The Application of Parallel Processing

to

Radiotherapy Dose Computation

A thesis

submitted in fulfilment

of the requirements for the Degree

of

Doctor of Philosophy

at the

University of Waikato

by

DAVID CAMPBELL MURRAY

University of Waikato

1991

Abstract

Delivery of the correct dose to patients undergoing high-energy x-ray radiotherapy treatment for cancer requires accurate simulation of photon beams. Current algorithms using equivalent depth or tissue-air ratio (TAR) correction techniques fail to accurately model electron transport, and hence show inaccuracy in inhomogeneous regions of the body (especially at high photon energies). Newer techniques such as superposition and Monte Carlo simulation do model these effects, but require a large computational effort.

Convolution in Fourier space is adequately fast using even standard microprocessor technology, but this approach lacks the flexibility offered by a density-scalable superposition kernel, available only in real space. However, in order for the real-space superposition algorithm to be clinically useful it is necessary to reduce planning times to acceptable levels. This has been achieved using a multicomputer system based on Inmos T800 transputer modules, where the superposition is calculated using a master/worker network employing synchronous message passing. The calculation is partitioned by decomposing the terma array into a number of command vectors or "grains." The master task sends a vector to each worker task as it becomes free, and the worker performs the superposition for those interation voxels in the vector, storing the result in a local copy of the dose array. Upon completion of the entire calculation, the worker dose arrays are collected and summed to form the complete dose distribution. This approach has a very small communication overhead and exhibits near-linear speedup with increasing processor number.

The superposition algorithm has been incorporated into the GRATIS Treatment Planning System developed at the University of North Carolina. This system, based on the UNIX and X standards, has been installed on a Sun SPARCstation network, with the transputer network attached to the Sbus port of one SPARCstation. The treatment planning system can access the transputers from anywhere in the network, using Sun's RPC (Remote Procedure Call) standard.

Energy deposition kernels required by the superposition algorithm have been produced using the EGS4 Monte Carlo Code System (Stanford Linear Accelerator Center). The user code `RTPEDK` has been developed to generate spherically symmetric kernels, which are then interpolated by the treatment planning system to form cartesian kernels for use in superposition. Two other user codes, `RTPCYL` and `RTPCART`, have been developed to model more general radiotherapy problems based on cylindrical and cartesian scoring geometries, respectively.

Preface

A significant proportion of cancer patients are treated using radiotherapy. In *external photon beam therapy* the patient is exposed to high-energy x-ray beams from cobalt-60 machines or linear accelerators. Usually several beams from different directions are used to maximise dose to the tumour while minimising dose to healthy tissue. Since it is impractical to monitor the patient dose directly during treatment, simulation or *treatment planning* must be used to predict dose prior to treatment.

Chapter 1 reviews the principles of radiotherapy and briefly outlines the major *calculation algorithms* used, including TAR correction methods such as the Batho power law, and equivalent tissue-air ratio (E-TAR) techniques. Although these are suitable for use at cobalt energies, they fail to accurately predict the dose in heterogeneous media at linear accelerator energies. This is due primarily to inadequate (or non-existent) modelling of the physical processes, in particular the scattering of electrons away from the photon interaction sites.

Chapter 2 discusses *Monte Carlo* modelling, which has the potential to account correctly for all particle transport, and hence provide a completely accurate assessment of dose. This approach models individual particles using random variables to determine the site and type of interaction for each particle. Although this is highly accurate in principle, many million photon histories must be modelled to give an adequate dose distribution in most cases. This requirement places severe limitations on the applicability of Monte Carlo to clinical treatment planning. An abridged version of Chapter 2 has been published in *Australasian Physical and Engineering Sciences in Medicine* **13** (**3**) 132–147 (1990), under the title "Using EGS4 Monte Carlo in medical radiation physics."

Chapter 3 discusses the *superposition/convolution* algorithm, which represents a compromise between the rigorous but computationally intensive approach of Monte Carlo and the empirical but computationally rapid approaches of currently used algorithms. Superposition/convolution is more intuitive and more physically justifiable than are currently used methods, but requires a large number of calculations (although many less than Monte Carlo). Chapter 3 briefly introduces the theory behind superposition and convolution, then discusses how these algorithms can be implemented in practice (the section on convolution has also been published in extended form in *Australasian Physical and Engineering Sciences in Medicine* **12** (**3**) 128–137 (1989), under the title "3-D superposition for radiotherapy treatment planning using fast Fourier transforms"). The final sections of Chapter 3 introduce the GRATIS treatment planning software, and describe how the superposition algorithm has been implemented under this system.

Chapter 4 discusses one approach to accelerating the superposition process, namely by employing *parallel* computer architectures. The particular parallel processing system im-

plemented at Waikato University, a multicomputer network of *Inmos T800 transputers*, is discussed in detail. Chapter 4 also discusses parallel implementation of the superposition, FFT convolution, and electron pencil-beam algorithms, and investigates the performance of a multicomputer network for these applications. The transputer network and parallel-processing algorithms have been described elsewhere in the *New Zealand Journal of Computing* **1 (2)** 30–38 (1989), under the title "Radiotherapy treatment planning using transputers," and in *Medical Physics* **18 (3)** 468–473 (1991), under the title "Superposition on a multicomputer system."

Chapter 5 reviews the transputer treatment planning system, and discusses future developments in treatment planning, in terms of both calculation algorithms and computer hardware.

D.C.M.
University of Waikato
Hamilton, New Zealand
September 1991

Acknowledgements

I wish to thank all those who helped me in the course of my work. Particular thanks must go to Drs Howell Round and Ian Graham for their guidance, support and organisational efforts, and to Peter Metcalfe for his helpful input throughout the project. I wish especially to thank my colleague and good friend Peter Hoban, whose work has been vital to the results presented in this thesis, and in the success of the project as a whole. The financial support of the Cancer Society of New Zealand (National and Waikato-Bay of Plenty Divisions), the Waikato Hospital Research Fund, the University of Waikato, and the Australasian College of Physical Scientists and Engineers in Medicine (who provided funding for travel to the EGS4 Monte Carlo Workshop at the National Physical Laboratory, U.K.) is acknowledged and appreciated. This thesis has been prepared using Donald Knuth's TEX typesetting package.

Lastly, I would like to thank my parents, Alastair and Sarah Murray and Morgan and Pauline Thomas, for their unwavering moral and financial support, and my wife Lee, who has provided me with the encouragement, motivation (and occasional distraction) which I have needed during the course of my research.

D.C.M.

Contents

Tables

Figures

Symbols and Abbreviations

Symbols

Φ	planar fluence
ϕ	fluence rate or flux density
Ψ	energy fluence
ψ	energy fluence rate or energy flux density
σ	total cross-section
μ	linear attenuation coefficient
$\frac{\mu}{\rho}$	mass attenuation coefficient
N_A	Avogadro's number
$\frac{\tau}{\rho}$	photoelectric component of $\frac{\mu}{\rho}$
$\frac{\sigma_c}{\rho}$	Compton component of $\frac{\mu}{\rho}$
$\frac{\kappa}{\rho}$	photoelectric component of $\frac{\mu}{\rho}$
S_{tot}	linear stopping power
$\frac{S_{\mathrm{tot}}}{\rho}$	mass stopping power
$\frac{S_{\mathrm{col}}}{\rho}$	collisional (ionizational) mass stopping power
$\frac{S_{\mathrm{rad}}}{\rho}$	radiative mass stopping power
$\mathrm{d}\overline{\theta^2} \,/\, (\rho\,\mathrm{d}z)$	mass angular scattering power
L_Δ	linear energy transfer or restricted linear collision stopping power
$\frac{S_\Delta}{\rho}$	restricted mass collision stopping power
$\frac{\overline{S(E)}}{\rho}$	mean stopping power
$\frac{\mu_{\mathrm{tr}}}{\rho}$	mass energy transfer coefficient
$\frac{\mu_{\mathrm{ab}}}{\rho}, \frac{\mu_{\mathrm{en}}}{\rho}$	mass energy absorption coefficient
T	total energy released by primary photons per unit mass
K	kinetic energy released in the medium
D	absorbed dose
H	dose equivalent
Q	quality factor
X	exposure
D_{max}	maximum dose
d_{max}	depth of maximum dose
ρ_e^w	electron density relative to water
$\mathcal{F}$	Fourier transform
$\mathcal{F}^{-1}$	inverse Fourier transform

λ	photon mean free path
AE	low energy threshold for delta-ray production
AP	low energy threshold for bremsstrahlung production
ECUT	charged particle low energy local deposition cut-off
PCUT	photon low energy local deposition cut-off
ESTEPE	fractional energy loss per electron step

Abbreviations

BCA	boundary crossing algorithm
Bq	becquerel
BSF	backscatter factor
CCR	conditional critical region
Ci	curie
CSDA	continuous slowing down approximation
CSP	Communicating sequential processes
CT	computerised tomography
DEC	Digital Equipment Corporation
DMA	direct memory access
DSA	dose spread array
dSAR	differential scatter-air ratio
EDK	energy deposition kernel
EGS4	"Electron Gamma Shower 4" Monte Carlo Code
E-TAR	equivalent tissue/air ratio
ETRAN	"Electron Transport" Monte Carlo Code
eV	electron-volt
FFT	fast Fourier transform
FSD	focus-to-surface distance
FTP	file transfer protocol
GMFP	gamma-ray mean free path
Gy	gray
ICRU	International Commission on Radiation Units and Measurements
ITS	"Integrated Tiger Series" Monte Carlo Code
kerma	kinetic energy released in the medium
keV	kilo-electron-volt

LCA	lateral correlation algorithm
LET	linear energy transfer or restricted linear collision stopping power
linac	linear accelerator
MeV	mega-electron-volt
MFlops	millions of floating-point instructions per second
MIMD	multiple instruction stream, multiple data stream
MIPS	millions of instructions per second
MISD	multiple instruction stream, single data stream
MV	megavolts (peak spectral energy)
OAR	off-axis ratio
PDD	percent depth dose
PDF	probability distribution function
PEGS4	preprocessor for EGS4
PET	positron emission tomography
PLC	path length correction
PRESTA	parameter reduced electron-step transport algorithm
PSF	peak scatter factor
R	roentgen
RAM	random access memory
RPC	remote procedure call
RTP	radiotherapy treatment planning
SAD	source-axis distance
SAR	scatter-air ratio
SCSI	small computer systems interface
SIMD	single instruction stream, multiple data stream
SISD	single instruction stream, single data stream
SSD	source-to-skin (or source-surface) distance
Sv	sievert
TAR	tissue-air ratio
terma	total energy released by primary photons per unit mass
TLD	thermoluminescent dosimetry/dosemeter
TMR	tissue-maximum ratio
TPR	tissue-phantom ratio
TRAM	transputer module
WFTA	Winograd Fourier transform algorithm
XDR	external data representation protocol

Chapter 1

Photon Beam Radiotherapy and Treatment Planning

1.1. Introduction

The term radiotherapy encompasses both internal and external treatment — using photons, electrons, or both. One commonly used modality is *external photon beam therapy*, which involves exposing the patient to high-energy x-ray beams from one of a variety of treatment machines (see Section 1.3.1), most commonly a linear accelerator. Usually several beams are directed at the tumour from differing directions, as illustrated in Figure 1.1. This is done in order to maximise dose to the target area while minimising dose to surrounding tissue.

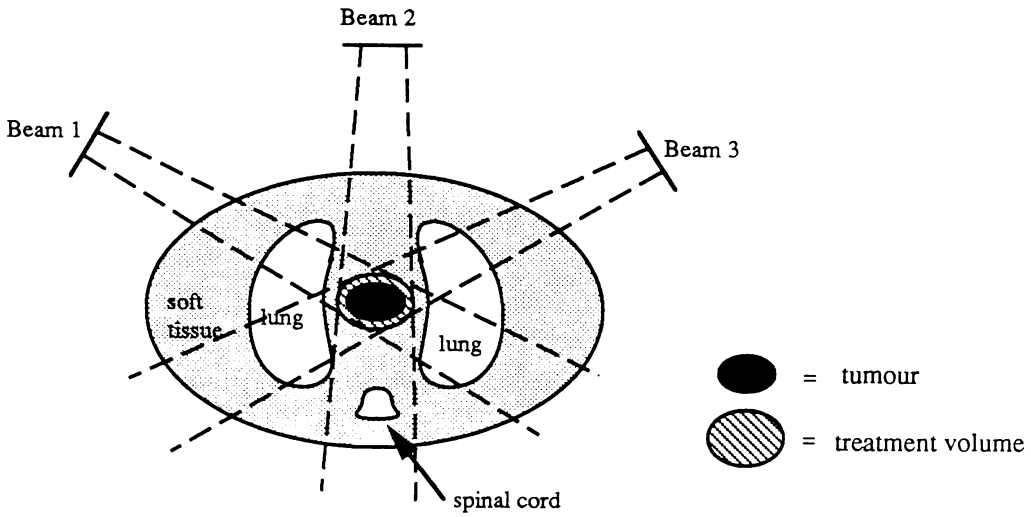

Figure 1.1. Therapy beams incident on patient lung region. Dark region indicates tumour and hatched region indicates target volume.

It is impractical to directly monitor tumour dose *in-vivo*. This restriction necessitates two important areas of radiation therapy:

(a) **Dosimetry.** Radiotherapy machines (especially linear accelerators) require constant calibration and monitoring to ensure the patient receives the intended dose. A means of quantifying the dose due to radiation produced from treatment machines must therefore be developed. This requires a knowledge of how dose is deposited in a medium (see Sections 1.2.2 and 1.2.3), a system of radiation units (see Sections 1.2.1 and 1.2.4), and a method for determining dose in air or water phantoms, or on the surface of a patient

(see Section 1.2.5). Other concepts such as tissue-air ratio and percentage depth dose have also been developed to aid in characterising photon beams of varying energy, size and shape. These will be discussed in Section 1.3.3.

(b) **Treatment Planning.** It must be possible to calculate beforehand the distribution of dose to a patient when using a particular treatment regime. In general this problem involves knowledge of the characteristics of the beams (see Sections 1.3.2 and 1.3.3), the composition of the patient, and the way in which the beams and the patient interact. The situation may be further complicated by the addition of beam modifiers, or by the use of rotating fields (see Section 1.3.4). Assuming that beam data is available and patient composition is known from a computerised tomography (CT) scan, accurate treatment planning requires a suitable *algorithm* on which to base the calculations (see Section 1.4). Many of these algorithms predict distribution of dose in homogeneous media well, but fail to accurately predict dose in heterogeneous cases. More recent algorithms such as superposition (see Section 1.4.4) and Monte Carlo modelling (see Section 1.4.5) can predict these doses more accurately.

Simulation systems are known as *treatment planning systems*, and are usually based around a mini- or micro-computer. Patient density data is usually obtained from diagnostic x-ray or CT scans and either digitised or read directly into the planning system. A simulation algorithm is used to model the distribution of dose within the patient for various beams whose placement is chosen by treatment planning personnel. Only when a satisfactory dose distribution has been achieved can treatment begin.

1.2. Radiation Physics and Dosimetry

1.2.1. Fundamental Measures of the Radiation Field

In order to quantify the effect which a radiation beam has on the patient undergoing treatment it is necessary to adopt a system of radiation units for dosimetry. The units listed in this section (except terma) are those defined and recommended by the International Commission on Radiation Units and Measurements,[1] although older units still commonly used will also be mentioned. More detailed description of these units can be found in texts by Johns and Cunningham,[2] and Greening.[3]

The following basic units are used to measure a radiation field:

(a) **Activity.** For radioactive sources such as those found in cobalt-60 machines the most

obvious measure is *activity* in *becquerels*, where

$$1 \text{ Bq} = 1 \text{ disintegration/second} \tag{1.1}$$

The old unit *curie* is also used, where 1 Ci $= 3.7 \times 10^{10}$ Bq.

(b) **Fluence.** Of more interest in external beam therapy is the quantity of ionizing radiation incident on a surface (in particular the patient or phantom surface). This is known as the **planar fluence** Φ, such that

$$\Phi = \frac{\mathrm{d}N}{\mathrm{d}a} \ (\mathrm{m}^{-2}) \tag{1.2}$$

where $\mathrm{d}N$ is the number of particles incident on a normal area $\mathrm{d}a$. Similarly, **fluence rate** or **flux density** ϕ can be defined as

$$\phi = \frac{\mathrm{d}\Phi}{\mathrm{d}t} \ (\mathrm{m}^{-2}\mathrm{s}^{-1}) \, . \tag{1.3}$$

All references to fluence and related quantities in this thesis represent *planar* fluence — as opposed to fluence defined by an elementary sphere, useful in considering radioactive sources.

(c) **Energy fluence.** Taking the energy of the incident particles into account, the *energy fluence* Ψ is defined as

$$\Psi = \frac{\mathrm{d}E_{\mathrm{fl}}}{\mathrm{d}a} \ (\mathrm{J\,m}^{-2}) \tag{1.4}$$

where $\mathrm{d}E_{\mathrm{fl}}$ is the sum of the energies of all the particles incident on $\mathrm{d}a$. **Energy fluence rate** or **energy flux density** ψ is defined similarly to the fluence rate:

$$\psi = \frac{\mathrm{d}\Psi}{\mathrm{d}t} \ (\mathrm{J\,m}^{-2}\mathrm{s}^{-1}) \, . \tag{1.5}$$

1.2.2. Interaction of Photons with Matter

The tendency of a photon to interact with matter is measured by the **total cross-section** σ of the matter, where

$$\sigma = \frac{\text{probability of interaction}}{\text{unit particle fluence}} \ (\mathrm{m}^2). \tag{1.6}$$

This quantity is commonly measured in *barns*, where 1 barn $= 10^{-28}\mathrm{m}^2$. It can also be considered as the area surrounding an interaction centre in which the photon will interact. If there are B interaction centres per unit volume exposed to a particle fluence Φ then there will be $B\sigma\Phi$ interactions per unit path length. The quantity $B\sigma$ is called the **linear**

attenuation coefficient μ (m^{-1}). It can be used to relate the fluence Φ_l after travelling a path length l to the initial fluence Φ_0, according to the relation

$$\Phi_l = \Phi_0 e^{-\mu l} \ (m^{-2}) . \tag{1.7}$$

This equation demonstrates the exponential nature of photon attenuation — which is vastly different from the way in which electrons are absorbed, as discussed in Section 1.2.3. The linear attenuation coefficient of an absorber depends on its density (which is proportional to B), so to obtain a measure characteristic of the *composition* of the absorber the **mass attenuation coefficient** $\frac{\mu}{\rho}$ is used. This quantity is the probability of interaction per unit path length *per unit density* ρ of absorber, related to the cross-section σ by the expression

$$\frac{\mu}{\rho} = \frac{N_A \sigma}{M} \ (m^2 kg^{-1}) , \tag{1.8}$$

where N_A is Avogadro's number (6.022×10^{23}) and M is the molar mass of the absorber.

There are many ways in which a photon can interact with matter. These will be mentioned briefly in the context of Monte Carlo modelling (see Chapter 2), but for radiotherapy energies only three processes need be considered:

(a) **The photoelectric effect,** in which the incident photon gives up all its energy $h\nu$ to an electron, which is ejected from the atom with corresponding kinetic energy less the binding energy of the electron's shell (see Figure 1.2a). The photoelectron escapes the atom, which is left in an excited state. An outer shell electron may then fill the inner shell vacancy, emitting *characteristic radiation*, or an *Auger* electron may carry away the extra energy. The photoelectric component of the mass attenuation coefficient, referred to as $\frac{\tau}{\rho}$, varies as Z^3 for high-Z materials (where Z is the atomic number of the material), and varies as $Z^{3.8}$ for low-Z materials. $\frac{\tau}{\rho}$ also varies as E^{-3}, where E is the photon energy. Thus the photoelectric effect is most important in high-Z materials at low photon energies.

(b) **The Compton effect** (incoherent scattering), in which a photon collides with an electron, transferring momentum to the electron as it recoils (see Figure 1.2b). The scattered photon continues on with a longer wavelength (less energy) than the incident one. The electron escapes from the atom, leaving a vacancy to be filled by the emission of characteristic radiation or Auger electrons. Since momentum is conserved, the recoil angle of the electron is uniquely related to the scattering angle of the photon. The formula of Klein and Nishina can be used to calculate the differential cross-section

per unit solid angle,[4] and hence the Compton component $\frac{\sigma_c}{\rho}$ of the mass attenuation coefficient. This value varies approximately as Z/M, and is therefore almost independent of atomic number. It slowly decreases with increasing photon energy. Compton scattering dominates at the energies used in radiotherapy.

(c) **Pair Production,** which may occur if the incident photon energy is greater than 2 electron masses, that is $2m_ec^2 = 1.022$ MeV (see Figure 1.2c). In this process the photon interacts with the field of the nucleus to produce an electron-positron pair. Any energy which the photon has in addition to the 1.022 MeV is shared between the positron and electron. The positron deposits energy in a manner similar to an electron, but when at rest or nearly at rest it quickly *annihilates* with an electron to produce two photons of energy 0.511 MeV travelling in opposite directions. The pair production component $\frac{\kappa}{\rho}$ of the mass attenuation coefficient varies linearly with Z and increases rapidly with increasing photon energy (above 1.022 MeV). Thus pair production dominates at high photon energies, since the photoelectric and Compton processes decrease with increasing energy.

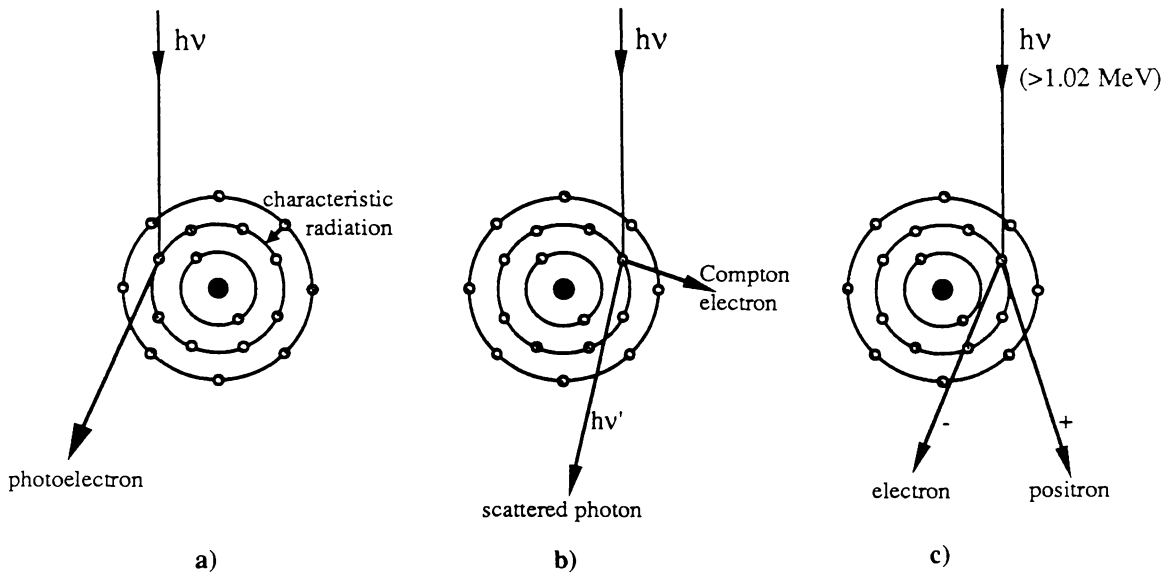

Figure 1.2. The three main photon interaction processes. (a) Photoelectric effect. (b) Compton effect. (c) Pair production.

It is now possible to rewrite the total mass attenuation coefficient of Equation 1.8 as the sum of the individual components discussed above:

$$\frac{\mu}{\rho} = \frac{N_A\sigma}{M} = \frac{\tau}{\rho} + \frac{\sigma_c}{\rho} + \frac{\kappa}{\rho} \ (\mathrm{m^2kg^{-1}}) \, . \tag{1.9}$$

Table 1.1 summarises the effects of photon energy E and atomic number Z on the significance of these processes, while Figure 1.3 illustrates the relative importance of the

cross-sections for these processes over a range of photon energy. It does not show the magnitude of the total cross-section, which decreases rapidly with photon energy in the 0–100 keV range, then more slowly at treatment energies. However, Figure 1.3 clearly illustrates that Compton processes dominate throughout radiotherapy energies, with the photoelectric contribution being significant at energies below 100 keV and pair production being significant above a few MeV (in water). Both photoelectric and pair production interactions tend to release more energy *per interaction* than the Compton process, so their relative contributions to dose deposition are somewhat understated by Figure 1.3.

Quantity	Effect of atomic number Z	Effect of photon energy E
photoelectric m.a.c. $\frac{\tau}{\rho}$	$\propto Z^3$ for $Z \leq 15$ $\propto Z^{3.8}$ for $Z \geq 16$	$\propto E^{-3}$
Compton m.a.c $\frac{\sigma_c}{\rho}$	$\propto Z$	slowly decreases with increasing E
pair-production m.a.c $\frac{\kappa}{\rho}$	$\propto Z$	increases rapidly above 1.02 MeV

Table 1.1. Effect of atomic number Z and photon energy E on mass attenuation coefficients.

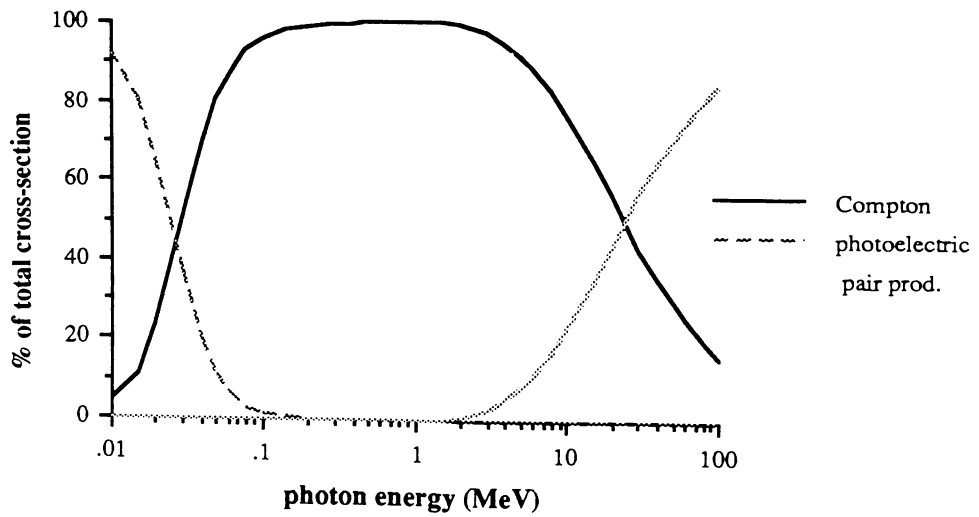

Figure 1.3. Relative importance of cross-sections for the three main photon interaction processes (data from Johns and Cunningham[2]).

1.2.3. Interaction of Electrons with Matter

Unlike photons, which do not lose energy as they travel, electrons are capable of directly *depositing* energy in a medium. Whether incident on the medium or liberated within it by photon interactions, the electrons set in motion interact in one of a number of ways (see Chapter 2). The measure of the rate of energy loss, analogous to mass attenuation coefficient for photons, is the **total mass stopping power.** Using the notation of Johns and Cunningham[2] the *linear* stopping power is denoted by $\frac{dE}{dx}$ and *mass* stopping power by S_{tot}. In this thesis the more usual notation of the ICRU,[1] Klevenhagen,[5] and others will be adopted, where total linear stopping power is denoted by S_{tot} and total mass stopping power by $\frac{S_{tot}}{\rho}$ (measured in MeV $g^{-1}cm^2$). $\frac{S_{tot}}{\rho}$ represents the energy which an electron loses per unit length per unit density. The concept of **mass angular scattering power** is also used to describe the way electrons interact in a medium: it is equal to the mean square angular deflection per unit mass per unit distance, written as $d\overline{\theta^2} \,/\, (\rho\, dz)$.

Energy is deposited in a medium by two different types of interaction:

(a) **Collisional energy loss,** in which an electron causes excitation or ionization of a bound electron. This is also known as *ionizational* energy loss or *Coulomb (Møller)* scattering. The excited electron deposits its extra energy in the medium as heat. Occasionally the excited electron has sufficient energy — approximately 100 eV — to be considered a distinct ionizing particle or *delta ray.* In general the incident electron continues on with only slightly reduced energy, so that an electron may undergo many thousands of collisions before coming to rest. The collisional (or ionizational) component $\frac{S_{col}}{\rho}$ of the mass stopping power varies approximately as E^{-2} for electron energies from 10 to 100 keV, reaches a minimum at approximately 1 MeV, then increases logarithmically with increasing energy. $\frac{S_{col}}{\rho}$ is less for high Z materials, primarily because the excitation energy of high-Z atoms is less. It is also affected by the density of medium, due to a phenomenon known as the *density effect.* This occurs because in dense materials the passage of an electron causes atoms to become polarised, resulting in a screening field which reduces the effect of distant atoms on the electron's energy loss. The density effect is usually corrected for using the theory of Sternheimer and Peierls.[6]

(b) **Radiative energy loss,** in which interaction of an electron with matter creates *photons.* In the *bremsstrahlung* process the electron is scattered by the field of the nucleus, emitting a photon as it is accelerated. If the electron interacts with an inner shell elec-

tron (much rarer than bremsstrahlung) then characteristic radiation may be emitted. At low energies the direction of maximum radiation is at right angles to that of the incident electron, while for linear accelerator energies the bremsstrahlung is emitted as a narrow forward-directed beam. Although characteristic radiation is a radiative energy loss, it is included in $\frac{S_{col}}{\rho}$ defined above, since the radiation is generally absorbed near the interaction site. The radiative component of the mass stopping power is that due to bremsstrahlung, denoted by $\frac{S_{rad}}{\rho}$. It increases linearly with Z, and also increases linearly with electron energy E.

The total mass stopping power can now be expressed as the sum of the collisional and radiative mass stopping powers:

$$\frac{S_{tot}}{\rho} = \frac{S_{col}}{\rho} + \frac{S_{rad}}{\rho} \quad (\text{MeV g}^{-1}\text{cm}^2) \,. \tag{1.10}$$

Figure 1.4 illustrates the relative importance of these two components in water. Note that bremsstrahlung becomes important only at energies above 10 MeV.

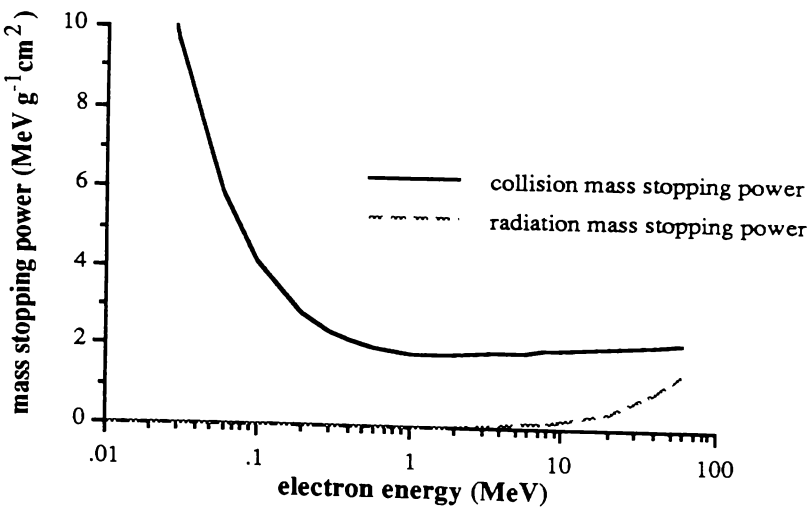

Figure 1.4. Relative importance of collisional and radiative energy loss in water (data from Berger and Seltzer[7]).

1.2.4. Quantities and Units for Radiotherapy

Having examined the ways in which photons and electrons interact with matter, and having use of the concepts of attenuation coefficient and stopping power, it is now possible to define some further quantities of use in external beam radiotherapy:

(a) **Linear energy transfer (LET)** defines that fraction of the stopping power which results in energy being deposited *locally* in the medium. All radiative interactions

are excluded, as well as collisions resulting in the production of a secondary electron capable of ranging away from the interaction site (a delta ray). LET is also known as the restricted linear collision stopping power, and is given the symbol L_Δ, where Δ represents the cutoff energy, above which electrons are considered to be delta rays. Similarly, the restricted mass collision stopping power $\frac{S_\Delta}{\rho}$ can also be defined. Note that $L_\infty = S_{\rm col}$, since L_∞ includes all delta rays, but excludes radiative energy loss.

(b) **Mean stopping power.** As an electron loses energy the stopping power of the medium changes. In order to describe an incident electron using a single quantity, it is necessary to define the mean ionizational mass stopping power of the particle $\frac{\bar{S}(E)}{\rho}$ as the mean stopping power experienced by the electron throughout its travel. Radiative energy loss is once again excluded, since it does not deposit energy locally.

(c) **Mass transfer and mass absorption coefficients.** When a photon interacts with an absorber, part of its energy is radiated away from the interaction site as photons, while the rest appears as the kinetic energy of excited or ionized electrons. $\bar{E}_{\rm tr}$ is used to denote the average energy transferred to kinetic energy of charged particles. Some of this energy is in turn radiated away as bremsstrahlung, but that which is absorbed by the medium as heat is denoted by $\bar{E}_{\rm ab}$. Two new coefficients can now be defined: the mass energy transfer coefficient $\frac{\mu_{\rm tr}}{\rho}$ and the mass energy absorption coefficient $\frac{\mu_{\rm ab}}{\rho}$ (sometimes denoted by $\frac{\mu_{\rm en}}{\rho}$). $\frac{\mu_{\rm tr}}{\rho}$ represents the rate at which energy is lost in collisions, while $\frac{\mu_{\rm ab}}{\rho}$ represents the rate at which heat is lost to the medium (deposited as dose). The difference between these two quantities is often expressed using the following formula:

$$\frac{\mu_{\rm ab}}{\rho} = \frac{\mu_{\rm tr}}{\rho}(1-g) \ \ ({\rm m^2 kg^{-1}}) \,, \tag{1.11}$$

where g is the fraction of energy converted to bremsstrahlung. The fraction g is small at low energies and in materials of low atomic number.

(d) **Terma.** As discussed above, when a photon first interacts in a medium some energy is given to electrons while the rest radiates away from the interaction site. The sum of these two components is known as the total energy released by primary photon interactions per unit mass, or terma.[8] For photons of energy E the terma T is given by

$$T = \Phi\left(\frac{\mu}{\rho}\right)E = \Psi\left(\frac{\mu}{\rho}\right) \ \ ({\rm J\,kg^{-1}}) \tag{1.12}$$

where Φ is the incident fluence, Ψ is the energy fluence, and $\frac{\mu}{\rho}$ is the mass attenuation coefficient in the medium at energy E. For a spectrum of incident energies the

expression for terma becomes

$$T = \int_0^{E_{max}} \frac{d\Phi(E)}{dE} \cdot \left(\frac{\mu(E)}{\rho}\right) \cdot E \; dE \;\; (\mathrm{J\,kg^{-1}}) \tag{1.13}$$

where E_{max} is the maximum energy in the spectrum. Terma is not a commonly used quantity in radiation physics, but when the superposition or convolution technique of dose calculation is used (see Section 1.4.4 and Chapter 3), terma becomes a fundamental quantity in the calculation.

(e) **Kerma.** If only that component of the terma which results in charged particles (usually electrons) gaining energy is considered, then a new quantity known as the kinetic energy released per unit mass, or kerma, can be defined. It is analogous to terma, except that in Equation 1.12 the total energy E is replaced by the average energy transferred per interaction $\bar{E}_{tr}$, and $\frac{\mu}{\rho}$ is replaced by the mass energy transfer coefficient $\frac{\mu_{tr}}{\rho}$, to yield the kerma K:

$$K = \Phi\left(\frac{\mu}{\rho}\right)\bar{E}_{tr} = \Psi\left(\frac{\mu_{tr}}{\rho}\right) \;\; (\mathrm{J\,kg^{-1}})\,. \tag{1.14}$$

Equation 1.13 can be similarly modified to produce K for an incident photon spectrum.

(f) **Absorbed Dose.** Although kerma describes that part of the incident energy which is converted to kinetic energy of electrons, it does not indicate how much energy is actually retained by the medium as heat. This is because electrons radiate away some of their energy as bremsstrahlung. Figure 1.5 demonstrates the relationship between kerma and absorbed dose: kerma released at a certain point in the medium causes absorbed dose to be deposited throughout the path of liberated electrons travelling *downstream* from the interaction site.

Absorbed dose D has the same dimensions ($\mathrm{J\,kg^{-1}}$) as terma and kerma, but is given the special unit *gray* (Gy), defined by

$$D = \frac{d\bar{E}_{ab}}{dm} \;\; (\mathrm{Gy})\,, \tag{1.15}$$

where $d\bar{E}_{ab}$ is the average energy imparted by ionizing radiation to a mass dm. The older unit *rad* is still commonly used, where 100 rad = 1 Gy = 1 $\mathrm{J\,kg^{-1}}$.

(g) **Dose Equivalent.** Different types of radiation may have widely differing biological effects on tissue. For this reason it is necessary to link absorbed dose to a quantity representing biological effect. This is done using dose equivalent H, which once

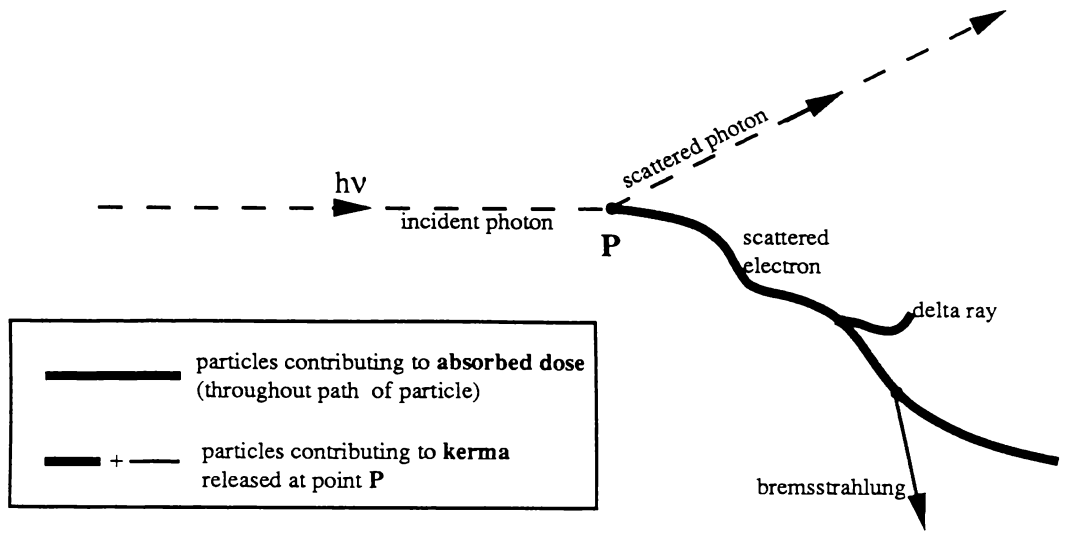

Figure 1.5. Distinction between kerma and absorbed dose.

again has units of J kg^{-1}. The special unit for dose equivalent is the *sievert*, where 1 Sv = 1 J kg^{-1}. It is related to absorbed dose D by

$$H = D \times Q \times N \quad (\mathrm{Sv}) , \tag{1.16}$$

where Q is the *quality factor* for the radiation and N (nominally 1.0) accounts for other factors such as dose fractionation, dose distribution and dose rate.[2] For photons and electrons Q is given the value 1.0, hence 1 Gy of absorbed dose results in a dose equivalent of 1 Sv.

(h) Exposure. Although the previously mentioned units can adequately define the effect of a radiation beam, one common method of calibrating beams is by the use of ionization chambers. These devices measure the amount of ionization which a beam causes in air, in units of exposure X:

$$X = \frac{\mathrm{d}Q}{\mathrm{d}m} \quad (\mathrm{C\ kg^{-1}}) , \tag{1.17}$$

where dQ is the total charge in coulombs produced by photons ionizing a mass dm of air. The older unit *roentgen* (R) is still commonly used, where 1 R = 2.58 × 10^{-4} C kg^{-1}. Exposure X can be converted to absorbed dose D_{med} in a medium, using the relation

$$D_{\mathrm{med}} = X \cdot f_{\mathrm{med}} \quad (\mathrm{Gy}) \tag{1.18}$$

where

$$f_{\mathrm{med}} = (33.85\ \mathrm{J\ C^{-1}}) \left(\frac{\mu_{\mathrm{ab}}}{\rho}\right)_{\mathrm{med}} \Big/ \left(\frac{\mu_{\mathrm{ab}}}{\rho}\right)_{\mathrm{air}} . \tag{1.19}$$

1.2.5. Dosimetry Techniques

In the introduction to this chapter it was mentioned that frequent and accurate calibration of linear accelerator beams is essential to the treatment process. A variety of methods for measuring dose delivered by a photon beam are available.[2,3,9] The most commonly used methods are:

(a) **Photographic dosimetry.** Thin sheets of film are placed between slabs of a phantom and exposed to a beam. The resulting optical density of the film is then measured using a densitometer, and this value is used as a measure of the dose received. Film has high spatial resolution, is quick to use and provides a good means of comparing dose (with earlier exposures). However, this method is not sufficiently accurate for absolute radiotherapy calibrations and monitoring, as sensitometric corrections are required for dose rate and spectral effects.

(b) **Ionization chambers.** Gas-filled ionization chambers can be used to accurately measure exposure, which can then be converted to absorbed dose. They rely for their operation on the Bragg-Gray principle, which states that a sufficiently small cavity does not affect the number or energy of electrons traversing the region occupied by the cavity. Alternatively, Fano's theorem permits large cavities in areas of electronic equilibrium, provided the cavity is of the same composition as the surrounding medium. In either case, several empirical corrections must be made, accounting for the finite size of the ion chamber, temperature and pressure, chamber wall composition and thickness, and ion recombination.[2] Ionization chambers are accurate and give reproducible results, but can usually be used only in air, water or specially drilled solid phantoms.

(c) **Solid state detectors.** Radiation also produces current in specially doped semiconductors. The high density of semiconductor material means that large numbers of ion pairs are generated per unit volume, and hence the detectors can be made very small, leading to excellent spatial resolution. Disadvantages of solid state detectors include aging of the semiconductor material, and sensitivity to photon energy.

