), SUM1, SUM2, ORFQ(NCL), ORZQ(NCL)
       REAL*8 SINT(NFG,NCL + 1)
       REAL*8 WF(NANG)
       REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER

       COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
       COMMON /R02/ ABC, AB, AO
```

```
      COMMON /R03/ BETA, BO
      COMMON /R04/ CV, CW, CHI
      COMMON /R05/ DL, DINT, PRNTM
      COMMON /R06/ DIST, DTIME, DSPR
      COMMON /R07/ ETA
      COMMON /R071/ FINT, FINL, FINR
      COMMON /R08/ FR, FQ
      COMMON /R11/ INTB, IINT, IZIN
      COMMON /R12/ ITRB, ITER
      COMMON /R15/ NATE, NINT
      COMMON /R16/ NFPT, NDIN, NZIN
      COMMON /R17/ PMAC, PINT, PRTIME
      COMMON /R18/ POIN, PFOL, PLANK
      COMMON /R181/ PDIN, PITE, PZIN
      COMMON /R21/ SIGM, ORSI
      COMMON /R22/ SOLU, SPRB
      COMMON /R23/ S, SUM1, SUM2, ORPQ, ORZQ
      COMMON /R231/ SINT
      COMMON /R27/ WF
      COMMON /R28/ ZLP, ZINT, ZQ, DIER

      IF (PRES .GE. 3 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1     THEN
       WRITE (8,*)'j k iter pite'
       WRITE (8,10) ((J,K,ITER(J,K),PITE(J,K),K=1,NC),J=1,NG)
   10  FORMAT (1X, I2, 1X, I2, 1X, 1PE10.3, 1X, 1PE10.3)
      END IF
      DO 60 J = 1, NG
       DO 20 K = 1, NCP1
        IF (K .EQ. 1) THEN
         A(K) = 0.0
         B(K) = SUM1 + SIGM(J,K) * BO(J,K) / 4 + 1 / (3*SIGM(J,K))
         C(K) = SIGM(J,K) * AO(J,K) / 4 - 1 / (3*SIGM(J,K))
        ELSE IF (K .EQ. NCP1) THEN
         A(K) = SIGM(J,K - 1) * BO(J,K - 1) / 4 - 1 / (3*SIGM(J,K -
     1    1))
         B(K) = -SUM2 + SIGM(J,K - 1) * AO(J,K - 1) / 4 + 1 / (3*
     1    SIGM(J,K - 1))
         C(K) = 0.0
        ELSE
         A(K) = SIGM(J,K - 1) * BO(J,K - 1) / 4 - 1 / (3*SIGM(J,K -
     1    1))
         B(K) = (SIGM(J,K - 1)*AO(J,K - 1) + SIGM(J,K)*BO(J,K)) / 4 +
     1    1 / (3*SIGM(J,K - 1)) + 1 / (3*SIGM(J,K))
         C(K) = SIGM(J,K) * AO(J,K) / 4 - 1 / (3*SIGM(J,K))
        END IF
   20  CONTINUE
       DO 30 K = 1, NCP1
        IF (K .EQ. 1) THEN
         R(K) = SUM1 * PDIN(J,K) + FINT(J,K) + 2 * (SINT(J,K + 1) -
     1    SINT(J,K)) / (3*SIGM(J,K)) + ZQ(J,K) * DL(K) / 2 - FQ(J,K) *
     2    DL(K) / SIGM(J,K) + ITER(J,K)
        ELSE IF (K .EQ. NCP1) THEN
         R(K) = -SUM2 * PDIN(J,K) - FINT(J,K) - 2 * (SINT(J,K) -
     1    SINT(J,K - 1)) / (3*SIGM(J,K - 1)) + ZQ(J,K - 1) * DL(K - 1)
     2    / 2 + FQ(J,K - 1) * DL(K - 1) / SIGM(J,K - 1) + ITER(J,K -
     3    1)
        ELSE
         R(K) = ITER(J,K - 1) + ITER(J,K) + S(J,K - 1)
        END IF
   30  CONTINUE
       IF (PRES .GE. 7 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1     THEN
        WRITE (8,*)'a b c r in group ', J
        WRITE (8,40) (K,A(K),B(K),C(K),R(K),K=1,NCP1)
   40   FORMAT (1X, 'k', I2, 1X, 1PE10.3, 1X, 1PE10.3, 1X, 1PE10.3,
     1      1X, 1PE10.3)
       END IF
       N = NCP1
C      iuvert the diffusion equation
       CALL LU3
       DO 50 K = 1, NCP1
        NDIN(J,K) = SOLU(K)
   50  CONTINUE
   60 CONTINUE
C**** keep track of the # of times negative diffusion intensity occurs
      DO 80 J = 1, NG
       DO 70 K = 1, NCP1
        IF (NDIN(J,K) .LT. 0) THEN
         NDICNT = NDICNT + 1
        END IF
   70  CONTINUE
   80 CONTINUE
      IF (PRES .GE. 6 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1     THEN
       WRITE (8,*)'dint ndin', TOIT, TIIT, OIT, IIT, TIT
       WRITE (8,90) ((J,K,DINT(J,K),NDIN(J,K),K=1,NCP1),J=1,NG)
   90  FORMAT (2(1X,'j',I2,1X,'k',I2,2(1PE10.3,1X)))
      END IF
      RETURN
      END
C
C     ****************************************************************
C     * this subroutine computes the spectral radius and determinant *
C     * of diffusion equation                                        *
C     ****************************************************************
C
      SUBROUTINE DPCONV

      PARAMETER (NANG=24,NFG=64,NCL=151)

      REAL*8 DDIFF, DPDIF, DNDIF
      REAL*8 DSPIN(NCL), DSNIN(NCL)
      REAL*8 TINT

      LO 4CAL TRN, DIFF, LGR, NLGR, SIMRUN, SMTST
      COMMON /L1/ TRN, DIFF, LGR, NLGR, SIMRUN, SMTST

      INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      INTEGER DRT, DIFBC, DIFACC, IIT, TITA
      INTEGER INSB, IRUN, LIN, MEDUSA
      INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      INTEGER USC, WFN, CONV, DBXL, DBXU, NUM
      INTEGER NCP1, NCM1, NAD2
      INTEGER SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

      COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
      COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
      COMMON /I3/ INSB, IRUN, LIN, MEDUSA
      COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM
      COMMON /I8/ NCP1, NCM1, NAD2
      COMMON /SMT/ SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

      REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
      REAL*8 DL(NCL), DINT(NFG,NCL + 1), PRNTM
      REAL*8 DIST(NCL), DTIME, DSPR
      REAL*8 FR(NFG + 1), FQ(NFG,NCL)
      REAL*8 NFPT(NANG,NFG,NCL), NDIN(NFG,NCL + 1), NZIN(NFG,NCL)
      REAL*8 WF(NANG)
      REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER

      COMMON /R02/ ABC, AB, AO
      COMMON /R05/ DL, DINT, PRNTM
      COMMON /R06/ DIST, DTIME, DSPR
      COMMON /R08/ FR, FQ
      COMMON /R16/ NFPT, NDIN, NZIN
      COMMON /R27/ WF
      COMMON /R28/ ZLP, ZINT, ZQ, DIER

      IF (DRT .LT. 1) THEN
       DO 20 K = 1, NC
        DSNIN(K) = 0.0
        DSPIN(K) = 0.0
        DO 10 J = 1, NG
         DSPIN(K) = DSPIN(K) + ABC(J,K) * DINT(J,K) * (FR(J + 1) -
     1    FR(J))
         DSNIN(K) = DSNIN(K) + ABC(J,K) * NDIN(J,K) * (FR(J + 1) -
     1    FR(J))
   10   CONTINUE
   20  CONTINUE
       DDIFF = 0.0
       DO 30 K = 1, NC
        DDIFF = DDIFF + (((DSNIN(K) - DSPIN(K))/2)**2) * DL(K)
   30  CONTINUE
       DPDIF = SQRT(DDIFF/4)
       DRT = 2
       DO 40 K = 1, NC
        DSPIN(K) = DSNIN(K)
   40  CONTINUE
       GO TO 120
      ELSE
       DO 60 K = 1, NC
        DSNIN(K) = 0.0
        DO 50 J = 1, NG
         DSNIN(K) = DSNIN(K) + ABC(J,K) * NDIN(J,K) * (FR(J + 1) -
     1    FR(J))
   50   CONTINUE
   60  CONTINUE
       DDIFF = 0.0
       DO 70 K = 1, NC
        DDIFF = DDIFF + (((DSNIN(K) - DSPIN(K))/2)**2) * DL(K)
   70  CONTINUE
       DNDIF = SQRT(DDIFF/4)
      END IF
      DSPR = DNDIF / DPDIF
      DPDIF = DNDIF
      IF (PRES .GE. 3 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1     THEN
       WRITE (8,80) DNDIF, DSPR
   80  FORMAT (1X, 'DINT ERR=', 1PE10.3, 5X, 'DIFF SPR=', 1PE10.3)
      END IF
      GO TO 110
      DO 100 J = 1, NG
       DO 90 K = 1, NC
        IF (NDIN(J,K)/DINT(J,K) .GT. 100.0 .OR. NDIN(J,K)/DINT(J,K) .
     1      LE. 0.01) THEN
         SMTST4 = SMTST4 + 1
        END IF
   90  CONTINUE
  100 CONTINUE
      IF (SMTST4 .GT. (NG*NC)*0.05) THEN
       SMTST = .TRUE.
       WRITE (99,*)'inner iteration terminated due to unphysical increa
     1se in ndin'
      END IF
  110 IF (DSPR .GE. 1.00) THEN
       SMTST3 = SMTST3 + 1
       IF (SMTST3 .GE. (NG*NC*0.01)) THEN
        WRITE (99,*)'inner iteration terminated due to dspr being > 1.
     1 0 too often'
        SMTST = .TRUE.
       END IF
      END IF
  120 DO 130 K = 1, NC
       DSPIN(K) = DSNIN(K)
  130 CONTINUE
      IF (CONVC .EQ. 0) THEN
```

```
        IF (DNDIF .GE. DIER) GO TO 190
       ELSE IF (CONVC .EQ. 1) THEN
        DO 150 J = 1, NG
         DO 140 K = 1, NCP1
          IF (ABS(NDIN(J,K)/DINT(J,K) - 1) .GE. DIER) GO TO 190
  140    CONTINUE
  150  CONTINUE
       ELSE IF (CONVC .EQ. 2) THEN
        DO 180 J = 1, NG
         TINT = 0.0
         DO 160 K = 1, NCP1
          TINT = TINT + NDIN(J,K)
  160    CONTINUE
         DO 170 K = 1, NCP1
          IF (ABS((NDIN(J,K) - DINT(J,K))/TINT) .GE. DIER)
     1       GO TO 190
  170    CONTINUE
  180  CONTINUE
       END IF
       CONV = 0
       RETURN
  190 CONV = 1
       RETURN
       END
C
C   ****************************************************************
C   * this subroutine solves the diffusion equation using SI       *
C   ****************************************************************
C
       SUBROUTINE DIFFSI

       PARAMETER (NANG=24,NFG=64,NCL=151)

       INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
       INTEGER DRT, DIFBC, DIFACC, IIT, TITA
       INTEGER INSB, IRUN, LIN, MEDUSA
       INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
       INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
       INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
       INTEGER USC, WFN, CONV, DBXL, DBXU, NUM
       INTEGER NCP1, NCM1, NAD2
       INTEGER SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

       COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
       COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
       COMMON /I3/ INSB, IRUN, LIN, MEDUSA
       COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
       COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
       COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
       COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM
       COMMON /I8/ NCP1, NCM1, NAD2
       COMMON /SMT/ SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

       REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFG,NCL), ATEM(NCL)
       REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
       REAL*8 BETA(NCL), BO(NFG,NCL)
       REAL*8 CV(NCL), CW, CHI(NFG,NCL)
       REAL*8 DL(NCL), DINT(NFG,NCL + 1), PRNTM
       REAL*8 DIST(NCL), DTIME, DSPR
       REAL*8 ETA(NCL)
       REAL*8 ITRB, ITER(NFG,NCL)
       REAL*8 NFFT(NANG,NFG,NCL), NDIN(NFG,NCL + 1), NZIN(NFG,NCL)
       REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER

       COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
       COMMON /R02/ ABC, AB, AO
       COMMON /R03/ BETA, BO
       COMMON /R04/ CV, CW, CHI
       COMMON /R05/ DL, DINT, PRNTM
       COMMON /R06/ DIST, DTIME, DSPR
       COMMON /R07/ ETA
       COMMON /R12/ ITRB, ITER
       COMMON /R16/ NFFT, NDIN, NZIN
       COMMON /R28/ ZLP, ZINT, ZQ, DIER

       DO 20 K = 1, NC
        AB(K) = 0.0
        DO 10 J = 1, NG
         AB(K) = AB(K) + ABC(J,K) * (AO(J,K)*NDIN(J,K + 1) + BO(J,K)*
     1   NDIN(J,K))
   10  CONTINUE
   20 CONTINUE
       IF (PRES .GE. 4 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1     THEN
        WRITE (8,*)'ab sidiff'
        WRITE (8,30) (K,AB(K),K=1,NC)
   30 FORMAT (1X, 4(1X,'k',I'  ' X,1PE10.3,1X))
       END IF
       DO 50 K = 1, NC
        DO 40 J = 1, NG
         ITER(J,K) = DL(K) * CHI(J,K) * ETA(K) * AB(K) / 4
   40  CONTINUE
   50 CONTINUE
       IF (DIFACC .EQ. 1 .AND. IIT .GE. 2) THEN
        DO 70 K = 1, NC
         DO 60 J = 1, NG
          DAF(J,K) = (AO(J,K)*NDIN(J,K + 1) + BO(J,K)*NDIN(J,K)) / (
     1    AO(J,K)*DINT(J,K + 1) + BO(J,K)*DINT(J,K))
   60    CONTINUE
   70  CONTINUE
        DO 90 K = 1, NC
         DO 80 J = 1, NG
          ITER(J,K) = ITER(J,K) * DAF(J,K)
   80    CONTINUE
   90  CONTINUE
       END IF
       RETURN
       END
C
C   ****************************************************************
C   *    this subroutine computes temperature in each cell         *
C   ****************************************************************
C
       SUBROUTINE TEMPER

       PARAMETER (NANG=24,NFG=64,NCL=151)
       PARAMETER (RK=7.5606E-16,PK=6.6261E-34)
       PARAMETER (BK=1.3807E-23,SP=3.0E08,EVB=1.6022E-19)
       PARAMETER (TCF=EVB/BK,OK=8.3144,PI=3.141593,AO=PK/(BK*TCF))

       REAL*8 NERED(NCL)

       INTEGER INSB, IRUN, LIN, MEDUSA
       INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
       INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
       INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
       INTEGER USC, WFN, CONV, DBXL, DBXU, NUM

       COMMON /I3/ INSB, IRUN, LIN, MEDUSA
       COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
       COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
       COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
       COMMON /I7/ USC, WFN, CONV, DBXL, DBXU, NUM

       REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
       REAL*8 BETA(NCL), BO(NFG,NCL)
       REAL*8 DIST(NCL), DTIME, DSPR
       REAL*8 ETA(NCL)
       REAL*8 INLB(NFG), INRB(NFG), ITSL, IERED(NCL), IERR
       REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
       REAL*8 PMAC(NCL), PINT(NANG,NFG,NCL), PRTIME
       REAL*8 TTIME, TOU, TERR, TELB

       COMMON /R02/ ABC, AB, AO
       COMMON /R03/ BETA, BO
       COMMON /R06/ DIST, DTIME, DSPR
       COMMON /R07/ ETA
       COMMON /R10/ INLB, INRB, ITSL, IERED, IERR
       COMMON /R15/ NATE, NINT
       COMMON /R17/ PMAC, PINT, PRTIME
       COMMON /R24/ TTIME, TOU, TERR, TELB

       DO 10 K = 1, NC
        NERED(K) = (ETA(K)/(SP*PMAC(K))) * (2*PI*AB(K) + IERED(K)/(
     1  DTIME*BETA(K)))
        NATE(K) = ((NERED(K)/RK)**0.25)
   10 CONTINUE
       IF (MEDUSA .EQ. 0) THEN
        DO 20 K = 1, NC
         NATE(K) = NATE(K) / TCF
   20   CONTINUE
         IF (PRES .GE. 1 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1       THEN
          WRITE (6,30) TTIME, TOIT, TIIT, TIT
   30     FORMAT (1X, 'temp in:', 1PE10.3, 5X, 'toit tiit tit=',
     1        3(1X,I5))
          WRITE (6,40) (DIST(K),NATE(K),K=1,NC)
   40     FORMAT (3(1PE10.3,1X,1PE10.3,1X))
         END IF
       END IF
       RETURN
       END
C
C ******************************************************************
C *    determine if temperature is converged                        *
C ******************************************************************
C
       SUBROUTINE TEMCON

       PARAMETER (NANG=24,NFG=64,NCL=151)

       INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
       INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
       INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN

       COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
       COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
       COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN

       REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFG,NCL), ATEM(NCL)
       REAL*8 DIST(NCL), DTIME, DSPR
       REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
       REAL*8 TTIME, TOU, TERR, TELB

       COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
       COMMON /R06/ DIST, DTIME, DSPR
       COMMON /R15/ NATE, NINT
       COMMON /R24/ TTIME, TOU, TERR, TELB

       LOGICAL TRN, DIFF, LOR, NLOR, SIMRUN, SMTST
       COMMON /L1/ TRN, DIFF, LOR, NLOR, SIMRUN, SMTST

       IF (TCONT .EQ. 0) THEN
        GO TO 30
       END IF
       DO 10 K = 1, NC
        IF (ABS(NATE(K)/ATEM(K) - 1) .GT. TERR) THEN
         CONT = 1
         GO TO 30
        END IF
   10 CONTINUE
       CONT = 0
```

```
      IF ( NOT SIMRUN) THEN
       DO 20 K = 1, NC
        IF (ABS(NATE(K)/ATEM(K) - 1) GT 0 01) THEN
         GO TO 30
        END IF
   20  CONTINUE
       DTIME = 2 * DTIME
      END IF
   30 DO 40 K = 1, NC
       ATEM(K) = NATE(K)
   40 CONTINUE
      RETURN
      END

C**** f1 function is used for computing planckian absorption coefficient
C**** and initial intensity (which is assumed to be planckian)

      FUNCTION FPLNK(X)

      REAL*8 X, FPLNK

      INTRINSIC EXP

      IF (ABS(X) .LT. 1 E-06) THEN
       FPLNK = X * X
      ELSE
       FPLNK = (X**3) / (EXP(X) - 1.0)
      END IF
      RETURN
      END

C**** f2 and f3 functions are used for computing rosseland absorp
C**** ion coefficient

      FUNCTION F2(X)
      REAL*8 X, F2
      INTRINSIC EXP
      IF (ABS(X) .LT. 1.E-6) THEN
       F2 = X * X * EXP(X)
      ELSE
       F2 = ((X**4)*EXP(X)) / ((EXP(X) - 1.0)**2)
      END IF
      RETURN
      END
C
      FUNCTION F3(X)
      REAL*8 X, F3
      INTRINSIC DEXP
      IF (ABS(X) .LT. 1.E-6) THEN
       F3 = X ** 4 * EXP(2.*X)
      ELSE
       F3 = ((X**7)*EXP(2.*X)) / ((EXP(X) - 1.0)**3)
      END IF
      RETURN
      END
C
      FUNCTION FSIG(X)

      REAL*8 X, FSIG

      INTRINSIC EXP

      IF (ABS(X) .LT. 1.E-06) THEN
       FSIG = -1 / X ** 2
      ELSE
       FSIG = (1 - EXP(-X)) / X ** 3
      END IF
      RETURN
      END
C
C *******************************************************************
C *    following subroutine computes the integral of a given fun-   *
C * ction using romberg method.                                *
C *******************************************************************
C
      SUBROUTINE ROMB(F, D, E, RES)

      REAL*8 D, DEL, DIFF
      REAL*8 ERR
      REAL*8 E, F
      REAL*8 RES
      REAL*8 X, Z(20,20)

      EXTERNAL F

      IF (D .EQ. 0) THEN
       D = 1.0E-06
      END IF
      ERR = 1.0E-06 * ((D/E)**4)
      I = 1
      DEL = E - D
      Z(1,1) = 0.5 * DEL * (F(D) + F(E))
   10 J = 2 ** (I - 1)
      DEL = DEL / 2
      I = I + 1
      Z(I,1) = 0.5 * Z(I - 1,1)
      DO 20 K = 1, J
       X = D + ((2.*K) - 1) * DEL
       Z(I,1) = Z(I,1) + DEL * F(X)
   20 CONTINUE
      DO 30 K = 2, I
       Z(I,K) = (4.**(K - 1)*Z(I,K - 1) - Z(I - 1,K - 1)) / (4.**(K -
     1 1) - 1.)
   30 CONTINUE
      DIFF = ABS(Z(I,I) - Z(I,I - 1))
      IF (DIFF .LT. ERR) GO TO 40
      IF (I .LT. 20) GO TO 10
      WRITE (6,*)'required precision could not be achieved'
      STOP
   40 RES = Z(I,I)
      RETURN
      END
C
C  *****************************************************************
C  * this subroutine inverts the tridiagonal matrix by LU        *
C  * decomposition                                       *
C  *****************************************************************
C
      SUBROUTINE LU3

      PARAMETER (NANO=24,NFO=64,NCL=151)

      LOGICAL TRN, DIFF, LOR, NLOR, SIMRUN, SMTST
      COMMON /L1/ TRN, DIFF, LOR, NLOR, SIMRUN, SMTST

      INTEGER DRT, DIFBC, DIFACC, IIT, TITA
      INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      INTEGER UBC, WFN, CONV, DBXL, DBXU, NUM
      INTEGER NCP1, NCM1, NAD2
      INTEGER SMTST1, SMTST2, SMTST3, SMTST4, SMTST5

      COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
      COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
      COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
      COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
      COMMON /I7/ UBC, WFN, CONV, DBXL, DBXU, NUM
      COMMON /I8/ NCP1, NCM1, NAD2
      COMMON /SMT/ SMTST1, SMTST2, SMTST3, SMTST4, SMTST5
      COMMON /RMT/ A, B, C, R

      REAL*8 DL(NCL), DINT(NFO,NCL + 1), PRNTM
      COMMON /R05/ DL, DINT, PRNTM

      REAL*8 A(NCL + 1), B(NCL + 1), C(NCL + 1), R(NCL + 1)
      REAL*8 SOLU(NCL + 1), SPRB
      REAL*8 OA(NCL + 1), OB(NCL + 1), OC(NCL + 1)
      REAL*8 OR(NCL + 1)
      REAL*8 NR(NCL + 1), ERR(NCL + 1)
      REAL*8 GAM(NCL + 1), BET

      COMMON /R22/ SOLU, SPRB

      COMMON /IMT/ J, N

      IF (PRES .GE. 6 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1   THEN
       WRITE (8,*)'values in lu of group ', J
       WRITE (8,*)'a b c r'
       WRITE (8,10) (K,A(K),B(K),C(K),R(K),K=1,N)
   10  FORMAT (1X, 'k', I2, 1X, 1PE11.4, 1X, 1PE11.4, 1X, 1PE11.4, 1X,
     1     1PE11.4)
      END IF

      DO 20 K = 1, N
       OA(K) = A(K)
       OC(K) = C(K)
       OB(K) = B(K)
       OR(K) = R(K)
   20 CONTINUE
      IF (B(1) .EQ. 0.0) THEN
       WRITE (8,*)'SOLVE SOLU(2) BY ELIMINATION, LEAVING N-1 EQUATION'
       PAUSE
      END IF
      BET = B(1)
      SOLU(1) = R(1) / BET
      DO 30 K = 2, N
       GAM(K) = C(K - 1) / BET
       BET = B(K) - A(K) * GAM(K)
       SOLU(K) = (R(K) - A(K)*SOLU(K - 1)) / BET
   30 CONTINUE
      DO 40 K = N - 1, 1, -1
       SOLU(K) = SOLU(K) - GAM(K + 1) * SOLU(K + 1)
   40 CONTINUE
      DO 50 K = 1, N
       IF (K .EQ. 1) THEN
        NR(K) = OB(K) * SOLU(K) + OC(K) * SOLU(K + 1)
       ELSE IF (K .EQ. N) THEN
        NR(K) = OA(K) * SOLU(K - 1) + OB(K) * SOLU(K)
       ELSE
        NR(K) = OA(K) * SOLU(K - 1) + OB(K) * SOLU(K) + OC(K) * SOLU(
     1  K + 1)
       END IF
       IF (NR(K) .EQ. 0) THEN
        ERR(K) = -999.9999
       ELSE
        ERR(K) = 1 - OR(K) / NR(K)
       END IF
   50 CONTINUE
      DO 60 K = 1, N
       IF (ABS(ERR(K)) .GT. 0.01) GO TO 70
   60 CONTINUE
      RETURN
   70 IF (NPR .EQ. 0) THEN
       WRITE (6,*)'soln is ill behaved for group', J, TOIT, IIT, TIT
       WRITE (6,80) (K,A(K),B(K),C(K),R(K),K=1,N)
   80  FORMAT (1X, I2, 1X, 'a=', 1PE10.3, 1X, 'b=', 1PE10.3, 1X, 'c=',
     1      1PE10.3, 1X, 'r=', 1PE10.3)
       WRITE (6,90) (K,OR(K),NR(K),SOLU(K),DINT(J,K),K=1,N)
   90  FORMAT (1X, I2, 'or=', 1PE10.3, 1X, 'nr=', 1PE10.3, 1X, 'solu=',
     1      1PE10.3, 1X, 'dint=', 1PE10.3)
```

```
          SMTST2 = SMTST2 + 1
          IF (SMTST2 .GT. INT((NG*NC)*.05)) THEN
            WRITE (6,*)'program terminated in LU smtst2'
            SMTST = .TRUE.
          END IF
          WRITE (6,*)'enter 1 if above message to be removed'
          READ (5,*) NPR
        END IF
        RETURN
        END
C
C ******************************************************************
C *    following subroutine provides zeroes and weight functions   *
C * of Legendre PoLynomial                                         *
C ******************************************************************
C
        SUBROUTINE LPL

        PARAMETER (NANG=24,NFG=64,NCL=151)

        INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
        INTEGER NCP1, NCM1, NAD2
        COMMON /I8/ NCP1, NCM1, NAD2

        REAL*8 S(NFG,NCL), SUM1, SUM2, GRFQ(NCL), GRZQ(NCL)
        REAL*8 WF(NANG)
        REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER

        COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST

        COMMON /R23/ S, SUM1, SUM2, GRFQ, GRZQ
        COMMON /R27/ WF
        COMMON /R28/ ZLP, ZINT, ZQ, DIER

        IF (NA .EQ. 2) THEN
          WF(1) = 1.000000
          ZLP(1) = 0.577353
        ELSE IF (NA .EQ. 4) THEN
          WF(1) = 0.652142
          WF(2) = 0.347858
          ZLP(1) = 0.339981
          ZLP(2) = 0.861136
        ELSE IF (NA .EQ. 6) THEN
          WF(1) = 0.467914
          WF(2) = 0.360762
          WF(3) = 0.171325
          ZLP(1) = 0.238619
          ZLP(2) = 0.661209
          ZLP(3) = 0.932470
        ELSE IF (NA .EQ. 8) THEN
          WF(1) = 0.362684
          WF(2) = 0.313707
          WF(3) = 0.222381
          WF(4) = 0.101229
          ZLP(1) = 0.183435
          ZLP(2) = 0.525532
          ZLP(3) = 0.796667
          ZLP(4) = 0.960290
        ELSE IF (NA .EQ. 10) THEN
          WF(1) = 0.295524
          WF(2) = 0.269267
          WF(3) = 0.219086
          WF(4) = 0.149451
          WF(5) = 0.066671
          ZLP(1) = 0.148874
          ZLP(2) = 0.433395
          ZLP(3) = 0.679410
          ZLP(4) = 0.865063
          ZLP(5) = 0.973907
        ELSE IF (NA .EQ. 12) THEN
          WF(1) = 0.249147
          WF(2) = 0.233493
          WF(3) = 0.203167
          WF(4) = 0.160078
          WF(5) = 0.106939
          WF(6) = 0.047175
          ZLP(1) = 0.125233
          ZLP(2) = 0.367832
          ZLP(3) = 0.587318
          ZLP(4) = 0.769903
          ZLP(5) = 0.904117
          ZLP(6) = 0.981561
        ELSE IF (NA .EQ. 16) THEN
          WF(1) = 0.189451
          WF(2) = 0.182603
          WF(3) = 0.169157
          WF(4) = 0.149596
          WF(5) = 0.124629
          WF(6) = 0.095159
          WF(7) = 0.062254
          WF(8) = 0.027152
          ZLP(1) = 0.095013
          ZLP(2) = 0.281604
          ZLP(3) = 0.458017
          ZLP(4) = 0.617876
          ZLP(5) = 0.755404
          ZLP(6) = 0.865631
          ZLP(7) = 0.944575
          ZLP(8) = 0.989401
        ELSE IF (NA .EQ. 20) THEN
          WF(1) = 0.152753
          WF(2) = 0.149173
          WF(3) = 0.142096
          WF(4) = 0.131689
          WF(5) = 0.118195
          WF(6) = 0.101930
          WF(7) = 0.083277
          WF(8) = 0.062672
          WF(9) = 0.040601
          WF(10) = 0.017614
          ZLP(1) = 0.076527
          ZLP(2) = 0.227786
          ZLP(3) = 0.373706
          ZLP(4) = 0.510867
          ZLP(5) = 0.636054
          ZLP(6) = 0.746332
          ZLP(7) = 0.839117
          ZLP(8) = 0.912234
          ZLP(9) = 0.963972
          ZLP(10) = 0.993129
        ELSE IF (NA .EQ. 24) THEN
          WF(1) = 0.127938
          WF(2) = 0.125837
          WF(3) = 0.121670
          WF(4) = 0.115506
          WF(5) = 0.107444
          WF(6) = 0.097619
          WF(7) = 0.086190
          WF(9) = 0.073346
          WF(10) = 0.044277
          WF(11) = 0.028531
          WF(12) = 0.012341
          ZLP(1) = 0.064057
          ZLP(2) = 0.191119
          ZLP(3) = 0.315043
          ZLP(4) = 0.433794
          ZLP(5) = 0.545421
          ZLP(6) = 0.648909
          ZLP(7) = 0.740124
          ZLP(9) = 0.886416
          ZLP(10) = 0.938275
          ZLP(11) = 0.974729
          ZLP(12) = 0.995187
        ELSE
          WRITE (6,10)
     10   FORMAT (1X, 'NA CAN NOT BE EQUAL TO ODD # OR 14,18 OR 22, CHANGE
     1    THE VALUE FOR NA')
          STOP
        END IF
        DO 20 I = 1, NAD2
          WF(NA + 1 - I) = WF(I)
          ZLP(NA + 1 - I) = -ZLP(I)
     20 CONTINUE
        SUM1 = 0.0
        SUM2 = 0.0
        DO 30 I = 1, NAD2
          SUM1 = SUM1 + ZLP(I) * WF(I)
     30 CONTINUE
        DO 40 I = NAD2 + 1, NA
          SUM2 = SUM2 + ZLP(I) * WF(I)
     40 CONTINUE
        RETURN
        END
C
C   ****************************************************************
C   * this subroutine runs this code for the model problem for     *
C   * cases.                                                       *
C   ****************************************************************
C
        SUBROUTINE RUNTST(IT)

        PARAMETER (NANG=24,NFG=64,NCL=151)

        INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
        INTEGER DRT, DIFBC, DIFACC, IIT, TITA
        INTEGER INSB, IRUN, LIN, MEDUSA
        INTEGER OIT, OITA, PRES, RT, SETBOU, SCR, TRNACC

        COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
        COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
        COMMON /I3/ INSB, IRUN, LIN, MEDUSA
        COMMON /I5/ OIT, OITA, PRES, RT, SETBOU, SCR, TRNACC

        LOGICAL TRN, DIFF, LOR, NLOR, SIMRUN, SMTST
        COMMON /L1/ TRN, DIFF, LOR, NLOR, SIMRUN, SMTST

        INTEGER NDB(8), IT
        REAL*8 TMEX(16)
        REAL*8 MAXT
        REAL*8 TTIME, TOU, TERR, TELB
        REAL*8 DIST(NCL), DTIME, DSPR

        COMMON /R14/ MAXT
        COMMON /R24/ TTIME, TOU, TERR, TELB
        COMMON /R06/ DIST, DTIME, DSPR

        IF (IT .EQ. 0) THEN
          NDB(1) = 2
          NDB(2) = 5
          NDB(3) = 10
          NDB(4) = 20
          NDB(5) = 50
          NDB(6) = 100
          NDB(7) = 500
          NDB(8) = 1000
          TMEX(1) = 1.00E-12
          TMEX(2) = 2.00E-12
          TMEX(3) = 5.00E-12
          TMEX(4) = 1.00E-11
          TMEX(5) = 2.00E-11
          TMEX(6) = 5.00E-11
          TMEX(7) = 1.00E-10
```

```
          TMEX(8) = 2.00E-10
          TMEX(9) = 5.00E-10
          TMEX(10) = 1.00E-09
          TMEX(11) = 2.00E-09
          TMEX(12) = 5.00E-09
          TMEX(13) = 1.00E-08
          TMEX(14) = 2.00E-08
          TMEX(15) = 5.00E-08
          TMEX(16) = 2.00E-07
          K = 0
        END IF
        WRITE (8,*)'trn db it=', TRN, DB, IT + 1
        IF ( .NOT. TRN) THEN
          IT = IT + 1
          DTIME = TMEX(IT)
          TTIME = DTIME
          TRN = .TRUE.
          DB = NDB(1)
          IF (TTIME .GE. MAXT) THEN
            WRITE (6,*)'analysis is complete'
            STOP
          ELSE
            RETURN
          END IF
        END IF
     10 IF ( .NOT. DIFF) THEN
          TRNACC = 1
          DIFACC = 0
          IF (DB .GT. 500) THEN
            K = 0
            DIFF = .TRUE.
            DIFACC = 2
            DB = NDB(1)
            LIN = 0
            GO TO 10
          ELSE
            K = K + 1
            DB = NDB(K)
            RETURN
          END IF
        ELSE IF ( .NOT. LGR) THEN
          IF (DB .GT. 500) THEN
            K = 0
            LGR = .TRUE.
            DB = NDB(1)
            LIN = 1
            GO TO 10
          ELSE
            K = K + 1
            DB = NDB(K)
            RETURN
          END IF
          WRITE (8,*)'nlgr dr', DIFF, DB
        ELSE IF ( .NOT. NLGR) THEN
          IF (DB .GT. 500) THEN
            K = 0
            NLGR = .TRUE.
            DB = NDB(1)
            GO TO 10
          ELSE
            K = K + 1
            DB = NDB(K)
            RETURN
          END IF
        ELSE
          WRITE (8,*)'trn diff lgr nlgr dr', TRN, DIFF, LGR, NLGR, SIMRUN,
     1        DB
          TRN = .FALSE.
          DIFF = .FALSE.
          LGR = .FALSE.
          NLGR = .FALSE.
          DB = NDB(1)
          GO TO 10
        END IF
        END
C
C     *****************************************************************
C     * this subroutine solves the grey equation                      *
C     *****************************************************************
C

        SUBROUTINE GRDE

        PARAMETER (NANO=24,NFO=64,NCL=151)
        PARAMETER (PI=3.141593)

        LOGICAL TRN, DIFF, LGR, NLGR, SIMRUN, SMTST
        COMMON /L1/ TRN, DIFF, LGR, NLGR, SIMRUN, SMTST

        INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
        INTEGER DRT, DIFBC, DIFACC, IIT, TITA
        INTEGER INBB, IRUN, LIN, MEDUSA
        INTEGER NA, NG, NC, NF, NDIT, NFPC, NDICNT, NFMUST
        INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
        INTEGER TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
        INTEGER UBC, WFN, CONV, DBXL, DBXU, NUM
        INTEGER NCP1, NCM1, NAD2

        COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
        COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
        COMMON /I3/ INBB, IRUN, LIN, MEDUSA
        COMMON /I4/ NA, NG, NC, NF, NDIT, NFPC, NDICNT, NFMUST
        COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
        COMMON /I6/ TB, TEMEXP, TIIT, TOIT, TEMPTD, TIT, TRIN
        COMMON /I7/ UBC, WFN, CONV, DBXL, DBXU, NUM
        COMMON /I8/ NCP1, NCM1, NAD2