(d) **Thermoluminescent dosimetry (TLD).** Some crystalline solids, in particular LiF, can trap electrons in high energy levels when irradiated. When heated in a TLD reader the electrons are elevated to even higher levels, from which they can return to the ground state, emitting light in doing so. By measuring this light the dose absorbed by the TLD can be estimated. TLDs can be used to measure dose over a wide range, but are less accurate than ionization chambers. Since there are no

electrical connections, TLDs are especially useful for monitoring dose to the patient during treatment, especially entry, exit, intracavity, and eye doses.

Other methods used for dosimetry include calorimetry (used by standards laboratories) and chemical dosimetry. The importance of dosimetry in radiotherapy is evidenced by the amount of time which the hospital physicist devotes to this task, and by the large amount of research effort spent in improving both the theory and implementation of dosimetry techniques.

1.3. Aspects of External Beam Radiotherapy

1.3.1. Radiotherapy Treatment Machines

The first available radiotherapy units were "teleradium" units utilising a small radium capsule as the source.[9] However, the radiation beams produced were of very low intensity, due to the prohibitive expense of radium and the self-absorbing properties of a radium source.

The availability of cobalt-60 sources signalled the real beginning of radiotherapy in 1952.[10] Cobalt-60 teletherapy units consist of a source of between 1000 and 5000 Ci contained in a steel capsule and surrounded by lead. The therapy beam exits through a multi-vaned collimator. The energy of photons produced by a cobalt-60 machine is approximately 1.25 MeV. Cesium units, producing lower energy radiation, were also developed to treat shallower lesions.

Betatrons also became available for radiotherapy. In the betatron, electrons are injected into an evacuated glass or porcelain cavity subject to an oscillating magnetic field. The electrons orbit the chamber, gaining energy from the magnetic field. Eventually they are made to exit the field and hit a target, producing bremsstrahlung photons. Betatrons produce particle energies as high as 45 MeV, but dose rates from these machines are low.

Most radiotherapy units in use today are linear accelerators (linacs). Linacs have several advantages over cobalt-60 machines: they can provide both electrons and photons at multiple energies, they can provide relatively high energy (deeply penetrating) beams, high dose rates can be achieved, and they are not radioactive while turned off. In addition, linacs have a sharper penumbra (except at very high energies), and generally have less beam divergence as a result of operating at a larger source-surface distance (see Section 1.3.2 for discussion of these quantities). Figure 1.6 illustrates schematically a typical medical linac, although there are many different designs. In each case an electron gun is used to inject electrons into

an evacuated accelerating tube of the travelling wave or standing wave variety. Microwave power, produced by a magnetron or klystron, is also supplied to the tube, and the resultant electric field accelerates the electrons to energies of between 4 and 35 MeV. The electron beam is usually bent through 90 or 270 degrees into the treatment head. A scattering foil is used to disperse the electrons into a flat beam, or a tungsten target may be placed in the path of electrons, producing bremsstrahlung photons for use as a photon beam. Since the bremsstrahlung photons form an energy spectrum, beam energy is referred to using the highest energy component and the notation MV: for example, a 10 MV beam has a maximum photon energy of 10 MeV but a mean energy of only approximately 3 MeV. Finally, the beam passes through a collimation system, flattening filter (for photons), and the beam defining jaws.

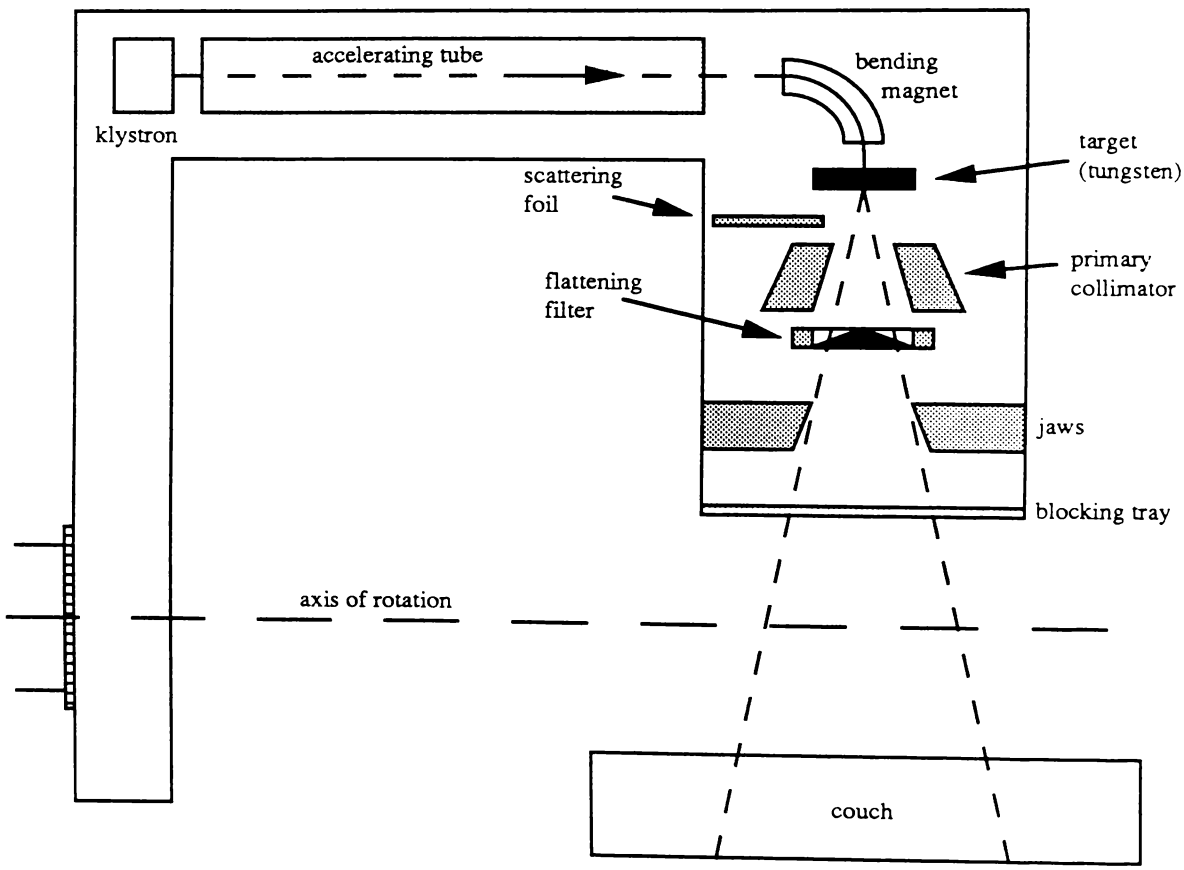

Figure 1.6. Schematic of a typical medical linear accelerator (photon treatment). For electron treatment the target and flattening filter are replaced by the scattering foil.

The significant advantages which linacs have over cobalt-60 units and betatrons mean that they are currently the preferred modality for most radiotherapy treatments. This is despite their high initial and running costs, higher staffing requirement and need for constant calibration.

1.3.2. Characteristics of the Dose Distribution Produced by a Photon Beam

As discussed in the previous section, linear accelerators produce photon beams by the collision of high energy electrons with a target, producing bremsstrahlung photons. These photons travel from the treatment head to the patient or phantom surface essentially unattenuated by the intervening air. When the photon beam passes through the target medium it is attenuated according to that medium's attenuation coefficient. The intensity of the beam falls due to absorption by the medium, and for point sources, which a linac approximates, there is also an inverse square attenuation due to increasing distance from the (virtual) source. For a photon beam whose source is a distance f from the surface of the medium,

$$\Phi(x) = \Phi_0 \, e^{-\mu x} \left(\frac{f^2}{(f+x)^2} \right) \quad (\mathrm{m}^{-2}) \,, \tag{1.20}$$

where Φ_0 is the fluence at the surface and x is the depth in the medium.

Although the incident photons do not deposit dose directly, they interact with matter to liberate *kerma*. In the region close to the surface of the medium the dose deposited is relatively low, due to the lack of electrons liberated upstream which normally contribute to the dose. At a depth slightly less than the maximum electron range the dose rises to a maximum D_{max} occurring at position d_{max}. The dose then decreases with depth in a manner closely allied to the kerma, as illustrated in Figure 1.7.

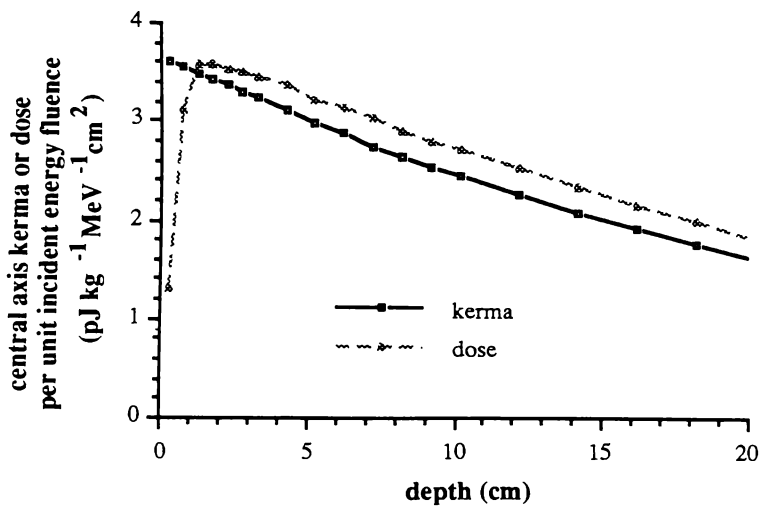

Figure 1.7. Kerma and dose deposition with depth. Results were generated by using EGS4 Monte Carlo user code RTPCYL to simulate a 2cm radius parallel beam of 3 MeV photons incident on a water phantom.

The region near the surface where dose rises to a maximum is known as the *buildup region*. The existence of this region is a major advantage of high energy photon beams, since

the dose to the surface (skin) is much less than to deeper tissue, and thus the incidence of radiation-induced skin reactions is reduced. For this reason this effect is also known as *skin-sparing*. It occurs because more electrons are travelling from the region than are travelling into the region, causing a local deficiency of electrons, and hence of dose deposition. This phenomenon is known as *electronic disequilibrium*, and is responsible for the failure of many treatment planning algorithms to accurately predict dose in areas of heterogeneity.

Figure 1.7 illustrated a depth dose curve for photons of energy 3 MeV. Higher energy photons tend to liberate higher energy electrons, and hence the range of the electrons is longer. This causes the position of D_{max} to shift deeper into the medium, while lower energy photons cause D_{max} to occur at a shallower depth. If incident photons form a spectrum then the dose peak broadens.

Field size also affects how dose delivered to the patient varies with depth. As well as ranging downstream from their site of liberation, electrons range *laterally*. Near the edge of a field the number of electrons scattering outside the field is greater than the number scattering towards the centre of the field. This *lateral electronic disequilibrium* causes dose at the edge of the field to fall less sharply than expected in the *penumbral region* (see Figure 1.8). There is also a component of the penumbra due to the finite size of the beam source, but in linear accelerators the virtual source size is very small, and this effect is overshadowed by electronic disequilibrium.

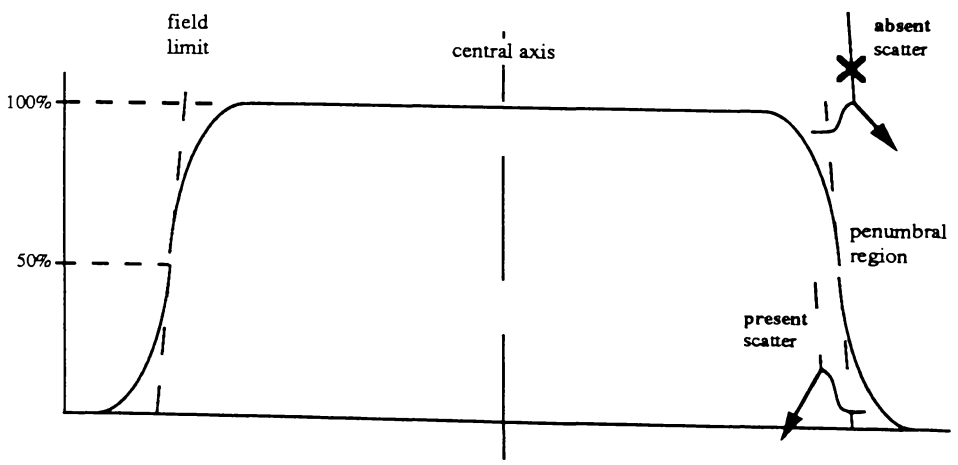

Figure 1.8. Beam profile at depth.

Another factor affecting the shape and intensity of a photon field is the *source-to-skin distance* or *SSD* (sometimes termed the source-surface distance, and also known as the focus-to-surface distance or FSD). For treatment using linear accelerators this distance is

typically 100 cm. By examining Equation 1.20 it is seen that a large SSD (f) will cause the fluence, and hence the dose, to drop off less rapidly with depth (since the inverse square factor is less significant). Increasing SSD will also reduce the geometrical penumbra for a given field size, although the penumbra due to electronic disequilibrium will remain unaffected.

One further complication of photon fields generated from linear accelerators is the possibility of *electron contamination*. This occurs when the photon beam interacts with the beam collimating jaws, flattening filter and block shadow tray, producing high energy electrons. These electrons then form (an unwelcome) part of the treatment beam, affecting the shape of the depth dose curve in the buildup region, and increasing surface dose.[11]

1.3.3. Measuring the Dose Distribution Produced by a Photon Beam

In order to quantify the effect of a photon beam on a patient or phantom, a number of radiological measures have evolved. Some measures developed for use in low energy beams are less suitable at photon energies generated by linear accelerators, so it has been necessary to develop additional quantities, illustrated in Figure 1.9 and discussed below. Several references examine the following measures in more detail,[2,9] and tables of these quantities are readily available.[2,12]

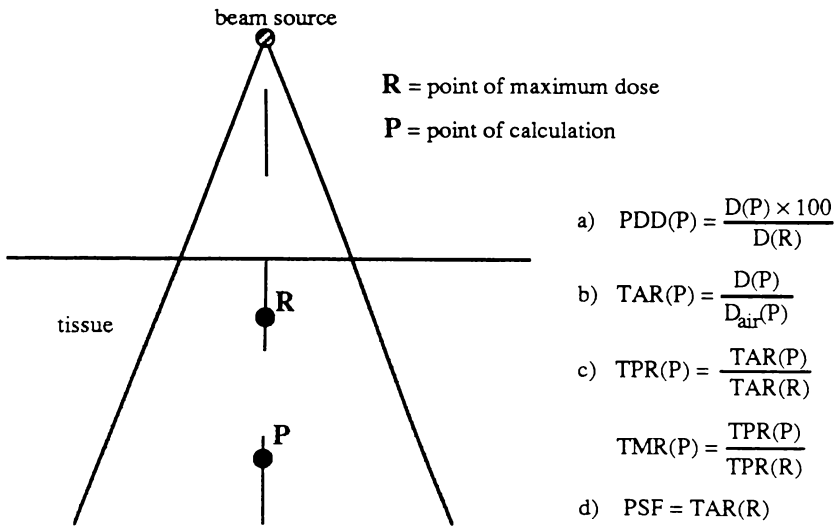

Figure 1.9. Measures of the dose distribution produced by a photon beam. (a) Percentage depth dose (PDD). (b) TAR. (c) TPR and TMR. (d) Peak scatter factor (PSF).

(a) **Percent depth dose (PDD).** Percent depth dose (usually at central axis) is the ratio of dose at a point to dose at a reference point in the patient or phantom. The reference dose is usually the dose D_{max} occurring at d_{max}, and the ratio is expressed as

a percentage. It is the most commonly used of all radiation measures, since it demonstrates the characteristics of the beam in an easily understood way. It is dependent upon depth, width of the beam, SSD, and beam energy. Depth doses using a reference point other than D_{max} are referred to as *relative depth doses.*

(b) **Tissue-air ratio (TAR) and scatter-air ratio (SAR).** The tissue-air ratio is a dimensionless quantity representing the ratio of the dose in tissue to the dose in air at the same point. It is a function of field size, depth and photon energy. TAR increases slowly with field size, decreases with increasing tissue depth, and increases with increasing photon energy (except at shallow depth, where TAR decreases). When measuring TARs the distance from the source to the point of measurement (source-axis distance or SAD) is kept constant, so that additional phantom material is stacked *on top* of the measuring device. TARs are useful in calculating depth dose curves from ionization measurements made in air, using a buildup cap to ensure electronic equilibrium in the chamber. Similarly, depth dose curves in water can be used to calculate TARs. TARs are often decomposed into two components: the *zero-area* TAR representing the contribution from primary radiation, and the *scatter-air ratio* (SAR) representing the contribution from scattered radiation. Although the zero-area TAR cannot be measured directly, it can be extrapolated from measured data, allowing SAR values to be determined. SARs are useful in some treatment planning algorithms (see Section 1.4.3).

(c) **Tissue-phantom ratio (TPR).** For high energy photons such as those produced by linear accelerators the buildup cap used to measure TARs must be made very large. Scatter from the buildup material affects the measurement, and for small fields TAR cannot be measured, since electronic equilibrium does not exist within the field. An alternative is to use the tissue-phantom ratio, which is the ratio of the dose at a point to the dose at a point which is the same distance from the source but at a reference depth d_{ref}. This approach avoids the need for a buildup cap. TPR and TAR are related in a simple way[2] (see Figure 1.9c).

(d) **Tissue-maximum ratio (TMR).** If the reference depth d_{ref} is set to d_{max}, then the TPR becomes known as a tissue-maximum ratio (TMR). It is also the ratio of the TPR at a point to the TPR at the point of maximum dose (see Figure 1.9c). Since tissue-equivalent material is stacked on top of the machine isocentre when making TPR measurements, the dose at d_{max} can be derived from TMR.[2] This provides a measure similar to fractional depth dose.

(e) **Backscatter factor (BSF) or Peak scatter factor (PSF).** The peak scatter factor (PSF) is equal to the TAR at the depth of maximum dose d_{max}. It is the factor by which energy *scattered* from the phantom increases the dose. Since for high energy photons D_{max} does not occur at the surface, not all scattered radiation reaching d_{max} is strictly speaking backscattered, and hence the more generally applicable term is peak scatter factor (PSF). TAR, TMR and PSF are linked by the relationship

$$\mathrm{TAR}(d, W_m) \;=\; \mathrm{TMR}(d, W_m) \times \mathrm{PSF}(W_m)\;, \tag{1.21}$$

where d is depth and W_m is the width of the (square) field at depth d_{max}.

Consider a point D which is d cm below the surface of a phantom irradiated by a field of width W_m at D_{max} and W_d at point D, as illustrated by Figure 1.10. For a given energy of radiation, Johns and Cunningham[2] interrelate these quantities by an expression similar to the following:

$$\begin{aligned}\mathrm{PDD}(d, W_m, \mathrm{SSD}) \;&=\; \frac{100 \times \mathrm{TAR}(d, W_d) \times (d - d_{max})}{\mathrm{PSF}(W_m)} \\ &= 100 \times \frac{\mathrm{TAR}(d, W_d)}{\mathrm{PSF}(W_m)} \times \left(\frac{\mathrm{SSD} + d_{max}}{\mathrm{SSD} + d}\right)^2 \;(\%)\;.\end{aligned} \tag{1.22}$$

Thus percentage depth doses can be calculated from TARs and PSFs, or *vice versa.* Since TAR is related to TPR (see Figure 1.9c), measurements taken in water phantoms can also be used to determine percentage depth doses and *vice versa.*

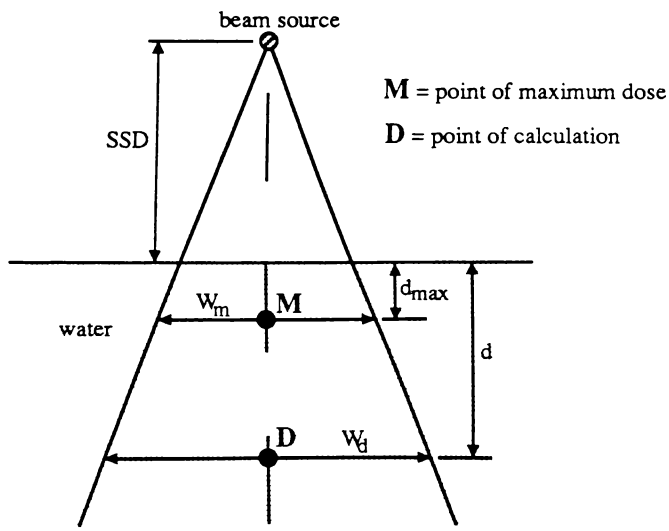

Figure 1.10. Parameters used in Equations 1.21 and 1.22 (adapted from Johns and Cunningham[2] page 339).

The quantities defined above are suitable primarily as a measure of how dose varies with depth in the centre of the field (where lateral electronic equilibrium exists). However, it is also useful to be able to describe the *shape* of the field: that is, how dose varies with distance from the central axis. This may be done using the following concepts:

(a) **Profiles.** If an ionization chamber is driven across a phantom at depth d in a plane perpendicular to the beam axis, then a *dose profile* at that depth can be generated. A dose profile gives a good indication of the beam shape at that depth, but on its own gives no information about how dose varies with depth. Sets of beam profiles are often used in conjunction with depth dose measurements to completely define the beam in one plane parallel to (and usually intersecting) the central axis. Figure 1.11a illustrates a beam profile, and also shows how the 50% point in the penumbral region can be used to define the *field size* at depth d. The ratio of dose at a particular off-axis distance to the central axis dose at the same depth is known as an *off-axis ratio* (OAR) or *off-centre ratio* (OCR).

(b) **Isodose curves.** Isodose curves provide a good way of visualising the overall shape of the beam. An isodose curve joins points of equal dose, so a set of isodose curves — one for each selected dose level — provides a 2-dimensional picture of the beam (see Figure 1.11b). For constant-SSD measurements, isodose curves are usually normalised to be 100% at the point of maximum dose, while for *isocentric* measurements, where the source to machine axis distance (SAD) is constant, curves are usually normalised to the *isocentre* (machine centre). However, in constant-SSD measurements involving heterogeneities, isodose curves are often normalised to the point of maximum dose *in water*. This allows direct comparison between heterogeneous and homogeneous cases.

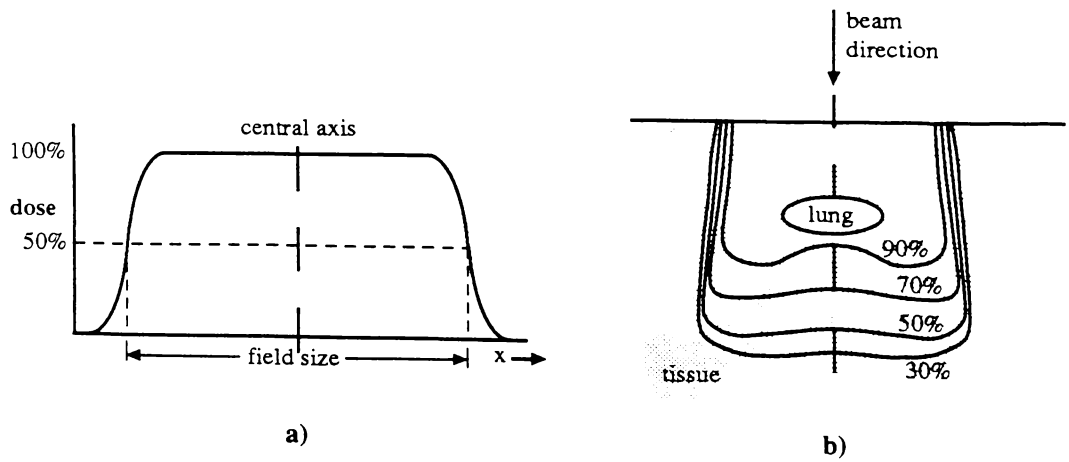

Figure 1.11. Measures of beam shape. (a) Dose profile. (b) Isodose curve.

Many of the above quantities, such as TAR, TPR and TMR, were developed primarily as tools for use in treatment planning. However, newer dose calculation methods discussed in Section 1.4, particularly Monte Carlo and superposition/convolution, do not require these quantities, relying instead on more fundamental concepts such as terma and energy deposition kernels. Percent depth doses, profiles and isodose curves will always be useful, however, since they are an intuitive and easily visualised description of the dose distribution, and hence provide a means of direct comparison between the various dose calculation methods.

1.3.4. Photon Treatment

The first step in radiotherapy treatment is to obtain patient anatomical data (usually from diagnostic x-rays or CT scans). The radiotherapist (with the aid of a physicist) then determines:

- A suitable *target volume*. This volume usually contains some healthy tissue as well as the tumour (see Figure 1.1), since the edge of the tumour may be difficult to define.
- Dose fractionation (many small doses tend to be more effective than one large dose in eradicating a tumour).

The planning radiographer (radiation therapist) must then decide:

- How many beams should be used, from what angles they should be directed and what their relative intensities should be.
- How each beam should be modified (if at all).

These last two considerations require knowledge of radiation physics — much of it already incorporated into the treatment planning system — but also rely on the medical knowledge and experience of the radiation therapist. However, the physicist can certainly advise the radiation therapist on how various treatment options are likely to affect the dose distribution — in particular the option of beam modification. Photon beams are usually modified for one of two reasons:

(a) **Correcting for missing tissue.** When the photon beam intersects the patient at an oblique angle the "missing" tissue causes the dose to be increased (see Figure 1.12a). This effect is often undesirable, in which case *bolus* may be used. Bolus is simply a tissue-like material which replaces the missing tissue, restoring the dose distribution to its normal shape. However, a bolus destroys the skin-sparing properties of a high energy photon beam. A similar effect can be achieved without loss of skin-sparing by placing a *wedge* or *compensating filter* away from the skin surface (this is known as

a *retracted compensator*). Figures 1.12b and 1.12c illustrate the use of a bolus and a wedge to restore the shape of the dose distribution.

(b) **"Tilting" isodose lines.** Rather than correct for a distorted dose distribution, it may be desirable to intentionally distort the contribution from a beam in order to achieve a more desirable dose distribution within the patient. Figure 1.12d illustrates the use of a wedge filter to "tilt" the isodose lines from a single beam. By combining two wedged fields at right angles it may often be possible to obtain a region of uniform dose where the fields intersect.

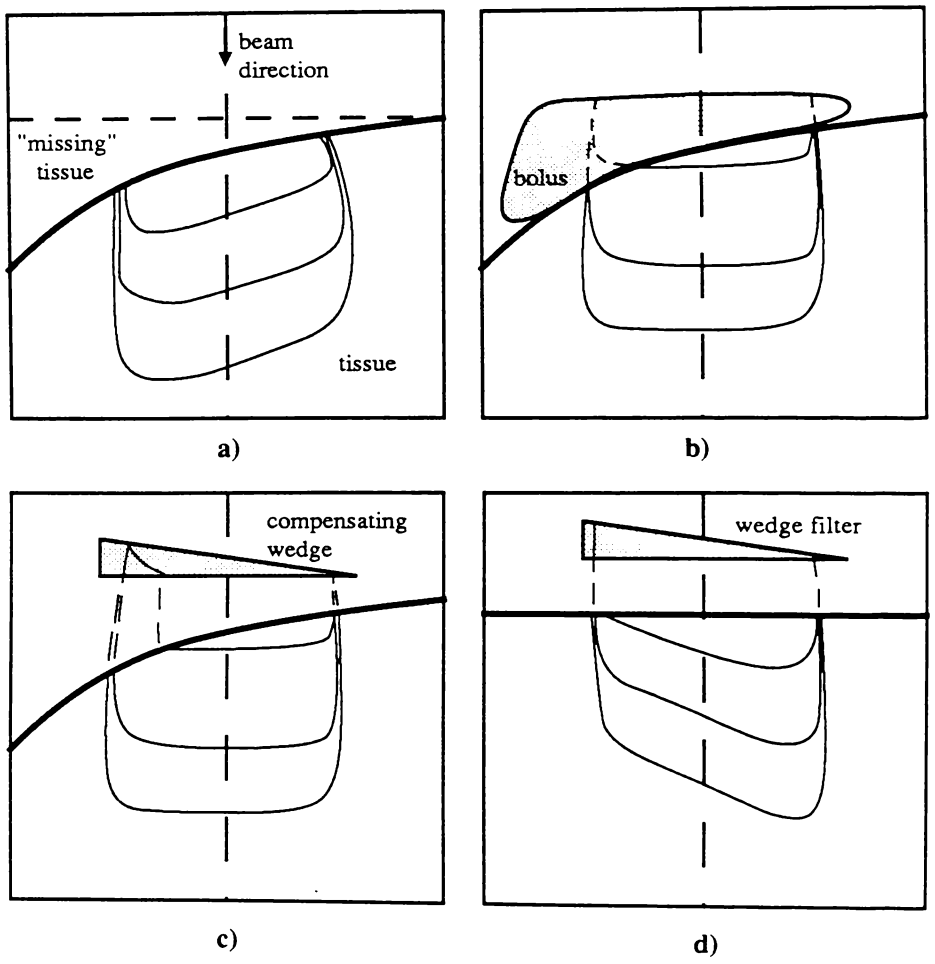

Figure 1.12. Modifying dose distribution from a single beam. (a) "Missing" tissue causing distortion of the dose distribution. (b) Correction using a bolus. (c) Correction using a wedge. (d) "Tilting" the isodose lines with a wedge filter.

The radiation therapist can use these techniques to help achieve a more satisfactory dose distribution than would otherwise be possible. They might employ a single field, a pair of opposing fields, three fields, two opposing pairs of fields directed at right angles, or six fields. The radiation therapist must also decide the relative weighting of these fields in

order to achieve optimal dose distribution. Figure 1.13 illustrates just two of the possible arrangements and the type of dose distributions that they produce. Detailed discussions of field placement can be found in many textbooks.[2,9,13]

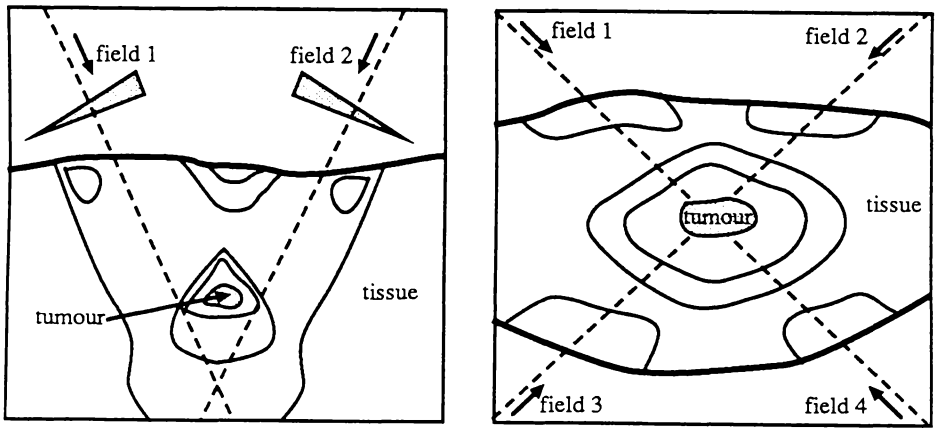

Figure 1.13. Treatment using multiple fields (solid lines are isodose curves). (a) Two wedged fields (for irradiating a tumour through one skin surface). (b) Two opposed pairs of fields.

In addition, other techniques such as *rotation therapy* are available for tumour treatment. In rotation therapy the treatment head of the linear accelerator is rotated around the patient during treatment. Rotation therapy gives some dose to all regions in the field, and thus a larger area of healthy tissue is exposed, but at a reduced dose.

Using the system of measures discussed in Section 1.3.3, it is possible to quantify the effects of various beam arrangements and modifications. Provided accurate anatomical patient information is known, the treatment planning system can now be presented with all the information necessary to accurately determine dose in a homogeneous medium. With further knowledge of the way in which photon beams behave in regions of heterogeneity it is possible, in theory at least, to predict dose in arbitrary regions of heterogeneity. The algorithms available to predict dose in both homogeneous and heterogeneous cases are discussed in the next section.

1.4. Photon Treatment Planning Algorithms

1.4.1. Introduction — The Heterogeneity Problem

In 1976 the International Commission on Radiation Units and Measurements (ICRU) recommended that the degree of accuracy for delivery of tumour dose should be $\pm 5\%$.[14] Cunningham estimates that an accuracy of $\pm 3.8\%$ is required from the dose calculation algorithm to achieve this.[15] Various algorithms have been used to carry out treatment planning

simulations, as documented by Cunningham[15] and the ICRU.[14] More sophisticated algorithms currently used by commercial systems for photon treatment planning include *scatter function* methods. The Batho power law and equivalent tissue-air ratio (E-TAR) techniques use data from TAR tables to modify standard measurements in water for use in heterogeneous cases. Although these corrections are suitable for use at cobalt energies, they fail to accurately predict the dose in heterogeneous cases at linear accelerator energies.[16] This is due primarily to inadequate modelling of the physical processes, in particular the scattering of *electrons* away from the photon interaction sites. This phenomenon results in electronic disequilibrium, which is particularly noticeable in regions of density greatly different from that of tissue, such as in body cavities, lung, and to a lesser extent, bone. Figure 1.14 illustrates the effect of lung cavities on the central axis depth dose curve for a 10 MV photon beam. Note the drop in dose (*build-down*) just before the first (tissue-lung) interface, and the rise in dose (*buildup*) just before the second (lung-tissue) interface. These effects are due to "longitudinal" electronic disequilibrium. The build-down case is caused by a deficiency of backscatter from the low density region, while in the buildup case there is extra backscatter from the nearby high density region. The other effect noticeable from Figure 1.14 is the drop in dose in the central lung region, caused by *lateral* electronic disequilibrium. Here electrons scatter several centimetres away from the central axis (since the density is low), and are not completely replaced by electrons scattering from the edge of the field.[17] This effect is not present at the central axis for larger field sizes, since electronic equilibrium is present. However, lateral electronic disequilibrium will always affect the penumbral shape of any field size.

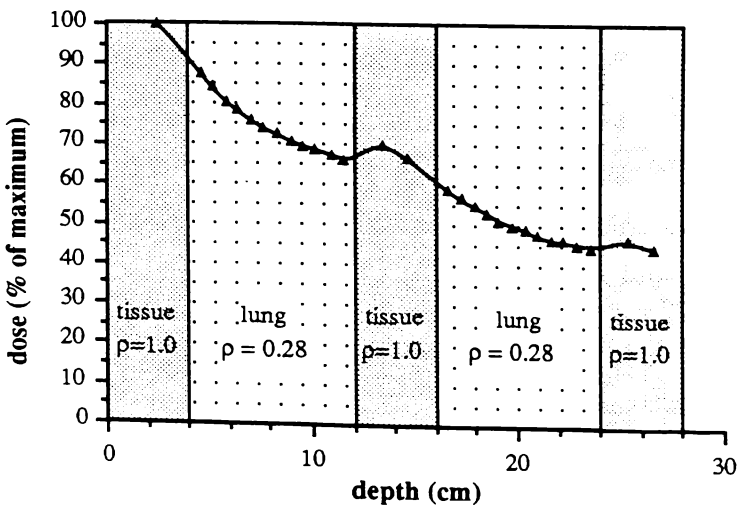

Figure 1.14. Central axis depth dose for a 5 cm × 5 cm photon field of energy 10 MV incident on a chest phantom. Results obtained using a Farmer ionization chamber (adapted from Hoban *et al.*[17]).

There exist methods which more successfully model radiation transport. The most physically realistic of these is *Monte Carlo* modelling, in which individual particles are modelled using random variables to determine the sites and types of interaction for each particle. *Superposition/convolution* methods can also accurately model the physical processes. However, both these methods are much more computationally intensive than currently used algorithms.

1.4.2. Effective Depth Methods

Effective depth methods are based upon calculating the *water equivalent depth* of the dose calculation points, also known as the *equivalent pathlength* or *radiological depth*. These methods include:

(a) **Effective attenuation coefficient method.** In this method the water equivalent depth is calculated at each dose point, and is used to calculate a correction factor for each point. For example, in Figure 1.15 the depth d of point P is 2.5+5.0+2.0 = 9.5 cm, while the water equivalent depth d' of point P is $2.5 + (5.0 \times 0.25) + 2.0 = 5.75$ cm. The correction factor C for point P when irradiated by a 10 MeV photon field is then calculated using[15]

$$\begin{aligned} C &= e^{\mu(d-d')} \\ &= e^{0.0222\times(9.5-5.75)} \\ &= 1.087 \ , \end{aligned} \tag{1.23}$$

where μ is the linear attenuation coefficient for water at that energy. Dose at point P is then obtained by multiplying C by the dose obtained in a homogeneous water phantom at depth d.

(b) **Effective SSD method.** This is similar to the previous method, except that the correction factor C is formed from the ratio of the percentage depth doses at depths d' and d, multiplied by an inverse square correction. The correction factor C is given by

$$C = \frac{\mathrm{PDD}(d', W_0, \mathrm{SSD})}{\mathrm{PDD}(d, W_0, \mathrm{SSD})} \times \left(\frac{\mathrm{SSD} + d'}{\mathrm{SSD} + d}\right)^2 \ , \tag{1.24}$$

where PDD is percentage depth dose, W_0 is the field width at the surface and SSD is the source-surface distance. This method accounts for beam divergence as well as depth and field size, and can also account for beam profile shape. This method has been commonly used in computerised treatment planning systems, in which case *off-axis ratios* (OARs) are employed to store the shape of the dose distribution off-axis. A simplification known as the *isodose shift method*[15] may be used when the calculation is to be done manually.

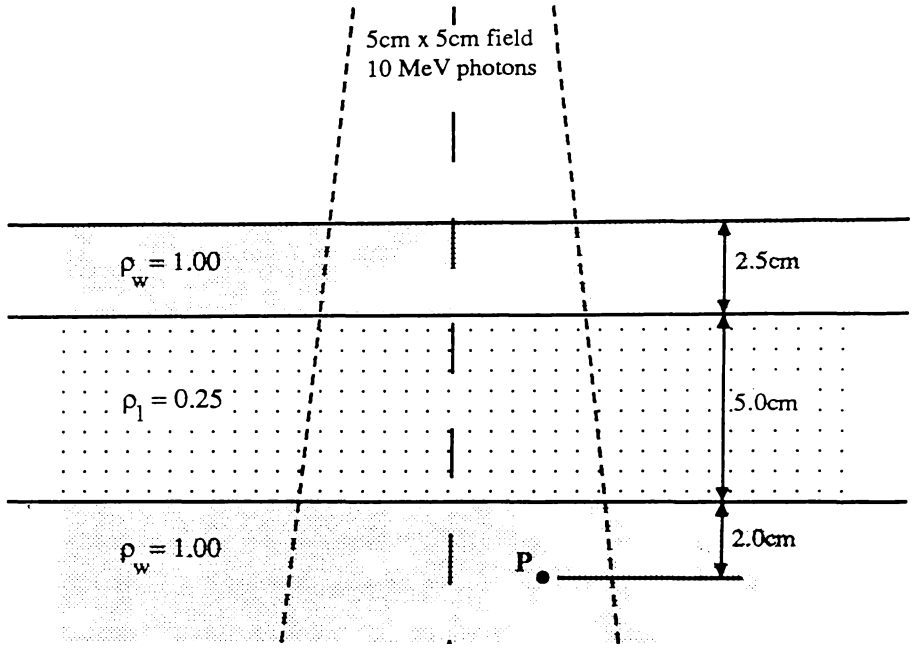

Figure 1.15. Phantom arrangement used for discussion of dose calculation methods. Material of density $\rho_l = 0.25$ g cm^3 represents lung, while material of density $\rho_w = 1.00$ g cm^3 represents tissue. A 5 cm × 5 cm field of monoenergetic photons (energy 10 MeV) is incident on the phantom.

1.4.3. TAR Correction Methods

There is a more sophisticated class of algorithms which use tissue-air ratios (TARs) to correct homogeneous dose distributions in order to account for heterogeneities. The first method mentioned below is essentially similar to the effective depth methods, in that it fails to consider the effect of scattered photons on the dose distribution. However, the other methods listed below apply increasingly complex inhomogeneity corrections which attempt to account for scattering (and hence they are also known as *scatter function models*).