        REAL*8 SUMTH(NCL), SUMPH(NCL)

        REAL*8 AAO(NCL), A(NCL + 1)
        REAL*8 BBO(NCL), B(NCL + 1)
        REAL*8 C(NCL + 1)
        REAL*8 DUMF(NCL)
        REAL*8 FUNA(NCL), FUNC(NCL), FUND(NCL), FUNE(NCL)
        REAL*8 FUNF(NFO,NCL), FUNG(NFO,NCL), FUNH(NCL)
        REAL*8 GINT(NCL + 1), GRFI(NCL + 1)
        REAL*8 ITRB, ITER(NFO,NCL)
        REAL*8 NGIN(NCL + 1)
        REAL*8 NFPT(NANO,NFO,NCL), NDIN(NFO,NCL + 1), NZIN(NFO,NCL)
        REAL*8 PHI(NFO,NCL)
        REAL*8 R(NCL + 1)
        REAL*8 GRS(NCL), SDDI(NCL), SUMZ(NCL), SL, SR
        REAL*8 THETA(NFO,NCL)
        REAL*8 ZETA(NFO,NCL)

        COMMON /IMT/ J, N
        COMMON /RMT/ A, B, C, R

        REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFO,NCL), ATEM(NCL)
        REAL*8 ABC(NFO,NCL), AB(NCL), AO(NFO,NCL)
        REAL*8 BETA(NCL), BO(NFO,NCL)
        REAL*8 CV(NCL), CW, CHI(NFO,NCL)
        REAL*8 DL(NCL), DINT(NFO,NCL + 1), PRNTM
        REAL*8 ETA(NCL)
        REAL*8 FR(NFO + 1), FQ(NFO,NCL)
        REAL*8 FINT(NFO,NCL + 1), FINL(NFO), FINR(NFO)
        REAL*8 INTB(NANO,NFO,NCL + 1), IINT(NANO,NFO,NCL), IZIN(NFO,NCL)
        REAL*8 PGIN(NCL + 1), PFGI(NCL + 1), PLANK(NFO,NCL)
        REAL*8 PDIN(NFO,NCL + 1), PITE(NFO,NCL), PZIN(NFO,NCL)
        REAL*8 PAB(NCL)
        REAL*8 SIGM(NFO,NCL), GRSI(NCL + 1)
        REAL*8 SOLU(NCL + 1), SPRB
        REAL*8 S(NFO,NCL), SUM1, SUM2, GRFQ(NCL), GRZQ(NCL)
        REAL*8 SINT(NFO,NCL + 1)
        REAL*8 TTIME, TOU, TERR, TELB
        REAL*8 ZLP(NANG), ZINT(NFO,NCL), ZQ(NFO,NCL), DIER

        COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
        COMMON /R02/ ABC, AB, AO
        COMMON /R03/ BETA, BO
        COMMON /R04/ CV, CW, CHI
        COMMON /R05/ DL, DINT, PRNTM
        COMMON /R07/ ETA
        COMMON /R071/ FINT, FINL, FINR
        COMMON /R08/ FR, FQ
        COMMON /R11/ INTB, IINT, IZIN
        COMMON /R12/ ITRB, ITER
        COMMON /R16/ NFPT, NDIN, NZIN
        COMMON /R18/ PGIN, PFGI, PLANK
        COMMON /R181/ PDIN, PITE, PZIN
        COMMON /R182/ PAB
        COMMON /R21/ SIGM, GRSI
        COMMON /R22/ SOLU, SPRB
        COMMON /R23/ S, SUM1, SUM2, GRFQ, GRZQ
        COMMON /R231/ SINT
        COMMON /R24/ TTIME, TOU, TERR, TELB
        COMMON /R28/ ZLP, ZINT, ZQ, DIER

        DO 20 K = 1, NCP1
          GINT(K) = 0.0
          DO 10 J = 1, NG
            GINT(K) = GINT(K) + NDIN(J,K)
     10   CONTINUE
     20 CONTINUE
        DO 40 K = 1, NC
          AAO(K) = 0.0
          BBO(K) = 0.0
          DO 30 J = 1, NG
            AAO(K) = AAO(K) + AO(J,K) * NDIN(J,K + 1)
            BBO(K) = BBO(K) + BO(J,K) * NDIN(J,K)
     30   CONTINUE
          AAO(K) = AAO(K) / GINT(K + 1)
          BBO(K) = BBO(K) / GINT(K)
     40 CONTINUE
        IF (PRES .GE. 3 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
          WRITE (8,*)'aao bbo'
          WRITE (8,50) (K,AAO(K),BBO(K),K=1,NC)
     50   FORMAT (1X, 2('k',I2,1X,2(1PE10.3,1X)))
        END IF
        DO 70 K = 1, NCM1
          GRS(K) = 0.0
          DO 60 J = 1, NG
            GRS(K) = GRS(K) + S(J,K)
     60   CONTINUE
     70 CONTINUE
        DO 90 K = 1, NC
          DO 80 J = 1, NG
            ZETA(J,K) = (1 + ALFA(K)*ABC(J,K)*ETA(K)) * TOU * DL(K)
            FUNF(J,K) = AO(J,K) * NDIN(J,K + 1) + BO(J,K) * NDIN(J,K)
     80   CONTINUE
     90 CONTINUE
        IF (PRES .GE. 4 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
          WRITE (8,*)'s'
          WRITE (8,100) (K,GRS(K),K=1,NCM1)
    100   FORMAT (1X, 4('k',I2,1X,1PE10.3))
          WRITE (8,*)'zeta funf'
          WRITE (8,110) ((J,K,ZETA(J,K),FUNF(J,K),K=1,NC),J=1,NG)
    110   FORMAT (1X, 2('j',I2,1X,'k',I2,2(1PE10.3,1X)))
        END IF
        DO 130 K = 1, NC
```

```fortran
          FUNH(K) = 0.0
          DO 120 J = 1, NG
           FUNH(K) = FUNH(K) + ABC(J,K) * (AAO(K)*GINT(K + 1) + BBO(K)*
     1     GINT(K))
  120     CONTINUE
  130   CONTINUE
        SL = PGIN(1) * SUM1 + DL(1) * GRZQ(1) / 2 - DL(1) * GRFQ(1) + 2 *
     1  GRSI(1) / 3 + PFGI(1)
        SR = PGIN(NCP1) * SUM2 - DL(NC) * GRZQ(NC) / 2 - DL(NC) * GRFQ(NC)
     1  + 2 * GRSI(NC) / 3 + PFGI(NCP1)
        IF (LIN .EQ. 0) THEN
C**** use the linear grey model for accelerating the diffusion
C**** equation
         DO 150 K = 1, NC
          DUMF(K) = 0.0
          DO 140 J = 1, NG
           DUMF(K) = DUMF(K) + CHI(J,K) / (ABC(J,K) + TOU)
  140     CONTINUE
  150    CONTINUE
         DO 170 K = 1, NC
          DO 160 J = 1, NG
           THETA(J,K) = CHI(J,K) / ((ABC(J,K) + TOU)*DUMF(K))
           PHI(J,K) = THETA(J,K)
  160     CONTINUE
  170    CONTINUE
         DO 190 K = 1, NC
          FUNA(K) = 0.0
          SUMZ(K) = 0.0
          DO 180 J = 1, NG
           FUNA(K) = FUNA(K) + PHI(J,K) / SIGM(J,K)
           SUMZ(K) = SUMZ(K) + ZETA(J,K) * THETA(J,K)
  180     CONTINUE
  190    CONTINUE
         DO 210 K = 1, NC
          SDDI(K) = 0.0
          FUNC(K) = 0.0
          DO 200 J = 1, NG
           SDDI(K) = SDDI(K) + (NDIN(J,K + 1) - NDIN(J,K)) / SIGM(J,K)
           FUNC(K) = FUNC(K) + ZETA(J,K) * FUNF(J,K)
  200     CONTINUE
          FUND(K) = FUNA(K) * (GINT(K + 1) - GINT(K))
          FUNE(K) = SUMZ(K) * (AAO(K)*GINT(K + 1) + BBO(K)*GINT(K))
  210    CONTINUE
         DO 220 K = 1, NCP1
          IF (K .EQ. 1) THEN
           IF (DIFBC .EQ. 0) THEN
            R(K) = SUM1 * PGIN(K) + PFGI(K) + SDDI(K) / 3 - FUND(K) /
     1      3 + DL(K) * GRZQ(K) / 2 - DL(K) * GRFQ(K) + 2 * GRSI(K) /
     2      3 - FUNC(K) / 4 + FUNE(K) / 4
           ELSE IF (DIFBC .EQ. 1) THEN
            R(K) = SUM1 * (PGIN(K) - GINT(K)) + PFGI(K) + SDDI(K) / 3
     1      - FUND(K) / 3 + DL(K) * GRZQ(K) / 2 - DL(K) * GRFQ(K) + 2
     2      * GRSI(K) / 3 - FUNC(K) / 4 + FUNE(K) / 4
           ELSE IF (DIFBC .EQ. 2) THEN
            R(K) = ETA(K) * DL(K) * FUNH(K) / 4 + SL
           END IF
          ELSE IF (K .EQ. NCP1) THEN
           IF (DIFBC .EQ. 0) THEN
            R(K) = -SUM2 * PGIN(K) - PFGI(K) - SDDI(K - 1) / 3 + FUND(
     1      K - 1) / 3 - 2 * GRSI(K - 1) / 3 + DL(K - 1) * GRZQ(K - 1)
     2      / 2 + DL(K - 1) * GRFQ(K - 1) - FUNC(K - 1) / 4 + FUNE(K
     3      - 1) / 4
           ELSE IF (DIFBC .EQ. 1) THEN
            R(K) = SUM2 * (GINT(K) - PGIN(K)) - PFGI(K) - SDDI(K - 1)
     1      / 3 + FUND(K - 1) / 3 - 2 * GRSI(K - 1) / 3 + DL(K - 1) *
     2      GRZQ(K - 1) / 2 + DL(K - 1) * GRFQ(K - 1) - FUNC(K - 1) /
     3      4 + FUNE(K - 1) / 4
           ELSE IF (DIFBC .EQ. 2) THEN
            R(K) = -ETA(K - 1) * DL(K - 1) * FUNH(K - 1) / 4 + SR
           END IF
          ELSE
           R(K) = (SDDI(K) - SDDI(K - 1)) / 3 - (FUND(K) - FUND(K - 1))
     1     / 3 - (FUNC(K) + FUNC(K - 1)) / 4 + (FUNE(K) + FUNE(K - 1))
     2     / 4 + GRS(K - 1)
          END IF
  220    CONTINUE
        ELSE
C**** use the nonlinear grey model for accelerating the diffusion
C**** equation
         DO 240 K = 1, NC
          DO 230 J = 1, NG
           FUNG(J,K) = NDIN(J,K + 1) - NDIN(J,K)
  230     CONTINUE
  240    CONTINUE
         DO 260 J = 1, NG
          DO 250 K = 1, NC
           THETA(J,K) = FUNF(J,K) / (AAO(K)*GINT(K + 1) + BBO(K)*GINT(
     1     K))
           PHI(J,K) = FUNG(J,K) / (GINT(K + 1) - GINT(K))
  250     CONTINUE
  260    CONTINUE
         DO 280 K = 1, NC
          SUMZ(K) = 0.0
          DO 270 J = 1, NG
           SUMZ(K) = SUMZ(K) + ZETA(J,K) * THETA(J,K)
  270     CONTINUE
  280    CONTINUE
         DO 300 K = 1, NC
          FUNA(K) = 0.0
          DO 290 J = 1, NG
           FUNA(K) = FUNA(K) + PHI(J,K) / SIGM(J,K)
  290     CONTINUE
  300    CONTINUE
         DO 330 K = 1, NCP1
          IF (K .EQ. 1) THEN
           IF (DIFBC .EQ. 0) THEN
            R(K) = SUM1 * PGIN(K) + PFGI(K) + DL(K) * GRZQ(K) / 2 -
     1      DL(K) * GRFQ(K) + 2 * GRSI(K) / 3
           ELSE IF (DIFBC .EQ. 1) THEN
            R(K) = 0.0
            DO 310 J = 1, NG
             R(K) = R(K) + SUM1 * PDIN(J,K) + FINT(J,K) + 2 * (SINT(
     1       J,K + 1) - SINT(J,K)) / (3*SIGM(J,K)) + ZQ(J,K) * DL(K)
     2       / 2 - FQ(J,K) * DL(K) / SIGM(J,K) + ITER(J,K)
  310       CONTINUE
           ELSE IF (DIFBC .EQ. 2) THEN
            R(K) = ETA(K) * DL(K) * FUNH(K) / 4 + SL
           END IF
          ELSE IF (K .EQ. NCP1) THEN
           IF (DIFBC .EQ. 0) THEN
            R(K) = -SUM2 * PGIN(K) - PFGI(K) + DL(K - 1) * GRZQ(K - 1)
     1      / 2 + DL(K - 1) * GRFQ(K - 1) - 2 * GRSI(K - 1) / 3
           ELSE IF (DIFBC .EQ. 1) THEN
            R(K) = 0.0
            DO 320 J = 1, NG
             R(K) = R(K) - SUM2 * PDIN(J,K) - FINT(J,K) - 2 * (SINT(
     1       J,K) - SINT(J,K - 1)) / (3*SIGM(J,K - 1)) + ZQ(J,K - 1)
     2       * DL(K - 1) / 2 + FQ(J,K - 1) * DL(K - 1) / SIGM(J,K -
     3       1) + ITER(J,K - 1)
  320       CONTINUE
           ELSE IF (DIFBC .EQ. 2) THEN
            R(K) = -ETA(K - 1) * DL(K - 1) * FUNH(K - 1) / 4 + SR
           END IF
          ELSE
           R(K) = GRS(K - 1)
          END IF
  330    CONTINUE
        END IF
        IF (PRES .GE. 3 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1   THEN
         DO 350 K = 1, NC
          SUMTH(K) = 0.0
          SUMPH(K) = 0.0
          DO 340 J = 1, NG
           SUMTH(K) = SUMTH(K) + THETA(J,K)
           SUMPH(K) = SUMPH(K) + PHI(J,K)
  340     CONTINUE
  350    CONTINUE
         WRITE (8,*)'dumf'
         WRITE (8,360) (K,DUMF(K),K=1,NC)
  360    FORMAT (4(1X,'k',I2,1X,1PE10.3))
         WRITE (8,*)'theta phi'
         WRITE (8,370) ((J,K,THETA(J,K),PHI(J,K),K=1,NC),J=1,NG)
  370    FORMAT (2(1X,'j',I2,1X,'k',I2,2(1PE10.3,1X)))
         WRITE (8,*)'sumth sumph'
         WRITE (8,380) (K,SUMTH(K),SUMPH(K),K=1,NC)
  380    FORMAT (2(1X,'k',I2,1X,2(1PE10.3)))
         WRITE (8,*)'funa sumz'
         WRITE (8,390) (K,FUNA(K),SUMZ(K),K=1,NC)
  390    FORMAT (2(1X,'k',I2,1X,2(1PE10.3)))
         WRITE (8,*)'sddi func fund fune'
         WRITE (8,400) (K,SDDI(K),FUNC(K),FUND(K),FUNE(K),K=1,NC)
  400    FORMAT (1X, 'k', I2, 1X, 1PE10.3, 1X, 1PE10.3, 1X, 1PE10.3, 1X,
     1    1PE10.3)
        END IF
        DO 450 K = 1, NCP1
         IF (K .EQ. 1) THEN
          IF (DIFBC .EQ. 0) THEN
           A(K) = 0.0
           B(K) = SUM1 + FUNA(K) / 3 + SUMZ(K) * BBO(K) / 4
           C(K) = -FUNA(K) / 3 + SUMZ(K) * AAO(K) / 4
          ELSE IF (DIFBC .EQ. 1) THEN
           A(K) = 0.0
           B(K) = 0.0
           C(K) = 0.0
           DO 410 J = 1, NG
            B(K) = B(K) + (SIGM(J,K)*BO(J,K)/4 + 1/(3*SIGM(J,K))) *
     1      NDIN(J,K) / GINT(K)
            C(K) = C(K) + (SIGM(J,K)*AO(J,K)/4 - 1/(3*SIGM(J,K))) *
     2      NDIN(J,K) / GINT(K)
  410      CONTINUE
           B(K) = B(K) + SUM1
          ELSE IF (DIFBC .EQ. 2) THEN
           A(K) = 0.0
           B(K) = 0.0
           C(K) = 0.0
           DO 420 J = 1, NG
            B(K) = B(K) + (SIGM(J,K)*BO(J,K)/4 + 1/(3*SIGM(J,K))) *
     1      NDIN(J,K)
            C(K) = C(K) + (SIGM(J,K)*AO(J,K)/4 - 1/(3*SIGM(J,K))) *
     1      NDIN(J,K + 1)
  420      CONTINUE
           B(K) = B(K) / GINT(K) + SUM1
           C(K) = C(K) / GINT(K + 1)
          END IF
         ELSE IF (K .EQ. NCP1) THEN
          IF (DIFBC .EQ. 0) THEN
           A(K) = -FUNA(K - 1) / 3 + SUMZ(K - 1) * BBO(K - 1) / 4
           B(K) = FUNA(K - 1) / 3 + SUMZ(K - 1) * AAO(K - 1) / 4 - SUM2
           C(K) = 0.0
          ELSE IF (DIFBC .EQ. 1) THEN
           A(K) = 0.0
           B(K) = 0.0
           DO 430 J = 1, NG
            A(K) = A(K) + (SIGM(J,K - 1)*BO(J,K - 1)/4 - 1/(3*SIGM(J,
     1      K - 1))) * NDIN(J,K) / GINT(K)
            B(K) = B(K) + (SIGM(J,K - 1)*AO(J,K - 1)/4 + 1/(3*SIGM(J,
     1      K - 1))) * NDIN(J,K) / GINT(K)
  430      CONTINUE
           B(K) = B(K) - SUM2
           C(K) = 0.0
          ELSE IF (DIFBC .EQ. 2) THEN
```