(a) **Ratio of tissue-air ratios method.** This is similar to the effective SSD method, except that the correction factor C is formed from the ratio of the TAR at depth d' to the TAR at depth d:

$$C = \frac{\mathrm{TAR}(d', W_d)}{\mathrm{TAR}(d, W_d)} \ , \tag{1.25}$$

where TARs are expressed as a function of W_d, the beam width at depth d. Unlike the effective attenuation coefficient method, this method accounts for both field size and depth. Both this method and the effective attenuation coefficient method were originally developed for calculation by hand.[14]

(b) **Power law tissue-air ratio method (Batho method).** This method, developed by Batho[18] and modified by Sontag and Cunningham,[19] employs a correction factor that takes heterogeneities into account. As an example consider point P in Figure 1.15,

lying at a depth $d_1 = 2.0$ cm below the upper surface of the second tissue region (density $\rho_1 = 1.0$ g cm^{-3}). It is also $d_2 = 7.0$ cm below the upper surface of the lung heterogeneity (density $\rho_2 = 0.2$ g cm^{-3}). These distances are incorporated into a new correction factor C such that

$$C = \frac{\mathrm{TAR}(d_1, W_d)^{\rho_1 - \rho_2}}{\mathrm{TAR}(d_2, W_d)^{1-\rho_2}} \quad . \tag{1.26}$$

In previous methods the equivalent pathlength used to calculate correction factors was unaffected by the *position* of the heterogeneity, but the power law method accounts for this, and can therefore be said to crudely model photon *scatter*. However, care must be taken in using this method near interfaces, and in buildup and build-down regions.[15]

(c) **Equivalent tissue-air ratio method (E-TAR).** The power law method works well in regions below a heterogeneity, but results in relatively large errors *within* regions of heterogeneity, especially for large field sizes.[15] The equivalent tissue-air ratio method was developed by Sontag and Cunningham[20] to more adequately model heterogeneities, and to enable the use of CT scan data without identification of structures (necessary for the power law method). It assumes that the dose to a point in a heterogeneous medium is equivalent to the dose at some point in a homogeneous phantom of unit density, where field size and depth may be scaled as necessary. The equivalent tissue-air ratio correction factor C can be written as[20]

$$C = \frac{\mathrm{TAR}(d', \tilde{r})}{\mathrm{TAR}(d, r)} \quad , \tag{1.27}$$

where d is depth, r is field radius and d' and $\tilde{r}$ are corresponding scaled quantities calculated by the E-TAR method. It can be shown that these quantities scale linearly with electron density.[21] More formally, $\mathrm{TAR}(d, r)_\rho$ — the TAR at depth d, field radius r in a homogeneous medium of density ρ — is equal to $\mathrm{TAR}(d\rho, r\rho)$ in a unit density homogeneous medium.[20]

Heterogeneous media require a more complicated correction for scattered photons. By writing scaled TAR as the sum of the zero-area TAR and the scatter-air ratio (SAR):

$$\mathrm{TAR}(d', \tilde{r}) = \mathrm{TAR}(d', 0) + \mathrm{SAR}(d', \tilde{r}) \quad , \tag{1.28}$$

the problem is decomposed into a simple scaling for zero-area TAR and a more complicated SAR scaling. The scaled distance d' is just a density-weighted average of the n voxels along the pathlength d:

$$d' = \frac{d \cdot \sum_{j=1}^{n} \rho_j}{n} \quad , \tag{1.29}$$

and the field size is scaled using

$$\tilde{r} \;=\; r \times \tilde{\rho} \;\;, \tag{1.30}$$

where $\tilde{\rho}$ is formed by weighting all voxels according to their importance in the scattering process, then averaging the result. Performing such an average over all voxels is a computationally intensive task, so to reduce calculation time the E-TAR method "collapses" the CT planes onto a single plane, using weighting factors for each plane (more remote planes contribute less to the scatter component). These $\tilde{\rho}_{ij}$ form a single scattering plane which is a distance Z_{eff} away from the calculation plane. The effective density $\tilde{\rho}$ is then calculated by a summation over the ij plane:

$$\tilde{\rho} \;=\; \frac{\sum_i \sum_j \tilde{\rho}_{ij} \cdot W_{ij}(Z_{\text{eff}})}{\sum_i \sum_j W_{ij}(Z_{\text{eff}})} \;\;, \tag{1.31}$$

where both Z_{eff} and the weighting factors W_{ij} are calculated using relationships derived by Sontag and Cunningham.[20]

E-TAR is superior to the power law method and other methods discussed previously because it takes into account the lateral extent of heterogeneities. It can also be directly applied to CT generated electron density data. However, it requires considerably more computation than the previous methods, and requires a large amount of empirical data.

(d) **Volume integration of differential scatter-air ratios (dSARs).** A scatter-air ratio (SAR) measures the amount of radiation scattering to a point from all of the surrounding volume. By differentiating this quantity, an expression for the amount of radiation ΔS scattered into a point from a volume ΔV can be calculated from measured SARs and TARs.[15] The dose at a point in a heterogeneous medium is then calculated by

$$D \;=\; D_A \left(\text{TAR}(d', 0) + \sum_V \left(e^{-\mu(a'-a)} \epsilon \frac{\Delta S}{\Delta V} e^{-\mu'(b'-b)} \Delta V \right) \right) \;\;, \tag{1.32}$$

where

D_A	is the dose in air at the calculation point,
d'	is the radiological depth of the calculation point,
$\text{TAR}(d', 0)$	is the zero-area TAR at radiological depth d',
μ	is the linear attenuation coefficient for primary photons,
a	is the pathlength to the scattering volume,
a'	is the radiological pathlength to the scattering volume,

ϵ	is the weight for the scattering site ($\propto$ relative electron density),
μ'	is the linear attenuation coefficient for scattered photons,
b	is the pathlength from scattering volume to calculation point,
b'	is the radiological pathlength from scattering vol. to calc. point.

The zero-area TAR term represents the dose contribution from primary radiation, while the second (summation) term represents the scatter contribution summed over the entire treatment volume. Along with E-TAR this system is a sophisticated and potentially accurate algorithm, but the volume integration requires a large amount of computation, and hence is suitable for implementation only on relatively powerful treatment planning computers. However, like all effective depth and TAR correction methods, it fails to model electrons ranging away from the interaction site.

1.4.4. Superposition and Convolution

The techniques discussed in the previous sections rely on experimentally obtained data, such as depth dose curves or TARs, and perform empirical corrections to these distributions in order to obtain dose in heterogeneous regions. The superposition/convolution technique is not really a *correction* technique at all — it seeks to model the physical processes *directly* from knowledge of only the beam characteristics. The method was first suggested by Dean[22] and developed by Mackie *et al.*,[23] Boyer and Mok,[24] and many others.

The first quantity needed in order to perform a superposition/convolution is the total energy released per unit mass, or *terma*. This is calculated by a ray-trace to determine radiological depth, coupled with a knowledge of attenuation coefficients of the medium. The second quantity required is an *energy deposition kernel,* also known as a dose spread array, differential pencil beam or point spread function. It represents the distribution of deposited dose caused by interactions at a single site, and is normally calculated using Monte Carlo techniques (see Section 1.4.5 and Chapter 2). The terma T and energy deposition kernel H can then be used to calculate the dose D deposited at a point (x, y, z) using convolution:

$$D(x,y,z) = \frac{1}{\rho(x,y,z)} \int_0^X \int_0^Y \int_0^Z \rho(x',y',z') \cdot T(x',y',z') \cdot H(x-x', y-y', z-z') \, \mathrm{d}x' \mathrm{d}y' \mathrm{d}z' \ , \tag{1.33}$$

where (x', y', z') is the interaction site and X, Y, and Z are the upper limits of the region of calculation. Equation 1.33 calculates the dose at a single point by summing dose scattered from every other point in the medium, which requires a very large computational effort. Assuming that the energy deposition kernel does not vary with position within the phantom enables Fourier transform techniques to be applied, speeding the process consid-

erably. However, the presence of heterogeneities and hardening of the beam spectrum with depth require that the energy deposition kernel be *scaled*, invalidating the Fourier transform technique. This real-space approach using a variant kernel is known as *superposition.*

As mentioned above, these methods — especially superposition — require extremely large amounts of computation. Another significant disadvantage is that the characteristics of the beam must be well known. Linear accelerator spectra are difficult to measure experimentally, but Mohan *et al.*[25] and others have had considerable success in simulating linear accelerator treatment heads using Monte Carlo techniques.

However, there are also several advantages to the convolution/superposition approach. It requires a minimum of experimental data and is conceptually simple and intuitive. The primary advantage is that *superposition/convolution models electronic disequilibrium*, which is necessary for accurate dose calculation in regions of heterogeneity when using high energy photon beams — especially for small field sizes and in low density regions such as air cavities and lung. It is this property which sets superposition/convolution (and Monte Carlo) apart from all previously discussed methods.

1.4.5. Monte Carlo Modelling

Monte Carlo modelling is the standard which all other dose calculation algorithms must seek to approach. Monte Carlo attempts to directly model the particle interactions which occur in photon and electron transport, and provided these are modelled adequately there is no theoretical limitation to the accuracy which this method can achieve. Only statistical fluctuations caused by the modelling of an insufficient number of particle histories can cause error in the dose distribution.

Monte Carlo techniques are so-called because they use random events to model the overall behaviour of a system. In Monte Carlo particle transport simulation an incident photon is modelled by first using a random number to select the distance to the site of first interaction, from a knowledge of the photon's mean free path in the medium. The type of interaction, resulting particles and their energies and scattering angles must then be randomly selected. All resulting particles must in turn be simulated until their energy drops below a predetermined cut-off level, in which case their energy is deposited locally. Photons undergo a small number of well-defined interactions, but charged particles (primarily electrons) that they produce may undergo many thousands of collisions, each of relatively small energy loss. It is simulation of these electron tracks which requires the bulk of the computational effort.

Many authors have written Monte Carlo particle transport simulation packages, but the ETRAN/ITS[26,27] and EGS4[28] codes are most commonly used in medical physics applications. These codes have been extensively benchmarked at linear accelerator energies, and provided they are applied correctly there is justifiable confidence in the results they produce.[29,30] Like superposition/convolution, Monte Carlo requires both an accurate knowledge of the incident photon beam and an extremely large amount of computational effort. Although this last factor makes Monte Carlo impracticable for routine treatment planning, the technique is still useful as a benchmark for other techniques, and also as a method of generating energy deposition kernels for the superposition/convolution approach.

1.4.6. Summary

In the previous sections a wide range of treatment planning algorithms have been discussed. Their properties are summarised in Table 1.2.

Algorithm	Method accounts for:					complexity/ speed
	photon path length	field size	inhomogeneity position	inhomogeneity shape	electronic equilibrium	
Effective attenuation coefficient	√	×	×	×	×	very simple/ very fast
Effective SSD	√	√	×	×	×	simple/fast
Ratio of TARs	√	√	×	×	×	very simple/ very fast
Power law TAR	√	√	√	×	×	simple/fast
E-TAR	√	√	√	√	×	complex/slow
Volume integration of dSARs	√	√	√	√	×	complex/slow
Convolution/ Superposition	√	√	√	√	√	simple/ very slow
Monte Carlo	√	√	√	√	√	very complex/ extremely slow

Table 1.2. Capabilities of various treatment planning algorithms (adapted from Cunningham[15]).

The table shows that the equivalent pathlength methods, which were originally developed for calculation by hand, are relatively poor at modelling dose distributions in heterogeneous media, although they are fast and easily implemented on a computer. TAR correction methods calculate dose near heterogeneities with varying degrees of success, but are more complicated to implement and require more powerful treatment planning computers. However, it should be noted that a method such as volume integration of dSARs is accurate when

modelling low energy beams such as that from cobalt-60. Only superposition/convolution and Monte Carlo modelling can account for electronic disequilibrium caused by high energy (linear accelerator) photon beams, but these methods are more computationally intensive than the other methods.

Of all the methods listed above, only Monte Carlo modelling has the capability of being arbitrarily accurate in calculating dose distribution. Although Monte Carlo is useful as a reference technique, present computer technology does not allow generation of Monte Carlo plans in just a few minutes, so the superposition/convolution option is a more realistic algorithm for routine clinical use. Provided the energy deposition kernels are scaled in a physically reasonable manner the accuracy of the superposition technique should closely approach that of Monte Carlo. The next two chapters discuss both the Monte Carlo and superposition/convolution techniques.

References

(1) International Commission on Radiation Units and Measurements, *Radiation Quantities and Units.* ICRU Report 33, Washington D.C. (1980).

(2) Johns H.E. and Cunningham J.R., *The Physics of Radiology (4th edition).* Charles C. Thomas, Springfield, Illinois (1983).

(3) Greening J.R., *Fundamentals of Radiation Dosimetry (2nd edition).* Medical Physics Handbook Number 15, Adam Hilger Ltd, Bristol (1985).

(4) Heitler W., *The Quantum Theory of Radiation.* Oxford University Press, New York (1954).

(5) Klevenhagen S.C., *Physics of Electron Beam Therapy.* Medical Physics Handbook Number 13, Adam Hilger Ltd, Bristol (1985).

(6) Sternheimer R.M. and Peierls R.F., General expression for the density effect for the ionization loss of charged particles. *Phys. Rev. B* (1971), **3** 3681.

(7) Berger M.J. and Seltzer S.M., *Stopping Powers and Ranges of Electrons and Positrons.* NBSIR 82-2550, National Bureau of Standards, Washington D.C. (1982).

(8) Ahnesjö A., Andreo P. and Brahme A., Calculation and application of point spread functions for treatment planning with high energy photon beams. *Acta Oncologica* (1987), **26** 49.

(9) Hendee W.R., *Radiation Therapy Physics.* Year Book Medical Publishers, Chicago (1981).

(10) Johns H., Bates L., and Watson T., 1000 Curie cobalt units for radiation therapy: I. The Saskatchewan cobalt-60 unit. *Br. J. Radiol.* (1952), **25** 296.

(11) Harper N.R., *The Design, Construction, and Use of a Thin Window Ionization Chamber.* University of Waikato M.Sc. Thesis (1990).

(12) British Institute of Radiology Supplement 17,*Central Axis Depth Dose Data for Use in Radiotherapy.* British Institute of Radiology, London (1983).

(13) Mould R.F., *Radiotherapy Treatment Planning (2nd edition).* Medical Physics Handbook Number 14, Adam Hilger Ltd, Bristol (1985).

(14) International Commission on Radiation Units and Measurements, *Determination of Absorbed Dose in a Patient Irradiated by Beams of X or Gamma Rays in Radiotherapy Procedures.* ICRU Report 24, Washington D.C. (1976).

(15) Cunningham J.R., Tissue inhomogeneity corrections in photon-beam treatment planning. In *Progress in Medical Radiation Physics, Volume 1, p103,* edited by Orton C.G. Plenum Press, New York (1982).

(16) Mackie T.R., El-Khatib E., Battista J.J., Scrimger J., Van Dyk J., and Cunningham J.R., Lung dose corrections for 6 and 10 MV x-rays. *Med. Phys.* (1985), **12** 327.

(17) Hoban P.W., Murray D.C., Metcalfe P.E. and Round W.H., Superposition dose calculation in lung for 10MV photons. *Australas. Phys. Eng. Sci. Med.* (1990), **13 (2)** 81.

(18) Batho H.F., Lung corrections in cobalt 60 beam therapy. *J. Can. Assoc. Radiol.* (1964), **15** 79.

(19) Sontag M.R. and Cunningham J.R., Corrections to absorbed dose calculations for tissue inhomogeneities. *Med. Phys.* (1977), **4** 431.

(20) Sontag M.R. and Cunningham J.R., The equivalent tissue-air ratio method for making absorbed dose calculations in a heterogenous medium. *Radiology* (1978), **129 (3)** 787.

(21) O'Connor J.E., The variation of scattered x-rays with density in an irradiated body. *Phys. Med. Biol.* (1957), **1** 352.

(22) Dean R.D., A scattering kernel for use in true three-dimensional dose calculations. *Med. Phys.* (1980), **7** 429.

(23) Mackie T.R., Scrimger J.W., and Battista J.J., A convolution method of calculating dose for 15-MV x rays. *Med. Phys.* (1985), **12** 188.

(24) Boyer A.L. and Mok E.C., A photon dose distribution model employing convolution calculations. *Med. Phys.* (1985), **12** 169.

(25) Mohan R., Chui C. and Lidofsky L., Energy and angular distributions of photons from medical linear accelerators. *Med. Phys.* (1985), **12** 592.

(26) Seltzer S.M., An overview of ETRAN Monte Carlo methods. In *Monte Carlo Transport of Electrons and Photons, p153,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(27) Halbleib J., Structure and operation of the ITS Code System. In *Monte Carlo Transport of Electrons and Photons, p249,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(28) Nelson W.R., Hirayama H. and Rogers D.W.O., *The EGS4 Code System.* Stanford Linear Accelerator Center Report SLAC-265 (1985).

(29) Berger M.J., ETRAN — experimental benchmarks. In *Monte Carlo Transport of Electrons and Photons, p183,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(30) Rogers D.W.O. and Bielajew A.F., Experimental benchmarks of EGS. In *Monte Carlo Transport of Electrons and Photons, p~~249~~ 301,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Chapter 2

Monte Carlo Simulation

2.1. Introduction

There is a wide range of problems in physics and mathematics which can be successfully simulated by the sampling of random variables. These methods are known collectively as *Monte Carlo* techniques, and are particularly well suited to the simulation of particle transport, since this element of chance is implicit in the behaviour of the particles themselves. Full or *analogue* Monte Carlo modelling is a direct simulation of the physical processes which occur in particle transport, and hence has the capability of giving results as accurate as statistical errors will allow.

Many scientists have reservations about the Monte Carlo procedure on the grounds that it is empirical rather than rigorously analytical. However, quite apart from the impossibility of treating all but the very simplest cases analytically, an equally valid view is that Monte Carlo simulation of radiation transport, with correctly determined physical simulation data, is a method whose results all other correct physical theories should approach.[1] Monte Carlo simulation has provided the physically correct answer in many cases where all other workable methods of calculation have been in error. Monte Carlo also allows more flexibility than is possible in physical situations. For example, in calculating the dose distribution in a phantom, primary and scattered particles can be scored separately.

Monte Carlo techniques first began to be used for particle transport simulation in the mid 1950s.[2–4] Development of the method by Kahn,[5] Berger[6] and others resulted in the Monte Carlo method outlined in reviews by Turner[7] and Raeside,[8] but here it is sufficient to say that two widely-used Monte Carlo code systems have evolved. The ETRAN[9] (**E**lectron **TRAN**sport) system was developed at the National Bureau of Standards (NBS) by Berger and Seltzer, leading to the Integrated TIGER Series[10] (ITS) codes developed at Sandia National Laboratories. The other system is the EGS4 (**E**lectron **G**amma **S**hower) code,[11] developed at the Stanford Linear Accelerator Center by Nelson, Hirayama and Rogers. These two codes are essentially similar, although there are significant differences in the way

An abbreviated form of this chapter appears in *Australas. Phys. Eng. Sci. Med.* (1990) **13** (**3**) 132–147, under the title "Using EGS4 Monte Carlo in Medical Radiation Physics."

electron transport is treated. However, the ease with which the EGS4 system can be used has resulted in it being the most popular Monte Carlo system. The intention here is to briefly outline the principles behind the Monte Carlo technique in general, then elaborate on particular aspects of the EGS4 system and its application in the medical radiation physics field.

2.2. Principles of Monte Carlo Particle Transport

2.2.1. Photon Transport

Consider a photon beam incident on the surface of a phantom or patient. Using the Monte Carlo technique this beam is modelled as a large but finite number of incident photons. The mean distance which a photon of a given energy travels in a given medium is known as the *mean free path* λ, given by[11]

$$\lambda = \frac{M}{N_a \rho \sigma_t} \quad , \tag{2.1}$$

where

M = molecular weight of the medium,

N_a = Avogadro's number,

ρ = density of the medium,

σ_t = total cross-section per molecule

(λ is also equivalent to $1/\Sigma_t$ where Σ_t is the *macroscopic cross-section* of the medium). If ζ is a random variable uniformly distributed between 0 and 1 then the number of mean free paths to the next interaction N_λ can be sampled by

$$N_\lambda = -\ln \zeta \quad , \tag{2.2}$$

using the inversion sampling method described in Section 2.2.3. The mean free path is dependent upon the medium, so that in the general case the number of mean free paths N_λ between two points x_0 and x_1 must be formed from the integral

$$N_\lambda = \int_{x_0}^{x_1} \frac{dx}{\lambda(x)} \quad . \tag{2.3}$$

In the EGS4 system a medium is modelled as a finite number of regions, each of which is homogeneous, so Equation 2.3 reduces to a sum over all regions intersected by the path of the particle, allowing for different values of λ in those regions.

Having been transported to the interaction site, the photon may then interact in one of a number of ways, including:

(a) **Compton scattering,** in which a photon of energy $h\nu$ collides with an effectively "free" electron, transferring momentum to the electron as it recoils.

(b) **The photoelectric effect,** in which a photon gives up all its energy $h\nu$ to an electron, which is ejected from the atom with the corresponding energy less the binding energy of the electron. If an inner shell electron is knocked out then *characteristic radiation* may also be emitted when an outer shell electron drops into the inner shell vacancy.

(c) **Pair production,** in which a photon with energy greater than the mass of two electrons ($2m_ec^2 = 1.02$MeV) interacts with the nucleus to produce an electron-positron pair. The positron may later *annihilate* with an electron to produce two 0.511 MeV photons.

There are also other ways in which the photon can interact, including *Thomson scattering* (important only at very low energies), *coherent (Rayleigh) scattering* (in which the photon is scattered elastically off a tightly bound electron, causing the entire atom to recoil), the *nuclear photoeffect* (in which a photon typically of energy 12–24 MeV interacts with a nucleus), and *triplet production* (similar to pair production except that interaction occurs with an electron, which recoils). The relative probabilities or *branching ratios* for these various types of interaction at x-ray photon energies are well known, so once again a random variable can be used to determine the type of interaction. For example, if there are n types of interaction possible, where the jth type of interaction has cross-section σ_j and the total cross-section is σ_t, then we can generate n divisions on the unit interval such that[11]

$$F(i) = \frac{\sum_{j=1}^{i} \sigma_j}{\sigma_t} \quad . \tag{2.4}$$

By sampling a uniform random variable on the interval [0,1) the type of interaction i can then be determined by finding $F(i)$ such that

$$F(i-1) < \zeta < F(i) \quad . \tag{2.5}$$

Finally it remains to determine any other necessary particle parameters. These might include scattering angles and product particle energies. Again this is done using random variables. The angles are transformed from the particle coordinate system to the original coordinate system, and the histories of all product particles are followed in turn.

2.2.2. Electron Transport

In order to determine energy deposited in a medium it is necessary to accurately model the transport of *electrons*. Electrons may be incident on the medium (as with an electron beam

produced by a linear accelerator) or generated by the interaction of photons with matter. They then interact in one of three principal ways:

(a) **Collisional energy loss,** occurring due to *Coulomb interactions* (Møller scattering), in which the electron causes excitation or ionization of a bound electron. The excited electron then deposits its extra energy in the medium as heat. Occasionally an ejected electron has sufficient energy (approximately 100 eV) to be considered a distinct ionizing particle or *delta ray.*

(b) **Radiative energy loss,** which does not deposit energy directly but instead generates further photons. In the *bremsstrahlung* process an electron (or positron) is scattered by a virtual photon from the nucleus and a free photon (which may then go on to interact) is created. Classically speaking the electron accelerates (changes direction) due to the electrostatic force from the nucleus and in so doing emits radiation. The electron continues on with reduced energy. *Characteristic* radiation may also occur, when an inner shell electron is ejected and replaced by an outer shell electron, emitting radiation of energy characteristic to the atom.

(c) **Elastic nuclear (Rutherford) scattering,** in which the electron is deflected due to the interaction of the electron's charge with the nuclear Coulomb field. Since the nucleus has vastly more mass than the electron, almost no kinetic energy is transferred to the nucleus and the scattering is therefore elastic.

Electrons may also undergo *annihilation*, in which an electron and a positron annihilate to produce two photons of energy 0.511 MeV; *Bhabha scattering*, in which an electron and positron interact without annihilation; *electron-electron bremsstrahlung* (unimportant at radiotherapy energies); and *Čerenkov emission* (also unimportant).

In general, the number of interactions which an electron undergoes is large (many thousand in medical radiation physics applications). To avoid long simulation times it is therefore necessary to treat many interactions together by using *multiple scattering* instead of single scattering, and the concept of *stopping power* instead of discrete energy loss. Stopping power can be further divided into *collision* and *radiation* stopping powers, representing the contributions of collision and radiation losses, respectively. If only those interactions producing products below a certain energy threshold are considered then *restricted* stopping power is used. Figure 2.1 shows the path of an electron being modelled as a sequence of straight lines, where the scattering angles θ_i are sampled using a multiple scattering theory and energy loss is calculated using *total* stopping power.

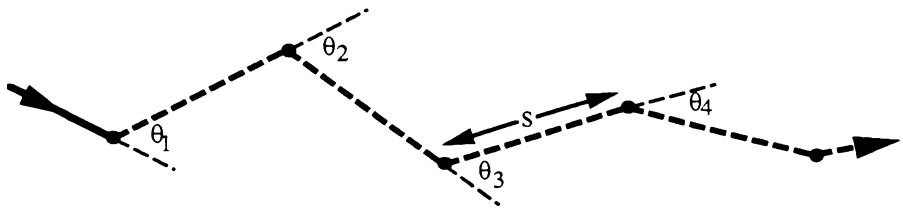

Figure 2.1. Simulation of electron transport using multiple scattering. Electrons are transported rectilinearly by step length s (dashed lines) and energy is deposited at sites indicated by solid dots. A multiple scattering theory is used to determine the scattering angles θ_i. Note that the path simulated by Monte Carlo is one of an infinite number of physically possible electron paths.

In the simplest Monte Carlo approach the total stopping power is used to determine the energy lost along an electron step, with a multiple scattering theory (such as the Gaussian, Goudsmit-Saunderson or Molière theory) used to determine the scattering angle.[12] However, these *Class I* schemes assume a *continuous slowing down approximation* (CSDA), in which the energy losses of the electron are continuous. In reality the energy loss per unit distance varies, with the stopping power representing the *average* energy loss. These fluctuations are known as *energy loss straggling*. In Class I schemes allowance for this must be made by building in an analytical energy loss straggling correction.

An alternative is a *Class II* or *Mixed* procedure, in which electron energy losses are grouped into two distinct categories:

(a) Energy losses below a certain threshold are treated using multiple scattering and *restricted* or *continuous* stopping power, which is the sum of the restricted radiation stopping power and restricted collision stopping power.

(b) Energy losses above that threshold are modelled as discrete collisions in a similar way to the modelling of photon interactions.

The random nature of the discrete (category b) events automatically introduces energy loss straggling into the simulation. However, the cut-off energy — above which discrete energy losses are modelled — must be sufficiently low to ensure an adequate amount of energy loss straggling is included. A high cut-off energy results in a faster simulation, but can give erroneous results if too much straggling is excluded by the modelling of continuous (category a) losses.

Another correction which may need to be built into the simulation is the *pathlength correction* (PLC) factor. This arises because the distance which the electron has travelled between electron steps is greater than the straight line distance between the beginning and end of the step, due to curvature of the electron track. The effect of this can be reduced by making

the electron steps very small, but for large step sizes the error can become significant unless a PLC factor is included.

Figure 2.2 (adapted from Seltzer *et al.*[13]) illustrates the concepts discussed above. It shows energy deposited with depth for a broad beam of electrons incident on a homogeneous medium. The curve labelled "unscattered CSDA" shows how energy would be deposited if continuous slowing down was assumed and no scattering was modelled. Note that since delta rays are not modelled, the electrons all reach their residual energy at exactly the same depth, leading to the Bragg peak at the end of the electron range. The curve labelled "multiple-scattered CSDA" shows that when multiple scattering is simulated the electrons travel obliquely and energy begins to be deposited at shallower depths. The bold curve labelled "straggling" shows what happens when energy loss straggling is also taken into account (by using a Class II scheme, for example). The region of low energy deposition near the surface is due to the creation of delta rays, which deposit energy downstream from the site of interaction. The bremsstrahlung contribution is also shown.

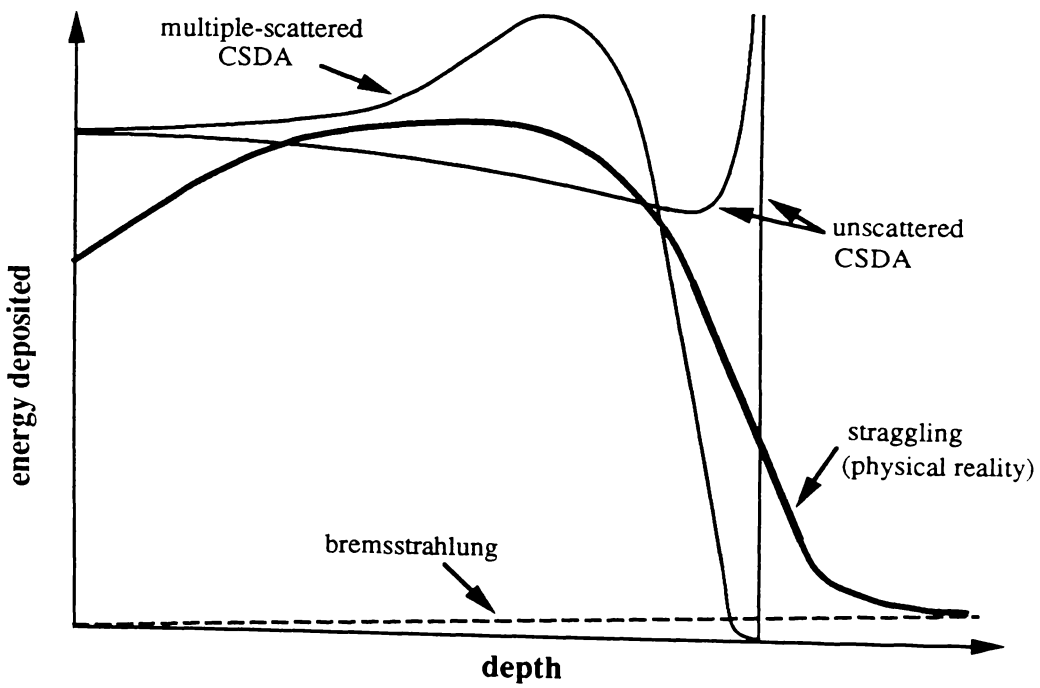

Figure 2.2. Energy deposition curves for a broad parallel beam of electrons incident on a medium (adapted from Seltzer *et al.*[13]) using various modelling schemes (see text).

In summary, photons undergo a relatively small number of well-defined interactions which may give rise to free electrons. These electrons then traverse the medium depositing energy by means of a far larger number of interactions, involving small average energy loss. The modelling of these electrons provides the greatest challenge for Monte Carlo theorists.

2.2.3. Sampling

The Monte Carlo technique relies upon the correct selection of particle parameters from associated probability distribution functions (PDFs) using *sampling techniques.*[14] For example, in Section 2.2.1 it was mentioned that the distance which a photon travels before the next interaction must be sampled using a random variable. The method which EGS4 and other codes such as ETRAN use to obtain this value is known as the *inversion method.* For photon transport the normalised probability distribution function $f(x)$ for the distance x to the next interaction is

$$f(x) = \mu e^{-\mu x} \quad , \tag{2.6}$$

where μ is the linear attenuation coefficient ($\lambda = 1/\mu$). From this the *cumulative probability distribution* $F(x)$ can be formed, such that

$$F(x) \;=\; \int_0^x \mu e^{-\mu x'} dx' \;=\; 1 - e^{-\mu x} \quad . \tag{2.7}$$

From Figure 2.3 it can be seen that $dF(x)$ is proportional to the *slope* of the cumulative probability distribution, and is therefore proportional to the original probability distribution $f(x)$. By sampling $F(x)$ with a random variable R uniform on the interval [0,1):

$$R = 1 - e^{-\mu x} \quad , \tag{2.8}$$

we can therefore solve for distance x:

$$x = -\frac{1}{\mu} \ln(1 - R) \quad . \tag{2.9}$$

Noting that $(1 - R)$ is distributed in the same way as R and $\lambda = 1/\mu$, the number of mean free paths N_λ to the next interaction can be sampled most simply by the expression

$$N_\lambda = -\ln R \quad . \tag{2.10}$$

The EGS4 implementation of this expression is discussed in Section 2.3.3.

In some cases the probability distribution function $f(x)$ cannot be integrated, or $x = F^{-1}(R)$ cannot be solved. In these cases *rejection sampling* can be used, providing the probability distribution function $f(x)$ is bounded between $x = a$ and $x = b$ with a maximum value M, as shown in Figure 2.4. Using two random numbers to sample the interval (a, b) on the x-axis and $(0, 1)$ on the y-axis, a point on the normalised graph in Figure 2.4 can be randomly selected. If it lies outside the probability distribution function then it is rejected and a new point is generated, otherwise the value of x is accepted.

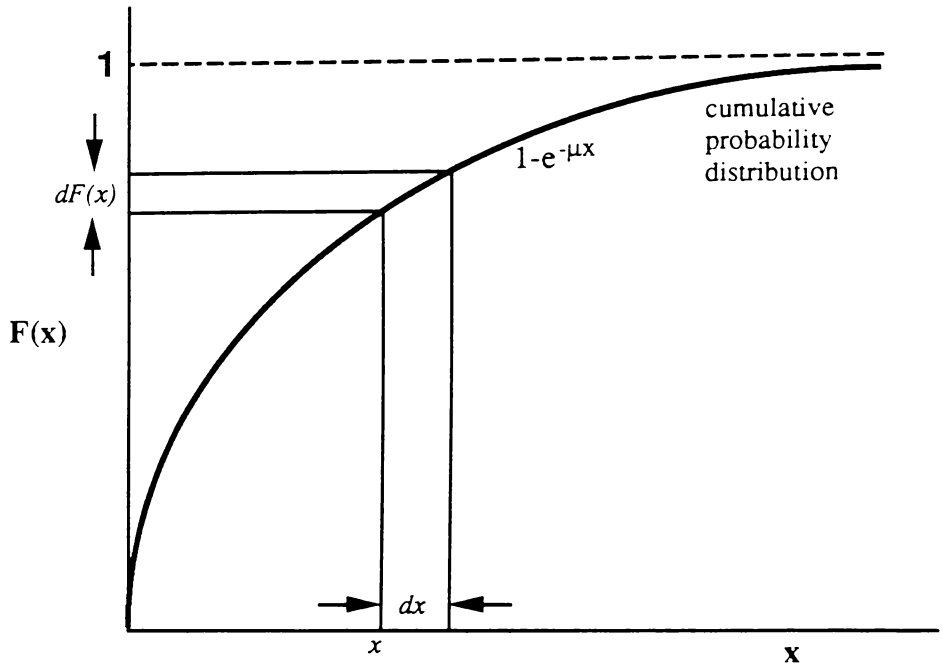

Figure 2.3. Inversion sampling. The normalised probability distribution $f(x)$ is first used to form the cumulative probability distribution $F(x)$ (illustrated). $dF(x)$ is proportional to the slope of $F(x)$ at x, and hence is proportional to $f(x)$ at x. Sampling $F(x)$ randomly with R and solving for x gives a correctly sampled value for x.

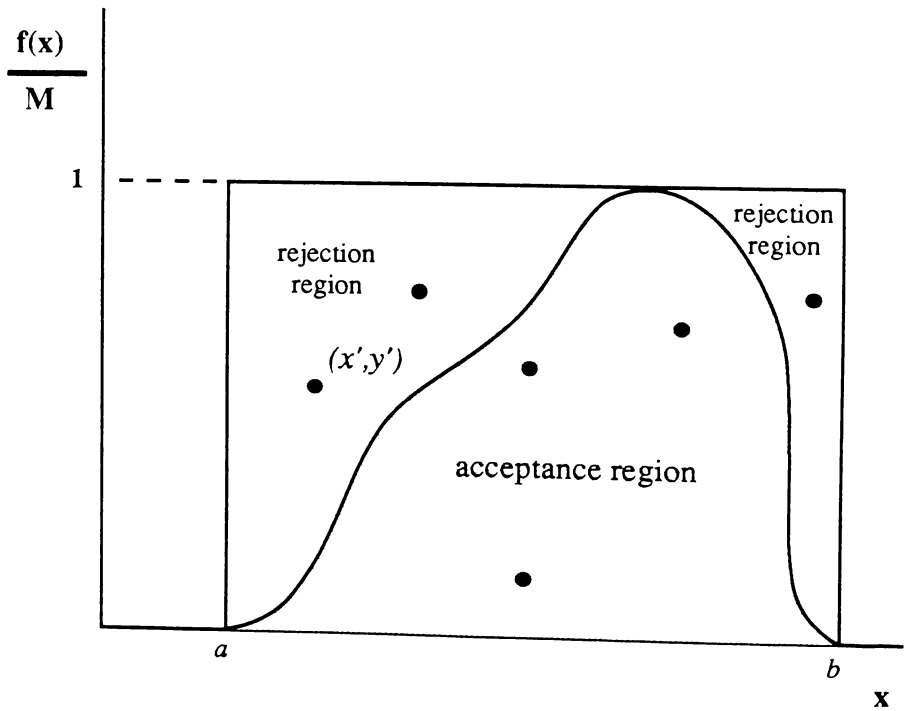

Figure 2.4. Rejection sampling. The probability distribution function (PDF) is bounded by a and b and has a maximum value of M. The PDF is normalised to a maximum value of 1, then to sample the distribution two random variables are chosen to select the coordinates of a point (x', y'). If this point lies within the PDF (shaded region) then x' is accepted: otherwise it is rejected and sampling is repeated.

A combination of the two methods may also be employed when the ratio of rejected to accepted values is very high using the rejection method. If a probability distribution function $f(x)$ can be expressed as the product of two functions $g(x)$ and $h(x)$, it may be possible to perform inversion sampling on $g(x)$ to determine a value x', followed by rejection sampling on $h(x')$. Carefully choosing $g(x)$ and $h(x)$ may dramatically increase the acceptance probability and hence speed up the simulation.

2.3. The EGS4 Monte Carlo System

2.3.1. Overview

The EGS code system was developed by Ford and Nelson[15] during the period 1972–1978, based on work by Nagel at the University of Bonn. Known as EGS3, its purpose was to simulate particles of energies up to a few thousand GeV. In an effort to adapt the code to low energy physics problems, particularly in the field of medical physics, Nelson (SLAC), Hirayama (KEK) and Rogers (NRC) extended the code to what is now known as EGS4.[11] Several hundred EGS4 distribution tapes are distributed each year, the majority to medical physics institutions. Nelson and Rogers[16] provide a more extensive review of EGS4 than is possible in this section.

The general structure of the EGS4 Monte Carlo System is illustrated in Figure 2.5. It consists of two distinct components: the PEGS4 preprocessor and the EGS4 simulation code. PEGS4 creates data sets for each element, compound or mixture used in the simulation, which are read in by the `HATCH` routine of the EGS4 code itself. The user is responsible for writing three routines, `MAIN`, `HOWFAR`, and `AUSGAB`, which form what is known as the *user code*. `MAIN` performs any initialisation necessary for the simulation, including the media to be used, particle parameters and cut-off energies and geometry of the simulation. Having called `HATCH` to obtain media data sets, `MAIN` then repeatedly calls `SHOWER`, once for each incident particle. `SHOWER` and its various subroutines simulate the particle and its offspring until they leave the region of interest, reach the end of their track or are discarded. Although these routines are never called directly by the user, they themselves frequently call two *user-written* subroutines, `HOWFAR` and `AUSGAB`. `HOWFAR` is used to determine the distance to the next medium boundary along the current path, and `AUSGAB` is called to score energy and any other parameters of interest.

2.3.2. Using PEGS4 to Create Material Data Sets

Before the EGS4 code can be run a preprocessor known as PEGS4[11] must be used to generate data sets for each material used in the simulation. PEGS4 begins by computing

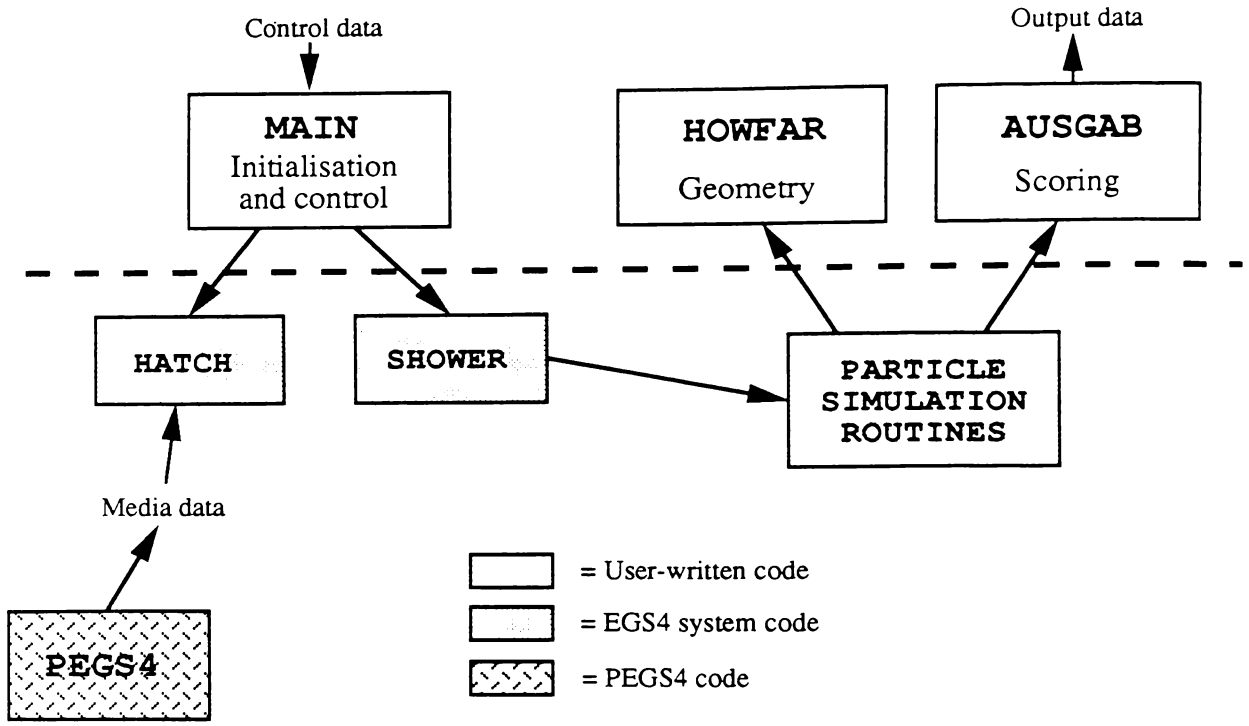

Figure 2.5. The EGS4 Monte Carlo System (adapted from Nelson *et al.*[11]). The user code (above dotted line) consists of subroutine **MAIN**, for general initialisation and control, **HOWFAR**, for defining the geometry of the simulation, and **AUSGAB**, for scoring dose and any other parameters of interest. The media data is obtained from PEGS4 by calling subroutine **HATCH**.

physical and mathematical constants, then reading in pair production, photoelectric and Rayleigh data from two files supplied with the EGS4 system. An input file is then used to specify the medium, cut-off energies and output options required. Tables of differential cross-sections for all types of interaction are produced, along with functions such as gamma-ray mean free path (GMFP) and electron stopping powers. These tables are then used as input to the EGS4 code itself, but may also be used independently of EGS4 if a user wishes to calculate these parameters for some other application.

As an example of the use of PEGS4, consider the following input file, used to create a data set for water between 10 keV and 15 MeV:

```
COMP
 &INP NE=2,RHO=1.0,PZ=2,1 &END
WATER (< 15 MEV)          H2O
H  O
ENER
 &INP AE=0.521,UE=15.511,AP=0.010,UP=15.0 &END
PWLF
 &INP &END
DECK
 &INP &END
```

The first input line (COMP) specifies that the medium is a compound consisting of two

elements whose density is 1.0 g cm^{-3}. The relative numbers of H and O atoms in the compound (H_2O) are `PZ=2,1`. The third line specifies the identifier to be used in EGS4 simulations (`WATER (< 15 MEV)`) while `H2O` specifies the identifier used by PEGS4 in accessing its Sternheimer density effect table. The symbols `H` and `O` specify the elements making up the compound, enabling PEGS4 to look up its table of atomic weights. The `ENER` input line then specifies the lower (`AE,AP`) and upper (`UE,UP`) cut-off energies for electron and photon transport. In this case data between 10 keV and 15 MeV kinetic energy for both electrons and photons is created — note that total energy is specified, so 0.511 MeV must be added to electron kinetic energy. The `PWLF` input line specifies that a piecewise linear fit of the functions is to be used, while `DECK` instructs PEGS4 to actually write the medium data set to a file.

Other options are also available in PEGS4. The `ELEM` and `MIXT` options can be used to specify elements and mixtures respectively, while adding a `TEST` input line will instruct PEGS4 to print out plots of all the functions that the `PWLF` option fits. Plots of any PEGS4 created function can be produced using the `PLTI` or `PLTN` options, or the `CALL` option can be used to precisely evaluate any function at any specified energy (as opposed to the piecewise linear fitted output).

2.3.3. The Mortran Preprocessor

The EGS4 system is written in a structured language called *Mortran3*,[11,17] developed at SLAC by Cook and Shustek. It is essentially a FORTRAN-type language with extensions, and compiles to either FORTRAN IV or FORTRAN 77 (two versions of the Mortran compiler are provided on the EGS4 distribution tape). Although it is possible to program EGS4 entirely in FORTRAN, use of Mortran results in very much shorter and more readable code.

The following list briefly describes the main features of Mortran:

(a) **Relaxation of FORTRAN formatting requirements.** Mortran statements are free-field — that is, column boundaries and card boundaries are ignored. Labels can be of (almost) arbitrary length. Comments can be inserted at any place in the program, including across lines and within block structures.

(b) **Abbreviation of FORTRAN syntax.** Certain FORTRAN commands can be abbreviated in Mortran by using special constructs. For example, a FORTRAN `WRITE` statement directed to standard output would normally require an associated label pointing to a format list. In Mortran this can be abbreviated to

`WRITE` *i/o list*; (*format list*);

Another commonly used abbreviation is *multiple assignment*, which performs several variable assignments in one Mortran statement. For example, the statement

```
/A,B,C/ = 0;
```

assigns the value zero to each of the variables `A`, `B` and `C`.

(c) **Nested block structures and conditionals.** Rather than use the unwieldy FORTRAN system, the programmer can define blocks using square brackets, as in this array initialisation:

```
DO I=1,10 [
  DO J=1,10 [
    DOSE(I,J) = 0.0;]]
```

which is much more concise than the equivalent FORTRAN statement. A large set of conditional and iteration statements are available which can also be nested, including `IF`, `ELSE`, `ELSEIF`, `WHILE`, `UNTIL`, `LOOP`, `FOR-BY-TO`, and `DO`.

(d) **Control cards.** Also known as *processor control directives*, these allow the user to control input files and output format. For example, `%F` and `%M` allow switching between FORTRAN and Mortran modes, `%L` and `%N` turn the Mortran listing file on and off, and `%In` indents the Mortran listing `n` places. Mortran input can be switched to another file (FORTRAN unit `n`) by the command `%Un`, performing a function similar to the FORTRAN statement `INCLUDE`. The only mandatory control card in an EGS4 user code is the `%%` card required to signal the end of Mortran input.

(e) **Macro replacement.** This is by far the most important feature of Mortran, the simplest form of which is *string replacement:*

`REPLACE` {*pattern*} `WITH` {*replacement*} .

This statement searches throughout the entire Mortran code, replacing any occurrence of the string *pattern* with the string *replacement*. For example, the maximum number of different regions in the simulation is normally represented by the string `$MXREG`,which can then be used in any statement requiring that constant, such as the array declaration `COMMON/ECUT($MXREG)`. If the string replacement

```
REPLACE {$MXREG} WITH {20}
```

is placed in the user code then Mortran will search the entire EGS4 code, replacing occurrences of `$MXREG` with the value `20` (allowing up to 20 regions to be simulated). A more complicated example of this is given in the following code extract, used to implement inversion sampling of the distance to next photon interaction, as described

by Equation 2.10:

```
REPLACE {$SELECT-PHOTON-MFP;} WITH
{$RANDOMSET RNN035; IF(RNN035=0.0)[RNN035=1.E-30;]
DPMFP=-ALOG(RNN035);} .
```

In this example the string or *macro* **$SELECT-PHOTON-MFP** is replaced by code to sample **DPMFP**, the number of mean free paths to the next interaction. **$RANDOMSET** is the random number generation macro, which is itself replaced using a more sophisticated replacement involving *parameters* (see Section 2.3.4). Note that this code also ensures that the random number is non-zero, so that the **ALOG** function will work correctly. EGS4 comes supplied with a set of *default* macros, defined in a similar manner to the above example. However, any one of these macros can be replaced or overridden by user-defined macros, so that features of the EGS4 system can be changed without explicitly altering the EGS4 code (only the user code is modified). It is this capability more than any other which makes Mortran so useful.

(f) **Conditional compilation.** Use of the **GENERATE** and **NOGENERATE** statements allow sections of code to be omitted if desired. For example, in an EGS4 user code developed by the author, if the following two lines are placed near the beginning of the program

```
REPLACE {$VAX;} WITH {NOGENERATE;}
REPLACE {$TRANSPUTER;} WITH {GENERATE;}
```

then the Mortran code will be compiled into FORTRAN code suitable for running on transputers[18] (fast RISC processors designed for performing computations in parallel[19]). Throughout the program any occurrences of **$VAX** will be replaced by **NOGENERATE** and hence not be compiled, whereas transputer code will be generated instead. If VAX code was to be generated, the position of the **GENERATE** and **NOGENERATE** commands would be swapped. The following is a sample of machine dependent commands which are found in the body of the code:

```
"VAX commands"
$VAX;
REPLACE {$MACHINE} WITH {'VAX'}
ENDGENERATE;

"TRANSPUTER commands"
$TRANSPUTER;
REPLACE {$MACHINE} WITH {'TRANSPUTER'}
REPLACE {CALL CPUTIME(#);} WITH {CALL ICLOCK({P1});{P1}={P1}*100;}
REPLACE {EXIT;} WITH {STOP;}
REPLACE {$RANDOMSET#;} WITH {$FIBONACCI-GENERATOR{P1};}
ENDGENERATE; .
```

$MACHINE is used in output statements to identify the host system. **CPUTIME** is a routine supplied with the EGS4 system which returns the elapsed CPU time in hundredths

of a second, while the equivalent in 3L Parallel FORTRAN[20] is the function `ICLOCK`, returning the elapsed time in seconds. This statement also illustrates use of the Mortran `REPLACE` command using *parameters*. Every occurrence of `CALL CPUTIME(#)` — where # is any string representing the name of a variable — is replaced with a call to `ICLOCK`, where `{P1}` represents that same variable. The last two statements replace the FORTRAN command `EXIT` with the equivalent 3L FORTRAN command `STOP`, and replace the default multiplicative congruential random number generator with a lagged-Fibonacci generator (see Section 2.3.4).

2.3.4. Random Number Generation

In order to perform a Monte Carlo simulation a supply of random numbers is required. However, due to the difficulties in generating truly random numbers, *pseudorandom number generators* are often used in computer simulations.[21,22] The most commonly used random number generators (RNGs) of this type are based on the *multiplicative congruential* or *power residue* method, where a pseudorandom number r_i is generated from the previous pseudorandom number r_{i-1} by using modulo arithmetic such that

$$r_i \equiv k r_{i-1} (\mathrm{mod}\ m) \quad , \tag{2.11}$$

where k is a constant and r_0 is the first element (seed) of the series. In using the EGS4 system, a RNG must be called every time the `$RANDOMSET` instruction is encountered. This instruction can be replaced by the RNG of the user's choice, but when running EGS4 on DEC VAX machines the RNG used is usually a multiplicative congruential one of the following form:

```
REPLACE {$RANDOMSET#;} WITH
{IXX=IXX*663608941;{P1}=0.5+IXX*0.23283064E-09;}    .
```

Once again the parameter feature of the `REPLACE` statement is used to replace every occurrence of `$RANDOMSET#;` — where # is any string representing the name of the pseudorandom variable — with the in-line pseudo RNG, where `{P1}` represents that same variable. Note that there is no explicit modulo arithmetic: this is achieved by instructing the compiler to ignore integer overflow during compilation. The value `663608941` is chosen for m so that the resulting pseudorandom number sequence will have a very long cycle on 32-bit computers such as VAX systems. The second part of the pseudo RNG simply normalises the resulting number to a real value between 0 and 1.

One disadvantage of this pseudo RNG is that the sequence length may not be enough for very large simulations (more than 10^8 random numbers). A repeating sequence will at best

give results which show poor statistics, and at worst give incorrect results. To avoid this a different pseudo RNG is chosen when performing simulations on the transputer network at Waikato University,[19] using a lagged-Fibonacci random number generator[21] as suggested by Duane.[23] This method uses 97 seeds and has an extremely long period. It also avoids the need to switch off overflow checking, but since it is a more complicated algorithm the simulation runs slightly slower than when using the standard generator.

2.3.5. EGS4 Electron Transport and PRESTA

As was mentioned in Section 2.2.2, the transport of electrons involves simulating many collisions, the majority of which involve small energy loss. EGS4 implements a Class II scheme, where small collisions are grouped together and large collisions are treated individually using differential cross-sections. The user can to some extent control the way in which this scheme is implemented by means of the five parameters in Table 2.1.

Parameter	*Significance*
AE	low energy threshold for discrete electron **collision** losses (delta-ray production)
AP	low energy threshold for discrete electron **radiation** losses (bremsstrahlung)
ECUT	**charged particle** low energy cut-off (energy deposited locally below this value)
PCUT	**photon** low energy cut-off (energy deposited locally below this value)
ESTEPE	fractional energy loss per electron step

Table 2.1. Major user-selectable parameters in EGS4 particle transport.

In EGS4 the threshold for collision losses (delta ray production) is represented by the variable **AE**, while for radiation losses (bremsstrahlung production) it is represented by **AP**. The values of these cut-offs are set when the PEGS4 data set is created. Choice of these thresholds is an important aspect of an EGS4 simulation. If they are set too low a needlessly large number of low-energy but discrete interactions will be simulated, resulting in a long simulation time. If they are set too high then a large proportion of the energy loss straggling discussed in Section 2.2.2 will be excluded, since no energy loss straggling correction is made for continuous interactions. Some simulations, such as the calculation of depth dose curves, are relatively insensitive to the choice of **AE** and **AP**, while others, such as calculation of the energy spectrum emerging from thin absorbers,[24] are much more sensitive. For most simulations a choice of 10 keV for **AP** and 521 keV (511 keV rest mass energy + 10 keV kinetic energy) for **AE** is satisfactory.

Having chosen **AE** and **AP** it is necessary to choose the energies below which electrons and photons will no longer be simulated, with their energy being deposited locally. Normally the electron cut-off **ECUT** is chosen so that the range of an electron of energy **ECUT** is a small proportion (say one third) of the minimum dimension of a scoring region. Thus the only energy capable of being deposited very far from the scoring region is that carried by bremsstrahlung generated from the electron when it has energy below **ECUT**, which in most cases is a small proportion of the electron energy. **PCUT**, the cut-off energy for photons, is sometimes chosen to be less than **ECUT** since the range of photons is much larger. The values of **ECUT** and **PCUT** are also set when the PEGS4 data set is created, although their values may be increased when the simulation is run. A separate set of values is maintained for each geometrical region in the simulation.

The final parameter in Table 2.1 is **ESTEPE**. It represents the maximum fraction of energy lost during each electron step due to continuous energy losses (those below the **AE** and **AP** thresholds). In default EGS4, **ESTEPE** represents a fraction of the *initial* energy, but the macro **FIXTMX**[25] modifies **ESTEPE** to represent a fraction of the *current* particle energy, which results in a more efficient and accurate simulation. The larger the value of **ESTEPE** the fewer steps will be taken per electron, and the faster the simulation will run. However, too large a value of **ESTEPE** can cause the electron to pass straight through a region without being simulated, so **ESTEPE** must be chosen carefully when simulating electrons passing through thin plates.

Although choosing a large **ESTEPE** will result in a faster simulation, the curved path length of the electron becomes significantly greater than the simulated (straight line) step length. This results in an underestimation of the electron path length per step, so that the electron is transported further than happens in reality. To correct for this EGS4 uses Fermi-Eyges transport theory to build in a *pathlength correction factor* (PLC).[11] If t is the average true pathlength for an electron step and s is the straight line step length then the PLC is expressed as

$$\mathrm{PLC} = \frac{(t-s)}{s} \quad . \tag{2.12}$$

Some simulations produce accurate results with **ESTEPE** set as high as 10%, while others require **ESTEPE** to be set at less than 1%. The selection of a suitable **ESTEPE** is a difficult problem for the EGS4 user, so for this reason *PRESTA* (the **P**arameter **R**educed Electron-**S**tep **T**ransport **A**lgorithm) was developed by Bielajew and Rogers.[25,26] It dynamically selects a suitable step length, which enables fast simulation while still providing accurate physics. It consists of three key components:

(a) **A new path length correction (PLC).** Simulations performed using the default EGS4 PLC algorithm developed from Fermi-Eyges theory show some dependence upon the electron step size (value of **ESTEPE**) chosen. PRESTA uses an improved PLC based on Molière multiple-scattering theory, where the limitations of the theory set an upper and lower limit on the electron step size.

(b) **A lateral correlation algorithm (LCA).** When electron steps become very large, the lateral displacement of the electron during a step becomes significant. Default EGS4 transports the electron in the initial direction of motion, whereas PRESTA also displaces the electron laterally.

(c) **A boundary crossing algorithm (BCA).** Large electron steps may transport the electron into another geometrical region which may be composed of a different medium, thereby violating the constraints of the underlying multiple scattering theory. To avoid this PRESTA reduces step sizes near a boundary. When the step size becomes sufficiently small the electron is transported across the boundary and the step size is allowed to increase again, up to the maximum value allowed by the Molière theory (or specified by the user). Figure 2.6 illustrates how the BCA alters step size as the electron approaches and then recedes from a boundary.

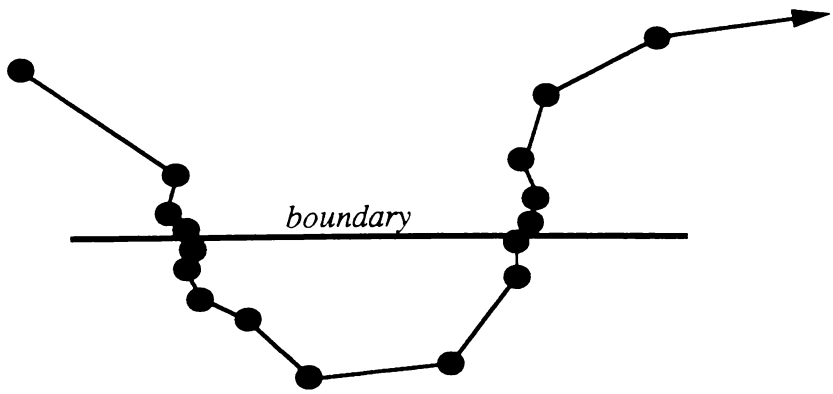

Figure 2.6. The PRESTA boundary crossing algorithm prevents large electron steps from transporting the electron into a different region. As the electron approaches a boundary the step size is reduced until the boundary is finally crossed with a very small electron step. As the electron recedes from the boundary the step size is allowed to increase.

These three components almost completely remove the step size dependence for the majority of EGS4 calculations. They remove the need to explicitly select a value for **ESTEPE** (usually set by trial and error), and in some simulations enable dramatic increases in the efficiency of calculation. For example, Figure 2.7 shows one electron track generated using the standard electron transport algorithm with **FIXTMX** invoked and **ESTEPE** set at 2% (lower track), and

another generated using the PRESTA algorithm (upper track). It is clearly seen that the standard algorithm (lower track) transports electrons in small step lengths which slowly decrease as the electron energy decreases, whereas PRESTA (upper track) uses large step sizes where possible. Only near scoring layer boundaries (grid lines) does PRESTA reduce step size. Using PRESTA for a dose spread array calculation using this geometry with 10 MeV incident photons results in a simulation which is 4.3 times more efficient than when using the standard algorithm. Bielajew and Rogers[26] provide a detailed discussion of the PRESTA algorithm and its significant impact on the speed and accuracy of EGS4 Monte Carlo calculations.

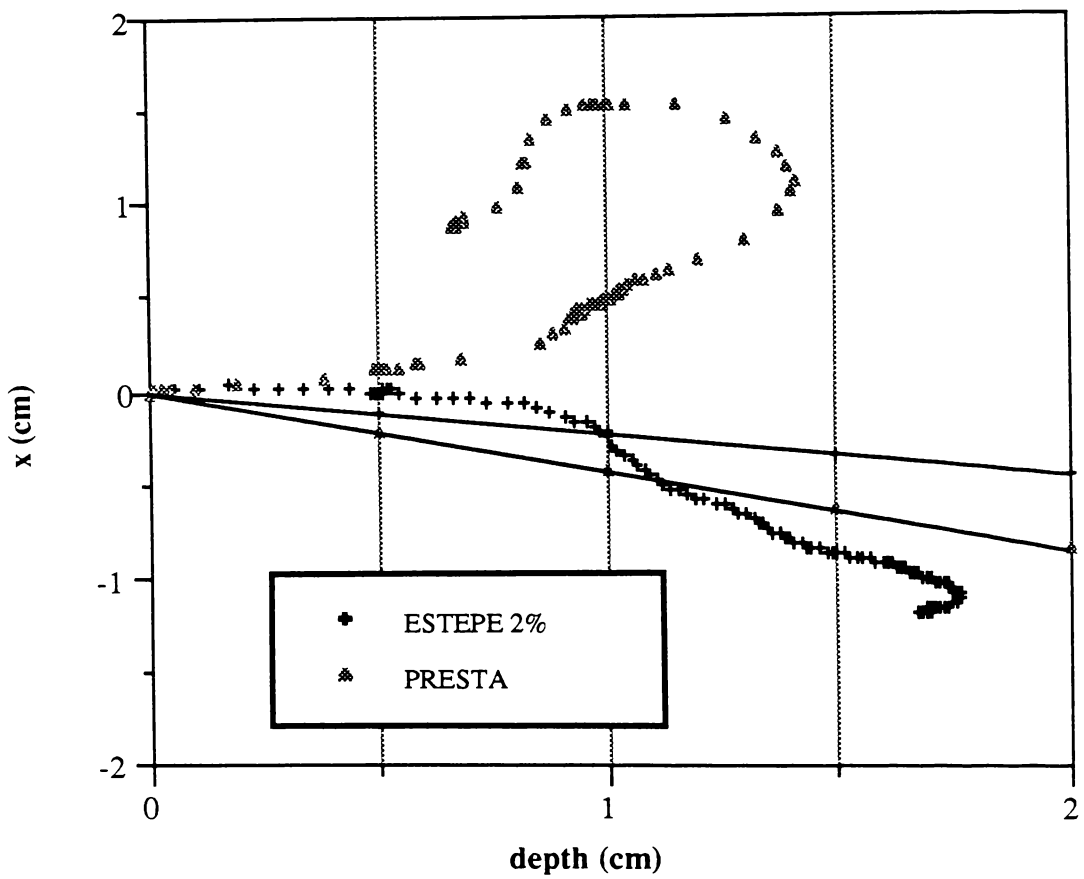

Figure 2.7. Particle tracks generated by two 10 MeV photons forced to interact at the origin of a water phantom (see Section 2.5d), generated using the DOTPLOT package developed at Waikato University. Layer boundaries are at 5 mm depth increments. Dots represent points at which AUSGAB has been called, and solid lines are photons. Grid lines indicate layer boundaries. **Lower track:** Generated using standard electron transport algorithm with ESTEPE set at 2%. Electron step length is small, decreasing slowly as electron energy decreases. Note delta ray produced at approximately 5 mm depth. 374 electron steps are modelled. **Upper track:** Generated using PRESTA transport algorithm. Step size is large, except near layer boundaries where the boundary crossing algorithm (BCA) is invoked. Only 162 electron steps need to be modelled.

2.3.6. EGS4 Variance Reduction

Calculation of electron energy deposition is computationally intensive, and at low energies is a good target for some form of variance reduction technique.[27,28] In general, to reduce the uncertainty in a result to $1/n$ of its former value, n^2 as many particles must be simulated. Variance reduction techniques seek to improve the *relative efficiency* of a simulation, where efficiency ϵ is defined by[27]

$$\epsilon = \frac{1}{s^2 T} \quad , \tag{2.13}$$

s^2 being the variance of the result.

The following are the major methods used, based on a review by Bielajew and Rogers.[27] Although they are discussed with particular reference to EGS4, these methods are applicable to any Monte Carlo code:

(a) **Geometry interrogation reduction.** Electrons generally travel only a small distance during each electron step. If this distance is small compared to the size of regions in the simulation then the geometry need be checked only relatively infrequently (when the electron approaches a boundary). In EGS4 this is done by maintaining a variable `DNEAR`, which represents an estimate of the distance to the nearest boundary. Only when `DNEAR` falls below a pre-set value is the geometry routine `HOWFAR` interrogated.

(b) **Zonal discard.** If the range of an electron is less than the distance to the closest scoring region boundary, the energy of the electron may be safely deposited in the current region, thereby saving further simulation of the electron. Many EGS4 simulations use the parameter `ESAVE`, representing the energy below which electrons will be subject to zonal discard. However, this technique does not account for energy which might be deposited outside the zone due to bremsstrahlung photons. This error is usually small.

(c) **Range rejection.** This technique is similar to zonal discard, except that the electron is discarded because it cannot reach a region of interest. Again the bremsstrahlung contribution is neglected.

(d) **Interaction forcing.** This technique is very useful when there is a high probability that a photon will pass through a region of interest without interacting, as in the calculation of energy deposited in a shallow tank of water. In these cases Equation 2.2 can be modified such that:

$$N_\lambda = -\ln\left(1 - \zeta(1 - \mathrm{e}^{-\Lambda})\right) \quad , \tag{2.14}$$

where N_λ is the number of mean free paths to the next interaction, ζ is random on the interval $[0,1)$ and Λ is the total number of mean free paths along the path to the end of the region of interest. Since all photons have been forced to interact, the *weight* of each photon w' must be reduced to

$$w' = w(1 - \mathrm{e}^{-\Lambda}) \ , \tag{2.15}$$

where w is the old weight of the photon. The factor $(1-\mathrm{e}^{-\Lambda})$ represents the probability that the photon would have interacted before travelling Λ mean free paths. It must be included to ensure that the photon maintains the correct amount of statistical significance compared to other photons which may not have had interactions forced.

(e) **Exponential transform, particle splitting and Russian roulette.** The exponential transform technique is useful for surface or deep-penetration problems, where only a small proportion of photons interact in the region of interest. The number of mean free paths N_λ is "stretched" or "shortened" according to the formula

$$N_\lambda = \frac{-\ln(\zeta)}{(1 - C\mu)} \ , \tag{2.16}$$

where μ is the cosine of the angle the photon makes with the z-axis and C is a scaling parameter such that normally $-1 \leq C \leq 1$. Again the influence of the photon must be correctly weighted:

$$w' = \frac{w\mathrm{e}^{-N_\lambda C\mu}}{(1 - C\mu)} \ . \tag{2.17}$$

For $C > 0$ the path length is "stretched," while for $C < 0$ it is "shortened." From Equation 2.17 it can be seen that for $C < 0$ any photons which do happen to penetrate deeply will receive very large weighting factors. These particles may increase the variance considerably if they happen to backscatter into the region of interest, so they may be *split* to avoid this. The particle is divided up into N particles, each with weight $w' = w/N$, so that each particle can have less of an influence on the overall statistics. Similarly, when particle weights become very small then the amount of computing time spent on those particles is reduced by playing *Russian roulette*. Only a proportion α of the particles are simulated, with a weight $w' = w/\alpha$, while the rest are discarded.

(f) **Leading particle biasing.** When simulating very high energy particles the simulation time per particle becomes very large (it increases approximately linearly with energy). If the high energy particles tend to contribute more to the final result then efficiency can be increased by simulating only one of the product particles in any collision, with a

bias towards the higher energy particle. The particles must once again be appropriately weighted. This technique requires that the relative importance of the physical processes be known, so that it can determined whether or not the variance can be reduced while maintaining an accurate result.

(g) **Sectioning.** If a large number of simulations involving a complicated geometry are to be performed, computing time can be saved by dividing the problem into sections. Results from the first section can be used repeatedly as input to the second section, the results of which might be used for a third section. This approach assumes that there is no interaction between the sections, which is a good approximation in many simulations, such as linear accelerator modelling.

(h) **Geometry equivalence.** In certain situations the *geometry equivalence* or *reciprocity* theorem can be applied to drastically reduce computation time. The most common example is that of a broad parallel beam incident on a semi-infinite geometry. In this case the dose in a cylindrical region of radius ρ_d due to an incident beam of radius ρ_b is equivalent to the dose in a region of radius ρ_b due to a beam of radius ρ_d. When calculating central axis depth doses the beam radius is much larger than the scoring region radius, so relatively few of the incident photons cause energy to be deposited in the region of interest. However, if the geometry equivalence theorem is employed then almost all photons contribute towards the dose scored, and hence the gain in efficiency can be very large.

(i) **Geometry symmetry.** In many simulations there is a degree of symmetry which may be employed to improve the statistics of the simulation. For example, when modelling a square field incident on a homogeneous medium there are eight equivalent voxels, since a square has four lines of symmetry. For cylindrical fields all energy deposited within a given cylindrical shell is equivalent, leading to even greater improvement in statistics for a given simulation time.

2.3.7. Benchmarking EGS4

In order to have confidence in Monte Carlo calculations, and to test the underlying theory, it is essential to perform benchmark calculations against known and trusted experimental results. EGS4 has been exhaustively compared to the ETRAN system in order to check the consistency of the two major Monte Carlo codes,[29] but it has also been extensively tested against experimental benchmarks in the low energy (less than 50 MeV) region.[30] The major experimental benchmarks used are:

(a) **Detector response functions.** Both photon detectors (such as NaI detectors) and small electron spectrometers have been simulated. Agreement is very good (within 1%) for photon detectors,[31] although even a poor Monte Carlo code will model these detectors quite well. However, electron detectors are a stringent test of a Monte Carlo code, since bremsstrahlung, backscatter and energy loss straggling must all be modelled accurately in thin absorbers. For very thin detectors both EGS4 and ETRAN predict the low-energy peak to be at a slightly higher energy than observed experimentally.[24]

(b) **Ion chamber response.** Since ion chamber response can be calculated accurately using cavity theory, this type of simulation is a good test for EGS4. Agreement is usually within 1%, although poor selection of electron transport parameters (such as using the default step-size algorithm) can result in large errors.[32,33] This type of simulation was used extensively in developing the PRESTA algorithm.[25]

(c) **Depth-dose curves.** EGS4 shows excellent agreement with experiment for both electron and photon depth-dose curves,[30] although there are difficulties in determining the incident spectrum from linear accelerators, and in removing the effect of electron contamination from photon depth-dose curves. Depth-dose curves are not a stringent test of Monte Carlo codes.

(d) **Bremsstrahlung production.** The calculation of bremsstrahlung spectra generated from linear accelerators is a good test of Monte Carlo. However, accurate experimental data is hard to obtain. Mohan *et al.*[34] have shown that EGS4 produces accurate depth-dose curves for linear accelerators, although the curves are not particularly sensitive to changes in incident spectrum.

2.4. Using the EGS4 System

2.4.1. The EGS4 Environment

EGS4 is supplied by Walter R. Nelson of the Stanford Linear Accelerator Center.[35] Magnetic tape formats suitable for DEC (VAX) or IBM machines can be requested. Also supplied with the tape is the EGS4 user manual,[11] of which Appendices 5 and 6 describe the installation process and tape contents, respectively. Source files are provided in both FORTRAN IV and FORTRAN 77 formats. Installation involves copying the files to a hard disk, compiling the FORTRAN source code of the Mortran processor, then compiling the Mortran source files for the PEGS4 and EGS4 systems. Command procedures are available to help with the installation process.

A version of the EGS4 system for running under OS–386 on PC–386 machines using Lahey FORTRAN is also available.[36] It requires minor modifications to run under other PC environments — Appendix A of this thesis outlines the procedure used by this author to install EGS4 on a PC–AT running Microsoft FORTRAN under MS–DOS. However, EGS4 running on a PC is likely to be slow. For example, a PC–AT will run EGS4 at about one-twentieth the speed of a VAX 11/780 with floating point accelerator. Even so, some useful simulations can still be performed in reasonable times. EGS4 can also be modified to run under other systems, such as SunOS on a Sun SPARCstation. Some features such as the random number generator (see Section 2.3.4) may have to be modified to suit the host system.

2.4.2. Structure of an EGS4 User Code

An EGS4 user code may be written entirely in FORTRAN, but is usually written in Mortran. Appendix 2 of the EGS4 user manual[11] describes the structure of a user code in detail, but the ten sections of a user code are briefly described here:

(a) **User override of EGS4 macros.** The maximum number of different media, maximum number of scoring regions, and other array bounds may be assigned here. For example, the Mortran statement

```
PARAMETER          $MXMED = 2;
```

sets the maximum number of different media in the simulation to two. Other Mortran macros such as the random number generator `$RANDOMSET` (see Section 2.3.4) and the distance to next interaction macro `$SELECT-PHOTON-MFP` may be overridden here. Common block definitions such as

```
REPLACE {;COMIN/GEOM/;} WITH {;COMMON/GEOM/ZBOUND;}
```

are also defined in this section. This example replaces every occurrence of `COMIN/GEOM` with a list of variables belonging to that common block (in this case just `ZBOUND`). This enables the user to control the contents of each common block, adding variables if necessary.

(b) **Pre-`HATCH` call initialisation.** Variables used by `HATCH` in reading in the PEGS4 data sets are initialised here. They include the number of media and media names, the array `MED` which defines the medium used in each geometrical region, and the particle transport cut-offs `ECUT` and `PCUT` (see Section 2.3.5).

(c) **`HATCH` call.** The required PEGS4 data sets (see Section 2.3.2) are obtained by calling the subroutine `HATCH`.

(d) **Initialisation for HOWFAR.** HOWFAR is the routine which determines the distance to the next region boundary (see Section 2.4.2i). Any arrays or variables needed to define the geometry may be initialised here. In general, the volume of interest is divided up into a number of different geometrical regions, each of which may be composed of a different medium.

(e) **Initialisation for AUSGAB.** AUSGAB is the routine which scores energy (or any other quantities of interest) into the appropriate scoring bins (see Section 2.4.2j). Scoring arrays and variables for scoring total energy may be initialised here. Scoring regions are distinct from geometrical regions, as in a water phantom simulation where there is only one geometrical region but many scoring regions.

(f) **Determination of incident particle parameters.** Prior to the actual simulation of particle transport, the characteristics of the incident particles must be defined. These include the particle charge, energy, coordinates, direction and weight. The region number in which the particles begin is also defined, and the random number seed and number of particle histories to be run (NCASE) are initialised.

(g) **SHOWER call.** The call to the EGS4 routine SHOWER represents the interface between the user code and the EGS4 system code itself. It is called once per incident particle, performing the simulation of that particle and all its products. It accepts ten parameters:

```
CALL SHOWER(IQ,E,X,Y,Z,U,V,W,IR,WT);
```

defining the particle's charge, energy, position, direction cosines, starting region and weight. The EGS4 code then interacts with the user code only by means of the two user-written routines HOWFAR and AUSGAB. When all product particles have been simulated SHOWER returns and another incident particle can be simulated.

(h) **Output of results.** When the simulation is complete, this section outputs the contents of the various scoring arrays. Output can also be displayed graphically if desired.[37]

(i) **Specification of HOWFAR.** As stated earlier, the HOWFAR routine is responsible for defining the geometry of the simulation. The EGS4 system code calls HOWFAR when it wishes the current particle to be transported by a distance USTEP in a straight line. If transport by this distance will not result in a region boundary being crossed then HOWFAR should simply return. If a boundary will be crossed then HOWFAR should identify the region on the other side of the boundary via the variable IRNEW, and set USTEP equal to the distance to the boundary. This enables the EGS4 system code

to transport the particle up to *but not over* the region boundary. Subsequent steps are performed by calling **HOWFAR** again. For complicated geometries, calculation of potential boundary crossings is computationally very intensive, and may slow down the simulation significantly. For this reason geometry interrogation reduction is often employed by maintaining the variable **DNEAR** (see Section 2.3.6a). A variable **IDISC** is also available within **HOWFAR**, which can be used to force a discard of the particle due to range rejection. Coding **HOWFAR** for a complex geometry is time consuming and difficult, so several geometry packages are available to simplify the problem. These include various macros for cylindrical, spherical and planar geometry,[38] and the MORSE-CG[39] combinatorial geometry package.

(j) **Specification of AUSGAB.** Subroutine **AUSGAB** is responsible for scoring quantities of interest to the user. **AUSGAB** is called with a parameter **IARG** indicating which of 25 situations it is being called under. Normally it is called when a particle is about to be transported or discarded, although flags may be set to force **AUSGAB** to be called when certain types of interaction take place. In medical physics applications the variable **EDEP** is commonly used, representing the energy deposited by the particle during the current step. **AUSGAB** checks to see if the particle is within the scoring region, then adds **EDEP** to the appropriate element of the energy scoring array. When the simulation is complete the array can be converted to dose and output.

Writing EGS4 user codes may appear a daunting task. Fortunately a series of tutor codes are distributed with the EGS4 tape, as well as a number of more complicated codes. These include **UCCYSL**, a cylindrical-slab user code; **UCBEND**, a user code simulating transport in a magnetic field; **UCSAMPCG**, a combinatorial geometry example; and **INHOM**, a general purpose user code for slab geometries.

2.4.3. Compiling and Running EGS4

A complete EGS4 application is built from at least five input files. As well as the Mortran user code there is **MORTRAN3.DAT**, which contains the Mortran system macros in hexadecimal format; **EGS4MAC.MOR**, which contains the default EGS4 macros; **EGS4BLOK.MOR**, which contains EGS4 block data; and **EGS4.MOR**, the EGS4 system code. In addition the application may also incorporate the NRCC macro set (**NRCC4MAC.MOR**) and the NRCC auxiliary routines (**NRCCAUX.MOR**), or the equivalent PRESTA versions. Developed by Rogers and Bielajew at the National Research Council of Canada, these two files contain macro overrides for many of the standard EGS4 macros (such as random number generation), and several routines for geometrical manipulation, monitoring and other functions (such as

FIXTMX discussed in Section 2.3.5). All the above files are normally incorporated into the user code by means of the unit switch %Un, where n is a special unit number assigned to each of the files (see Section 2.3.3d). On VAX or IBM systems a command file is available both to compile the complete application and to execute it. Appendix 5 of the EGS4 user manual[11] provides a more detailed explanation of the entire process.

2.5. Some Applications of EGS4

The EGS4 Monte Carlo code has been used in a wide range of medical physics problems. A few important examples are:

(a) **Dosimetry.** Responses of various ion-chambers and associated interface effects[33] and A_{wall} correction factors[40] have been modelled using EGS4. However, such simulations are extremely sensitive to the electron transport parameters (especially step size) selected, due to the very small size of scoring regions.[33] In addition to these difficulties, extremely large amounts of computing time are required to produce acceptable statistics. Stopping-power ratios have also been calculated using EGS4.[41] A future application of EGS4 is the simulation of solid-state detector response.

(b) **Positron emission tomography.** Monte Carlo codes such as EGS4 can be used to simulate a given PET design, and to investigate the effects of positron range, detector resolution, Compton scattering within the patient and other parameters affecting the performance of the PET machine.[42]

(c) **Radiotherapy treatment head simulation.** Knowledge of the energy spectra emitted from linear accelerators and cobalt units is necessary for treatment planning algorithms such as convolution (see Section 2.5d). However, direct measurement of high energy photon spectra is difficult due to high beam intensity, so Monte Carlo is an ideal technique. Monte Carlo simulation of treatment heads is also useful for predicting the effects of treatment head and flattening filter design changes, field size, beam modifiers, and spectral variation across the beam. Electron contamination of photon beams can also be modelled using Monte Carlo. For ^{60}Co teletherapy units, only the source and treatment head geometry must be defined, but for linear accelerators the situation is much more complex. The electron target, primary collimator, flattening filter, secondary collimator and jaws must all be modelled.[34]

(d) **Radiotherapy treatment planning.** The potentially high accuracy of the Monte Carlo method makes it well suited to radiotherapy dose calculation. Although the time required to calculate dose to the required accuracy currently prohibits its use in routine

treatment planning, electron pencil beams[43] and energy deposition kernels (dose spread arrays)[44] can be calculated using Monte Carlo. Using EGS4, dose spread arrays are calculated by forcing incident photons to interact at the phantom origin. This is done by modifying the `$SELECT-PHOTON-MFP` macro. Incident photons (charge `IQ=0`) enter the phantom travelling in the z direction, so if the initial particle coordinates are set to the origin then the macro

```
REPLACE {$SELECT-PHOTON-MFP;} WITH
{$RANDOMSET RNNO35; IF(RNNO35 = 0.0)[RNNO35=1.E-30;]
DPMFP=-ALOG(RNNO35);
"IF INCIDENT PHOTON THEN FORCE TO INTERACT IMMEDIATELY AT ORIGIN"
IF ((U(NP)=0.0).AND.(V(NP)=0.0).AND.(W(NP)=1.0).AND.
(X(NP)=0.0).AND.(Y(NP)=0.0).AND.(Z(NP)=0.0).AND.(IQ(NP)=0))
[DPMFP=0.0;]}
```

will cause a dose spread array to be generated. `NP` is the subscript of the current particle on the stack, `(X(NP),Y(NP),Z(NP))` are the coordinates of that particle and `((U(NP),V(NP),W(NP))` are the corresponding direction cosines. As an example, Figure 2.7 illustrates the paths taken by particles set in motion due to two photons forced to interact at the origin of a water phantom (typically several million would be used to generate an accurate energy deposition kernel). The differences between the default (lower track) and PRESTA (upper track) simulations are discussed in Section 2.3.5. Figure 2.7 has been generated using DOTPLOT, a simple graphics package developed at Waikato University. Photons are represented by straight lines and electrons (and positrons) by dotted tracks. The EGS4 user code and input file used to generate Figure 2.7 is discussed in Section 2.6.2.

In addition to the above applications, Monte Carlo techniques have also been applied to diagnostic radiology,[45] nuclear medicine,[46] and many aspects of radiation therapy.[47] Monte Carlo is also a potentially useful tool for understanding radiobiological effects,[48] through accurate simulation of electron track structure on a cellular level. However, this requires more detailed treatment of electron transport than is available in EGS4, and also requires knowledge of electron interaction cross-section data in the range 10 eV to 1 MeV. Little is known about these cross-sections.

2.6. General Purpose EGS4 User Codes for Radiotherapy

2.6.1. Introduction — The User Code INHOM(P)

Although it is possible to write a separate user code to solve any specific EGS4 simulation problem, many simulations are sufficiently alike that a general purpose code capable of taking parameters as input would be useful. One such code is `INHOM`, developed by Rogers

at the National Research Council of Canada. Distributed on the EGS4 tape, it is designed to model an electron or photon beam incident on one or more semi-infinite plates, each composed of one of two different materials. Energy is scored in a set of concentric cylindrical shells divided by the plate boundaries, as illustrated in Figure 2.8a. The code was later enhanced by incorporating the PRESTA algorithm discussed in Section 2.3.5. This INHOM + PRESTA code is known as INHOMP. The codes have been used in a variety of simulations, including slab geometries, electron detectors, and penetration of thin foils. These applications and the effect of transport parameter selection are discussed by Rogers.[24]

The key element of INHOMP is an input file of parameters used to specify details of the simulation, including:

- Materials used for the simulation.
- Details of scoring regions, including densities, scoring radii and plate thicknesses.
- Transport parameters such as ECUT and PCUT, and switches for multiple scattering, Rayleigh scattering, and the CSDA approximation.
- Incident particle number, charge, direction, and energy (which may be spectral).
- Control of a monitor procedure to allow study of individual particle histories.
- Simulation type, including parallel beam, point source, broad parallel beam and central axis simulations.
- Type of variance reduction, including geometry symmetry (included by default since cylindrical symmetry exists), geometry equivalence (reciprocity) as described in Section 2.3.6h, Russian roulette, and exponential transform.
- PRESTA parameters, including individual control of BCA,LCA,PLC and FIXTMX options.

The primary output from INHOMP is the total dose scored in each voxel. It is available via a table of dose distribution, expressed in units of Gy MeV^{-1} cm^2 in order to be independent of the incident energy fluence. With each voxel dose is an associated percentage uncertainty, calculated by dividing the simulation into ten batches and determining the standard deviation of the individual batch doses. This division also enables automatic termination of the simulation if the next batch will exceed the CPU job limit. Other output includes a graph of central axis dose and a table of region masses. For incident electrons, dose deposited due to secondary electrons and dose due to stoppers (electrons falling below ECUT)

are also scored. For incident photons, dose from electrons beginning in the other material, dose from all particles interacting in the other material, and stoppers dose are scored.

Versatile as the INHOMP code may be, it is not suitable for generating energy deposition kernels or modelling square radiotherapy fields. These and other requirements prompted refinement of the original INHOMP code to yield RTPCYL, a more general-purpose code particularly suitable for treatment planning simulations and cylindrical energy deposition kernel generation. In addition, many radiotherapy simulations do not exhibit cylindrical symmetry, and require that dose be scored in a cartesian coordinate system. Another user code called RTPCART has been developed to model these situations. A third code RTPEDK, which is a cut-down and modified version of RTPCYL, has been developed specifically for generating spherical energy deposition kernels for use in the GRATIS treatment planning system. Figure 2.8 illustrates schematically the difference between these three codes.