```
            A(K) = 0 0
            B(K) = 0 0
            DO 440 J = 1, NG
             A(K) = A(K) - (SIGM(J,K - 1)*BO(J,K - 1)/4 - 1/(3*SIGM(J,
     1        K - 1))) * NDIN(J,K - 1)
             B(K) = B(K) - (SIGM(J,K - 1)*AO(J,K - 1)/4 + 1/(3*SIGM(J,
     1        K - 1))) * NDIN(J,K)
  440      CONTINUE
            A(K) = A(K) / GINT(K - 1)
            B(K) = B(K) / GINT(K) + SUM2
            C(K) = 0 0
          END IF
        ELSE
          A(K) = -FUNA(K - 1) / 3 + SUMZ(K - 1) * BBO(K - 1) / 4
          B(K) = (FUNA(K - 1) + FUNA(K)) / 3 + (SUMZ(K - 1)*AAO(K - 1) +
     1     SUMZ(K)*BBO(K)) / 4
          C(K) = -FUNA(K) / 3 + SUMZ(K) * AAO(K) / 4
        END IF
  450 CONTINUE
        IF (PRES .GE. 4 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
         IF (LIN .EQ. 0) THEN
          WRITE (8,*)'a b c r in lin grey'
         ELSE
          WRITE (8,*)'a b c r in nonlin grey'
         END IF
         WRITE (8,460) (K,A(K),B(K),C(K),R(K),K=1,N)
  460   FORMAT (1X, 'k', I2, 1X, 1PE10.3, 1X, 1PE10.3, 1X, 1PE10.3, 1X,
     1       1PE10.3)
        END IF
        N = NCP1
        CALL LU3
        DO 470 K = 1, NCP1
         NGIN(K) = SOLU(K)
  470 CONTINUE
        IF (PRES .GE. 3 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
         WRITE (8,*)'gint ngin ngin/gint'
         WRITE (8,480) (K,GINT(K),NGIN(K),(NGIN(K)/GINT(K)),K=1,NCP1)
  480   FORMAT (1X, 'k', I2, 1X, 1PE10.3, 1X, 1PE10.3, 1X, 1PE10.3)
        END IF
        IF (PRES .GE. 5 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
         WRITE (8,490) ((K,J,THETA(J,K)*(AAO(K)*NGIN(K + 1) + BBO(K)*
     1  NGIN(K)),THETA(J,K)*(AAO(K)*GINT(K + 1) + BBO(K)*GINT(K)),FUNF(
     2  J,K),J=1,NG),K=1,NC)
  490   FORMAT (2(1X,'k',I2,1X,'j',I2,1PE10.3,1X,1PE10.3,1X,1PE10.3))
        END IF
        DO 510 K = 1, NC
         AB(K) = 0.0
         DO 500 J = 1, NG
          AB(K) = AB(K) + ABC(J,K) * (THETA(J,K)*(AAO(K)*NGIN(K + 1) +
     1    BBO(K)*NGIN(K)) + FUNF(J,K) - THETA(J,K)*(AAO(K)*GINT(K + 1) +
     2    BBO(K)*GINT(K)))
  500   CONTINUE
  510 CONTINUE
        IF (PRES .GE. 3 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
         WRITE (8,*)'ab grey'
         WRITE (8,520) (K,AB(K),K=1,NC)
  520   FORMAT (4(1X,'k',I2,1X,1PE10.3))
        END IF
        DO 540 K = 1, NC
         DO 530 J = 1, NG
          ITER(J,K) = DL(K) * CHI(J,K) * ETA(K) * AB(K) / 4
  530   CONTINUE
  540 CONTINUE
        RETURN
        END
C
C     ****************************************************************
C     *  prints time of simulation on a pc with microsoft compiler  *
C     * BUT "C*" comments must be removed before time can be obtained *
C     ****************************************************************
C
        SUBROUTINE CPUTM

        PARAMETER (NANG=24,NFG=64,NCL=151)

        INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
        INTEGER DRT, DIFBC, DIFACC, IIT, TITA
        INTEGER INSB, IRUN, LIN, MEDUSA
        INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
        INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
        INTEGER TB, TEMEXP, TBIT, TOIT, TEMPTD, TIT, TRIN

        COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
        COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
        COMMON /I3/ INSB, IRUN, LIN, MEDUSA
        COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
        COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
        COMMON /I6/ TB, TEMEXP, TBIT, TOIT, TEMPTD, TIT, TRIN

        REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
        REAL*8 DIST(NCL), DTIME, DEPR
        REAL*8 TTIME, TOU, TERR, TELB

        COMMON /R06/ DIST, DTIME, DEPR
        COMMON /R15/ NATE, NINT
        COMMON /R34/ TTIME, TOU, TERR, TELB

        LOGICAL TRN, DIFF, LGR, NLGR, SIMRUN, SMTST
        COMMON /L1/ TRN, DIFF, LGR, NLGR, SIMRUN, SMTST
        REAL*8 STIME, FTIME, RUNTM
        COMMON /TMS/ STIME, FTIME, RUNTM

        WRITE (99,10) TOIT, TBIT, TIT, OIT, IIT
   10 FORMAT (1X, 'TOIT=', I8, 1X, 'TBIT=', I8, 1X, 'TIT=', I4, 1X,
     1   'OIT=', I4, 1X, 'IIT=', I3)
        WRITE (99,20) (DIST(K),NATE(K),K=1,NC)
   20 FORMAT (2(1X,1PE10.3))
C*    CALL GETTIM(IHR, IMIN, ISEC, I100SEC)
C*    FTIME = (IHR*360000 + IMIN*6000 + ISEC*100 + I100SEC)
        RUNTM = FTIME - STIME
        WRITE (99,40) TRNACC, DIFACC, LIN, DTIME
   40 FORMAT (1X, 'trnacc=', I2, 1X, 'difacc=', I2, 1X, 'lin=', I2, 1X,
     1   'dtime=', 1PE11.4)
        WRITE (99,50) RUNTM / 100
        WRITE (6,50) RUNTM / 100
   50 FORMAT (1X, 'runtime =', F12.2, ' sec')
        IF (RUNTM/100 .GT. 1000.0) THEN
         WRITE (99,60) RUNTM / 6000.0
   60  FORMAT (1X, 'runtime =', F12.2, ' min')
        ELSE IF (RUNTM/100 .GT. 10000.0) THEN
         WRITE (99,70) RUNTM / 360000.0
   70  FORMAT (1X, 'runtime =', F12.2, ' hrs')
        END IF
        END
C
C     ****************************************************************
C     * this subroutine reads the tables and evaluates all the      *
C     * variables needed for the simulation of this code with MED6UA *
C     * this subroutine is equivalent of subroutien model, thisA    *
C     * over rides any predefined variables, which are redeifne here.*
C     ****************************************************************
C
        SUBROUTINE TABLES