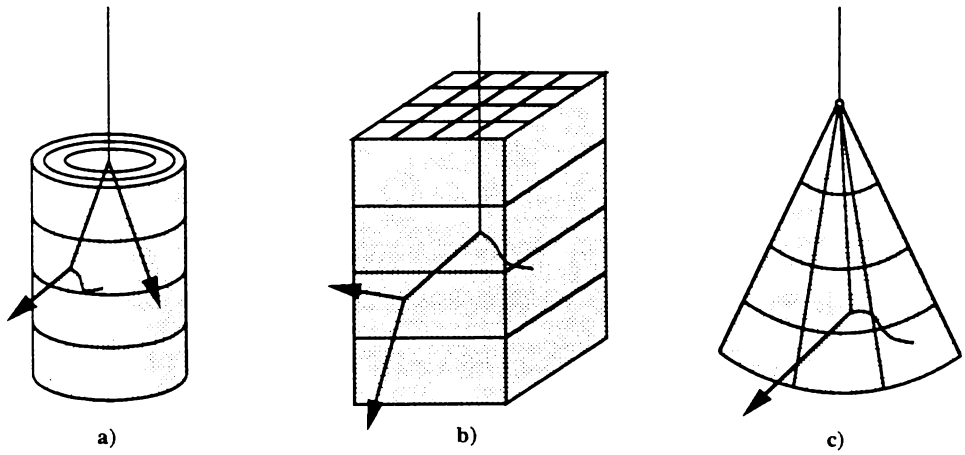

Figure 2.8. User codes developed by the author. (a) RTPCYL, for central axis measurements and cylindrical energy deposition kernels. (b) RTPCART, for cartesian geometry simulations and cartesian energy deposition kernels. (c) RTPEDK, for spherical energy deposition kernels.

2.6.2. RTPCYL — for Cylindrically Symmetric Geometries

Features of RTPCYL

As mentioned in the previous section, RTPCYL is an INHOMP-based code designed to model square or circular beams incident on a cylindrically symmetric geometry, and can also be used to generate cylindrical and polar energy deposition kernels. Features incorporated into RTPCYL to achieve this are as follows:

(a) **Square field** modelling. This is done by specifying the parameter ISOURC as 4, with a beam half-width RBEAM. The (x, y) coordinates of the incident photon are selected

by sampling two random numbers and scaling appropriately — this is in contrast to the rejection sampling used by INHOMP and RTPCYL for circular fields. Since a square field is not cylindrically symmetric, scoring shells near the edge of the field will contain an "average" of energy scored inside and outside the field. However, the square field option was included primarily for determination of central axis energy deposition, where the effects of cylindrical asymmetry are negligible for all except the smallest field sizes. The geometry equivalence variance reduction offered by the cylindrical symmetry of the RTPCYL scoring scheme offers a very significant advantage over RTPCART when determining central axis energy deposition.

(b) **Energy deposition kernel** modelling, with energy being scored in cylindrically or spherically oriented bins. Interactions are forced to occur at (x, y, z) = (0,0,ORIGINDEPTH), where ORIGINDEPTH is the distance below the phantom surface of the kernel origin. This is done by modifying the $SELECT-PHOTON-MFP macro in a manner similar to that discussed in Section 2.5d. A further consideration when generating energy deposition kernels using polyenergetic spectra is that not all photons have the same probability of interacting — this probability is proportional to $\mu(E)$ where E is the energy of the incident photon. In RTPCYL and RTPCART this is accounted for using the GMFP (gamma mean free path) function available in PEGS, measured in radiation lengths. This function can be sampled using the $SET INTERVAL and $EVALUATE commands available in Mortran. Since $\mu \propto \frac{1}{\text{GMFP}}$, each incident particle can be weighted by 1/GMFP to yield a kernel where each spectral component is correctly weighted. Prior to calling SHOWER with a new incident photon, the incident particle weight WTIN is modified by the following code:

```
MEDIUMOLD=MEDIUM;     "save current medium"
MEDIUM=MED(2);        "find medium at interaction site"
Ensure energy above cut-off"
IF (EINN<PCUT(2)) [GAMMALOG=ALOG(PCUT(2));]
ELSE [GAMMALOG=ALOG(EINN);]
"Determine weighting factor"
$SET INTERVAL GAMMALOG,GE;
$EVALUATE GAMMAMFP USING GMFP(GAMMALOG);
WTIN=1.0/GAMMAMFP;
TOTALMEV=TOTALMEV+EINN*WTIN;     "add to total incident energy"
MEDIUM=MEDIUMOLD;                "restore current medium"
```

Note that the total incident energy TOTALMEV is also recorded by the above section of code. This enables the energy scored in the completed kernel to be normalised to the total incident energy, so that the kernel can be represented by an array of dimensionless numbers.

(c) **A radial binning option** for energy deposition kernels. Work by Metcalfe *et al.*[49] required a kernel represented in spherical coordinates, in an approach similar to that used by Mackie *et al.*[44] In this option (**ISOURC=6**), two additional parameters are required: **DELTATHETA**, representing the angle between successive conical scoring zones; and **DELTAR**, representing the distance between successive spherical shells (more complete discussion of this geometry can be found elsewhere[49,50]). Radial binning has been implemented using the same scoring array, but with **AUSGAB** modified so that the array coordinates are (R_r, θ) instead of (R_a, z), where R_r is the radial distance from the kernel origin and R_a is the shortest distance to the central axis of the kernel. The relevant excerpt from **AUSGAB** is as follows:

```
IF (ISOURC=6) ["score in radial bins"
 R=SQRT(X(NP)**2+Y(NP)**2+(Z(NP)-ORIGINDEPTH)**2); "calculate R in cm"
 IF (R<0.000001) [R=0.000001;] "avoid division by zero in arcsin calc."
 THETA=ASIN(SQRT(X(NP)**2+Y(NP)**2)/R)*180/3.1415926; "THETA in deg."
 IF (Z(NP)<ORIGINDEPTH) [THETA=180-THETA;] "ensure 0<=THETA<=180"
 IZ=INT(R/DELTAR)+1;              "calculate Z voxel index"
 IF (IZ>NLAYER) [RETURN;]         "don't score if below phantom"
 IRAD=INT(THETA/DELTATHETA)+1;    "calculate R voxel index"
 IF (IRAD>NRAD) [RETURN;]         "don't score if outside phantom"
]
```

This method is not ideal for scoring spherical energy deposition kernels, since the PRESTA boundaries do not coincide with the radial scoring regions (and hence PRESTA must be turned off). Also, **RTPCYL** is much more complex than necessary for generating spherical EDKs. The user code **RTPEDK**, discussed in Section 2.6.4, has been written specifically for this purpose.

(d) **Geometry equivalence for divergent fields.** In general, the reciprocity theorem can be used only when the incident beam is parallel (non-divergent). However, work done by Hoban[51] has shown that it *can* be used for divergent fields, provided that the distance of each electron step from the central axis is appropriately modified. In **RTPCYL** this option is selected by setting **ISOURC=7**, and is coded within **AUSGAB** in the following manner:

```
"Modify RSQ (radius squared) if using divergent field"
IF (ISOURC=7) [RSQ=RSQ*(DISTZ/(DISTZ+Z(NP)))**2;]
```

where **DISTZ** is the source-to-surface distance. Additionally, the energy deposited in each voxel must be modified by an inverse square correction at the end of the calculation:

```
"If ISOURC=7, modify dose by an inverse square correction"
IF (ISOURC = 7) [DOSE(IZ,IRAD,IT) = DOSE(IZ,IRAD,IT)
     * (DISTZ/(DISTZ+(ZBOUND(IZ)+ZBOUND(IZ+1))/2.0))**2;]
```

where ZBOUND(i) is the z-coordinate of the boundary before region $i+1$ (layer i). Refer to Hoban[51] for a full discussion of this approach.

(e) **Division into primary and scattered dose.** In some radiotherapy simulations, especially energy deposition kernel calculations, it is sometimes desirable to divide deposited energy into primary energy (deposited by electrons released from the primary interaction site) and scattered energy (deposited by all other interactions). For incident photons, total, primary, and scattered energy are scored in the first three "planes" of the array DOSEIS. The fourth plane is reserved for kerma (see Section 2.6.2f). Primary and scattered energy are distinguished using $FLAG4, an additional flag based on the stack variable LATCH(p) associated with each particle p on the stack. It is defined in the following manner:

```
REPLACE {$SET-FLAG4(#);} WITH
{;IF (LATCH({P1})>=0)
 [LATCH({P1})=LATCH({P1})+1000000;]
ELSE
 [LATCH({P1})=LATCH({P1})-1000000;];
}
REPLACE {$FLAG4} WITH {ABS(MOD(LATCH(NP),10000000))/1000000}
```

LATCH may be positive or negative, and its value is passed on automatically to any product particles. $FLAG4 is signified by the millions digit of LATCH, which is modified and tested by the above macros independently of all other digits of LATCH (used by other flags). When SHOWER is called with a new incident photon, $FLAG4 is initially zero. Immediately after the first interaction, AUSGAB increments each $FLAG4 once or twice, depending upon whether the associated product particle contributes to primary or scattered dose (note that bremsstrahlung is scored with scattered dose):

```
IF (IQIN=0) [ "for incident photons only"
 IF ($FLAG4=0 & NP>1) [ "more than one product particle"
  DO I=1,NP [ "for each product particle"
   IF (IQ(I)=0)
    [$SET-FLAG4(I);$SET-FLAG4(I);] "scattered photon"
   ELSE
    [$SET-FLAG4(I);] "electron is primary energy"
 ]
 ]
 IF($FLAG4=1 & IQ(NP)=0) [$SET-FLAG4(NP);] "brem = scattered dose"
 "Ensure photoelectrons are scored as primary energy"
 IF ($FLAG4=0 & NP=1 & IQ(1)<>0) [$ SET-FLAG4(1);]
]
```

Each subsequent time AUSGAB is called the energy EDEP deposited in the step is assigned to the total energy array. $FLAG4(NP) is then tested to determine whether EDEP is also assigned to the primary energy or scattered energy array.

(f) **Scoring of kerma.** As a check of superposition algorithms it is useful to have RTPCYL calculate the kerma (kinetic energy released in the medium due to primary interactions) deposited in each voxel. $FLAG5, formed from the ten-millions digit of LATCH, has been defined in a manner similar to $FLAG4 defined in Section 2.6.2e. It is initially set to zero, but whenever an electron is produced AUSGAB assigns the electron's kinetic energy to the terma array (plane 4 of DOSEIS), and increments $FLAG5 so that no further assignment of energy takes place:

```
IF ($FLAG4=1 & $FLAG5=0 & IQ(NP)=0) [
 "Primary electron with unassigned kerma"
 DOSEIS(IZ,IRAD,4,IS)=DOSEIS(IZ,IRAD,4,IS)+WT(NP)*(E(NP)-0.511);
 $SET-FLAG5(NP);
]
```

Note that the kinetic energy deposited is the total energy E(NP) less the rest mass energy of the electron, and that the particle weight WT(NP) must be taken into account.

(g) **Effective range calculation** for energy deposition kernels. Metcalfe *et al.*[50] also wished to calculate effective electron ranges when generating energy deposition kernels, for comparison with the calculations of Mackie *et al.*[44] If ϵ_n is the energy deposited by the nth transport step of a primary electron, then the effective charged particle range r_{eff}, effective longitudinal range z_{eff}, and effective lateral range y_{eff} are defined by Metcalfe *et al.*[50] as:

$$r_{\text{eff}} = \frac{\sum_n \epsilon_n \sqrt{(x_n^2 + y_n^2 + z_n^2)}}{\sum_n \epsilon_n} \tag{2.18}$$

$$z_{\text{eff}} = \frac{\sum_n \epsilon_n z_n}{\sum_n \epsilon_n} \tag{2.19}$$

$$y_{\text{eff}} = \frac{\sum_n \epsilon_n \sqrt{(x_n^2 + y_n^2)}}{\sum_n \epsilon_n} \quad , \tag{2.20}$$

where (x_n, y_n, z_n) is the particle position at the midpoint of the nth electron step. RTPCYL records the numerators in these expressions in AUSGAB:

```
RRANGE=RRANGE+DBLE(R*WT(NP)*EDEP);
ZRANGE=ZRANGE+DBLE(((Z(NP)-ORIGINDEPTH)*WT(NP)*EDEP);
YRANGE=YRANGE+DBLE(SQRT(X(NP)**2+Y(NP)**2)*WT(NP)*EDEP);
TOTALPRIMARY=TOTALPRIMARY+WT(NP)*DBLE(EDEP);
```

Note that total primary energy is also scored — this value represents the denominator in Equations 2.18–2.20 and is used to normalise RRANGE, ZRANGE and YRANGE to yield effective ranges in centimetres. This implementation of the effective range calculations yields results slightly different from those reported by Mackie, but the author considers

that this approach more closely resembles physical reality, since r_{eff}, z_{eff}, and y_{eff} are treated as fully independent vector quantities. Metcalfe[52] discusses this point in more detail.

One further consideration is that energy is usually deposited at the beginning of an electron step, which leads to a consistent underestimation of range equal to half the mean step length. To circumvent this problem particle histories alternate between depositing energy at the beginning and the end of electron steps. Immediately before SHOWER is called, flags 1 and 6 of the IAUSFL array are toggled to change the conditions under which AUSGAB is called:

```
IF (ISOURC>5) [ "kernel calculation"
 IF (IAUSFL(1)=1) [IAUSFL(1)=0;IAUSFL(6)=1;] "call AUSGAB at step end"
 ELSE [IAUSFL(1)=1;IAUSFL(6)=0;] "call AUSGAB at step start"
]
```

This approach ensures that the mean position of energy deposition is the midpoint of the electron steps.

(h) **Control of table and graph plotting options.** Input card 12 of the RTPCYL code has options for controlling the output of various dose categories. The array PLOTIT contains flags for each of the four dose categories — for incident photons these are total dose, primary dose, scattered dose, and kerma, while for incident electrons they are total dose, secondary electron dose, dose from particles from the other medium, and dose from stoppers and discards. For each category IT, PLOTIT(IT)=0 will cause no output, PLOTIT(IT)=1 will print a table of doses, and PLOTIT(IT)=2 will print a table of doses and a graph of central axis dose.

(i) **Option to output raw dose data.** Dose distributions and energy deposition kernels generated using RTPCYL are frequently used as input to superposition code or display software. To simplify this process a raw data output option has been added to RTPCYL, activated by setting the IRAWDATA flag. When selected, this option outputs one dose per line to file unit 14, with one blank line between each row of voxels. This allows other programs to pick up data directly, rather than sifting through a table generated by the PLOTIT option. For IRAWDATA=1, total dose data is output in $(z_1, r_1), (z_1, r_2) \ldots (z_n, r_n)$ order, where z_i is the ith layer and r_j the jth cylindrical shell. For spherical kernel scoring (ISOURC=6), output is $(\theta_1, R_1), (\theta_1, R_2) \ldots (\theta_n, R_n)$, where R_j is the jth *spherical* shell. Two other options are available: IRAWDATA=2 outputs raw data for primary and scattered dose only, while IRAWDATA=3 outputs raw data for total dose, primary dose, scattered dose, and kerma.

(j) **DOTPLOT output.** This feature produces a data file which can be used to trace the path of individual particles within the phantom. Each time `AUSGAB` is called, the variable `IDOTPLOT` is tested and appropriate particle data is output to file unit 13, with the default option (`IDOTPLOT=0`) causing no output. The second option (`IDOTPLOT=1`) records (q_i, x_i, z_i) for each particle step, where q_i is the particle charge, x_i is the particle x-coordinate, and z_i is the particle z-coordinate (depth) for the ith particle step. A third option (`IDOTPLOT=2`) records (q_i, x_i, z_i,`U(NP)`,`W(NP)`,`USTEP`), where `U(NP)` and `W(NP)` are the x and z direction cosines of the current particle and `USTEP` is the particle step length. This approach allows more faithful graphical representation of particle tracks at the expense of a larger data file.

(k) **A long-cycle random number generator.** The lagged-Fibonacci pseudorandom number generator described in Sections 2.3.3f and 2.3.4 has been added to `RTPCYL` to provide a longer sequence of pseudorandom numbers when dealing with very large simulations.

The above modifications have produced a useful and versatile EGS4 user code which is routinely used for many different types of simulation. Two examples of its use follow: Example 1 describes the input file used to generate the DOTPLOT picture in Figure 2.7, while Example 2 describes the input file used to generate central axis depth dose output in the lung phantom illustrated in Figure 1.14.

Example 1: DOTPLOT Simulation

To generate the data for plotting the upper track in Figure 2.7, the following input file to `RTPCYL` was used (comments are listed to the right):

`10 MeV photon into H2O`	simulation title
`WATER`	material 1
`DUMMY`	material 2 (not used)
`10.0,-1,3`	phantom thickness (cm), no. of depth groups (– indicates groups follow), no. of radial scoring regions
`0.5,20,1,1.0,0.561,0.050`	depth increment (cm), number of depth regions, material index, region density, `ECUT`, `PCUT`
`0.0`	inner radius for 1st radial region (cm)
`0.5`	inner radius for 2nd radial region (cm)
`1.5`	inner radius for 3rd radial region (cm)
`2.0`	outer radius for 3rd radial region (cm)
`10.0,0.561,0.050`	K.E. of incident particle (MeV), global `ECUT`,`PCUT`
`5`	indicates EDK generation (interaction forcing)
`0.0`	depth in phantom of kernel origin (cm)
`1,0,0,12.0,0,123456789`	no. of histories, incident charge, monitor routine off, CPU limit (hrs), histogram plot off, random no. seed
`10.0,0.2`	depth limit (cm) and survival prob of Russian roulette
`0,0,0,0`	PRESTA PLC,BCA,LCA and `FIXTMX` turned on
`0,0,0,0,0,1,0`	output options — no data files, DOTPLOT file only.

In the above example a water phantom is divided into 20 scoring layers each 5 mm deep, and 3 radial shells of thickness 5 mm. **ECUT** is set at 0.561 MeV and **PCUT** at 0.050 MeV in all scoring regions. The incident particles have energy 10.0 MeV and zero charge (photons). Russian roulette is played with any particles travelling beyond a depth of 10.0 cm, with a survival probability of 0.2. All components of PRESTA are turned on, and the **FIXTMX** macro is invoked. In this particular simulation there is no output except a DOTPLOT file containing particle charge and coordinates whenever **AUSGAB** is called.

Example 2: Central Axis Depth Dose

RTPCYL is especially useful for calculating central axis depth dose in cylindrically symmetric phantoms, particularly semi-infinite slab geometries such as the lung phantom illustrated in Figure 1.14. Although a square beam is not cylindrically symmetric, central axis depth dose can be scored in cylindrical voxels provided the scoring voxels remain in the essentially "flat" region of the beam. The extent of this flat region can be determined by scoring dose in a number of concentric cylinders, and checking for a fall-off as the cylinder radii become larger.

The example input file below illustrates simulation of a 5 cm × 5 cm square photon beam with 10 MV spectrum and 100 cm SSD, incident on the lung phantom of Figure 1.14:

`10 MeV photons into H2O`	simulation title
`WATER`	material 1
`DUMMY`	material 2 (not used)
`28.0,-5,1`	phantom thickness (cm), no. of depth groups (– indicates groups follow), no. of radial scoring reg.
`0.5,8,1,1.0,0.561,0.050`	first region (tissue) — depth increment (cm),number of depth regions, material index, density, ECUT, PCUT
`0.5,16,1,0.2,0.561,0.050`	second region (lung)
`0.5,8,1,1.0,0.561,0.050`	third region (tissue)
`0.5,16,1,0.2,0.561,0.050`	fourth region (lung)
`0.5,8,1,1.0,0.561,0.050`	fifth region (tissue)
`0.0`	inner radius for 1st radial region (cm)
`1.0`	outer radius for 1st radial region (cm)
`-1.0,0.561,0.050`	K.E. of incident particle (– indicates spectrum follows), global ECUT,PCUT
`0.0`	lower energy of 1st spectral bin
`1.0,17.5`	upper energy of 1st spectral bin, spectral weight
`2.0,22.0`	2nd spectral bin
`3.0,20.4`	3rd spectral bin
`4.0,12.9`	4th spectral bin
`5.0,9.2`	5th spectral bin
`6.0,7.1`	6th spectral bin
`7.0,4.9`	7th spectral bin
`8.0,3.2`	8th spectral bin
`9.0,1.9`	9th spectral bin
`10.0,0.9`	10th spectral bin
`0.0`	indicates end of spectrum

`4`	indicates square field, point source on axis
`100.0,2.5`	SSD, beam half-width (cm)
`100000,0,0,12.0,0,123456789`	no. of histories, incident charge, monitor routine off, CPU limit (hrs), histogram plot off, random no. seed,
`28.0,0.2`	depth limit (cm), and survival prob of Russian roulette
`0,0,0,0`	PRESTA PLC,BCA,LCA and `FIXTMX` turned on
`2,1,1,1,0,0,0`	output options — total dose table and graph, tables for primary dose, scattered dose and kerma, no region masses, no DOTPLOT, no RAWDATA

Note that tissue is modelled using water, and lung is modelled by a water-like material of density 0.2 g cm^{-3}. Only one radial scoring region is used (of outer radius 0.5 cm), and depth increments are 0.5 cm to a depth of 28 cm. For larger field sizes the radius of the scoring could safely be extended to several centimetres, which would vastly improve the efficiency of the simulation (and hence reduce statistical variation). However, many geometries are either not cylindrically symmetric or require dose information based on a cartesian framework — these simulations can be performed using the user code `RTPCART` described in the following section.

2.6.3. RTPCART — for Cartesian Geometries

Features of RTPCART

Although `RTPCART` is designed to model cartesian geometries, it shares many of the features of `RTPCYL`, including options to:

- Model parallel or point-source square beams, and energy deposition kernels,
- Score total, primary or scattered dose,
- Generate dose table, region mass table, DOTPLOT, and raw data output (graph output is not available),
- Monitor individual particle histories using subroutine `WATCH`, and
- Utilise a long-cycle random number generator.

However, there are many areas which are significantly different from `RTPCYL`, particularly in relation to scoring geometry and energy deposition kernel calculation. These areas include:

(a) **Source geometry options.** Three options are available for specifying the source arrangement. Setting `ISOURC=0` specifies a parallel beam, with the next input line indicating the (x, y) coordinates of the beam centre (`XCENTRE` and `YCENTRE`), and the beam width in centimetres (`BEAMWIDTH`). The second option (`ISOURC=1`) specifies a

point source with (x, y, z) = (XCENTRE,YCENTRE,-DISTZ) and beam width BEAMWIDTH (at the phantom surface). The third option (ISOURC=2) indicates an energy deposition kernel is to be generated and requires five additional parameters: XCENTRE and YCENTRE, representing the (x, y) coordinates of the kernel origin; ORIGINDEPTH, the depth of the kernel origin below the phantom surface; ISMEAR, a flag for indicating which type of voxel smearing is to be performed; and IDOSETYPE, indicating whether total, primary, or scattered dose is to be scored.

(b) **Definition of phantom geometry.** RTPCYL is based on a cartesian coordinate system which divides the patient or phantom into a (usually large) number of identically shaped boxes or *voxels*. Each voxel is a separate *geometrical* region, and is also a separate *scoring* region. The extent of the entire phantom is defined prior to compilation using Mortran PARAMETER statements:

```
PARAMETER $XMAX = 15;
PARAMETER $YMAX = 15;
PARAMETER $ZMAX = 60;
PARAMETER $MXREG=COMPUTE $XMAX*$YMAX*$ZMAX+2;
```

Note that $MXREG, the number of regions in the simulation, is also defined such that there is a unique region number for each voxel, plus one region above the phantom and one region below. The values defined above for these parameters are typical for a full-field simulation, and in this case each region array is composed of 13500 voxels. These large array sizes are a problem on machines without virtual memory or with limited virtual memory allocation, so to reduce total memory requirements the number of batches used to calculate statistical uncertainty ($STAT) has been reduced to five in RTPCART. In addition, only one dose category can be scored at a time — this can be total, primary, or scattered dose (see item a). Voxel dimensions in centimetres are specified in the parameter file by the variables XSIZE, YSIZE, and ZSIZE.

(c) **Definition of phantom composition.** On many occasions RTPCART is used to model a homogeneous phantom, while for patient simulations each voxel may be of a different density or composition than its neighbours. To allow maximum flexibility in the way phantom composition can be specified, there are four different options available. These are flagged by the signs of XSIZE, YSIZE and ZSIZE in the following manner:

(i) XSIZE<0,YSIZE<0,ZSIZE<0. All voxels are composed of the same medium and have the same density. The next input line indicates the medium number MED and density RHOR in $\mathrm{g\,cm^{-3}}$. This option is commonly used for energy deposition kernel calculation, and simulations in homogeneous media.

(ii) XSIZE>0,YSIZE>0,ZSIZE<0. Voxels *in each layer* are composed of the same medium and have the same density. There is one (MED,RHOR) input pair for each *layer* of voxels in the simulation. This option is used for slab phantoms such as the lung phantom illustrated in Figure 1.14.

(iii) XSIZE>0,YSIZE>0,ZSIZE>0. All voxels have potentially different density and composition. There is one (MED,RHOR) input pair for *each voxel* in the simulation. This option is used for heterogeneous patient simulations.

(iv) XSIZE<0,YSIZE<0,ZSIZE>0. All voxels are of the same density and composition, *except for a rectangular block.* There is one (MED,RHOR) pair for voxels in the phantom, and another for voxels within the block. Six other parameters (IXLOW,IXHIGH,IYLOW,IYHIGH,IZLOW and IZHIGH) are needed to specify the boundaries of the block (in terms of voxel indices). This option is used for phantoms having a single, regularly-shaped heterogeneity, such as the air-cavity phantom used in Example 2 below.

As with RTPCYL, a maximum of two different media can be used in any one simulation using RTPCART. However, since the human body can to a good approximation be divided into tissue-like and bone-like substances, it is possible to represent almost all radiotherapy geometries by using water and bone as the two materials, with the density appropriately scaled to represent lungs, air cavities, and other heterogeneities.

(d) **Interaction voxel smearing** for energy deposition kernel calculations. Energy deposition kernels are generated by forcing incident photons to interact in the voxel specified by the parameters XCENTRE, YCENTRE and ORIGINDEPTH. In RTPCYL these interactions occur at a single point, but for generating cartesian kernels these interactions should be spread throughout the interaction voxel, as illustrated in Figure 2.9.
This "smearing" is required because terma is liberated from *all* points within each patient or phantom voxel, so kernel interaction sites should be "smeared" in a similar manner to ensure that the correct dose distribution is produced when performing a superposition. In fact three options are available: ISMEAR=0 selects interactions at a single point, ISMEAR=1 selects two-dimensional smearing across the surface of the voxel, and ISMEAR=2 selects three-dimensional smearing throughout the interaction voxel. Two-dimensional smearing is used to generate electron pencil beams, while three-dimensional smearing is normally used to generate photon energy deposition kernels. If IXBASE, IYBASE and IZBASE are the coordinates of the interaction voxel

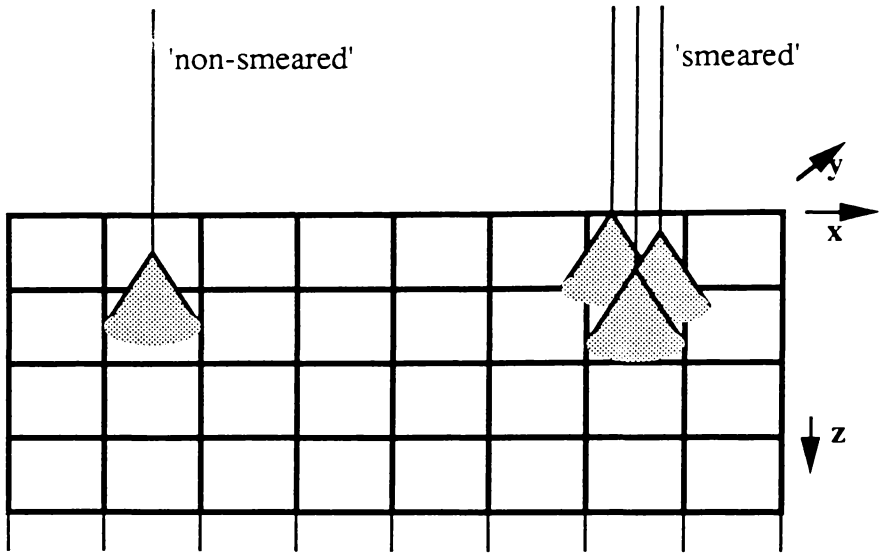

Figure 2.9. Interaction voxel "smearing." In the non-smeared approach (left) all incident photons are forced to interact at the centre of the interaction voxel. In the smeared approach (right) interactions are randomly distributed throughout the interaction voxel.

and XSIZE, YSIZE and ZSIZE are the voxel dimensions in centimetres, then smearing is achieved by the following code, located before the call to SHOWER:

```
IF (ISOURC=2) [ "kernel calculation"
 IF (ISMEAR=2) [ "2-D smearing"
  $RANDOMSET XIN; $RANDOMSET YIN;
  XIN=(IXBASE-1+XIN)*XSIZE;
  YIN=(IYBASE-1+YIN)*YSIZE;
 ]
 IF (ISMEAR=3) [ "3-D smearing"
  $RANDOMSET XIN; $RANDOMSET YIN; $RANDOMSET ZIN;
  XIN=(IXBASE-1+XIN)*XSIZE;
  YIN=(IYBASE-1+YIN)*YSIZE;
  ZIN=(IZBASE-1+ZIN)*ZSIZE;
 ]
]
```

Kernel smearing slightly modifies kernel shape (see Section 3.3.3c of Chapter 3), but has almost no effect on the resulting dose distribution, especially in homogeneous media.

(e) **Geometry equivalence** for energy deposition kernel calculation. Since energy deposition kernels generated in homogeneous media are symmetrical about the $x = 0$ and $y = 0$ planes, energy deposited in the four (x, y) quadrants can be "folded over" into one quadrant. Figure 2.10 illustrates this folding process, and also shows the multiplication factors necessary to maintain conservation of total energy deposited. Note that voxels along the x or y axes have only one other equivalent voxel, and that central voxel has no other equivalent voxels. This folding approach means that only one quarter of the complete kernel need be stored, and also means that inaccuracies due to

the statistical nature of Monte Carlo simulations are reduced by a factor of $\sqrt{4} = 2$ for off-axis voxels. If the voxels have a square cross-section (**XSIZE=YSIZE**) then statistical variance and storage requirements can be further reduced by averaging pairs of voxels (i, j, k) and (j, i, k), where i, j and k are the x, y and z voxel indices, respectively.

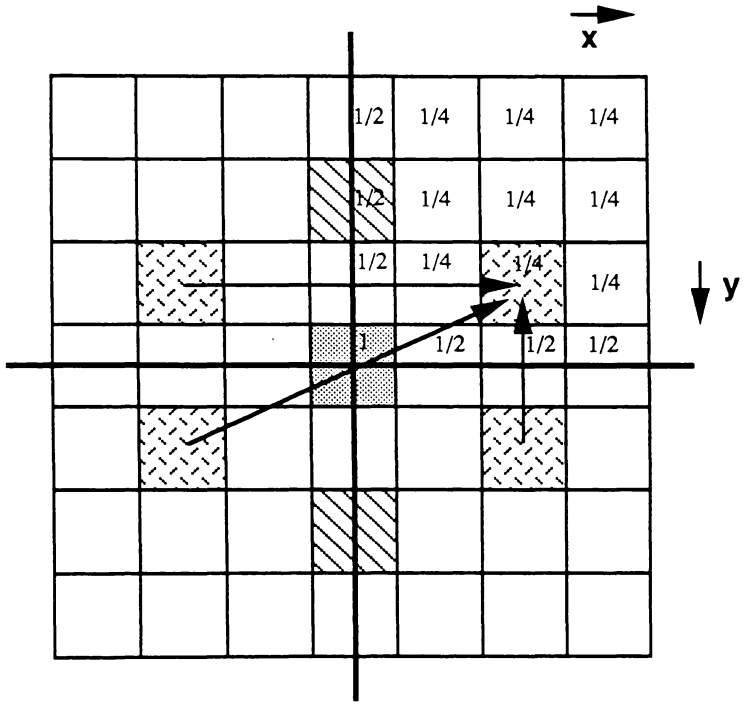

Figure 2.10. The "folding quadrant" technique for energy deposition kernel generation. Similarly shaded voxels are folded onto a single quadrant, then when the simulation is complete the voxels are multiplied by the factors shown to conserve total energy deposited. If voxels have a square cross-section (as shown here) then further savings can be achieved by averaging *(i,j,k)* and *(j,i,k)* voxels.

In **RTPCART**, energy is scored in all quadrants during the simulation. If an energy deposition kernel has been generated then the four quadrants are automatically "folded" before being output to a file, and if the voxel dimensions **XSIZE** and **YSIZE** are equal then equivalent voxels within the same quadrant are also averaged. In either case, one complete quadrant is printed in the output table or sent to the raw data file.

(f) **Coding of HOWFAR.** Since there are very many different regions in a typical **RTPCART** simulation, it is not practicable to individually code the geometry for each region in **HOWFAR**. Instead, the distance **TVAL** to the next region boundary is determined in a more general way, by making use of the fact that the region boundaries are parallel to the axes and evenly spaced. For example, the following code determines the distance to the next plane perpendicular to the z-axis:

```
TVAL=10000.0; "set TVAL large to begin"
IF (W(NP)>0.0 [ "particle moving in +z direction"
 TEMP=(ZSIZE-AMOD(Z(NP),ZSIZE))/W(NP);
 IF (TEMP<$SMALLDIST) [TEMP=ZSIZE/W(NP);]
 IF (TEMP<TVAL) [TVAL=TEMP; IDIRN=1;]]
ELSEIF (W(NP)<0.0) [ "particle moving in -z direction"
 TEMP=(AMOD(Z(NP),ZSIZE))/-W(NP);
 IF (TEMP<$SMALLDIST) [TEMP=ZSIZE/-W(NP);]
 IF (TEMP<TVAL) [TVAL=TEMP; IDIRN=2;]
]
```

where (X(NP),Y(NP),Z(NP)) are the coordinates of the current particle with corresponding direction cosines (U(NP),V(NP),W(NP)), IDIRN is a flag used to store the direction to the next boundary, and $SMALLDIST is a very small value used to prevent round-off error from producing an incorrect TVAL. Each clause of the IF statement determines the distance to the boundary in the relevant direction, then tests to see if that distance is less than the current value of TVAL, in which case TVAL is assigned the new value. Similar statements are used to test the x and y planes, so each of the six voxel boundaries is tested. TVAL then contains the distance to the *closest* region boundary along the current particle direction and IDIRN indicates which of the six planes ($+z, -z, +x, -x, +y$ or $-y$) is intersected.

Each voxel in the three-dimensional array is mapped onto the one-dimensional medium array MED using the following function:

```
MED((IX-1)*$YMAX*$ZMAX + (IY-1)*$ZMAX + IZ + 1) = IMED;
```

where IMED is the medium number of the voxel with indices (IX,IY,IZ), and ($XMAX,$YMAX,$ZMAX) are the maximum voxel indices. This mapping provides a simple means of determining the new region number IRNEW in HOWFAR:

```
IF (TVAL<=USTEP) [ "update region number"
 USTEP=TVAL; "set USTEP to TVAL"
 IF (IDIRN=1) [IRNEW=IRL+1;] "increment z layer"
 ELSEIF (IDIRN=2) [IRNEW=IRL-1;] "decrement z layer"
 ELSEIF (IDIRN=3) [IRNEW=IRL+($YMAX*$ZMAX);] "increment x layer"
 ELSEIF (IDIRN=4) [IRNEW=IRL-($YMAX*$ZMAX);] "decrement x layer"
 ELSEIF (IDIRN=5) [IRNEW=IRL+$ZMAX;] "increment y layer"
 ELSEIF (IDIRN=6) [IRNEW=IRL-$ZMAX;] "decrement y layer"
 "Discard particle if outside phantom limits"
 IF (IRNEW<1 | IRNEW>$MXREG) [IDISC=2;]
]
```

Finally, the PRESTA algorithm also requires that the macro $CALL-HOWNEAR be defined so as to calculate the smallest distance to *any* region boundary. Since the region array is cartesian, the closest distance to a surface is always normal to that surface, and $CALL-HOWNEAR can be defined very simply as:

```
REPLACE {$CALL-HOWNEAR(#);} WITH
{;IF (IRL>1) [ "if particle has reached phantom"
 XTEMP=AMOD(X(NP),XSIZE); "distance to x boundary"
 YTEMP=AMOD(Y(NP),YSIZE); "distance to y boundary"
 ZTEMP=AMOD(Z(NP),ZSIZE); "distance to z boundary"
 {P1}=AMIN1(XTEMP,XSIZE-XTEMP,YTEMP,YSIZE-YTEMP,ZSIZE,ZSIZE-ZTEMP);];
}
```

The above points have illustrated the main differences between RTPCART and its predecessor RTPCYL, but a full listing of RTPCART can be found in Appendix B. The following examples describe the use of RTPCART in two key areas — energy deposition kernel generation (using photons) and phantom simulation (using electrons).

Example 1: Energy Deposition Kernel

This example illustrates the use of RTPCART to generate a kernel for use in superposition dose calculations. The input parameter file used is as follows (comments to the right):

`EDK in H2O (10 MeV)`	simulation title
`WATER`	material 1
`DUMMY`	material 2 (not used)
`-0.5,-0.5,-0.5`	x,y,z voxel sizes (negs indicate homogeneous phantom)
`1,1.0`	material number and density of all voxels
`-1.0,0.561,0.050`	K.E. of incident particle (– indicates spectrum follows), global ECUT,PCUT
`0.0`	lower energy of 1st spectral bin
`1.0,17.5`	upper energy of 1st spectral bin, spectral weight
`2.0,22.0`	2nd spectral bin
`3.0,20.4`	3rd spectral bin
`4.0,12.9`	4th spectral bin
`5.0,9.2`	5th spectral bin
`6.0,7.1`	6th spectral bin
`7.0,4.9`	7th spectral bin
`8.0,3.2`	8th spectral bin
`9.0,1.9`	9th spectral bin
`10.0,0.9`	10th spectral bin
`0.0`	indicates end of spectrum
`2`	indicates EDK generation (interaction forcing)
`5.25,5.25,1.25,2,0`	x,y,z coordinates of interaction point, 3–D smearing of kernel origin, total energy scored
`200000,0,0,12.0,123456789`	no. of histories, incident charge, monitor routine off, CPU limit (hrs), random number seed
`20.0,0.2`	depth limit (cm), and survival prob of Russian roulette
`0,0,0,0`	PRESTA PLC,BCA,LCA and FIXTMX turned on
`1,0,0,2`	output options — total energy table, no region masses, no DOTPLOT data, raw data output (central plane only)

Note that specification of XSIZE, YSIZE, and ZSIZE as all negative (line 4) implies a homogeneous phantom (see Section 2.6.3c), while the next line specifies that the voxels are of unit density water. Line 20 places the interaction site at the centre of a voxel in the third layer of the phantom (1.25 cm deep), allowing 2 voxels for backscatter. Three-dimensional

kernel smearing is turned on, and total energy is scored.

The last line of the input file controls output options for the simulation. Part of the output file generated by this simulation is listed in Appendix C, showing the energy distribution table with associated uncertainties. **RTPCART** chooses a single (x, z) plane for table output, usually coinciding with the central axis of the phantom. Setting the **IRAWDATA** flag to 2 (last line of the input file) selects an another option available in **RTPCART** — storing raw data for the central (x, z) plane only. This feature reduces the size of raw data files and allows easier processing by graphics utilities which display energy or dose distribution in a single plane, such as the transputer-based display software developed at Waikato University. Figure 2.11 shows one such plot of an energy deposition kernel generated using the above input file.

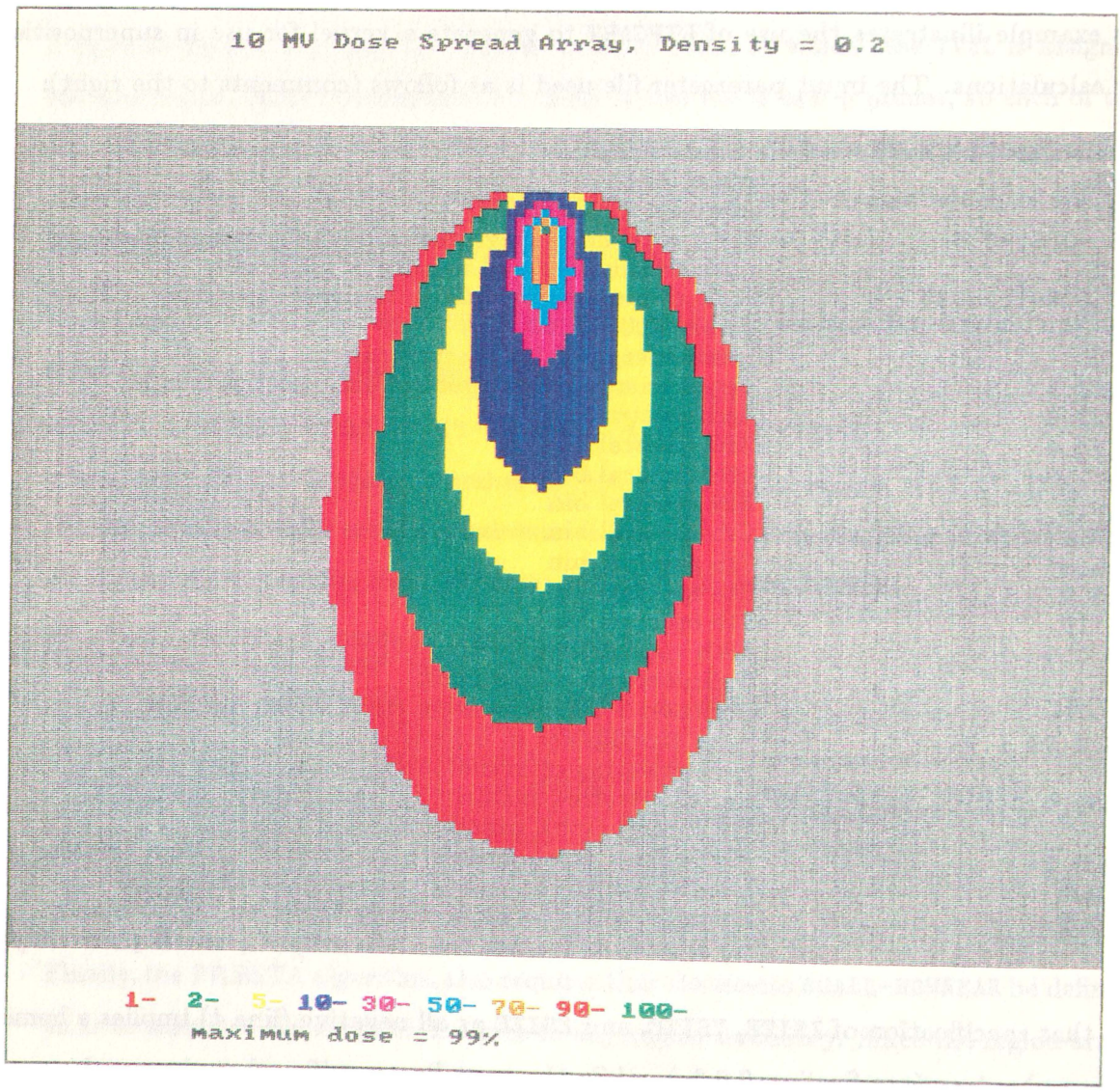

Figure 2.11. Colour-fill plot of energy deposition kernel generated using **RTPCART**. Interaction site is at the central green dot (just below the top of the kernel). A 10 MV spectral beam was incident from the top. Density of the grey area is 0.2 g cm^{-3}, representing lung, and its depth is 10 cm.

Example 2: Cavity Phantom

The second example of **RTPCART** is a simulation of point source *electron beam* into a water phantom containing a small air cavity. This phantom was required by Hoban[51] for work involving electron pencil beams. Figure 2.12 is a two-dimensional schematic of the (three-dimensional) beam and phantom geometry used in the simulation.

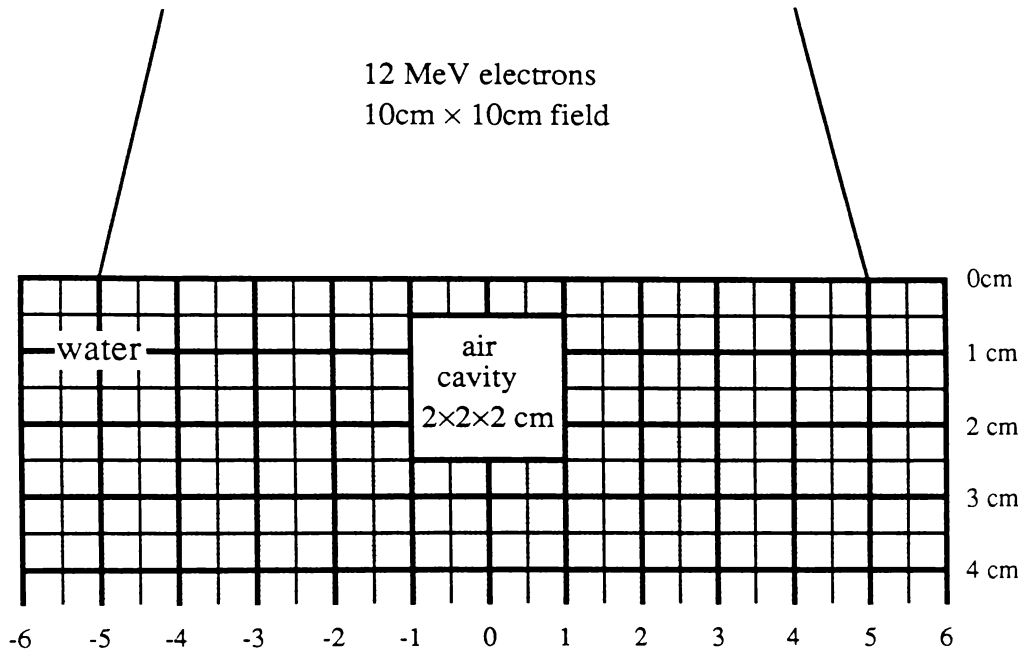

Figure 2.12. Schematic of cavity simulation. A 10 cm × 10 cm square field of 10 MeV monoenergetic electrons is incident on a unit density (water) phantom with a small square air cavity on the beam central axis and 0.5 cm below the surface.

The input file for the simulation is as follows:

`12 MeV electrons into cavity`	simulation title
`WATER`	material 1
`DUMMY`	material 2 (not used)
`-0.5,-0.4,0.5`	x,y,z voxel sizes (x,y negs indicate cavity phantom)
`1,1.0`	material number and density of non-cavity voxels
`1,0.001`	material number and density of cavity voxels
`13,16,11,15,2,5`	voxel indices of cavity boundaries
`12.0,0.561,0.050`	K.E. of incident particle,**ECUT**,**PCUT**
`1`	indicates point source square beam
`7.0,5.0,10.0,100.0`	x,y coordinates of beam centre, beam width, SSD
`75000,-1,0,12.0,123456789`	no. of histories, incident charge, monitor routine off, CPU limit (hrs), random number seed,
`8.0,0.1`	depth limit (cm), and survival prob of Russian roulette
`0,0,0,0`	PRESTA PLC,BCA,LCA and **FIXTMX** turned on
`1,0,0,2`	output options — total energy table, no region masses, no DOTPLOT data, raw data output (central plane only)

Note that the voxels are positioned so that there are four voxels beneath the cavity in the x-direction (as illustrated in Figure 2.12), and five under the cavity in the y-direction (with

each voxel having a y-dimension of 0.4 cm). This means that there is a plane of voxels directly below the *centre* of the cavity in the y-direction, rather than slightly off-centre. Specification of **XSIZE** and **YSIZE** negative, **ZSIZE** positive (line 4) indicates a homogeneous phantom *with a block of different material* (see Section 2.6.3c). The next two lines specify the composition and density of the phantom and block respectively, and in this case the block is a cavity of air, modelled as a water-like material of density 0.001 g cm^{-3}. The next line specifies the cavity boundaries using voxel indices as coordinates. The remander of the input file is similar to that in Example 1, except that the incident particles are monoenergetic 12 MeV *electrons*.

One further modification to this simulation is the addition of *particle splitting* (see Section 2.3.6e). Figure 2.13 illustrates the technique: incident particle coordinates are sampled randomly as before, but if the particle is incident on a 4 cm × 4 cm area near the cavity then it is *split* into 5 identical particles, each with weight 0.2. This increases the number of particles interacting near the cavity, and therefore reduces statistical fluctuations in that region, at the expense of reduced accuracy in regions of the phantom away from the cavity.

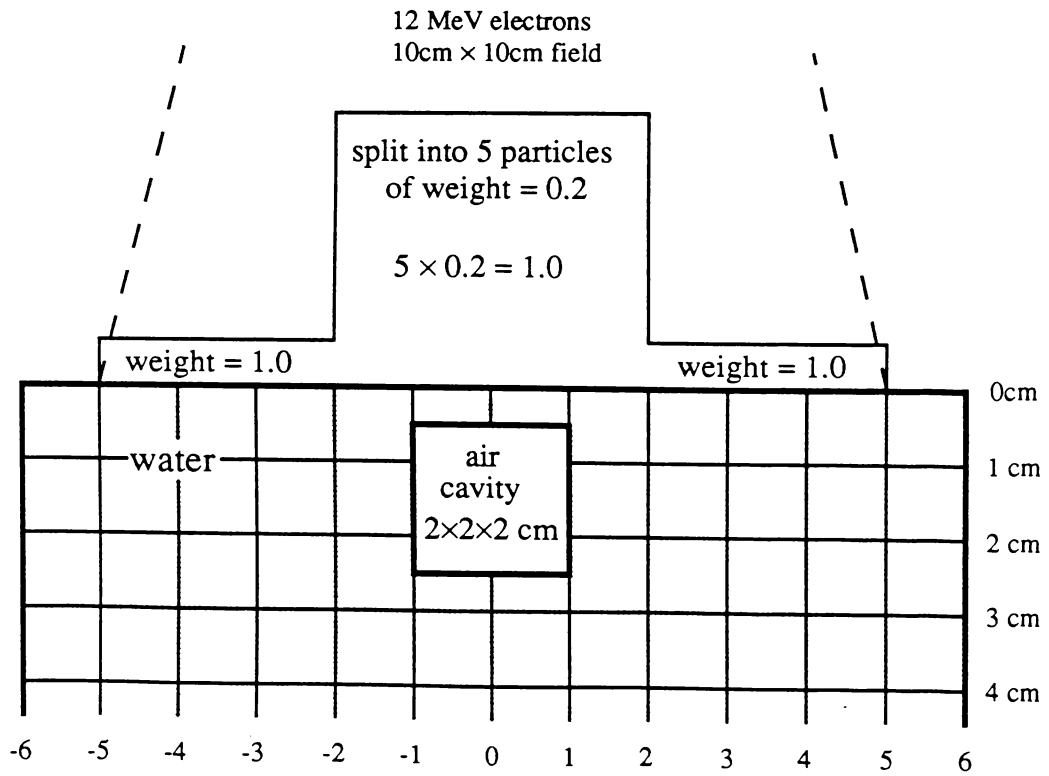

Figure 2.13. Particle splitting near the cavity. The coordinates of an incident particle are randomly selected, then the particle is split if it falls within a 4 cm × 4 cm square region near the cavity. This is done by replacing the particle with 5 identical particles, each of weight 0.2.

Particle splitting was hard-coded into RTPCART by testing the incident particle x and y offsets (XPROJ and YPROJ) prior to calling SHOWER:

```
IF (ABS(XPROJ)<2.0 & ABS(YPROJ)<2.0) [IPARTNO=5;WTIN=0.2;]
ELSE [IPARTNO=1;WTIN=1.0;]
```

then calling SHOWER repeatedly if the particle had been split:

```
"Split particle if option has been selected"
IF (IPARTNO>1) [
 DO I=1,IPARTNO [
  CALL SHOWER(IQIN,EI,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN);
 ]
]
```

The results of the cavity simulation are illustrated in Figure 2.14. This colour-fill plot clearly illustrates that low-density heterogeneities produce significant perturbation of electron beam dose distributions. The Monte Carlo technique illustrates these well, whereas the majority of electron beam dose calculation algorithms either fail to model, or inadequately model, these effects.

2.6.4. RTPEDK — for Spherical Energy Deposition Kernels

Features of RTPEDK

RTPEDK has been written with one purpose in mind: to generate energy deposition kernels for the GRATIS treatment planning system. It is essentially a cut-down version of RTPCYL, with modifications to the scoring geometry, PRESTA macro and output format. Many of RTPCYL's features remain, such as the ability to use an input spectrum, and Russian roulette for particles travelling beyond the phantom limits. A complete listing of RTPEDK can be found in Appendix D, but in summary the main areas of modification are as follows:

(a) Only **energy deposition kernels** are modelled — that is, *photons* are forced to interact at the origin of a *homogeneous* phantom. The $SELECT-PHOTON-MFP macro forces interactions in a manner similar to that described in Section 2.5d, and spectral components are μ-weighted as described in Section 2.6.2b. Energy deposition is swapped from beginning to end of each electron step (see Section 2.6.2g), and energy deposited in each voxel is divided by the total incident energy at the end of the simulation (see Section 2.6.2b).

(b) A **spherical scoring geometry** is used. The phantom is divided into NRADIALS radial shells spaced a distance DELTAR apart, and is also divided into NTHETAS conical regions originating at the origin and symmetric about the incident particle direction.

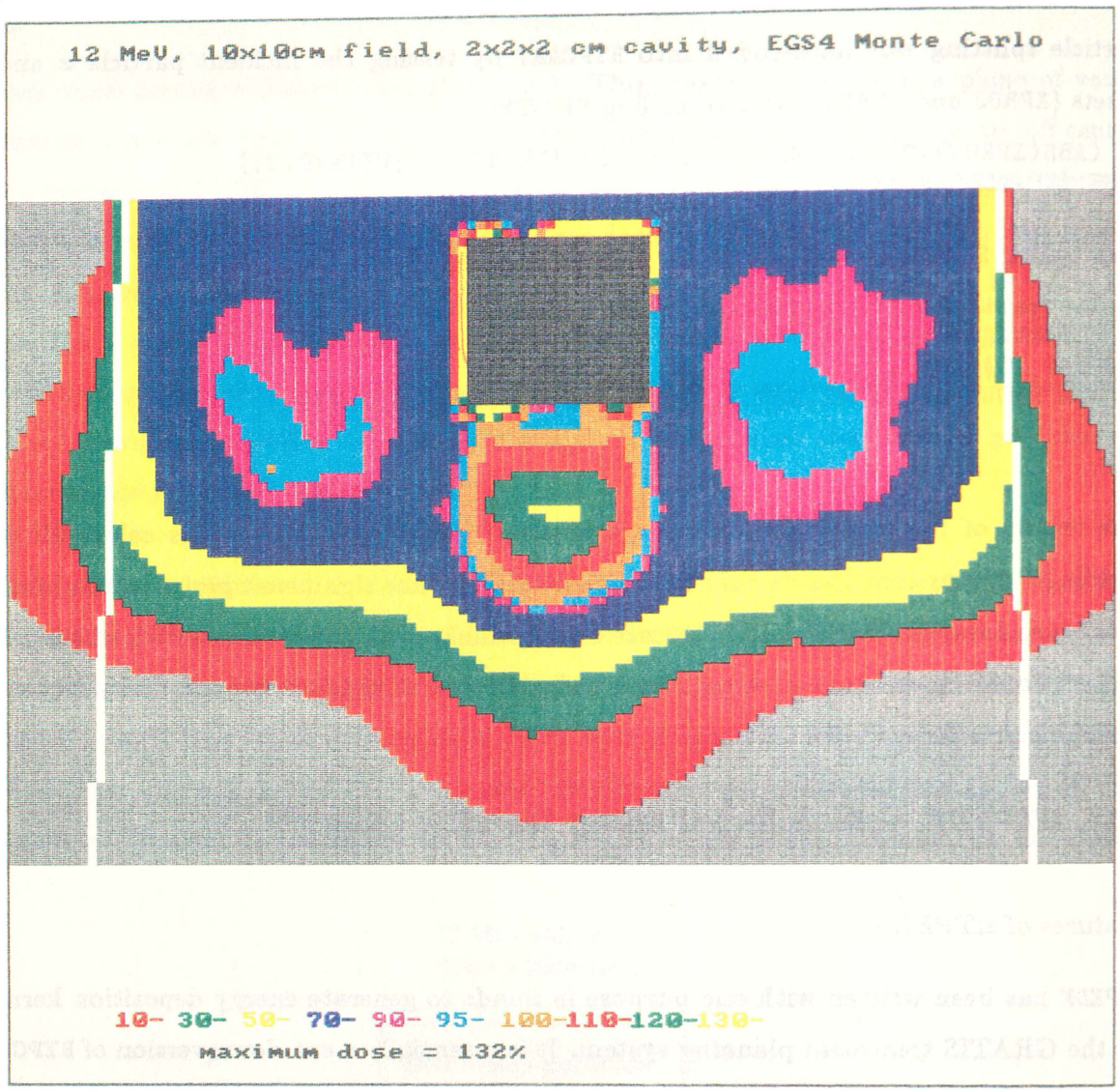

Figure 2.14. Colour-fill plot of dose distribution in cavity simulation. Light grey region represents tissue, dark grey region cavity, and white lines the field limits. Noticeable is the "hot spot" below the cavity and depression of the isodose lines due to the smaller radiological path length in the cavity.

This geometry is essentially that same as that described by Mackie *et al.*[44] for their code **SCASPH**. The polar angle θ between each conical region boundary is implicitly defined as $\theta = 180/N_\theta$, where N_θ is the number of conical regions (**NTHETAS**). Scoring parameters for scattered energy, termed **NRADSCAT**, **DELRSCAT**, and **NTHSCAT**, can be specified independently of the parameters for total and primary energy. Figure 2.15 illustrates the scoring geometry.

(c) **A spherical geometry is used for PRESTA boundaries.** In **RTPCYL**, PRESTA has to be turned off when using spherical coordinates, since the PRESTA boundaries lie on planes between *cylindrical* scoring regions. In **RTPEDK**, PRESTA boundaries are defined to be coincident with the scoring region boundaries. This is done by defining

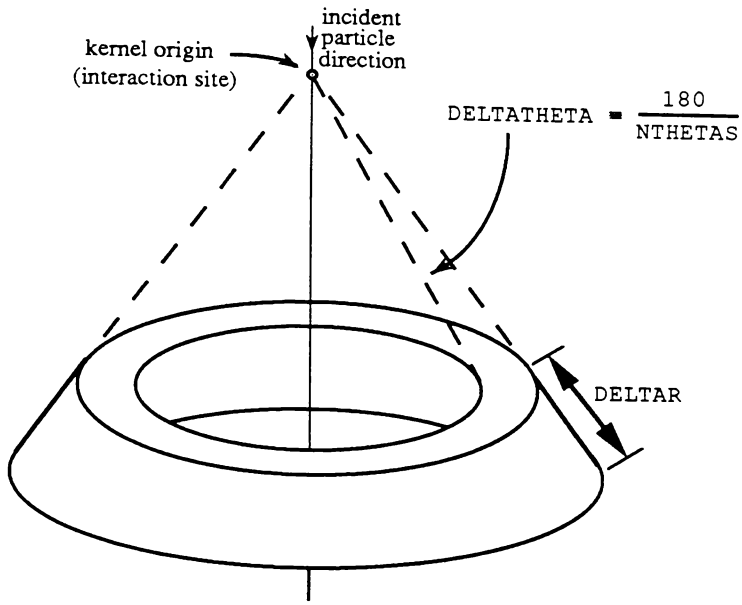

Figure 2.15. Spherical scoring geometry used in RTPEDK.

the PRESTA macro $CALL-HOWNEAR as:

```
REPLACE {$CALL-HOWNEAR(#);} WITH
{;IF ($FLAG1<=1)[ "primary energy"
    RFRAC=AMOD(R,DELTAR);
    THFRAC=AMOD(TH,180.0/NTHETAS);
    {P1}=MIN(MIN(RFRAC,DELTAR-RFRAC),
         R*SIN(MIN(THFRAC,(180.0/NTHETAS)-THFRAC)*(3.14159/180.0)));]
  ELSE[ "scattered energy"
    RFRAC=AMOD(R,DELRSCAT);
    THFRAC=AMOD(TH,180.0/NTHSCAT);
    {P1}=MIN(MIN(RFRAC,DELRSCAT-RFRAC),
         R*SIN(MIN(THFRAC,(180.0/NTHSCAT)-THFRAC)*(3.14159/180.0)));]
;}
```

The above routine is called from within PRESTA to determine the distance to the closest boundary. Note that the particle coordinates R and TH are required in the above macro — they are calculated within HOWFAR:

```
R=SQRT(X(NP)**2+Y(NP)**2+Z(NP)**2);    "R in cm"
IF(R<0.000001)[R=0.000001;]   "avoid division by zero in next line"
TH=ASIN(SQRT(X(NP)**2+Y(NP)**2)/R)*180/3.1415926;    "Theta in degrees"
IF(Z(NP)<0.0)[TH=180.0-TH;] "ensure 0<=TH<=180"
```

and made available to PRESTA via the common block GEOM.

(d) **There is a reduced set of output options.** A reduced set of output options is available when using RTPEDK. DOTPLOT data is available as in RTPCYL (see Section 2.6.2j), and raw dose data for total, primary and/or scattered dose can be produced. For each dose category in the raw data file, the dose category ("TOTAL", "PRIMARY", or "SCATTERED") is listed, followed by an associated set of values for RHOR, DELTAR,

DELTATHETA, NRADIALS and NTHETAS. This information is required to enable GRATIS to perform cartesian interpolation correctly, and also to check that the kernel density (RHOR) expected by GRATIS is consistent with that used in RTPEDK. In addition, a table of doses for the first seven angular bins is produced in the RTPEDK log file (along with associated uncertainties).

Example: Spherical EDK Generation

To generate a raw data file containing a spherical EDK for a 10 MV photon beam, the following input file to RTPEDK was used (comments are listed to the right):

```
10 MV EDK into H2O          simulation title
WATER                       phantom material
1.0                         density of medium
30,0.2,18,60,0.5,18         NRADIALS,DELTAR,NTHETAS,NRADSCAT,DELRSCAT,NTHSCAT
-1.0,0.561,0.050,0.5        K.E. of incident particle (- indicates spectrum follows),
                            global ECUT,PCUT, maximum step size (cm)
0.0                         lower energy of 1st spectral bin
1.0,17.5                    upper energy of 1st spectral bin, spectral weight
2.0,22.0                    2nd spectral bin
3.0,20.4                    3rd spectral bin
4.0,12.9                    4th spectral bin
5.0,9.2                     5th spectral bin
6.0,7.1                     6th spectral bin
7.0,4.9                     7th spectral bin
8.0,3.2                     8th spectral bin
9.0,1.9                     9th spectral bin
10.0,0.9                    10th spectral bin
0.0                         indicates end of spectrum
250000,0,12.0,0,123456789   no. of histories, monitor routine off, CPU limit (hrs)
                            Rayleigh scattering off, random number seed
20.0,0.2                    depth limit (cm) and survival prob of Russian roulette
0,0,0,0                     PRESTA PLC,BCA,LCA and FIXTMX turned on
0,1,1,0                     output options — primary & scattered dose only
```

In the above example a unit density water phantom is divided into 30 spherical shells (each 0.2 cm thick) and 18 cones (spaced every 10 degrees), which are used to score primary energy. Scattered energy is scored in 60 shells (each 0.5 cm thick) with cones also spaced every 10 degrees. 250,000 photons sampled from a 10 MV spectrum are forced to interact at the origin, producing a file containing primary and scattered energy normalised to total incident energy. The primary energy category in the raw data file consists of 30 sets of 18 normalised energies, preceded by header information identifying important parameters in the calculation (see Section 2.6.4d):

```
PRIMARY
RHOR,DELTAR,DELTATHETA,NRADIALS,NTHETAS=
   1.000000     0.200000    10.000000    30    18
0.3511E-01
0.2962E-01
```

```
0.2395E-01
0.1886E-01
0.1430E-01
0.1018E-01
0.6909E-02
0.4270E-02
0.2378E-02
.
.
.
```

A similar array for scattered energy is also contained in the raw data file.

2.7. Summary

Monte Carlo techniques in general — and EGS4 in particular — seek to accurately simulate the physical interactions between particles. Although the details of photon and electron transport may be different (as with Class I and Class II electron transport schemes), EGS4, ETRAN and other Monte Carlo codes are essentially similar in their approaches. They utilise the sampling of cross-sections and mean free paths to completely simulate the interactions of photons, but apply multiple scattering theory to reduce the computation needed for calculating electron paths.

The EGS4 Monte Carlo code has gained wide acceptance in the medical physics community, due primarily to its ease of use. Features such as the PEGS4 preprocessor, the macro facility provided by the Mortran precompiler, availability of geometry packages, and the ease with which simulation geometry and input parameters can be modified has enabled such wide use. Weaknesses in the system such as step-size dependence have been well documented, with improvements such as PRESTA continuing to be made. The system has been extensively benchmarked, so that users can have confidence in the results that EGS4 produces. However, although EGS4 is comparatively easy to use, it is advantageous that the user has a knowledge of the underlying theory (which may help the detection of errors in the user code) and available short-cuts (such as variance reduction techniques) which may allow more efficient simulation.

The Monte Carlo technique can simulate many experimental situations which may not be physically achievable or measurable. It requires a minimum of experimental apparatus (a computer) and produces reproducible results given an adequate number of particle histories. As the computational power of computers increases and the problems tackled become more complex, Monte Carlo will continue to become either the only available method, or the method of choice for a wide variety of medical physics problems.

References

(1) Nahum A.E., Overview of photon and electron Monte Carlo. In *Monte Carlo Transport of Electrons and Photons, p3*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(2) Wilson R.R., Monte Carlo study of shower production. *Phys. Rev.* (1952), **86** 261.

(3) Meyer H.A. (editor), *Symposium on Monte Carlo Methods.* John Wiley and Sons, New York (1954).

(4) McCracken D.D., The Monte Carlo method. *Sci. Am.* (1955), **192** 90.

(5) Kahn H., *Applications of Monte Carlo.* USAEC Report AECU-3259 (1954), 62.

(6) Berger M.J., Monte Carlo calculations of the penetration and diffusion of fast charged particles. In *Methods in Computational Physics, Vol. 1, p135*, edited by Alder B., Fernbach S., and Rotenberg M., Academic Press, New York (1963).

(7) Turner J.E., Wright H.A., and Hamm R.N., Review article: A Monte Carlo primer for health physicists. *Health Phys.* (1985), **48** 717.

(8) Raeside D.E., Monte Carlo principles and applications. *Phys. Med. Biol.* (1976), **21** 181.

(9) Seltzer S.M., An overview of ETRAN Monte Carlo methods. In *Monte Carlo Transport of Electrons and Photons, p153*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(10) Halbleib J., Structure and operation of the ITS Code System. In *Monte Carlo Transport of Electrons and Photons, p249*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(11) Nelson W.R., Hirayama H. and Rogers D.W.O., *The EGS4 Code System.* Stanford Linear Accelerator Center Report SLAC-265 (1985).

(12) Berger M.J. and Wang R., Multiple-scattering angular deflections and energy-loss straggling. In *Monte Carlo Transport of Electrons and Photons, p21*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(13) Seltzer S.M., Hubbell J.H., and Berger M.J., Some theoretical aspects of electron and photon dosimetry. In *National and International Standardisation of Radiation Dosimetry, Vol. II, p3*, IAEA publication STI/PUB/471 (1978).

(14) Williamson J.F., Monte Carlo simulation of photon transport phenomena: Sampling techniques. In *Monte Carlo Simulation in the Radiological Sciences, p53*, edited by Morin R.L., CRC Press (1988).

(15) Ford R.L. and Nelson W.R., *The EGS Code System: Computer Programs for the Monte Carlo Simulation of Electromagnetic Cascade Showers (Version 3).* Stanford Linear Accelerator Center Report SLAC-210 (1978).

(16) Nelson W.R. and Rogers D.W.O., Structure and operation of the EGS4 Code System. In *Monte Carlo Transport of Electrons and Photons, p287*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(17) Cook A.J., *Mortran3 Users Guide.* SLAC Computation Research Group Technical Memorandum CGTM-209 (1983).

(18) Inmos, *The Transputer Reference Manual.* Prentice-Hall, U.K. (1988).

(19) Murray D.C., Hoban P.W., Round W.H., Graham I.D, and De Vel O.Y., Radiotherapy treatment planning using transputers. *N.Z. Journal of Computing* (1989), **1 (2)** 30.

(20) 3L Limited, *Parallel FORTRAN User Guide.* 3L Limited, Livingston, Scotland (1988).

(21) Knuth D.E., *The Art of Computer Programming, Volume II.* Addison Wesley, Reading, Massachusetts (1981).

(22) Morin R.L., Raeside D.E., Goin J.E., and Widman J.C., Random number generation and testing. In *Monte Carlo Simulation in the Radiological Sciences, p37*, edited by Morin R.L., CRC Press (1988).

(23) Duane S., (National Physical Laboratory, Teddington, U.K.). *Private communication* (1989).

(24) Rogers D.W.O., Low energy electron transport with EGS. *Nucl. Instr. and Meth.* (1984), **A227** 535

(25) Bielajew A.F. and Rogers D.W.O., Electron step-size artefacts and PRESTA. In *Monte Carlo Transport of Electrons and Photons, p115*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(26) Bielajew A.F. and Rogers D.W.O., PRESTA: The parameter reduced electron-step transport algorithm for electron Monte Carlo transport. *Nucl. Instr. and Meth.* (1987), **B18** 165.

(27) Bielajew A.F. and Rogers D.W.O., Variance-reduction techniques. In *Monte Carlo Transport of Electrons and Photons, p407*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(28) McGrath E.J. and Crawford D.F., *Techniques for Efficient Monte Carlo Simulation, Vols I,II and III.* Radiation Shielding Information Center, Oak Ridge National Laboratory Report ORNL-RSIC-38 (1975).

(29) Rogers D.W.O. and Bielajew A.F., A comparison of EGS and ETRAN. In *Monte Carlo Transport of Electrons and Photons, p323*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(30) Rogers D.W.O. and Bielajew A.F., Experimental benchmarks of EGS. In *Monte Carlo Transport of Electrons and Photons, p307*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(31) Rogers D.W.O, More realistic Monte Carlo calculations of photon detector response functions. *Nucl. Inst. and Meth.* (1982), **199** 531.

(32) Nahum A.E., Simulation of dosimeter response and interface effects. In *Monte Carlo Transport of Electrons and Photons, p523*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(33) Bielajew A.F., Rogers D.W.O. and Nahum A.E., The Monte Carlo simulation of ion chamber response to ^{60}Co — resolution of anomalies associated with interfaces. *Phys. Med. Biol.* (1985), **30** 419.

(34) Mohan R., Chui C. and Lidofsky L., Energy and angular distributions of photons from medical linear accelerators. *Med. Phys.* (1985), **12** 592.

(35) Write to W.R. Nelson, Radiation Physics Group, BIN 48, Stanford Linear Accelerator Center, P.O. Box 4349, Stanford, California 94309, U.S.A.

(36) Write to Susan Walker, Institute of Applied Physiology and Medicine, Seattle, Washington 98122, U.S.A.

(37) Cowan R. and Nelson W.R., *Producing EGS4 Shower Displays with Unified Graphics.* Stanford Linear Accelerator Center Report TN-87-3 (1987).

(38) Nelson W.R. and Jenkins T.M., *Writing Subroutine HOWFAR for EGS4.* Stanford Linear Accelerator Center Report TN-87-4 (1988).

(39) Straker E.A., Stevens P.N., Irving D.C., and Cain V.R., *MORSE-CG, General Purpose Monte-Carlo Multigroup Neutron and Gamma-ray Transport Code with Combinatorial Geometry.* Radiation Shielding Information Center, Oak Ridge National Laboratory Report Number CCC-203 (1976).

(40) Rogers D.W.O., Bielajew A.F. and Nahum A.E., Ion chamber response and A_{wall} correction factors in a ^{60}Co beam by Monte Carlo simulation. *Phys. Med. Biol.* (1985), **30** 429.

(41) Andreo P., Stopping-power ratios for dosimetry. In *Monte Carlo Transport of Electrons and Photons, p485*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(42) Del Guerra A. and Nelson W.R., Positron emission tomography applications of EGS. In *Monte Carlo Transport of Electrons and Photons, p469*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(43) Bielajew A.F., Rogers D.W.O., Cygler J. and Battista J.J., A comparison of electron pencil beam and Monte-Carlo calculational methods. In *The Use of Computers in Radiation Therapy, p65*, edited by Bruinvis I.A.D. *et al.*, Elseviere Science Publishers, Holland (1987).

(44) Mackie T.R., Bielajew A.F., Rogers D.W.O. and Battista J.J., Generation of photon energy deposition kernels using the EGS Monte Carlo Code. *Phys. Med. Biol.* (1988), **33** 1.

(45) Chan H.-P. and Doi K., Monte Carlo simulation in diagnostic radiology. In *Monte Carlo Simulation in the Radiological Sciences, p103*, edited by Morin R.L., CRC Press (1988).

(46) Widman J.C., Monte Carlo simulation in nuclear medicine. In *Monte Carlo Simulation in the Radiological Sciences, p193*, edited by Morin R.L., CRC Press (1988).

(47) Nath R., Monte Carlo simulations in radiation therapy. In *Monte Carlo Simulation in the Radiological Sciences, p209*, edited by Morin R.L., CRC Press (1988).

(48) Ito A., Electron track simulation for microdosimetry. In *Monte Carlo Transport of Electrons and Photons, p361*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

(49) Metcalfe P.E., Hoban P.W., Murray D.C. and Round W.H., Modelling polychromatic high energy photon beams by superposition. *Austral. Phys. Eng. Sci. Med.* (1989), **12 (3)** 138.

(50) Metcalfe P.E., Hoban P.W., Murray D.C. and Round W.H., Beam hardening of 10MV x-rays: Analysis using a convolution/superposition method. *Phys. Med. Biol.* (1990), **35** (11) 1533.

(51) Hoban P.W., *Lateral Electron Disequilibrium in Radiotherapy Treatment Planning.* University of Waikato D.Phil. Thesis (1991).

(52) Metcalfe P.E., *X-ray Beam Modelling in Radiotherapy: The Effect of Lung Inhomogeneities.* University of Waikato D.Phil. Thesis (1990).

Chapter 3

Superposition and Convolution

3.1. Introduction

In Chapter 1 a variety of radiotherapy dose calculation algorithms were discussed. The majority of these algorithms assume that electrons liberated by incident and scattered photons deposit their energy at the site of interaction. In reality they range away from the interaction site, depositing energy along tracks of significant length, especially when considering high energy (linear accelerator) x-rays. One approach which *does* model this phenomenon is the *Monte Carlo* technique, discussed in Chapter 2. Although the potential accuracy of this technique is attractive, the computational effort required is a barrier to its implementation in a clinical situation.

Another method which models the physical process of electron ranging is the *superposition* technique, often referred to as *convolution* when performed with invariant kernels. First proposed by Dean,[1] this method was developed by Mackie *et al.*,[2] Boyer and Mok,[3] Mohan *et al.*,[4] Ahnesjö *et al.*,[5] and others, and has since been investigated by several other authors.[6–9] It has also been the subject of considerable research at this author's institution.[10–15] The need for truly three-dimensional treatment planning has been demonstrated,[16] and superposition is also well suited to this approach, as Mohan[17] and Mohan *et al.*[18] have discussed.

In the superposition/convolution technique a distribution of total primary energy released at each interaction site is convolved with one or more distributions describing the spread of energy from that point. In this chapter the first of these distributions will be called the *terma*[5] and the second the *energy deposition kernel* (EDK).[3,19] Energy deposition kernels have also been termed dose spread arrays,[2] differential pencil beams,[4] and point spread functions.[5] The `RTPCART` and `RTPEDK` Monte Carlo codes used to generate energy deposition kernels have been presented in Chapter 2, and the mathematical and physical basis for the superposition approach is presented briefly in Section 3.2 of this chapter. More detailed treatment of superposition mathematics, including density-scaling of the energy deposition kernel, can be found elsewhere.[14,15] Section 3.3 describes some key aspects

Sections of this chapter appear in *Australas. Phys. Eng. Sci. Med.*, **12** (**3**) 128–137 (1989), under the title "3-D Superposition for Radiotherapy Treatment Planning Using Fast Fourier Transforms."

concerning implementation of the superposition/convolution algorithm — in particular the determination of terma, formation of energy deposition kernels, and calculation of dose using real space superposition and Fourier space convolution.

A dose calculation algorithm, although critical, is just a small part of a complete treatment planning system. One such system, GRATIS, has been installed by this author at Waikato University — refer to Section 3.4 for a brief description of this system. Section 3.5 describes how the superposition algorithm has been implemented under GRATIS.

3.2. Superposition Theory

Several reviews of the theory behind superposition with respect to external beam radiotherapy can be found in the literature, in particular Ahnesjö *et al.*,[5] and Boyer and Mok.[3] The particular formalism used at the author's institution is described more fully by Hoban[15] and Hoban *et al.*,[12] but in summary the dose $D(\mathbf{r})$ to a point $\mathbf{r}$ from a monoenergetic beam is given by a superposition integral over the entire volume V:

$$D(\mathbf{r}) = \frac{1}{\rho(\mathbf{r})} \int_V T(\mathbf{r}')\rho(\mathbf{r}')H(\mathbf{r}-\mathbf{r}',\mathbf{r}')\,\mathrm{d}v \quad , \tag{3.1}$$

where $T(\mathbf{r}')$ is the terma at the interaction site $\mathbf{r}'$, $\rho(\mathbf{r}')$ is the mass density at that site and $H(\mathbf{r}-\mathbf{r}',\mathbf{r}')$ is the fraction of energy liberated at $\mathbf{r}'$ which is deposited per unit volume at a point a position $\mathbf{r}-\mathbf{r}'$ from the interaction site. Note that in general the kernel H is a function of the position $\mathbf{r}'$ of the interaction voxel. In Equation 3.1 the kernel must be normalised such that:

$$\int_V H(\mathbf{r}-\mathbf{r}')\,\mathrm{d}v = 1 \quad . \tag{3.2}$$

In general, both the terma and energy deposition kernel are determined using *polyenergetic* beam spectra. Terma is calculated by first determining the fluence differential in energy $\Phi_E(\mathbf{r}')$ for each spectral component,[5] which is simply the fluence at $\mathbf{r}'$ due to photons of energy E:

$$\Phi_E(\mathbf{r}') = \Phi_E(\mathbf{r}'_0)\left(\frac{r'_0}{r'}\right)^2 \mathrm{e}^{-\mu(E)\left(d(\mathbf{r}')-d(\mathbf{r}'_0)\right)} \quad , \tag{3.3}$$

where $\mathbf{r}'_0$ is a point at the surface of the calculation region intersected by a ray from the source to $\mathbf{r}'$, $d(\mathbf{r}')$ and $d(\mathbf{r}'_0)$ are the *radiological depths* (equivalent depths in water) at $\mathbf{r}'$ and $\mathbf{r}'_0$ respectively, and $\mu(E)$ is the linear attenuation coefficient of water at energy E. The terma differential in energy $T_E(\mathbf{r}')$ can then be written as

$$T_E(\mathbf{r}') = \left(\frac{\mu(E)}{\rho(\mathbf{r}')}\right)_{\mathbf{r}',E} \cdot E \cdot \Phi_E(\mathbf{r}') \quad . \tag{3.4}$$

For a beam modelled by N spectral components the total terma $T(\mathbf{r}')$ can now be expressed by

$$T(\mathbf{r}') = \sum_{i=1}^{N} T_{E_i}(\mathbf{r}') \quad , \tag{3.5}$$

where E_i is the energy of the ith spectral component.

In a similar manner the polyenergetic energy deposition kernel must be formed from a weighted sum of spectral components. If the terma liberated at the interaction site $\mathbf{r}$ due to a spectral component of energy E_i is $T_{E_i}(\mathbf{r})$ then

$$H(\mathbf{r}-\mathbf{r}') = \frac{\sum_{i=1}^{N} T_{E_i}(\mathbf{r}) H_{E_i}(\mathbf{r}-\mathbf{r}')}{\sum_{i=1}^{N} T_{E_i}(\mathbf{r})} \quad , \tag{3.6}$$

where $H_{E_i}(\mathbf{r}-\mathbf{r}')$ is fraction of energy liberated at $\mathbf{r}'$ by the spectral component of energy E_i which is deposited at $\mathbf{r}$. This treatment is less rigorous than performing a separate superposition for each spectral component, but this method can be modified to account for beam hardening,[13,15] and requires considerably less computation.

3.3. Implementation of the Superposition/Convolution Algorithm

3.3.1. Introduction

As demonstrated in Section 3.2, superposition is a conceptually simple approach to radiotherapy dose computation. However, as with many other algorithms, *implementation* of superposition is less simple, requiring consideration of the *practicalities* of calculation using a computer.

One of the most important decisions to be made in implementing superposition is the choice of a coordinate system. The superposition algorithm is presented in Equations 3.1 to 3.6 using a *vector* representation — the computational analogue of this approach would be to transform the underlying cartesian coordinate system used for patient (CT) data into a spherical polar coordinate system centred on each point of calculation. The energy deposition kernel, also represented in spherical polar coordinates, could then be superimposed on each terma point, so that the origins of both coordinate systems coincide. Rectilinear density scaling of the kernel would then be a simple matter.

However, a unique representation of the spherical polar terma array would be required for each interaction point. Considerable computation is required for coordinate transformations of this type, so a less ideal but spatially invariant coordinate system must be used in

a practical treatment planning system. A cartesian system is an obvious choice since it is suitable for superposition without further transformation, and is compatible with CT scanner input and display terminal output. In such a cartesian coordinate system, Equation 3.1 is replaced by the following equation, used to calculate the dose $D(i,j,k)$ to a rectangular volume element or *voxel* of mass density $\rho(i,j,k)$:

$$D(i,j,k) \;=\; \frac{1}{\rho(i,j,k)} \sum_{i'} \sum_{j'} \sum_{k'} \rho(i',j',k') \cdot T(i',j',k') \cdot H(i-i', j-j', k-k') \quad , \quad (3.7)$$

where $T(i',j',k')$ is the terma at the interaction site, and $H(i-i', j-j', k-k')$ is the value of the energy deposition kernel at the deposition site. In general H is also a function of the interaction voxel position (i',j',k'), due to beam hardening and the presence of inhomogeneities. The summation in Equation 3.7 is performed over all voxels in the medium which contribute scattered dose to (i,j,k). Equations 3.2 to 3.6 can be similarly modified, replacing $\mathbf{r}$ by (i,j,k) and $\mathbf{r}'$ by (i',j',k').

The following sections outline the implementation of the three main components of the superposition process: terma calculation (Section 3.3.2); energy deposition kernel calculation (Section 3.3.3); and dose computation using real space superposition (Section 3.3.4) and Fourier space convolution (Section 3.3.5).

3.3.2. Terma Calculation

The first step in terma calculation is to form an array containing the electron densities of each voxel in the region of interest. This is done by accessing data files created by a Computerised Tomography (CT) scanner, such as the Waikato Hospital's *Siemens DRH* scanner, then converting the CT numbers into relative electron densities using the tissue and bone calibration lines as determined by Metcalfe,[14] Metcalfe and Beckham,[20] and Metcalfe *et al.*[21] In this section it is assumed that the terma grid is aligned with the beam central axis — although in general this may not be so, in which case a coordinate transformation is required (see Section 3.5).

The next step is to determine the radiological depth of each voxel. In a homogeneous medium this calculation is trivial, but for a heterogeneous medium a ray traced from the beam source to the voxel centre may pass through many regions, each of differing composition. To adequately model this situation the author has developed a technique designed to produce highly accurate radiological depths using relatively little computation. Figure 3.1 illustrates how rays are traced from the source of the beam through the centre of each *surface* voxel and beyond to the lower limit of the patient or phantom. If these rays intersect both the upper *and* lower surfaces of a voxel then radiological depth is assigned to

that voxel by averaging the radiological depths of the upper and lower surfaces (for the ray shown in Figure 3.1 this occurs for the shaded voxels). This process is then repeated with any *second*-layer voxels which have not already been assigned a radiological depth by the top-layer ray traces. The third and subsequent layers are processed in the same way, and as each successive layer is processed a rapidly increasing proportion of the voxels have already been calculated from previous layers, so computation is less with each layer. Any voxels with more than one intersecting ray can be assigned the average of those depths, although in practice this is not done since the depth assignments have been found to be almost identical.