        PARAMETER (NANG=24,NFG=64,NCL=151)
        PARAMETER (RK=7.560667E-16,PK=6.6261E-34)
        PARAMETER (BK=1.3807E-23,SP=3.0E08,EVE=1.6022E-19)
        PARAMETER (TCP=EVE/BK,PI=3.141593,A0=PK/(BK*TCP))

        INTEGER ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
        INTEGER DRT, DIFBC, DIFACC, IIT, TITA
        INTEGER INSB, IRUN, LIN, MEDUSA
        INTEGER NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
        INTEGER OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
        INTEGER TB, TEMEXP, TBIT, TOIT, TEMPTD, TIT, TRIN
        INTEGER UBC, WFN, CONV, DBXL, DBXU, NUM
        INTEGER NCP1, NCM1, NAD2
        INTEGER SMTST1, SMTST2, SMTST3, SMTST4, SMTST5
        INTEGER MDT, NITT

        LOGICAL RTEON

        COMMON /I1/ ABVL, AUTACL, CONVC, COUNT, CONT, TCONT, DB, DCOUNT
        COMMON /I2/ DRT, DIFBC, DIFACC, IIT, TITA
        COMMON /I3/ INSB, IRUN, LIN, MEDUSA
        COMMON /I4/ NA, NG, NC, NF, NDIT, NFFC, NDICNT, NFMUST
        COMMON /I5/ OIT, OITA, PRES, RT, SETSOU, SCR, TRNACC
        COMMON /I6/ TB, TEMEXP, TBIT, TOIT, TEMPTD, TIT, TRIN
        COMMON /I7/ UBC, WFN, CONV, DBXL, DBXU, NUM
        COMMON /I8/ NCP1, NCM1, NAD2
        COMMON /SMT/ SMTST1, SMTST2, SMTST3, SMTST4, SMTST5
        COMMON /IMD/ MDT, NITT, RTEON

        REAL*8 ALFA(NCL), AC, ATMASS, DAF(NFG,NCL), ATEM(NCL)
        REAL*8 ABC(NFG,NCL), AB(NCL), AO(NFG,NCL)
        REAL*8 BETA(NCL), BO(NFG,NCL)
        REAL*8 CV(NCL), CW, CHI(NFG,NCL)
        REAL*8 DL(NCL), DINT(NFG,NCL + 1), PRNTM
        REAL*8 DIST(NCL), DTIME, DEPR
        REAL*8 ETA(NCL)
        REAL*8 FINT(NFG,NCL + 1), FINL(NFG), FINR(NFG)
        REAL*8 FR(NFG + 1), FQ(NFG,NCL)
        REAL*8 HELIM
        REAL*8 INLB(NFG), INRB(NFG), ITSL, IERED(NCL), IERR
        REAL*8 INTB(NANG,NFG,NCL + 1), IINT(NANG,NFG,NCL), IZIN(NFG,NCL)
        REAL*8 ITRB, ITER(NFG,NCL)
        REAL*8 L, LFL, LOWLIM
        REAL*8 MAXT
        REAL*8 NATE(NCL), NINT(NANG,NFG,NCL)
        REAL*8 NFFT(NANG,NFG,NCL), NDIN(NFG,NCL + 1), NZIN(NFG,NCL)
        REAL*8 PMAC(NCL), PINT(NANG,NFG,NCL), PRTIME
        REAL*8 PGIN(NCL + 1), PFGI(NCL + 1), PLANK(NFG,NCL)
        REAL*8 PDIN(NFG,NCL + 1), PITE(NFG,NCL), PZIN(NFG,NCL)
        REAL*8 PAB(NCL)
        REAL*8 Q(NANG,NFG,NCL)
        REAL*8 SPH, SIRC, SPR
        REAL*8 SIGM(NFG,NCL), GRSI(NCL + 1)
        REAL*8 SOLU(NCL + 1), SPRB
        REAL*8 S(NFG,NCL), SUM1, SUM2, GRPQ(NCL), GRZQ(NCL)
        REAL*8 SINT(NFG,NCL + 1)
        REAL*8 TTIME, TOU, TERR, TELB
        REAL*8 TRNFVR(NCL), TEMP(NCL)
        REAL*8 UFL
        REAL*8 WF(NANG)
        REAL*8 ZLP(NANG), ZINT(NFG,NCL), ZQ(NFG,NCL), DIER

        COMMON /R01/ ALFA, AC, ATMASS, DAF, ATEM
        COMMON /R02/ ABC, AB, AO
        COMMON /R03/ BETA, BO
        COMMON /R04/ CV, CW, CHI
        COMMON /R05/ DL, DINT, PRNTM
        COMMON /R06/ DIST, DTIME, DEPR
        COMMON /R07/ ETA
        COMMON /R071/ FINT, FINL, FINR
        COMMON /R08/ FR, FQ
        COMMON /R09/ HELIM
        COMMON /R10/ INLB, INRB, ITSL, IERED, IERR
        COMMON /R11/ INTB, IINT, IZIN
```

```
      COMMON /R12/ ITRB, ITER
      COMMON /R13/ L, LFL, LOWLIM
      COMMON /R14/ MAXT
      COMMON /R15/ NATE, NINT
      COMMON /R16/ NFFT, NDIN, NZIN
      COMMON /R17/ PMAC, PINT, PRTIME
      COMMON /R18/ POIN, PFOI, PLANK
      COMMON /R181/ PDIN, PITE, PZIN
      COMMON /R182/ PAB
      COMMON /R19/ Q
      COMMON /R20/ SPH, SIRC, SPR
      COMMON /R21/ SIGM, GRSI
      COMMON /R22/ SOLU, SPRB
      COMMON /R23/ S, SUM1, SUM2, GRFQ, GRZQ
      COMMON /R231/ SINT
      COMMON /R24/ TTIME, TOU, TERR, TELB
      COMMON /R25/ TRNFVR, TEMP
      COMMON /R26/ UFL
      COMMON /R27/ WF
      COMMON /R28/ ZLP, ZINT, ZQ, DIER

      INTEGER IFL, NTR
      EXTERNAL FPLNK, F2, F3

      INTEGER NRN, NCODE, NCOUNT, PRNT
      COMMON /RDC/ NRN, NCODE, NCOUNT, PRNT

      REAL*8 ABB(NCL)
      REAL*8 DD, DMIN
      REAL*8 EE, ENERGL(NFG), ENERGR(NFG)
      REAL*8 INIBR(NCL)
      REAL*8 JUNK(NFG,NCL), JUNK2(NFG,NCL), SUMJNK(NCL)
      REAL*8 PV, POVER(NFG,NCL), PLNKN(NFG,NCL)
      REAL*8 RESPLN
      REAL*8 SPLAN(NCL), SUMCHI(NCL)
      REAL*8 TMIN
      REAL*8 XA, XSA(NFG,NCL), XAA
      REAL*8 ZFAC, ZFAC1, ZTMAX
      REAL*8 NI0(151)
      REAL*8 RSPH(150)
      COMMON /RCV/ RSPH
      IMPLICIT REAL*8(A - H,O - Z)
C****  begining of medusa common blocks
C
CL        C2.1  HYDRODYNAMICS
C  VERSION 1A     1/8/73    JPC    CULHAM
      COMMON /COMHYD/ DM, P3, PINI, R1, R3, R5, RHO1, RHO3, RHO5,
     1   RHOINI, RHOR, RINI, TIME, U2, U4, UEDGE, V1, V3, V5
      DIMENSION DM(151), P3(151), R1(151), R3(151), R5(151), RHO1(151),
     1      RHO3(151), RHO5(151), U2(151), U4(151), V1(151),
     2      V3(151), V5(151)
C
CL        C2.2  THERMODYNAMICS
C  VERSION 1A   1/8/73     JPC      CULHAM
      COMMON /COMTH/ DDROE1, DDROE3, DDROI1, DDROI3, DDTE1, DDTE3,
     1   DDTI1, DDTI3, GAMMAE, GAMMAI, KAPPAE, KAPPAI, PE1, PE3,
     2   PI1, PI3, TE1, TE3, TEINI, TI1, TI3, TIINI
      REAL*8 KAPPAE, KAPPAI
      DIMENSION DDROE1(151), DDROE3(151), DDROI1(151), DDROI3(151),
     1      DDTE1(151), DDTE3(151), DDTI1(151), DDTI3(151),
     2      KAPPAE(151), KAPPAI(151), PE1(151), PE3(151), PI1(151),
     3      '13(151), TE1(151), TE3(151), TI1(151), TI3(151)
C
CL        C2.3  IONS AND ELECTRONS
C  VERSION 1A    1/8/73    JPC     CULHAM
      REAL*8 LC, MIEFF, NE, NI
      COMMON /COMIE/ BREMS1, BREMS3, DEGEN, DEGMAX, DEGMIN, EFFZ,
     1   EIXCH2, LC, MIEFF, NI, NE, OMEGA1, FZ1, FZ3, FZSQ1, FZSQ3,
     2   DDZ1, DDZ3, FIDASH, PMASS
      DIMENSION BREMS1(151), BREMS3(151), DEGEN(151), EFFZ(151),
     1      EIXCH2(151), LC(151), MIEFF(151), NE(151), NI(151),
     2      FZ1(151), FZ3(151), FZSQ1(151), FZSQ3(151), DDZ1(151),
     3      DDZ3(151), FIDASH(151), OMEGA1(151)
C
CL        C2.4  LASER VARIABLES
C  VERSION 1A    1/8/73    JPC     CULHAM
C   REAL*8     LAMDA1,  LASER1, NECRI1
      COMMON /COMLAS/ ALPHA1, ELAS1, LAMDA1, LASER1, PLAS1, RABS1,
     1   ROCRI1, XL1, XL3, NECRI1, FRACLS, NABS1
      DIMENSION ALPHA1(151), LASER1(100), XL1(151), XL3(151)
C
CL        C2.5  THERMONUCLEAR REACTIONS
C  VERSION 1A    1/8/73    JPC      CULHAM
      COMMON /COMFUS/ DEUTER, F1D, F3D, F1H, F3H, F1HE3, F3HE3, F1HE4,
     1   F3HE4, F1NEU, F3NEU, F1NTRL, F3NTRL, F1T, F3T, F1X, F3X,
     2   HELIU3, HELIU4, HYDROG, NETRAL, NTRLMS, PNEUT1, PNEUT3,
     3   R1DD, R3DD, R1DHE3, R3DHE3, R1DT, R3DT, RNEUT1, RNEUT3,
     4   TINUCL, TOTNEU, TRITIU, XMASS, XTRA, XZ, YE1, YE3, YI1,
     5   YI3, YIELD, XTRA1, XTRA2, XTRA3, XZ1, XZ2, XZ3, XMASS1,
     6   XMASS2, XMASS3, XTRA4, XZ4, XMASS4, F1X1, F1X2, F1X3, F3X1,
     7   F3X2, F3X3, F1X4, F3X4
      REAL*8 NETRAL, NTRLMS
      DIMENSION F1D(151), F3D(151), F1H(151), F3H(151), F1HE3(151),
     1      F3HE3(151), F1HE4(151), F3HE4(151), F1NEU(151),
     2      F3NEU(151), F1NTRL(151), F3NTRL(151), F1T(151),
     3      F3T(151), F1X(151), F3X(151), F1X1(151), F1X2(151),
     4      F1X3(151), F1X4(151), F3X1(151), F3X2(151), F3X3(151),
     5      F3X4(151), R1DD(151), R3DD(151), R1DHE3(151),
     6      R3DHE3(151), R1DT(151), R3DT(151), YE1(151), YE3(151),
     7      YI1(151), YI3(151)
C
CL        C2.8 NON-LTE DATA FOR AVRG-Z,RADIATIVE COOLING ETC...
C  VERSION JUNE 1986 Y.T. LEE
      COMMON /LEEXRY/ HNUEDG(64), DTEMP(19), DDENS(22), ENION(19,22),
     1   TBLZ(19,22), TBLZSQ(19,22), POWER(19,22,64),
     2   SIGMA(19,22,64), HNUMIN, HNUMAX, ZATOM, NT, NR, NGHIGH,
     3   NGLOW, NGRMAX, NTMAX
C
C------------------------------------------
CL  C0.0. AUXILIARY X-RAY TRANSPORT ARRAYS FOR USE WITH LEE'S TABLES
C
      REAL*8 NIB
      COMMON /LEEXR2/ RADTL, RADTR, PPL1, PPR1, DT2B, PRDLL1(64),
     1   PRDLR1(64), PRADLL(64), PRADLR(64), RADLL(64), RADLR(64),
     2   TE1B(151), NIB(151), REDGEB
C------------------------------------------------
CL       C3.2    MESH AND NUMERICAL METHODS
C  VERSION 1A   1/8/73      JPC          CULHAM
      COMMON /COMNUM/ AE, AI, BE, BI, CE, CI, DE, DI, E, F, GE, GI, Q2,
     1   Q4, TEITE, TIITE, UITE, MESH, NJ, NJM1, NJP1, NL, NLM1,
     2   NLP1
      DIMENSION AE(151), AI(151), BE(151), BI(151), CE(151), CI(151),
     1       DE(151), DI(151), E(151), F(151), GE(151), GI(151),
     2       Q2(151), Q4(151), TEITE(151), TIITE(151), UITE(151)
C
CL        C2.9 RADIATION REABSORPTION
C  VERSION JUNE 1984, D. SALZMANN
      COMMON /COMABS/ YDNS(151), DO(20,15)
C
CL        C2.7  PHYSICS  CONTROL
C  VERSION 1A   1/8/73      JPC        CULHAM
      COMMON /COMCON/ PMIN, RHOMIN, TEMIN, TIMIN, UMIN, MSTEP, NCASE,
     1   NGEOM, MLBRMS, MLECON, MLFUSE, MLICON, MLX, NLABS, NLBRMS,
     2   NLBURN, NLCRI1, NLDEPO, NLECON, NLFUSE, NLICON, NLMOVE,
     3   NLPFE, NLPFI, NLX
      LOGICAL MLBRMS, MLECON, MLFUSE, MLICON, MLX, NLABS, NLBRMS,
     1    NLBURN, NLCRI1, NLDEPO, NLECON, NLFUSE, NLICON, NLMOVE,
     2    NLPFE, NLPFI, NLX
C
CL        C3.1  NUMERICAL CONTROL PARAMETERS
C  VERSION 1A   1/8/73     JPC        CULHAM
      COMMON /COMNC/ AK0, AK1, AK2, AK3, AK4, AK5, BNEUM, DELTAT, DT2,
     1   DT3, DT4, DTEMAX, DTIMAX, DTMAX, DTMIN, DUMAX, RDT2, RDT3,
     2   RDT4, NCELDT, NCONDT, NIT, NITMAX, BREAK, NLGOON, NLITE
      LOGICAL BREAK, NLGOON, NLITE
C
CL        C4.1   ADMINISTRATIVE VARIABLES
C  VERSION 1A   1/8/73     JPC      CULHAM
      COMMON /COMADM/ PIQ, TSTOP, MAXDIM, MAXRUN, NDUMP, NREP, NLDUMP,
     1   NLEMP
      LOGICAL NLDUMP, NLEMP
      DIMENSION PIQ(100)
C
      COMMON /COMUTI/ ZWMAR1(151), ZWMAR2(151), ZWMAR3(151),
     1   ZWMAR4(151), ZWMAR5(151), ZWMAR6(151), ZWMAR7(151),
     2   ZWMAR8(151), ZWMAR9(151), ZWMA10(151), ZWMA11(151),
     3   ZWMA12(151), ZWMA13(151), ZWMA14(151), ZWMA15(151),
     4   ZWMA16(151)
C------------------------------------------
C   COMMON/SHELL/RSHELL(11),NSHELL(11),QSHELL(11),NSMAX
      COMMON /SHELL/ RSHELL, QSHELL, NSHELL, NSMAX
      DIMENSION RSHELL(11), QSHELL(11), NSHELL(11)
      EQUIVALENCE (PIQ(63),ZPLAS), (PIQ(62),ROPLAS), (PIQ(61),DRPLAS),
     1       (PIQ(59),STATE), (PIQ(58),SAHA), (PIQ(56),PONDF),
     2       (PIQ(57),RHOT), (PIQ(37),GAUSS), (PIQ(52),ANABS),
     3       (PIQ(22),FTHOT), (PIQ(21),FHOT), (PIQ(14),FNE),
     4       (PIQ(13),ZGLAS), (PIQ(12),ROGLAS), (PIQ(11),DRGLAS),
     5       (PIQ(10),FLIMIT), (RHOINI,RHOGAS), (LASER1(1),TON),
     6       (LASER1(2),PMULT), (LASER1(3),ANPULS), (LASER1(4),
     7       PLENTH), (LASER1(5),TOFF), (LASER1(6),PMAX),
     8       (PIQ(60),SIMULT), (PIQ(64),OUTAMI), (PIQ(66),DTPRNT)
      EQUIVALENCE (PIQ(67),NLEDO)
C------------------------------------------
CL          C1.1.   BASIC SYSTEM PARAMETERS
C   VERSION 2B       14.8.73    KVR/MHH       CULHAM
C------------------------------------------
CL          C1.1.   BASIC SYSTEM PARAMETERS
C   VERSION 2B       14.8.73    KVR/MHH       CULHAM
      COMMON /COMBAS/ ALTIME, CPTIME, NLEDGE, NLEND, NLRES, NONLIN,
     1   NOUT, NPRINT, NREAD, NREC, NRESUM, NSTEP, STIME, LABEL1,
     2   LABEL2, LABEL3, LABEL4, LABEL5, LABEL6, LABEL7, LABEL8,
     3   NDIARY, NIN, NPUNCH, NRUN
      LOGICAL NLEND, NLRES
      DIMENSION LABEL1(12), LABEL2(12), LABEL3(12), LABEL4(12),
     1      LABEL5(12), LABEL6(12), LABEL7(12), LABEL8(12)