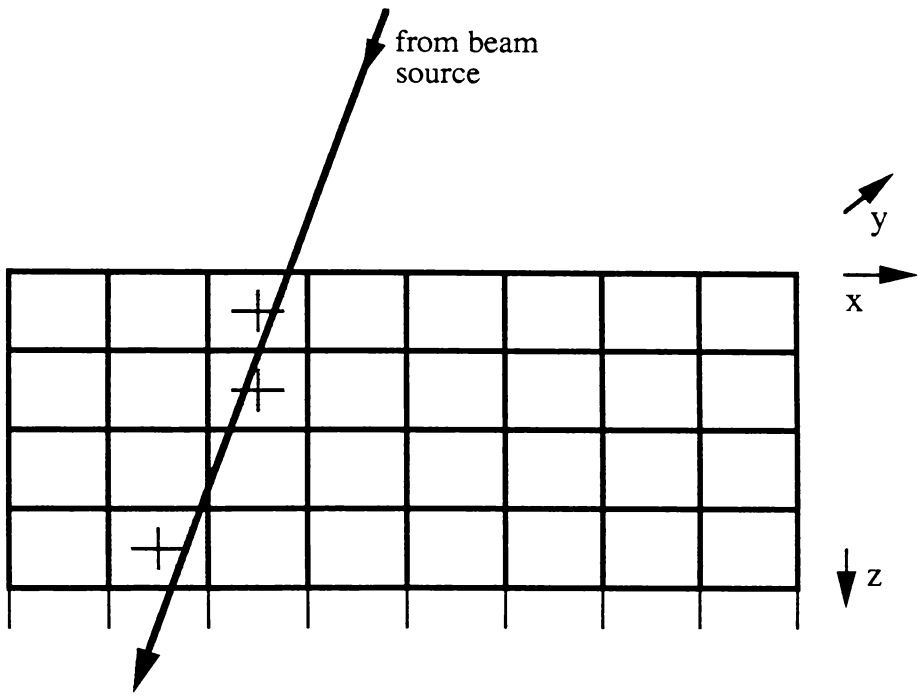

Figure 3.1. Ray tracing to determine radiological depth. Rays are traced from the beam source to the lower limit of the medium. If both the upper and lower voxel boundaries are intersected (shaded voxels) then radiological depth is assigned to that voxel.

Use of the above approach results in only relatively few rays being traced through the medium. This is desirable because ray tracing in general can require large amounts of computing time. To further reduce this problem the ray trace algorithm used is that described by Siddon,[22] in which the ray is described by a set of three parametric equations (one for each dimension). Intersections with voxel planes are easily described in terms of the parameter, so by merging the three parametric sets in order, the lengths of voxel intersections and hence the radiological depth can readily be calculated. The author's implementation of this algorithm is presented in Appendix E. It is significantly faster than conventional ray

tracing techniques for large array sizes,[22] and allows accurate determination of radiological depth in a relatively small computation time (8.3 CPU seconds on a DEC VAX-6200 series computer for a $16 \times 16 \times 100$ terma array; or 8.0 CPU seconds on a Sun SPARCstation IPC for a $22 \times 22 \times 69$ terma array).

Figure 3.2 presents a pseudocode description of the complete process of radiological depth determination. Note that the *electron* density relative to water $\rho_e^w(i,j,k)$ has been used instead of mass density — this approach is valid because Compton interactions dominate at linear accelerator energies. It is also convenient to use ρ_e^w, since CT numbers are more readily converted to this parameter.

Having determined the radiological depth of each voxel it is now possible to calculate the *terma* released at each voxel. *Kerma* is also required if the beam hardening correction described by Hoban[15] is to be used. In addition to radiological depth, the following entities are also needed:

(a) Beam size and shape, beam origin coordinates, and incident fluence. Incident fluence is typically represented by a square matrix or *fluence map* defined at the source-to-surface distance.

(b) The incident photon spectrum, where the spectrum is divided up into a number of weighted components. In general this spectrum will be different for each point in the fluence map. It can be determined by modelling the linear accelerator treatment head using Monte Carlo techniques, as described by Mohan *et al.*[23]

(c) Mass attenuation coefficients for water, obtained by interpolating data from Johns and Cunningham.[24] Mass absorption coefficients are also required if kerma is to be calculated.

(d) An array of patient electron densities relative to water, obtained from CT data by using an appropriate CT calibration line. This array has already been calculated when determining radiological depth.

Figure 3.3 shows a pseudocode representation of the terma calculation process. Note that since *electron* densities have been obtained from the original CT data, and *electron* densities have been used to calculate radiological depth, the mass density ρ used to calculate Φ_E (Equation 3.3) and T_E (Equation 3.4) must be replaced by relative electron density ρ_e^w.

The first step in the calculation is to determine the source-to-surface distance, and also the sum of all the spectral component weights. Then the algorithm steps through each voxel

```
constant X,Y,Z;
type point =  record
                  x,y,z : real
              end;
var   depth : array[0..X,0..Y,0..Z] of real;
      relative_electron_density : array[0..X,0..Y,0..Z] of real;

procedure radiological_depths;
   var   start_point, end_point : point;
         i,j,k : integer;
   start_point := beam origin;
   {Trace surface voxels}
   do i := 0..X
      do j := 0..Y
         end_point := centre of voxel (i,j,0);
         extend end_point to beyond limits of patient;
         ray_trace(start_point,end_point)
      end do
   end do;
   {Trace all other unassigned voxels}
   do k := 0..Z
      do i := 0..X
         do j := 0..Y
            if (voxel has not been assigned)
               end_point := centre of voxel (i,j,k);
               extend end_point to beyond limits of patient;
               ray_trace(start_point,end_point)
            end if
         end do
      end do
   end do;
   end;

procedure ray_trace(point start,end);
   var   l,m,n : integer;     {voxel coordinates}
         distance : real;     {distance between successive intersections}
         rpl : real;          {total radiological path length so far}
   compile list of voxel intersections using Siddon's method;
   rpl := 0.0;
   do (for each voxel intersection)
      calculate distance between intersection and previous intersection;
      determine intersected voxel coordinates (l,m,n);
      rpl := rpl + (distance × relative_electron_density[l,m,n] / 2);
      if (ray intersects upper and lower surfaces of voxel (l,m,n))
         depth[l,m,n] := depth[l,m,n] + rpl
      end if;
      rpl := rpl + (distance × relative_electron_density[l,m,n] / 2)
   end do
   end;
```

Figure 3.2. Pseudocode description of the algorithm to determine radiological depth. The procedure *radiological_depths* determines the radiological depth of each voxel in the region of interest by repeatedly calling *ray_trace*, passing as parameters the end points of the ray to be traced.

in the region of interest, tracing rays back to the beam origin from each of the four top corners of the voxel. These rays intersect with the fluence map, and the fluence map pixels closest to each point of intersection are averaged to yield the corresponding surface fluence — this approach is necessary to account accurately for regions of high fluence gradient, especially near the beam edges. Then the contribution to terma and kerma from each

```
constant X,Y,Z,COMPONENTS;
type spectral_component =   record
                              energy,weight,mu_water,mu_ab_water : real;
                            end;
     array_type = array[0..X,0..Y,0..Z] of real;
var  rad_depth,relative_electron_density,terma,kerma : array_type;
     spectrum : array[0..COMPONENTS - 1] of spectral_component;

procedure calculate_terma;
  var  ssd,sum_of_weights,source_distance : real;
       fluence,component,terma_diff,kerma_diff : real;
       i,j,k,l : integer;
  ssd := distance from beam source to patient surface;
  sum_of_weights := 0.0;
  do l := 0..COMPONENTS-1
    sum_of_weights := sum_of_weights + spectrum[l].weight;
  end do
  do k := 0..Z
    do i := 0..X
      do j := 0..Y
        terma[i,j,k] := 0.0;
        kerma[i,j,k] := 0.0;
        fluence := average of four nearest voxels in fluence map;
        source_distance := distance from beam source to voxel (i,j,k);
        do l := 0..COMPONENTS-1
          component := spectrum[l].weight ×
                exp(-spectrum[l].mu_water × rad_depth[i,j,k]) ×
                spectrum[l].energy;
          terma_diff := terma_diff + component × spectrum[l].mu_water;
          kerma_diff := kerma_diff + component × spectrum[l].mu_ab_water
        end do
        terma[i,j,k] := terma_diff × fluence ×
              (ssd^2 / source_distance^2) /
              sum_of_weights;
        kerma[i,j,k] := kerma_diff × fluence ×
              (ssd^2 / source_distance^2) /
              sum_of_weights;
        end do
      end do
    end do
  end do
  end;
```

Figure 3.3. Pseudocode description of terma and kerma calculation.

spectral component is calculated by forming the product of the spectral component weight, an attenuation factor to account for radiological depth, the spectral component energy, and the appropriate mass attenuation coefficient (μ for terma, μ_{ab} for kerma). Finally, these values are multiplied by the surface fluence and an inverse square correction factor, and divided by the sum of the component weights.

The above approach is essentially the same as that described in Equations 3.3 to 3.5, except that some terms have been moved outside the summation to make the calculation more efficient. Use of a number of spectral components automatically accounts for beam hardening with increasing depth, which Metcalfe *et al.*[13] have shown to be necessary for accurate results when using superposition.

3.3.3. Energy Deposition Kernel Calculation

Since there is no physical way in which photons can be forced to interact experimentally at only a single point in a homogeneous medium, a more indirect approach is necessary in order to obtain energy deposition kernels. A deconvolution technique used by Chui and Mohan[25] to obtain pencil beams could be adapted to this problem, but a more common approach is the use of Monte Carlo simulation. Since the incident photon spectrum must already be known in order to accurately calculate terma, the Monte Carlo technique requires no additional experimentally obtained data. The remainder of this section describes two Monte Carlo methods which can be used to generate cartesian kernels.

I. Direct Generation of Cartesian Kernels

The most obvious way of generating cartesian energy deposition kernels is to generate them directly using a Monte Carlo code based on a cartesian geometry. This approach has been used to generate kernels for most of the superposition work at Waikato University, including that described by Metcalfe *et al.*[11] and Hoban *et al.*[12] Originally this was done with a purpose-built EGS4[26] user code written exclusively for the computation of energy deposition kernels, using EGS4 simulation parameters described by Metcalfe *et al.*[11] However, for later simulations the general-purpose user code **RTPCART** (written by this author) was used. Chapter 2 discusses details of phantom and scoring geometry, electron transport using PRESTA, interaction forcing, variance reduction techniques and other technical aspects of the simulations using **RTPCART**, but some important characteristics of the simulation are also noted here:

(a) The incident photon spectrum used for the 10 MV beam of the Clinac–18 linear accelerator comprises 10 components, as documented by Metcalfe *et al.*[11] It is based on a 23-component spectrum calculated by Mohan *et al.*,[23] who used EGS4 to simulate the Clinac–18 treatment head.

(b) Since energy deposition kernels generated in homogeneous media are symmetrical about $x = 0$ and $y = 0$, energy deposited in the four (x, y) quadrants can be "folded over" into one quadrant, as discussed in Chapter 2, Section 2.6.3e. This approach increases the efficiency of kernel generation. In many simulations the voxels have a square (x, y) cross-section, and statistical variance can be further reduced by averaging pairs of voxels (i, j, k) and (j, i, k), where i, j and k are the x, y and z voxel indices, respectively.

(c) In early simulations at Waikato University, all incident photons were forced to interact at the *centre* of the kernel's interaction voxel. However, since terma is liberated from *all*

points within each patient or phantom voxel, photon interactions are more faithfully modelled if similar "smearing" of interaction sites is employed during kernel calculation. RTPCART does this using an approach described in Chapter 2, Section 2.6.3d. Smearing the interaction sites results in only a small difference to the energy deposition kernel, even in voxels close to the interaction voxel (see Figure 3.4). This is because the electrons liberated from the interaction site deposit energy at an approximately constant rate as they traverse the medium, and hence the energy deposited by a photon interacting at the voxel centre is a good approximation to the *mean* energy deposited in the voxel by smeared photons. However, photons interacting near the edge of the interaction voxel result in more energy being scattered out of the central axis voxels, so that the smeared energy deposition kernel has slightly lower central axis values and correspondingly higher off-axis values. These differences are illustrated quantitatively in Figure 3.4 for a kernel generated using a 10 MV Clinac–18 spectrum.

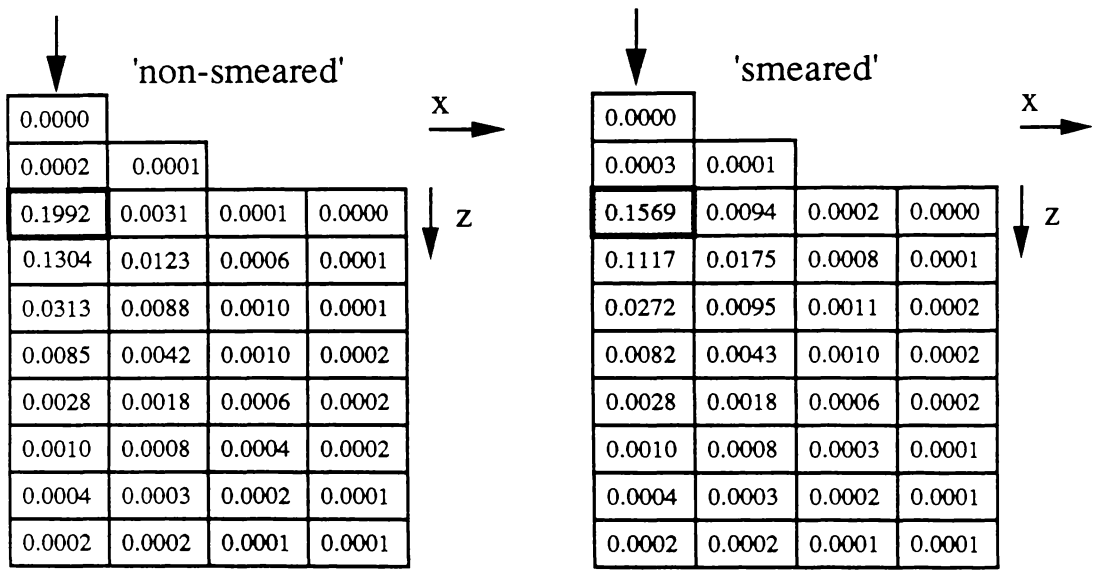

Figure 3.4. The effect of voxel "smearing" on energy deposition kernels (central plane) for a 10 MV photon beam generated by a Clinac–18 linear accelerator. Interaction voxels are shaded. In the smeared kernel (right) more energy is deposited in voxels adjacent to the interaction voxel but off the central axis. Kernels were generated using the EGS4 user code RTPCART, with 2×10^5 incident photons (simulation time 10.6 hours on a VAX 6300 series processor).

(d) When the spectrum of incident photons is examined at various depths within the phantom (using Monte Carlo simulation), it is observed that the mean energy of the beam increases with depth. This so-called *beam-hardening* occurs because the attenuation coefficient of low energy photons is higher, so more low energy photons are removed from the beam with depth. This phenomenon is automatically accounted for when

using the terma calculation technique described in Section 3.3.2, but it also requires that for rigorous treatment of superposition the kernel must be altered with depth to account for the changing spectrum. This could only be done using a separate superposition for each spectral component, which would slow the superposition computation down severalfold. However, the beam hardening effect has been carefully examined by Metcalfe *et al.*,[13] who have shown that using a single (unmodified) kernel results in only a small error depth dose. Hoban[15] has shown that this error can be reduced further by accounting for the increase in proportion of primary photon energy transferred to electrons as the beam hardens with depth. Thus, a kernel *weighting* factor varying with depth, rather than a change in kernel *shape*, is sufficient to accurately model the beam hardening process. Mohan *et al.*[23] demonstrated that the spectrum also "softens" as the off-axis distance becomes larger, but once again the resulting change in kernel shape has very little effect on the resultant dose distribution.

(e) Since a low energy photon has a greater probability of interacting per unit path length, it is important that each spectral component be weighted by this probability when generating an energy deposition kernel. The $T_{E_i}(\mathbf{r})$ terms in Equation 3.6 automatically account for this, but when performing a Monte Carlo simulation it is more convenient to sample from the frequency distribution of the incident spectrum and then give each particle a statistical weight proportional to μ at that energy (see Chapter 2, Section 2.6.2b).

(f) In general, statistical variations in the energy deposited near the interaction voxel are small, but increase in voxels more remote from the interaction site, since less energy is deposited there. Simulating more photon histories will reduce the variation, but remote voxels will still display larger variance than voxels near the interaction site. However, these statistical variations have an extremely small effect on dose distribution in a homogeneous medium, since the contributions from all the terma voxels tend to "average" these errors. In heterogeneous media the error could be more significant, since the effect of a small scale heterogeneity could be masked by a statistical variation in the kernel.

The simulation times required to generate highly accurate kernels are very large. Several tens of CPU hours on a Vax 6300 series processor are required to achieve an accuracy of even 2% in each voxel in a typical simulation (such as 5 mm $\times$ 5 mm $\times$ 5 mm voxels in a unit density medium). This long simulation time is due substantially to the fact that there is very little geometry symmetry in the simulation, even when applying the technique described

in (b). Another significant disadvantage is that the voxel size used in the superposition calculation is fixed by the voxel size used to generate the kernels. Both these difficulties can be overcome using the technique described below.

II. Interpolation of Spherical Polar Kernels

Since the medium in which energy deposition kernels are generated is homogeneous, the kernels are cylindrically symmetrical about the incident particle axis. This allows highly efficient Monte Carlo simulation of kernels using cylindrical or spherical-polar coordinate systems. The spherical-polar approach, first described by Mackie *et al.*,[19] has an additional advantage: the scoring voxels are automatically smaller near the kernel origin, where the energy gradient is high.

Monte Carlo generation of spherical-polar energy deposition kernels has been performed at Waikato University using the purpose built EGS4 user code `RTPEDK`, developed by this author. This code and its use are described in Chapter 2, Section 2.6.4. The resultant primary and scattered kernels must then be interpolated into a cartesian system prior to superposition (see Hoban[15]), and this interpolation process readily allows selection of arbitrary cartesian voxel dimensions, which is useful in a planning system where the dose grid size and resolution are modifiable by the user.

3.3.4. Dose Computation Using Superposition

The previous two sections of this chapter have described how the terma and energy deposition kernel are formed, using a cartesian coordinate system. To enable the superposition product to be easily calculated, it is convenient to make the following two assumptions:

(a) **Beam-hardening of the kernel is ignored.** As mentioned in Section 3.3.3, the spectrum of the incident beam alters with depth, and therefore in principle the spectrum used to generate the kernel should also vary with depth. However, since Metcalfe *et al.*[13] have shown that this phenomenon has only a very small effect on the dose distribution, it has been ignored here. A small beam-hardening correction applied to the terma can very satisfactorily account for this discrepancy.[15]

(b) **An infinite SSD is assumed when orienting kernels.** In order to avoid coordinate transformation, the kernels are directed along the axes of the terma array, so that kernel voxels coincide directly with terma voxels. That is, the kernels are directed as for a parallel beam (with infinite source-to-surface distance), instead of being skewed slightly outwards. This approximation, illustrated in Figure 3.5, has essentially no effect on

central axis dose, but has a more pronounced effect in the penumbral region of large fields, where the "skew" angle is larger.

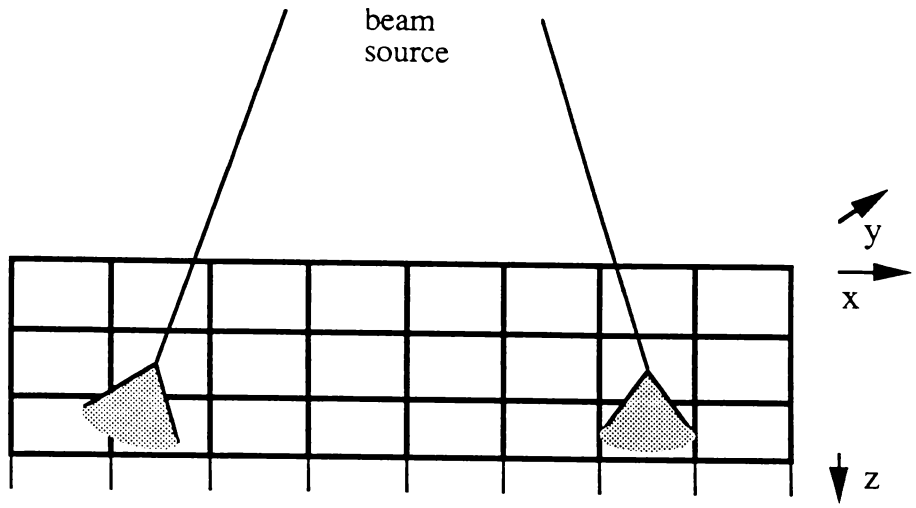

Figure 3.5. The infinite-SSD assumption. A correct treatment of the superposition process would be to skew the kernels outwards when away from the central axis of the beam (left). The assumption made here is to direct the kernels along the z-axis (right).

It is now a relatively simple procedure to perform the superposition. Equation 3.7 describes the superposition process using the "deposition point of view," but a more suitable approach for a practical computation is to use the "interaction point of view." In this case the dose deposited in a voxel (l, m, n) due to terma released at a single voxel (i, j, k) is given by

$$D(l,m,n) \;=\; \frac{1}{\rho_e^w(l,m,n)}\left(\rho_e^w(i,j,k)\cdot T(i,j,k)\cdot H(l-i,m-j,n-k)\right) \;. \qquad (3.8)$$

Note once again that the mass density $\rho(i, j, k)$ has been replaced by $\rho_e^w(i, j, k)$. Figure 3.6 is a pseudocode description of the real space superposition process, with no density scaling of the kernel. Note that the factor $\frac{1}{\rho_m^w(l,m,n)}$ can be removed from the calculation loop and incorporated at the end of the calculation, reducing the amount of computation required.

Energy deposition kernels are calculated in homogeneous media — in heterogeneous media the question of *scaling* these kernels must be addressed. A useful initial assumption, first suggested by O'Connor,[27] is that energy deposition scales inversely with density. For example, in a lung of relative density 0.2, the kernel should be scaled to 5 times its size in a water medium (relative density 1.0). However, in reality the primary and multiple-scattered energy distributions need to be scaled slightly differently, since the way in which primary *electrons* are transported differs from *photon* transport, which is the dominant influence in the distribution of multiple-scattered energy. Also, since primary energy has a high gradient near the interaction site and rapidly drops away, while multiple-scattered energy is

```
constant X,Y,Z;
type array_type = array[0..X,0..Y,0..Z] of real;
var terma,kernel,dose,relative_electron_density : array_type;

procedure calculate_dose;
   var i,j,k,l,m,n : integer;
   do i := 0..X
      do j := 0..Y
         do k := 0..Z
            if (terma[i,j,k] > 0.0)
               do l := 0..X
                  do m := 0..Y
                     do n := 0..Z
                        if ((l-i), (m-j) and (n-k) are within kernel limits)
                           dose[l,m,n] := dose[l,m,n] +
                                 (relative_electron_density[i,j,k] ×
                                 terma[i,j,k] ×
                                 kernel[abs(l-i)][abs(m-j)][n-k])
                        end if
                     end do
                  end do
               end do
            end if
         end do
      end do
   end do
   do l := 0..X
      do m := 0..Y
         do n := 0..Z
            dose[i,j,k] := dose[i,j,k] / relative_electron_density[i,j,k]
         end do
      end do
   end do
   end;
```

Figure 3.6. Pseudocode description of real space superposition, with no density scaling of the kernel. The kernel is stored as one forward-directed quadrant.

much more widespread in its distribution, the kernel is often separated into primary and scattered components. This is the approach used in generating the spherical-polar kernels at Waikato University, and in principle this allows a more efficient computation of the dose distribution, since a small, localised grid can be used for primary energy and a coarser, more expansive grid for multiple scatter.

The method of density scaling used at Waikato is similar to that described by Mackie *et al.*,[2] where an estimate is made of the average density between each pair of voxels. This figure is then used to interpolate between a number of kernels generated for water-like materials of differing densities (for example, of relative densities 0.25, 0.5, 0.75, 1.0, and 1.25). This approach achieves the effect of density scaling by interpolating between "pre-scaled" kernels, rather than by scaling a single kernel. It thereby avoids the difficult and computationally intensive problem of scaling a cartesian array, which involves interpolating between scaled voxels. It is also more physically correct in that it automatically accounts for any non-linearity in the way the kernel scales with density (provided the Monte Carlo

transport code faithfully models the physical situation). Mackie *et al.*[2] and Hoban[15] discuss the technique in more detail.

Estimation of the average relative density ρ^{w}_{ave} between the interaction and deposition voxels is implemented in varying levels of sophistication at Waikato University. The simplest approach is assumption of a homogeneous, water-like medium ($\rho^{w}_{ave} = 1.0$ everywhere), in which case there is effectively no kernel density scaling, and the Fourier transform convolution method discussed in the next section is more suitable than superposition. The next simplest technique used at Waikato University is a type of scaling where the mean density down the z-axis (parallel to the beam) from the interaction site to the deposition plane is calculated.[12] This scaling factor is then used for every voxel on the deposition plane. A third approach is to trace from every interaction voxel to every deposition voxel, using a method similar to that described by Mackie.[28] More sophisticated approaches (such as a Siddon-type ray trace between each pair of voxels) are also possible. With each increasingly complex scaling algorithm there is an associated penalty in computational effort. This compromise — between a computationally efficient but relatively crude scaling algorithm or a more sophisticated but computationally intensive approach — is one of the central issues of real space superposition.

3.3.5. Dose Computation Using Convolution

The major disadvantage of using the real space superposition technique to calculate dose is that the process is computationally intensive. To increase the efficiency of computation it was proposed that fast Fourier transform techniques be used to perform the convolution.[3] Although considerably faster,[29] the Fourier space approach cannot be used unless the convolution kernel is spatially invariant. This requires that the kernel be identical at each point in the terma array, which rules out any form of density scaling. Even so, the Fourier space approach can be usefully applied to the calculation of dose in homogeneous regions and some authors are working on correction techniques to account for heterogeneities.[6,9] However, such corrections detract from the simplicity of the technique, which is one its most appealing aspects. This section ignores these correction techniques, concentrating instead on how transforms can be utilised to perform multi-dimensional convolutions in homogeneous media.

For small convolutions the direct Agarwal-Cooley convolution technique[30] is generally more efficient than Fourier transform techniques. This method utilises the nesting of several short convolutions to calculate one longer convolution. However, when applied to larger array sizes typical in dose calculations, discrete Fourier transform techniques offer more significant

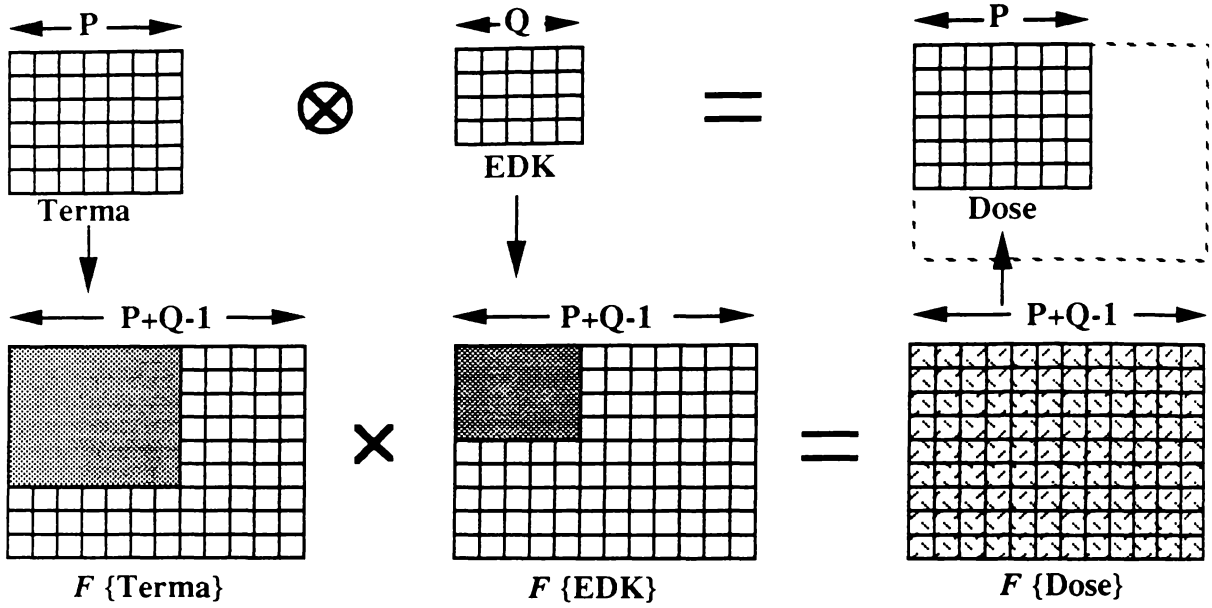

Figure 3.7. The FFT convolution framework. Convolution (or superposition) may be performed directly (top row), or by Fourier transform techniques (bottom row). Arrays must be padded out to ensure that the convolution does not "wrap around."

savings. This approach involves transforming the terma array and the convolution kernel into Fourier space representations, performing a voxel by voxel complex multiplication of these arrays and then inverse transforming to yield the real-space result. Denoting a Fourier transform as $\mathcal{F}$ and an inverse Fourier transform as $\mathcal{F}^{-1}$, Equation 3.7 can be replaced by

$$D(x,y,z) = \frac{1}{\rho_m^w(x,y,z)}\left\{\mathcal{F}^{-1}\left(\mathcal{F}(\rho_e^w T)\cdot\mathcal{F}(H)\right)\right\}(x,y,z) \quad , \tag{3.9}$$

where $\rho_e^w T$ is the product of terma and relative electron density, and H is the energy deposition kernel as before. In practice the transformed kernel $\mathcal{F}(H)$ can be pre-calculated and stored, so that only one transform and one inverse transform need be calculated for each beam.

One property of the discrete Fourier transform method is that it produces *circular* convolutions, where the input arrays are considered to be *periodic* in real space. To ensure that overlap known as the *end effect* is eliminated, the arrays must be padded out to length $N = P + Q - 1$, where P and Q are the number of samples in the input arrays.[31] Using a standard fast Fourier transform the condition $N = 2^\gamma$ where γ is an integer must also be satisfied, although the use of a mixed-radix FFT algorithm allows this restriction to be relaxed. Additionally, both arrays must be shifted to the origin prior to convolution, then back again after the convolution has been performed. This shifting process minimises the number of zeros which must be added to the arrays, thereby improving the efficiency of the convolution.

The simplest method of extending one-dimensional convolutions to three dimensions is *row-column decomposition*, so called because each dimension of the array is treated by a set of one-dimensional transforms. Using a standard radix-2 FFT algorithm on a three-dimensional array of dimensions $N \times N \times N$, the number of complex multiplications C_{rc} required is[32]

$$C_{rc} = \frac{3}{2} N^3 \log_2 N \quad . \tag{3.10}$$

Figure 3.8 illustrates schematically row-column decomposition in two dimensions, while Figure 3.9 is a pseudocode description of how row-column decomposition is implemented for a 3-dimensional calculation. Appendix F presents the fast Fourier transform subroutine used to implement the convolution algorithm.

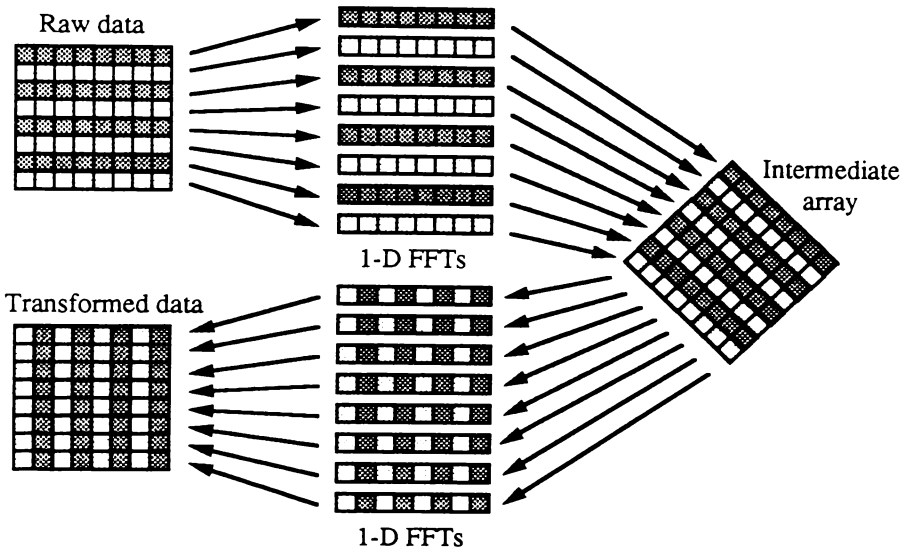

Figure 3.8. Schematic of row-column decomposition in two dimensions.

An alternative approach is to break the $N \times N \times N$ transform down into successively smaller transforms until only trivial ($2 \times 2 \times 2$) transforms need to be evaluated. One iteration of this process for two dimensions is illustrated in Figure 3.10. The number of (butterfly) multiplications C_{vr} required for this *vector-radix* transformation[32] is only 58% of C_{rc}:

$$C_{vr} = \frac{7}{8} N^3 \log_2 N \quad . \tag{3.11}$$

Neither the row-column nor vector-radix techniques described above are optimal for multidimensional convolutions. For example, a significant improvement can be obtained using the Winograd Fourier transform algorithm (WFTA).[33] In this technique a large multidimensional transform is computed by nesting short transforms, calculated using optimally

```
constant X,Y,Z,XFFT,YFFT,ZFFT;
type  complex = record
                   real,imag : real;
                end;
      real_array_type = array[0..X,0..Y,0..Z] of real;
      fft_array_type = array[0..XFFT,0..YFFT,0..ZFFT] of real;
var   terma,kernel,dose,relative_electron_density,relative_mass_density : real_array_type;
      fft1,fft2,fft3 : fft_array_type;

procedure calculate_dose;
   var i,j,k : integer;
   copy relative_electron_density × terma into low corner of fft1 (shifting if necessary);
   transform(fft1,x,forward);
   transform(fft1,y,forward);
   transform(fft1,z,forward);
   copy kernel into low corner of fft2 (shifting if necessary);
   transform(fft2,x,forward);
   transform(fft2,y,forward);
   transform(fft2,z,forward);
   do i := 0..XFFT
      do j := 0..YFFT
         do k := 0..ZFFT
            fft3[i,j,k].real := fft1[i,j,k].real × fft2[i,j,k].real -
                  fft1[i,j,k].imag × fft2[i,j,k].imag;
            fft3[i,j,k].imag := fft1[i,j,k].real × fft2[i,j,k].imag +
                  fft1[i,j,k].imag × fft2[i,j,k].real;
         end do
      end do
   end do
   transform(fft3,x,inverse);
   transform(fft3,y,inverse);
   transform(fft3,z,inverse);
   copy real part of fft3 into dose (performing shifts opposite to those above);
   do l := 0..X
      do m := 0..Y
         do n := 0..Z
            dose[i,j,k] := dose[i,j,k] / relative_mass_density[i,j,k];
         end do
      end do
   end do
   end;

procedure transform(var array : fft_array_type;
size : integer; axis : (x,y,z); dirn : (forward,inverse));
   perform one-dimensional forward FFTs on array in direction of axis;
   if (dirn = inverse)
      do l := 0..XFFT
         do m := 0..YFFT
            do n := 0..ZFFT
               array[l,m,n].real := array[l,m,n].real / size;
               array[l,m,n].imag := - array[l,m,n].imag / size
            end do
         end do
      end do
   end if
   end;
```

Figure 3.9. Pseudocode description of fast Fourier transform convolution using row-column decomposition. *XFFT*, *YFFT* and *ZFFT* are the dimensions of the FFT arrays (allowing for padding to avoid "wrap-around").

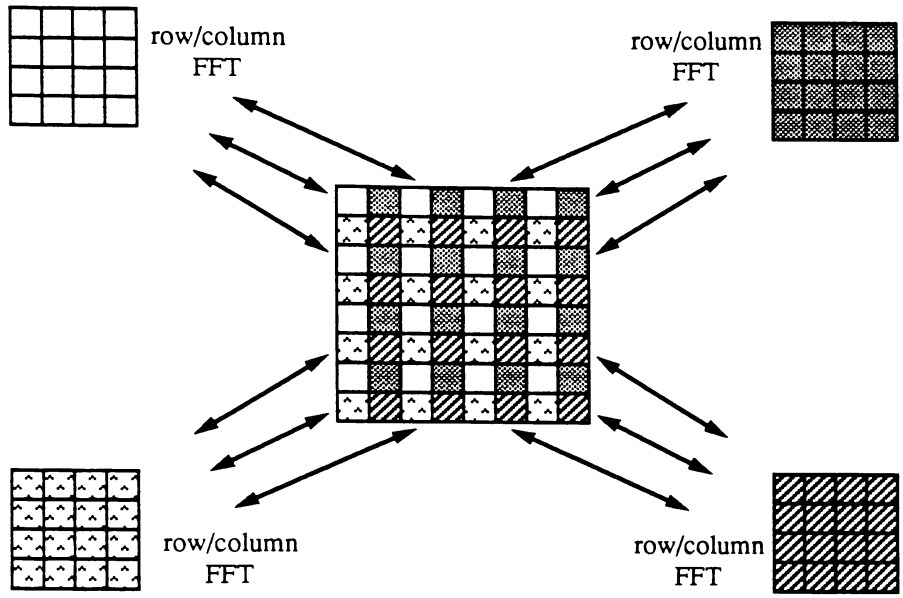

Figure 3.10. Schematic of vector-radix decomposition in two dimensions.

derived *small-N* transforms. Even more efficient than the WFTA are polynomial transform techniques, which can be viewed as discrete Fourier transforms defined in rings of polynomials.[34] These techniques can result in significant reduction in the number of multiplications needed. For instance, evaluating a $120 \times 120 \times 120$ discrete Fourier transform using a polynomial transform requires 2.50 multiplications per point, compared with 21.8 multiplications per point using the Agarwal-Cooley method[34] and 11.5 multiplications per point using the Good prime factor algorithm.[35] However, the considerable speed increase is at the expense of more complicated implementation.

Table 3.1 illustrates the effect which one such transform technique can have on the execution time of two medium-sized convolutions. Real space superposition (with no density scaling) was used to calculate dose for the two array sizes. Dose calculated was then repeated using a standard radix-2 FFT algorithm employing row-column decomposition, with sine, cosine and bit-reversal lookup tables. Note that the array sizes used for FFT calculation are larger than the equivalent real space array sizes, because sufficient zeros must be added to the FFT arrays to prevent wrap-around.

The computation times quoted in Table 3.1 are for one processor of a DEC VAX 6200 series computer. It can be seen that using the FFT approach results in a 7-fold increase in speed for the smaller calculation and an 11-fold increase for the larger one. This improvement would be even more significant for larger array sizes, since for an $N \times N \times N$ array FFT computation time increases as $N^3 \log_2 N$, whereas time required for real space superposition increases as N^6 (without density scaling) or as N^7 (with full density scaling).

Superposition size	*time (s)*	*FFT convolution size*	*time (s)*
1. 16 × 16 × 100 terma array, 16 × 16 × 72 kernel	1560	32 × 32 × 256 arrays	213
2. 32 × 32 × 50 terma array, 15 × 15 × 50 kernel	6132	64 × 64 × 128 arrays	542

Table 3.1. Computation times on a VAX 6200 series processor for two dose calculations. The first algorithm used (left half of table) is real space superposition and the second (right half) is radix-2 FFT convolution (using row-column decomposition). FFT times quoted are for performing a forward transform of the terma, multiplying with a pre-calculated kernel transform, and inverse transforming to obtain dose.

3.3.6. Discussion

This section has examined the way in which the superposition and convolution techniques have been implemented at Waikato University. The validity of these approaches has not been directly verified here, but these algorithms have been used in related work, with the following important results:

(a) Murray *et al.*[10] showed initially that the superposition and convolution techniques outlined above produce good results *in homogeneous media*, irrespective of the spectrum used to generate the energy deposition kernels. Beam profiles also agreed well with experiment, although a small error could be seen in the penumbral region of large fields. It was speculated that this was due to the lack of kernel "skewing."

(b) Metcalfe *et al.*[11,13] examined in more detail the effect of incident spectrum on superposition results. They concluded that a polyenergetic treatment of terma calculation is essential, and that a similar treatment of kernel calculation is desirable but less critical. However, it was found that altering spectral composition with depth for kernel calculation was unnecessary, since the dose distribution was relatively insensitive to the kernel spectrum chosen.

(c) The application of the superposition technique to an *heterogeneous* situation has been examined by Hoban *et al.*[12] Using a simple density scaling algorithm based on the approach of Mackie *et al.*,[2] superposition was demonstrated to be superior to scatter function models in a lung phantom, especially for small field sizes.

The works listed above have demonstrated that the superposition technique is a potentially accurate and hence clinically useful technique. However, on conventional radiotherapy workstations the computational complexity of the real space approach makes planning times unacceptably long, so the Fourier transform technique was developed. Although the FFT approach is relatively fast, the assumption of kernel invariance is a severe limitation

in heterogeneous media. Another solution to the speed problem is the use of more powerful processing hardware to accelerate real space calculations. A parallel processing approach has been investigated at the University of Waikato by Murray *et al.*,[36,37] and is the subject of Chapter 4. However, the remaining sections of this chapter introduce the GRATIS Treatment Planning System and discusses its adaption to the superposition algorithm.

3.4. The GRATIS Treatment Planning System

3.4.1. Overview

GRATIS (George's **RA**diotherapy **T**reatment Des**I**gn **S**ystem) is a research-based treatment planning system suitable for brachytherapy and external beam therapy. It has been written by George Sherouse and others at the University of North Carolina (Chapel Hill) during the period 1986–91. Formerly known as **usr/planning**, it is currently in its third release. It is designed to run on a variety of UNIX-based graphics workstations with at least a pseudocolour (eight plane) display and 20 Mbytes of available disk space. A Hewlett Packard or compatible plotter (for plan output) and Summagraphics or compatible digitising tablet (for manual contouring) are also recommended. All of GRATIS's visual tools are based on the X11 windowing standard. At least thirty copies of GRATIS have been distributed throughout the world.

Installation

GRATIS is distributed on an industry standard magnetic tape, in the form of a UNIX **tar** saveset. Due primarily to its multi-platform nature, installation of GRATIS is not completely straightforward. Several aspects of GRATIS need to be tailored to individual systems, in particular:

- Configuring GRATIS for the available scanners and treatment machines, including supplying necessary physics data for those machines.
- Modifying the UNIX **make** files to allow for any non-standard features of the particular workstation vendor's UNIX-like operating system.
- Ensuring that all UNIX libraries (particularly the X10 compatibility library) are present and installed in the places expected by GRATIS.
- Specifying whether Sun's XDR (external data representation) format is to be used.