C****   end of medusa common blocks
      IF (MDT .EQ. 0) THEN
C****  initialization of the variables
C**** mdt is set to 1 once this subroutine returns to main subroutine
       DATA NTR /0/, AN /6.022E24/, EPS, EPS1, EPS2 /1.E-9, 1.E-3, 1.E-
     1    1/
       IF (NTR .EQ. 0) THEN
        NTR = 1
        NC = NL
        NCP1 = NC + 1
        NCM1 = NC - 1
        NG = NGHIGH - NGLOW
        TMIN = 0.5 * (10.0**DTEMP(1))
        DMIN = 10.0 ** DDENS(1)
        IZP1 = ZATOM + 1.
        IF (TIME .EQ. 0.0) THEN
         DT2B = DT2
         DO 10 K = 1, NC
          NIB(K) = NI0(K)
          TE1B(K) = TE1(K)
   10     CONTINUE
         REDGEB = R1(NCP1)
         RADTL = 0.0
         RADTR = 0.0
         PPL1 = 0.0
         PPR1 = 0.0
```

```
            DO 20 J = 1, NG + 1
             PRADLL(J) = 0.0
             PRADLR(J) = 0.0
             PRDLL1(J) = 0.0
             PRDLR1(J) = 0.0
             RADLL(J) = 0.0
             RADLR(J) = 0.0
   20       CONTINUE
          END IF
        END IF
        DTIME = DT2
        ATMASS = XMASS
        TOU = 1 / (SP*DTIME)
        NITT = NIT
        IF (NIT .NE. 1) RETURN
C**** extrapolate te3 using the slope of the previous step
        DO 30 K = 1, NC
         Z1 = (TE1(K) - TE1B(K)) * DT2 / DT2B
         IF (ABS(Z1) .GT. 0.5*TE3(K)) Z1 = 0.0
         TE3(K) = TE1(K) + Z1
         NI0(K) = NI(K)
         Z1 = (NI(K) - NIB(K)) * DT2 / DT2B
         IF (ABS(Z1) .GT. 0.5*NI(K)) Z1 = 0.0
         NI(K) = NI(K) + Z1
   30   CONTINUE
        Z1 = (R1(NCP1) - REDOEB) * DT2 / DT2B
        IF (ABS(Z1) .GT. 0.5*R1(NCP1)) Z1 = 0.0
        REDOEB = R1(NCP1) + Z1
        DO 40 K = 1, NC
         BREMS3(K) = 0.0
   40   CONTINUE
        DO 50 J = 1, NG + 1
         PRADLL(J) = 0.0
         PRADLR(J) = 0.0
   50   CONTINUE
        ZTMAX = TE3(NABS1)
        DO 60 K = NABS1, NC
         ZTMAX = DMAX1(ZTMAX,TE3(K))
   60   CONTINUE
        IF (ZTMAX .LE. TMIN) THEN
C**** skip radiation code
         RTEON = .FALSE.
         WRITE (6,70) NCOUNT, TIT, TIME
   70    FORMAT ('radiation code skipped due to low temp in rts.f',
     1         2(1X,I4), 1X, 1PE11.4)
         GO TO 670
        ELSE
         RTEON = .TRUE.
        END IF
        DO 80 K = 1, NC
         ATEM(K) = TE3(K)
C**** rsph is obtained from EOS subroutine in MEDUSA
         CV(K) = RSPH(K) * RHO3(K) / 10.0
         DIST(K) = R1(K)
         DL(K) = ABS(R1(K + 1) - R1(K))
   80   CONTINUE
C**** define the frequency groups
        DO 90 JJ = NGLOW, NGHIGH
         J = JJ - NGLOW + 1
         FR(J) = HNUEDG(JJ)
   90   CONTINUE
C**** find the power and x-sectional area at given temperature
C**** and density by interpolation. (power=W/m**2, x-area=m**2 and
C**** abs coef=1/m)
        DO 120 K = 1, NC
         IF (TE3(K) .GE. TMIN) THEN
          DO 100 JJ = NGLOW, NGHIGH
           J = JJ - NGLOW + 1
           XAA = CIGMA(DLOG10(0.5*(HNUEDG(J) + HNUEDG(J + 1))),ZATOM)
     1      * ATMASS / AN
C**** zfac1 = 0.0 if te3 < tmin, otherwise zfac1 = 1.0
           ZFAC1 = DMAX1(0.0D0,DMIN1(1.0D0,(TE3(K) - TMIN)/TMIN))
C**** zfac = 0.0 if te3 < tmin, otherwise zfac =< 1.0
           ZFAC = ZFAC1 * DMIN1(1.0D0,NI(K)/DMIN)
           CALL DINTRP(DLOG10(TE3(K)), DLOG10(NI(K)), SIGMA(1,1,JJ),
     1        XA, IFL)
           XSA(J,K) = (1.0 - ZFAC1) * XAA + ZFAC1 * 10.0 ** XA
           XSA(J,K) = 10.0 ** XA
           ABC(J,K) = NI(K) * XSA(J,K)
CC          ABC(J,K) = (RHO3(K)*AN*XSA(J,K)) / ATMASS
           CALL INTRP(POWER(1,1,JJ), PV)
           PLANK(J,K) = ZFAC * 10.0 ** PV * NI(K) * DL(K) / (4*PI)
CC          PLANK(J,K) = ZFAC*10.0**PV/(4*PI*XSA(J,K))
  100     CONTINUE
C**** rtables
         ELSE
          DO 110 JJ = NGLOW, NGHIGH
           J = JJ - NGLOW + 1
           XAA = CIGMA(DLOG10(0.5*(HNUEDG(J) + HNUEDG(J + 1))),ZATOM)
     1      * ATMASS / AN
           ZFAC1 = DMAX1(0.0D0,DMIN1(1.0D0,(TE3(K) - TMIN)/TMIN))
           ZFAC = ZFAC1 * DMIN1(1.0D0,NI(K)/10.0**DDENS(1))
           CALL DINTRP(DLOG10(TE3(K)), DLOG10(NI(K)), SIGMA(1,1,JJ),
     1        XA, IFL)
           XSA(J,K) = (1.0 - ZFAC1) * XAA + ZFAC1 * 10.0 ** XA
C           XSA(J,K) = XAA
           ABC(J,K) = NI(K) * XSA(J,K)
CC          ABC(J,K) = (RHO3(K)*AN*XSA(J,K)) / ATMASS
           DD = HNUEDG(JJ) * 11604.0 / TE3(K)
           EE = HNUEDG(JJ + 1) * 11604.0 / TE3(K)
           IF (DD .GT. LOWLIM) THEN
            DD = LOWLIM
           END IF
           IF (EE .GT. HILIM) THEN
            EE = HILIM
           END IF
           CALL ROMB(FPLNK, DD, EE, RESPLN)
           PLANK(J,K) = (2*PK/SP**2) * ((BK*TE3(K)/PK)**4) * RESPLN
  110     CONTINUE
         END IF
  120   CONTINUE
        DO 150 K = 1, NC
         AB(K) = 0.0
         DO 140 J = 1, NG
          ZINT(J,K) = 0.0
          DO 130 I = 1, NA
           PINT(I,J,K) = PLANK(J,K)
           IINT(I,J,K) = PLANK(J,K)
           ZINT(J,K) = ZINT(J,K) + IINT(I,J,K) * WF(I)
  130     CONTINUE
          IZIN(J,K) = ZINT(J,K)
          AB(K) = AB(K) + ABC(J,K) * ZINT(J,K)
  140    CONTINUE
  150   CONTINUE
        IF (PRES .GE. 5 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
         WRITE (8,160) (K,DL(K),CV(K),ATEM(K)/11604,IERED(K),K=1,NC)
  160    FORMAT (2('k',I2,1X,'dl=',1PE10.3,1X,'cv=',1PE10.3,1X,'temp=',
     1       1PE10.3,1X,'Ue=',1PE10.3,1X))
         WRITE (8,*)'freq group limits in tables ', DTIME
         WRITE (8,170) (J,FR(J),J=1,NG + 1)
  170    FORMAT (8('g',I2,1X,F11.4,1X))
         WRITE (8,*)'i j k iint in tables'
         WRITE (8,180) (((I,J,K,IINT(I,J,K),K=1,NC),J=1,NG),I=1,NA)
  180    FORMAT (6('i',I2,1X,'j',I2,1X,'k',I2,1X,1PE10.3,1X))
         WRITE (8,*)'j k zint in tables '
         WRITE (8,190) ((J,K,ZINT(J,K),K=1,NC),J=1,NG)
  190    FORMAT (8(I2,1X,I2,1X,1PE10.3,1X))
        END IF
        DO 210 K = 1, NC
         ABB(K) = 0.0
         SPLAN(K) = 0.0
         DO 200 J = 1, NG
          ABB(K) = ABB(K) + ABC(J,K) * PLANK(J,K)
          SPLAN(K) = SPLAN(K) + PLANK(J,K)
  200    CONTINUE
  210   CONTINUE
        DO 230 K = 1, NC
         SUMCHI(K) = 0.0
         DO 220 J = 1, NG
          CHI(J,K) = ABC(J,K) * PLANK(J,K) / ABB(K)
          SUMCHI(K) = SUMCHI(K) + CHI(J,K)
          IERED(K) = 4 * PI * SPLAN(K) / SP
  220    CONTINUE
  230   CONTINUE
        IF (PRES .GE. 1) THEN
         WRITE (8,240) TE3(1), DDTE3(1), DDTI3(1), RHO3(1), CV(1),
     1   RSPH(1), DM(1), DL(1), ATMASS, V3(1)
  240    FORMAT ('t3 dte dti ro cv sp dm dl atm v= ', 10(1X,1PE10.3))
        END IF
        IF (PRES .GE. 21) THEN
         WRITE (8,*)'sumchi in tables '
         WRITE (8,250) (K,SUMCHI(K),K=1,NC)
  250    FORMAT (8(I2,1X,1PE11.4,1X))
        END IF
        DO 260 K = 1, NC
         BETA(K) = 4 * RK * (TE3(K)**3) / CV(K)
         PMAC(K) = ABB(K) / SPLAN(K)
         ALFA(K) = 1 / (PMAC(K)*BETA(K))
         TRNFVR(K) = SP * DTIME / DL(K)
         ETA(K) = 1 / (1 + ALFA(K)*TOU)
  260   CONTINUE
        SIRC = 0.0
        DO 270 K = 1, NC
         SIRC = SIRC + ETA(K)
  270   CONTINUE
        SIRC = SIRC / NC
        DO 300 J = 1, NG
         DO 290 K = 1, NC
          FQ(J,K) = 0.0
          DO 280 I = 1, NA
           Q(I,J,K) = CHI(J,K) * IERED(K) * ETA(K) / (2*DTIME*BETA(K)
     1     ) + TOU * IINT(I,J,K)
           FQ(J,K) = FQ(J,K) + IINT(I,J,K) * WF(I) * ZLP(I)
  280     CONTINUE
          SIGM(J,K) = (ABC(J,K) + TOU) * DL(K)
          ZQ(J,K) = CHI(J,K) * IERED(K) * ETA(K) / (DTIME*BETA(K)) +
     1    TOU * IZIN(J,K)
  290    CONTINUE
  300   CONTINUE
        IF (PRES .GE. 5 .AND. TOIT .GE. DBXL .AND. TOIT .LE. DBXU)
     1      THEN
         WRITE (8,*)'sum of chis and abbchi in tables '
         WRITE (8,310) (K,SUMCHI(K),SUMINK(K),K=1,NC)
  310    FORMAT (5(I2,1X,2(1PE11.4,1X)))
         WRITE (8,*)'j k chi abc plank in tables '
         WRITE (8,320) ((J,K,CHI(J,K),ABC(J,K),PLANK(J,K),J=1,NG),K=1,
     1   NC)
  320    FORMAT (3(I2,1X,I2,1X,3(1PE10.3,1X)))
         WRITE (8,*)' i j k q in tables '
         WRITE (8,330) (((I,J,K,Q(I,J,K),K=1,NC),J=1,NG),I=1,NA)
  330    FORMAT (6(I2,1X,I2,1X,I2,1X,1PE10.3,1X))
         PAUSE
        END IF
        DO 350 K = 1, NC
         GRZQ(K) = 0.0
         GRFQ(K) = 0.0
         DO 340 J = 1, NG
          GRZQ(K) = GRZQ(K) + ZQ(J,K)
          GRFQ(K) = GRFQ(K) + FQ(J,K) / SIGM(J,K)
  340    CONTINUE
  350   CONTINUE
```

```
        DO 360 J = 1, NG
          INLB(J) = 0.0
          INRB(J) = 0.0
    360 CONTINUE
        IF (NFFC .EQ. 0) THEN
          DO 370 K = 1, NC
            TRNFVR(K) = 0.5
    370   CONTINUE
          TRNFV = (SP*DTIME) / DL(1)
        ELSE IF (NFFC .EQ. 1) THEN
          DO 380 K = 1, NC
            TRNFVR(K) = 2.0
    380   CONTINUE
          TRNFV = (3P*DTIME) / DL(1)
        ELSE IF (NFFC .EQ. 2) THEN
          DO 390 K = 1, NC
            TRNFVR(K) = (SP*DTIME) / DL(K)
    390   CONTINUE
        END IF
        DO 410 K = 1, NC
          INIBR(K) = 0.0
          DO 400 J = 1, NG
            INIBR(K) = INIBR(K) + 4 * PI * PLANK(J,K) / DM(K)
    400   CONTINUE
    410 CONTINUE
        RETURN
       END IF
       IF (PRES .GE. 4) THEN
        WRITE (8,*)'i j k iint nint nint/iint in table'
        WRITE (8,420) (((I,J,K,IINT(I,J,K),NINT(I,J,K),NINT(I,J,K)/IINT(
     1  I,J,K),I=1,NA),J=1,NG),K=NC - 5,NC)
    420 FORMAT (3('i',I2,1X,'j',I2,1X,'k',I2,1X,3(1PE10.3,1X)))
       END IF
C**** brems - power emitted per unit mass  watts/Kg
       DO 450 K = 1, NCP1
        DO 440 J = 1, NG
          DINT(J,K) = 0.0
          DO 430 I = 1, NA
            DINT(J,K) = DINT(J,K) + INTB(I,J,K) * WF(I) * ZLP(I)
    430   CONTINUE
    440 CONTINUE
    450 CONTINUE
       DO 480 K = 1, NC
        BREMS3(K) = 0.0
        DO 470 J = 1, NG
          DO 460 I = 1, NA
            BREMS3(K) = BREMS3(K) + (2*PI/DM(K)) * (INTB(I,J,K + 1) -
     1      INTB(I,J,K)) * WF(I)
    460   CONTINUE
    470 CONTINUE
    480 CONTINUE
C**** flux emitted from right end of the plasma watts/m^2
       DO 510 J = 1, NG
        PRADLR(J) = 0.0
        DO 500 K = NC, NC
          DO 490 I = 1, NAD2
            PRADLR(J) = PRADLR(J) + (2*PI) * INTB(I,J,K + 1) * WF(I) *
     1      ZLP(I)
    490   CONTINUE
    500 CONTINUE
        FLUXR = FLUXR + PRADLR(J)
    510 CONTINUE
C**** flux emitted from left end of the plasma watts/m^2
       DO 540 J = 1, NG
        PRADLL(J) = 0.0
        DO 530 K = 1, 1
          DO 520 I = NAD2, NA
            PRADLL(J) = PRADLL(J) + (2*PI) * INTB(I,J,K) * WF(I) * ABS(
     1      ZLP(I))
    520   CONTINUE
    530 CONTINUE
        FLUXL = FLUXL + PRADLL(J)
    540 CONTINUE
       DUMTME = DUMTME + TIME
       IF (DUMTME .GE. DTPRNT/2.5) THEN
        DUMTME = 0.0
        WRITE (7,550) TIME, FLUXL, FLUXR
    550 FORMAT ('fluxl & fluxr in rte.f = ', 3(1PE11.4,1X))
        DO 580 J = 1, NG
          ENERGL(J) = 0.0
          ENERGR(J) = 0.0
          DO 570 I = 1, NA
            DO 560 K = NABS1, NABS1
              ENERGL(J) = ENERGL(J) + NINT(I,J,K - 2)
              ENERGR(J) = ENERGR(J) + NINT(I,J,K + 2)
    560     CONTINUE
    570   CONTINUE
    580 CONTINUE
        WRITE (11,590) (HNUEDG(J),ENERGL(J),ENERGR(J),J=1,NG)
    590 FORMAT (3(1PE12.5,1X))
       END IF
       IF (PRES .GE. 2) THEN
        CALL TEMPER
        WRITE (6,*)'after solving rte te3 te1 nate brems inibr in rte.f'
        WRITE (6,600) (K,TE1(K),TE3(K),NATE(K),BREMS3(K),INIBR(K),K=1,
     1  NC)
    600 FORMAT (2(I2,1X,5(1PE10.3,1X)))
        IF (PRES .GE. 2) THEN
          DO 620 K = 1, NC
            DO 610 JJ = NGLOW, NGHIGH
              J = JJ - NGLOW + 1
              DD = HNUEDG(JJ) * 11604.0 / NATE(K)
              EE = HNUEDG(JJ + 1) * 11604.0 / NATE(K)
              IF (DD .GT. LOWLIM) THEN
                DD = LOWLIM
              END IF
              IF (EE .GT. HILIM) THEN
                EE = HILIM
              END IF
              CALL ROMB(FPLNK, DD, EE, RESPLN)
C             WRITE (6,*)'rval ', RESPLN
              PLNKN(J,K) = (2/(SP**2)) * (NATE(K)**4) * (BK*TCF) *
     1        RESPLN / (A0**3)
    610     CONTINUE
    620   CONTINUE
          WRITE (8,*)'plankian before after solving rte'
          WRITE (8,630) ((J,K,IZIN(J,K),PLNKN(J,K),K=1,NC),J=1,NG)
    630   FORMAT (4(2(I2,1X),2(1PE10.3,1X)))
        END IF
        PAUSE
       END IF
       NPR = NPR + 1
       IF (NPR .GE. 100) THEN
        WRITE (6,640) TIME, DTIME, TRNACC, DIFACC, TOIT, TIIT, TIT, SPR
    640 FORMAT (1X, 't dt tr df tot tit tit spr=', 2(1PE10.3,1X),
     1        2(I1,1X), 2(I7,1X), I6, 1X, 1PE9.2)
        NPR = 0
       END IF
    650 CONTINUE
C**** return original values
       DT2B = DT2
       DO 660 K = 1, NC
        NIB(K) = NI0(K)
        NI(K) = NI0(K)
        TE1B(K) = TE1(K)
        TE3(K) = TE1(K)
    660 CONTINUE
       REDGEB = R1(NCP1)
       RETURN
       END
```