In the case of installation on the Waikato University Sun network, the two most significant problems concerned the X10 compatibility library (which had to be designed by this author,

since it was absent from Sun's *OpenWindows* release), and keyboard input focus (which operated incorrectly due to differences between *OpenWindows* and UNC's X11 windowing system). Appendix G describes in detail the installation of GRATIS on Waikato University's Sun network.

The GRATIS File Hierarchy

When installation is complete, the GRATIS file system has the structure illustrated in Figure 3.11.

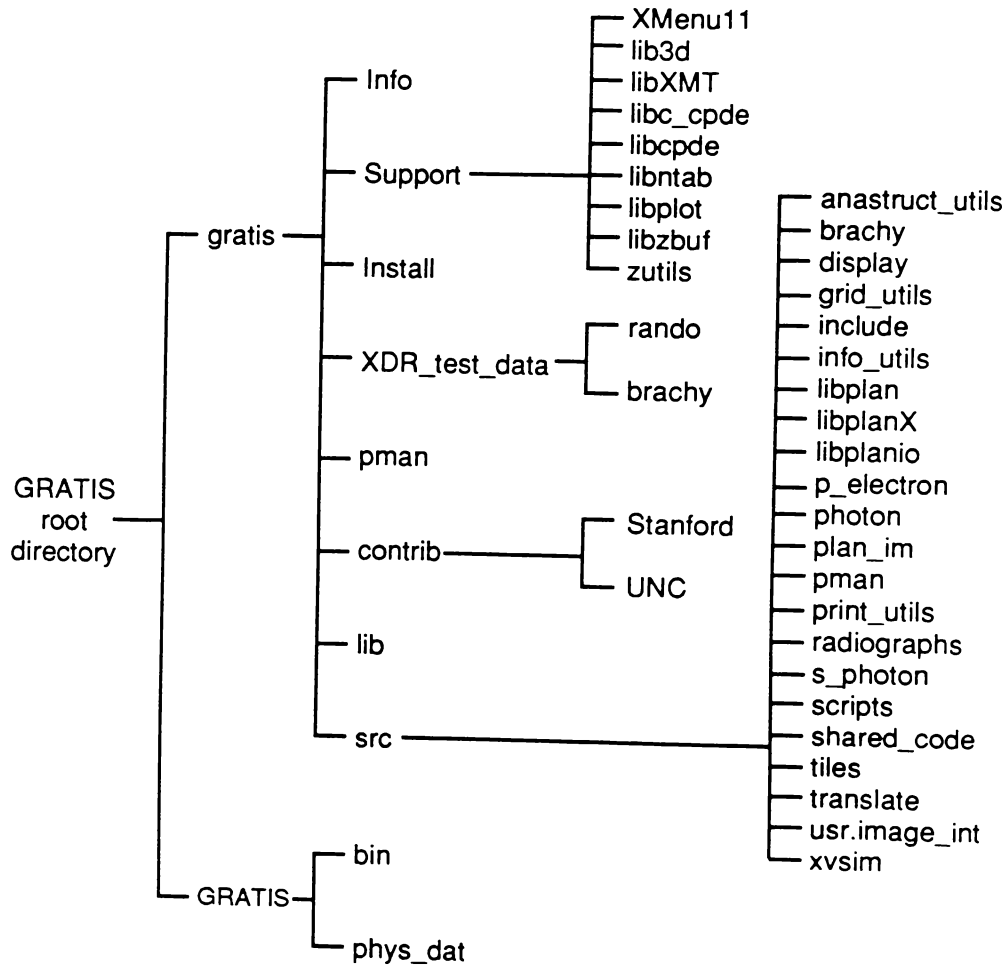

Figure 3.11. Organisation of the GRATIS system.

The contents of each of the subdirectory hierarchies are as follows:

- `Info`: GRATIS documentation, including release notes.
- `Support`: GRATIS support libraries, including `XMenu11` (routines for implementing X11 menus), `lib3d` (routines for manipulating and transforming 3-D entities, as described by Newman and Sproull[38] and Siddon[39]), `libXMT` (X11 mini-toolkit for cre-

ating and manipulating X11 windows and widgets), **libc_cpde** (data-entry routines based on the UNIX **curses** package), **libcpde** (data-entry routines based on the UNIX **termcap** facility), **libntab** (routines for Summagraphics-compatible digitising tablets), **libplot** (routines for Hewlett Packard or compatible plotters), **libzbuf** (routines for manipulating z-buffered images), and **zutils** (utilities for z-buffered images).

- **Install**: GRATIS installation scripts for tailoring **make** files to the appropriate workstation and operating system.
- **XDR_test_data**: Example data sets stored using Sun's XDR data representation. In particular, a full (42 slice) image of a Rando phantom, and example brachytherapy data.
- **pman**: GRATIS manual pages.
- **contrib**: User contributions to the GRATIS system (currently a tek4662 plotter library and a contour editing program).
- **lib**: Support library images. These are normally installed in **/usr/local/lib**, but in order to keep the GRATIS system completely self-contained this author chose to install them here.
- **src**: GRATIS source code. There are 22 subdirectory hierarchies located below **src**, including the **s_photon** (superposition) hierarchy discussed in Section 3.5, and the **p_electron** (electron pencil beam) hierarchy discussed by Hoban.[15] There are also three *library* subdirectories: **libplanio** (for input/output of GRATIS data structures); **libplanX** (for drawing objects using X); and **libplan** (miscellaneous routines useful in the planning process).
- **bin**: Scripts and executable images created when the GRATIS system is installed.
- **phys_dat**: Physics data (in binary format) used for simulation and dose calculation.

Coordinate Systems and Conventions

A number of distinct cartesian coordinate systems are defined in GRATIS. The most important of these are:

- The **patient** coordinate system, aligned with the treatment couch. If the patient is lying face-up on the couch when at its normal position, then x is positive to the patient's left, y is positive upward, and z is positive towards the patient's feet. This

coordinate system moves *with the patient*, and the origin is usually located at the intersection of the patient midline, couch surface and first CT scan.

- The **beam** coordinate system. This is a left handed coordinate system, fixed with respect to the collimator and with the coordinate system origin at the beam origin. This coordinate system is important when calculating dose using superposition, since the terma and kernels must be aligned with the beam.
- The **grid** coordinate system. This is a right handed coordinate system used to define dose calculation (and other) grids. Normally it is identical to the patient coordinate system.

There are also the **unit** and **filter** coordinate systems, and a set of conventions for collimator, table, and gantry angles. More details can be found in the GRATIS Programmer's Manual.[40]

3.4.2. Elements of the GRATIS System

GRATIS is a *set of tools* for radiotherapy treatment planning, linked together using standard UNIX shell scripts. The relationship of the major GRATIS tools used in external beam therapy is illustrated in Figure 3.12. Full descriptions of these tools can be found in the manual pages of the GRATIS Programmer's Manual,[40] and they are also described in more detail by Hoban.[15] However, brief descriptions of the most important tools are provided here.

GRATIS Tools Used in External Beam Radiotherapy

(a) `build_image`. This utility converts raw CT data files (obtained from a Siemens, Technicare, or possibly other type of scanner) to an image stored using the GRATIS image format, which is independent of scanner type. `build_image` can also be used to interpolate or remove CT slices, alter the table height, and retrieve CT scans across a network using FTP (file transfer protocol). `build_image` source code for a particular scanner must be obtained by permission of the scanner manufacturer.

(b) `imex`. The `imex` utility is an extensive image processing system developed by the University of North Carolina, and is used in many areas other than treatment planning and GRATIS. Its chief purpose in the context of GRATIS is to create and modify *contours* and *anatomes*. A *contour* is a set of points defining a two dimensional polygon, and an *anatome* (also known as an *anastruct*) is a group of contours which together define an anatomical structure, such as the skin, lungs, spinal cord, or stomach. `imex`

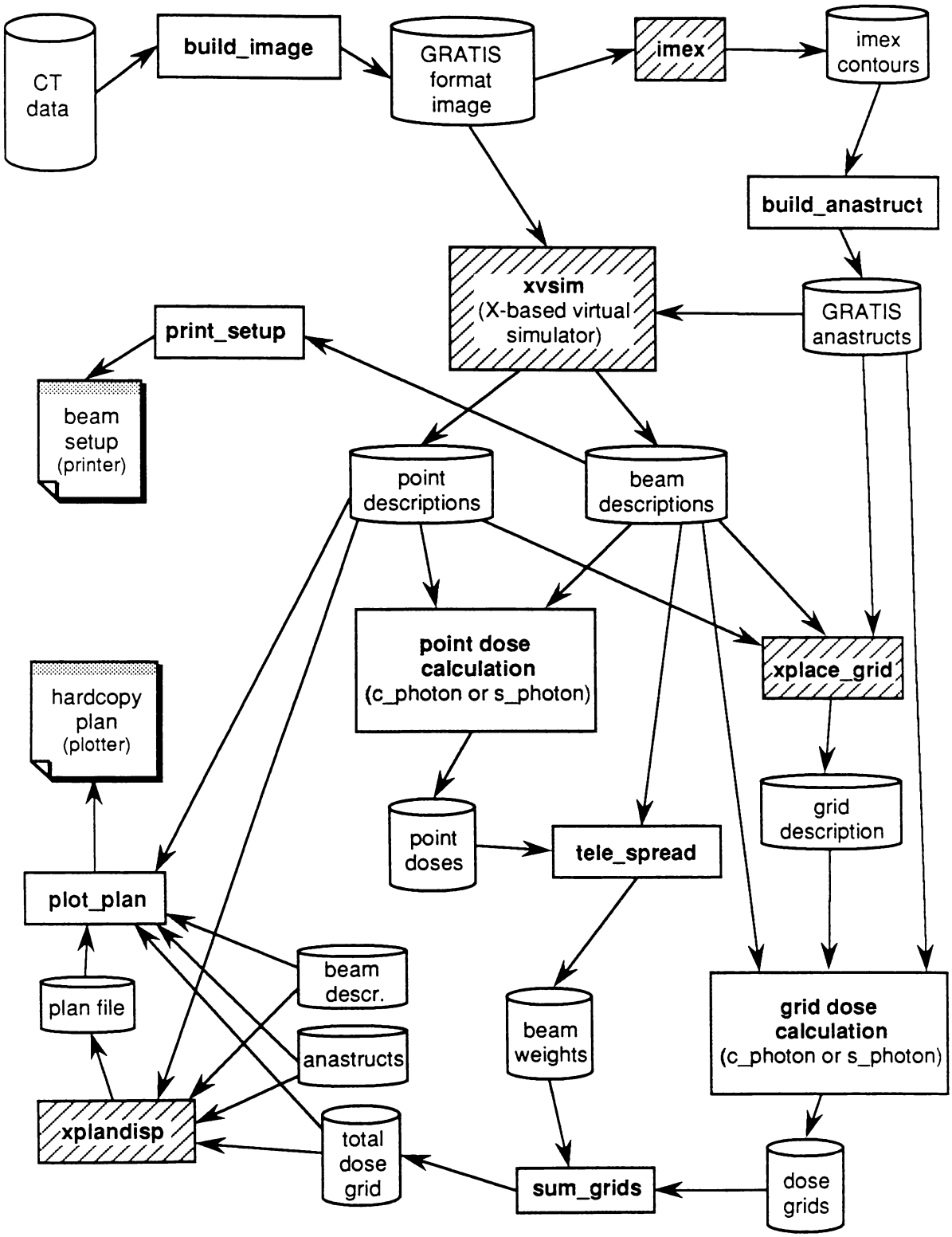

Figure 3.12. Elements of the GRATIS Treatment Planning System, and their relationship in external beam therapy. Diagonally hatched boxes are tools based on the X standard.

is capable of *autocontouring*, or automatically generating these contours, but the contouring process can also be controlled manually. **imex** is an X-windows based utility, displaying each CT slice as a postage-stamp (64 × 64 pixel) view. An individual slice can then be selected, enlarged, and contoured, with each contour being assigned to a particular anastruct.

(c) **build_anastruct**. This utility converts **imex** contours to anatomical structures in the GRATIS *anastruct* format, for use by **xvsim** and other utilities.

(d) **xvsim**. The X-based virtual simulator (**xvsim**) is a tool for placing external beams on a patient or phantom. Usually **xvsim** opens a GRATIS format image, overlays a set of one or more anastructs (one of which must be the skin), and allows the operator to position beams on the patient. Figure 3.13 shows **xvsim** with two beams incident on a Rando phantom — note that there are several distinct panels on the screen at any one time. The top panel in Figure 3.13 is used to select the treatment machine, machine parameters and display attributes, and the large panel on the right is a "beam's eye view" from the collimator down the currently selected beam. The XVIEW and YVIEW panels present orthogonal views of the patient, and the partially obscured IMAGE panel shows the beams superimposed on the currently selected CT slice, overlaid by anastructs. The small icons in the lower left corner show the current position of the treatment machine, and the UNIT control panel (bottom) allows adjustment of the SSD/SAD, couch position, and beam dimensions. Note the operator's placement of two points (labelled *tissue* and *lung*) on the phantom — the dose to these points (and also to the beam isocentres) will be calculated in the dose calculation step. Figure 3.14 illustrates **xvsim** with several of the panels closed down, revealing the complete IMAGE panel and many of the postage stamp CT slices. The way in which individual panels can be manipulated, and the intuitive way in which **xvsim** operates, are two of the most significant features of the GRATIS system. Note from Figure 3.12 that the output produced by **xvsim** is a set of beam descriptions and a set of point descriptions — these are required as input to the dose calculation utility.

(e) **print_setup**. This utility produces printer output describing the beam geometry details, for use in setting up the simulator or linear accelerator.

(f) **c_photon** or **s_photon**. Dose deposited due to an external beam can be calculated using the default GRATIS algorithm (**c_photon**) or the superposition algorithm (**s_photon**) developed at Waikato University. **c_photon** is a variant of the differential scatter-air

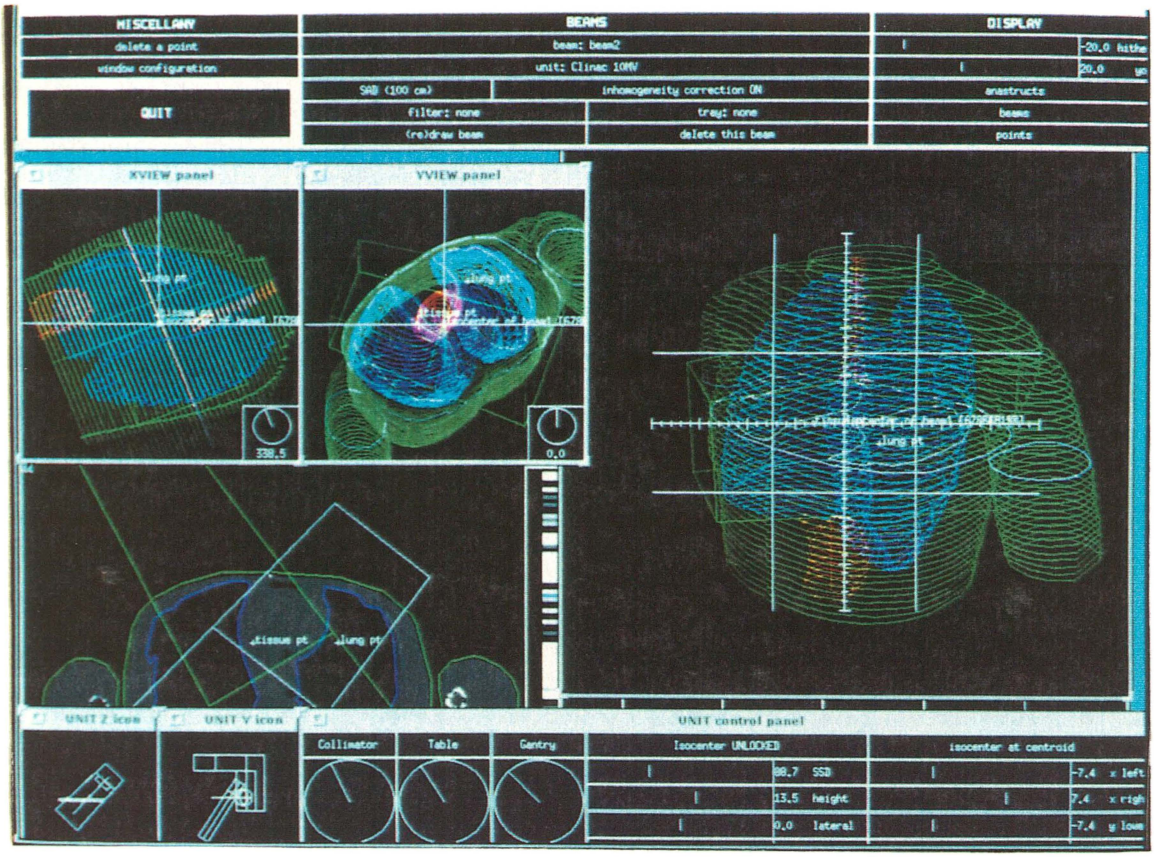

Figure 3.13. xvsim showing two beams incident on a Rando phantom.

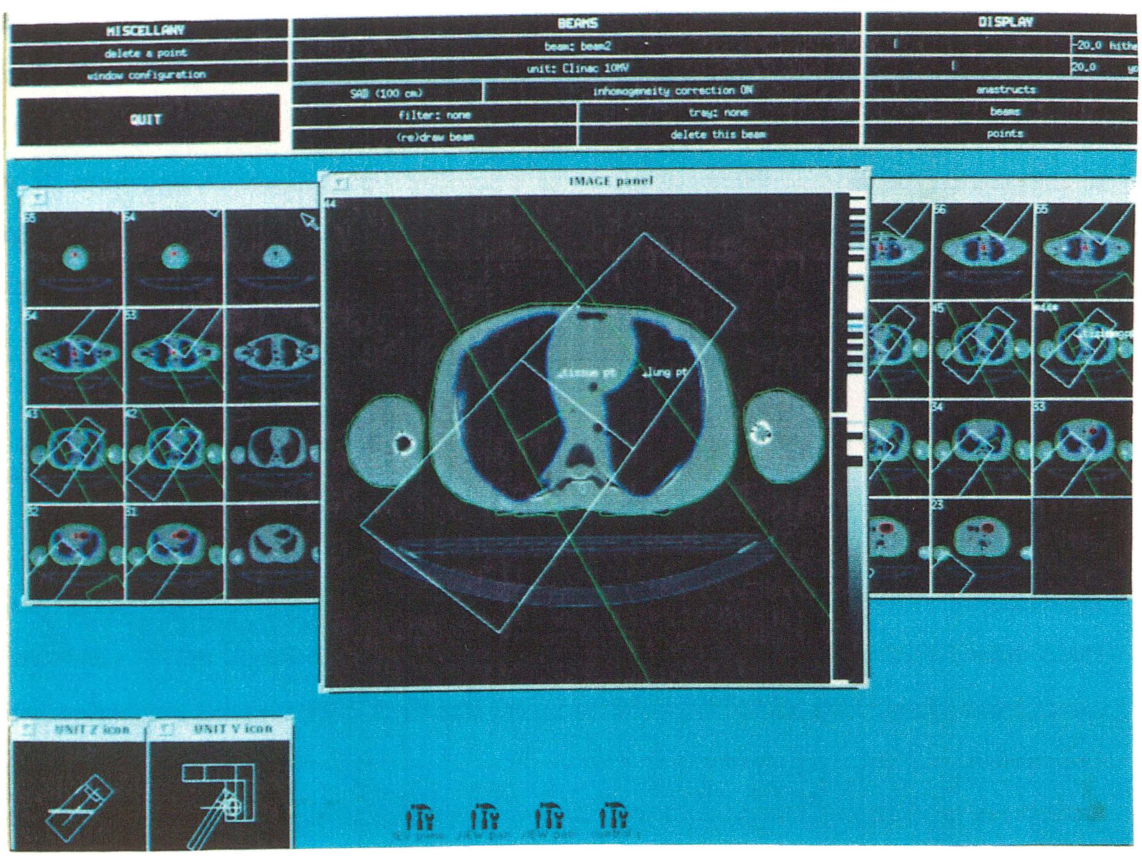

Figure 3.14. xvsim showing IMAGE panel and "postage stamp" CT slices.

ratio method described in Chapter 1, Section 1.4.3d, where scatter is integrated over sectors of annuli centred on and perpendicular to the ray from the source through the interaction point (see the GRATIS documentation for a more detailed explanation of this "dart-board" model). This method does *not* come supplied with an inhomogeneity correction, yet still requires several minutes computation (on a Sun workstation) for reasonably sized beams. The superposition algorithm (`s_photon`) developed at Waikato University is described in Section 3.5. Both algorithms operate in one of two modes: *point dose* mode, where doses to the beam isocentres and any other user-defined points are calculated; and *grid dose* mode, where the dose to every point in the grid (defined using `xplace_grid`, below) is calculated. `s_photon` also has a third mode, used for calculating machine output factors.

(g) `tele_spread`. This is a `curses` or `termcap` based spreadsheet utility for beam weighting and normalisation. The relative weight of the beams can be specified, as can the overall dose to be delivered, and the number of treatment fractions. The plan can be normalised to any of the user-defined calculation points, or to a beam isocentre.

(h) `xplace_grid`. This X-based utility is used after point dose calculation, to specify the dose grid limits and resolution. Three orthogonal views of the patient are presented (see Figure 3.15), and the dose "box" can be enlarged or reduced in any of the three views. The patient can also be "dragged" to alter the position of the dose grid relative to the patient, or rotated using the knobs in the lower right corner of each view. Three vertical bars (lower right in Figure 3.15) can be adjusted to alter the number of grid points along each of the three axes, and the resultant grid spacings and total number of calculation points are displayed. Anatomical structures can also be placed on the views, as in Figure 3.15.

(i) `sum_grids`. Since `c_photon` and `s_photon` produce a separate dose grid for each beam, the `sum_grids` utility must be invoked to produce a combined dose distribution.

(j) `xplandisp`. The complete dose distribution can be evaluated by the planner using the X-based tool `xplandisp`, illustrated in Figure 3.16. The large, upper view can be altered to represent an arbitrary cut through the patient, by dragging the patient using the mouse and rotating it using the three knobs, as with `xplace_grid`. Anatomical structures in both the foreground and background can be displayed (see Figure 3.16), although the isodose lines can be seen more clearly if these options are turned off. The dose "shoebox" and dots showing the dose grid resolution can also be shown. Isodose

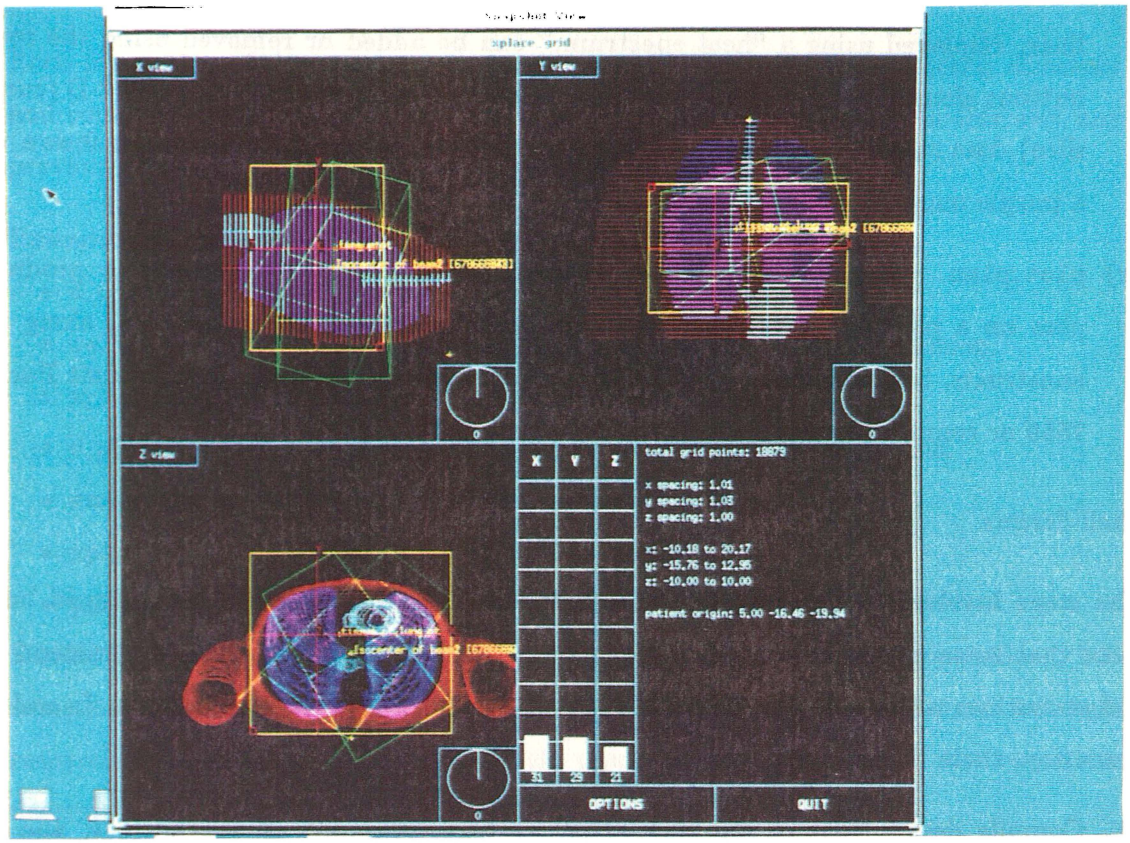

Figure 3.15. The xplace_grid tool.

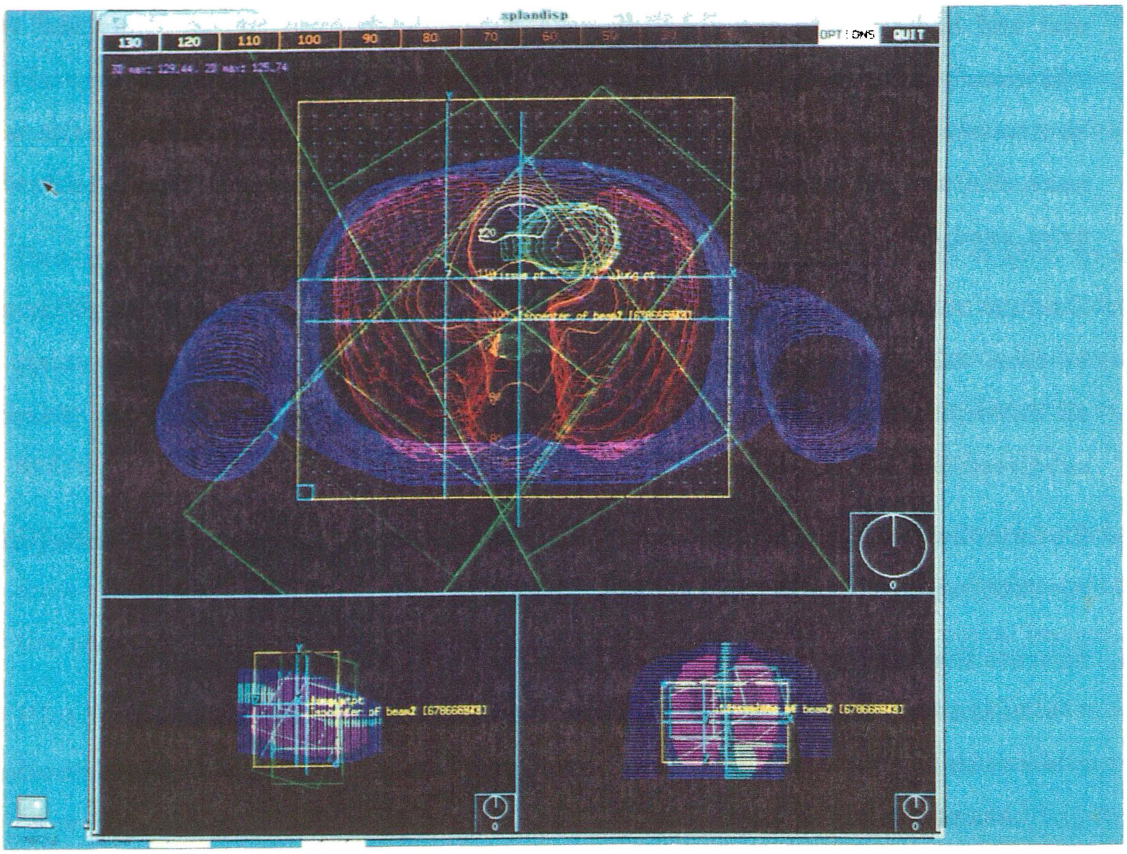

Figure 3.16. The xplandisp tool.

lines, displayed using a "heat spectrum," can be added or removed using the row of buttons along the top of the upper view. In the top left corner the 3-D and 2-D (current slice) dose maxima are displayed, with the position of the 2-D maximum shown with a purple marker.

(k) `plot_plan`. The final step in the external beam planning process is to produce a plan on a plotter. `plot_plan` uses the plan file created by `xplandisp`, along with anastructs, beams, points, and the dose grid, to produce a plan on a Hewlett Packard 7550 or compatible plotter.

Shell Scripts

The steps involved in the complete external beam planning process are outlined below. Note that some of the commands executed are *shell scripts* — these are command language routines which call the GRATIS tools discussed in the previous section. In order of execution, these steps are:

1. Compile descriptions of any new filters or trays required by the patient, and execute `make_filter` and `make_unit`, which transform the descriptions into binary format and add them to the GRATIS data base. Often appropriate filters and trays will already have been defined (in previous treatments), and hence this step will not be necessary. Similar utilities exist for creating new treatment machines (`make_unit`, `make_sar`, and `make_time_calc`), and for new brachytherapy sources (`make_ls_dat` and `make_seed_dat`).

2. Execute `build_image` to convert the CT data to GRATIS image format. This need be done only once per patient. Each new patient is assigned a unique directory, and all subsequently generated files (such as beam, grid, and point files) are placed in subdirectories of that directory.

3. Execute `imex` to generate contours. This would normally be done only once (unless the contours are later deemed to be inadequate).

4. Execute the `simulate` shell script. This script leads the user through:
   - `build_anastruct` (executed only in the first iteration of `simulate`).
   - `xvsim` (repeated as many times as necessary until the desired dose distribution is obtained using `extbm`).
   - `print_setup` (executed only when the planner is satisfied with the dose distribution obtained using `extbm`).

Each invocation of the **simulate** script automatically creates a new backup directory and copies all data files associated with the patient to that directory. In this way, any previous iteration of **xvsim** can be retrieved if desired.

5. Execute the **extbm** shell script. This script leads the user through:
   - **c_photon**, executed once per beam, to calculate point doses.
   - **tele_spread**, to weight the beams and normalise the dose distribution.
   - **xplace_grid**, to select the dose grid size and resolution.
   - **c_photon** again, executed once per beam, to calculate the dose grid.
   - **sum_grids**, to add the doses from individual beams.
   - **xplandisp**, to display the resultant dose distribution.
   - **plot_plan**, to output the plan to a plotter.

   Similar scripts **sextbm** and **transbm** perform the same sequence of operations, but using *superposition* (**s_photon**) on a Sun SPARC processor and a transputer network, respectively. All three scripts recalculate point or grid doses only if the respective descriptor files have been modified since the last dose calculation.

Other GRATIS Tools

The above sections have discussed the essential elements of GRATIS used when planning external beam radiotherapy. However, there are several other groups of utilities available in the GRATIS system, as documented in the GRATIS Programmer's Manual:[40]

- **Brachytherapy.** The shell script **brachy** guides the user through a set of tools in a similar manner to **extbm**. The tools are **f_brachy_film** (for creating an implant description), **xplace_grid**, **brachy_spread** (a spreadsheet for adjusting source loading), **c_brachy** (for calculating dose distribution using a Sievert line integral for line sources, and exponential fall-off for seeds), **sum_grids**, **xplandisp**, and **plot_plan**.

- **Digitally constructed radiographs.** The shell script **rgraph1** can be used to generate radiographs for each beam. These can then be displayed on an X11 display using the **show_rgraphs** script, and compared with simulator radiographs or portal images.

- **Plotting.** The shell script **plot_anastruct** enables the user to plot anatomical structures on an Hewlett Packard or compatible plotter without entering the external beam or brachytherapy planning processes. The utility **anastruct_info** can be used to produce summary statistics for anatomical structures. Beams can also be plotted using **plot_beam_outline**.

- **Obtaining planning data.** Utilities are available to print summaries of beam descriptions (`beam_info`), dose grids (`grid_stats`), planning images (`plan_image_info`), beam names (`print_beam_name`), and treatment units (`unit_info`). Raw data from dose grids can be printed using `cat_grid` and `cat_2D_grid`.

- **Manipulating images, contours, and grids.** Utilities are available for concatenating image files (`plan_im_cat`), interpolating between CT slices (`interpol`), removing CT slices (`plan_im_rm_scan`), reducing image resolution (`plan_im_shrink`), modifying image z-coordinates (`plan_im_z_fiddle`), setting image table height (`xplace_table`), reducing the number of points in a contour (`reduce_con`), scaling a dose grid (`scale_grid`), slicing through a dose grid (`slice_grid`), and tiling for rendering in 3-D (`plan_to_grid`, `tile_beam`, `anastruct_tile`, and `grid_tile`).

- **Displaying images.** The X-based utility `xdisp` can be used to view planning images and overlaid anastructs, without entering the brachytherapy or external beam therapy processes.

From this section it is clear that GRATIS is a comprehensive and powerful treatment planning system. Its well-documented structure and available source code make it an ideal basis for radiotherapy research such as that conducted at Waikato University. The following sections deal with how GRATIS and the superposition algorithm can be combined to yield a clinically useful, superposition-based treatment planning system.

3.5. Implementation of Superposition Under GRATIS

3.5.1. Invoking the Superposition Algorithm

The executable image for superposition is called `s_photon`. It is normally called from the shell scripts `sextbm` (for execution on a Sun) or `transbm` (for execution on a Sun-based *transputer* network). The only differences between these two scripts are the values given to the parameters *processor_type*, *processor_number*, and *vector_size*. `sextbm` supplies `sun` and `1` for *processor_type* and *processor_number*, leaving *vector_size* unspecified, while the `transbm` script supplies `transputer` for *processor_type* and accepts command line parameters for *processor_number* and *vector_size*, passing them on to `s_photon`.

Within each shell script, `s_photon` may be called in one or both of the following modes:

1. **Calculating point doses.** Doses to the user-defined calculation points and beam isocentres are calculated by calling `s_photon` in the following manner:

s_photon -a *skin_anastruct* **-b** *beam_file* **-p** *point_description* **-O** *output_point_dose* **-i** *plan_image* **-S** *scaling_type* **-T** *processor_type* **-N** *processor_number* **-V** *vector_size*

These parameters are similar to those supplied to **c_photon**, except that in **s_photon** the planning image file *plan_image* is required for terma calculation, and there is kernel range scaling of type *scaling_type*.

2. **Calculating dose to a grid.** In this second mode the point description and output point dose parameters are replaced by corresponding grid description and output grid parameters:

   s_photon -a *skin_anastruct* **-b** *beam_file* **-p** *point_description* **-O** *output_point_dose* **-i** *plan_image* **-S** *scaling_type* **-T** *processor_type* **-N** *processor_number* **-V** *vector_size* **-R** *remote_server_node*

   with all other parameters remaining the same. The additional parameters *processor_type*, *processor_number*, communication *vector_size*, and *remote_server_node* may optionally be specified — these are designed for use with a transputer network (see Chapter 4), but default to **sun**, **1**, **0** and the null string respectively, which are sensible values for operation on a Sun SPARCstation.

s_photon is also called under a third circumstance: when making superposition data. In order to correctly normalise dose distributions for a given machine, the output factors for that machine must be known, and these are calculated by calling **s_photon** with a single parameter:

s_photon -U unit_id

where *unit_id* is the GRATIS unit number of the machine. This need be executed only once per machine (but must be re-executed every time the machine parameters of the dose calculation algorithm are altered). See Hoban[15] for a more detailed discussion of this process.

3.5.2. Synopsis of the Superposition Process

Figure 3.17 illustrates schematically how superposition is carried out under the GRATIS system. The first step in the calculation is to load all necessary data, including the beam description, treatment unit information, time parameters, trays, filters, and dose grid or point descriptor files. Then patient density information is loaded from the GRATIS image file, and cartesian kernels are interpolated from spherical-polar kernels stored on disk.

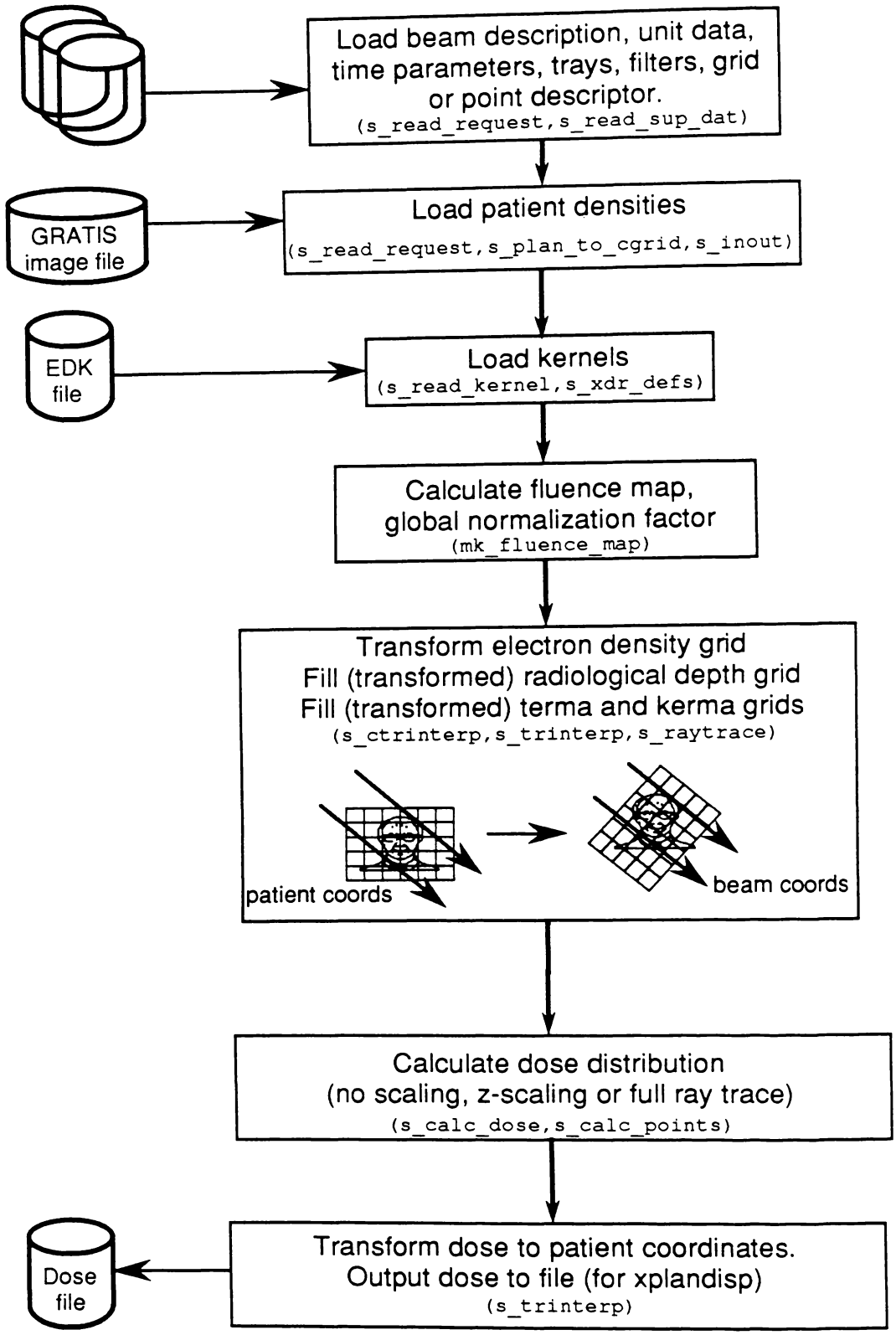

Figure 3.17. Schematic of the superposition process under GRATIS.

The fluence map and global normalisation factors are calculated, then the patient electron density grid is transformed into the beam coordinate system. Next, the (transformed) radiological depth, terma, and kerma grids are calculated using the techniques described in Section 3.3.2. The dose delivered to the grid or points is then calculated, and finally the dose grid or point list is transformed from beam to patient coordinates, and written to a file for input to **xplandisp**.

3.5.3. Organisation of the Superposition Code

The directory structure used for the superposition code is illustrated in Figure 3.18. Note that the parent directory **s_photon** is one of the subdirectories of **src**, shown in Figure 3.11. Source code modules are located in the **c** subdirectory — except those for making the kernels, which are located in the **phys_dat** hierarchy; and the module **mk_fluence_map()**, which is part of the **c_photon** source distributed with the GRATIS system (not shown in Figure 3.18). There are also some raw data files associated with each machine, located under the **MACHINES** directory. These are EDK files (for several densities); a header file containing the machine identification number, kernel densities, and details of the incident photon spectrum; and a file containing superposition normalisation data. The EDK and header files are compiled into binary format by **s_make_kernel** during the installation process, while the normalisation file is accessed during each superposition calculation.

The bracketed module names in Figure 3.17 illustrate where many of the source modules are used in the superposition process. These and other modules which make up the superposition code are now briefly described (principal author or authors in brackets):

(a) **s_main()** (Murray and Hoban): Top-level procedure for **s_photon**.

(b) **s_read_request()** (Murray and Hoban). Reads in command line parameters to **s_photon** discussed above (also refer to Appendix H for a description of these parameters). Allocates and reads external beam description, treatment unit description, time_calc parameters, tray and filter descriptions (if present), SAR file, skin anastruct, dose grid descriptor or point descriptor file, energy deposition kernels, and superposition parameters. Also opens output dose grid or dose point file. This routine is based on **read_request()** found in the **c_photon** code.

(c) **s_read_kernel()** (Hoban). Reads cartesian kernel header (using **xdr_EDKS_header()**) and spherical polar EDKs (using **xdr_PEDK()**). The spherical polar EDKs have previously been generated using **s_make_kernel()** (see below). The cartesian kernels are then interpolated from the spherical kernels by stepping through r and θ in small

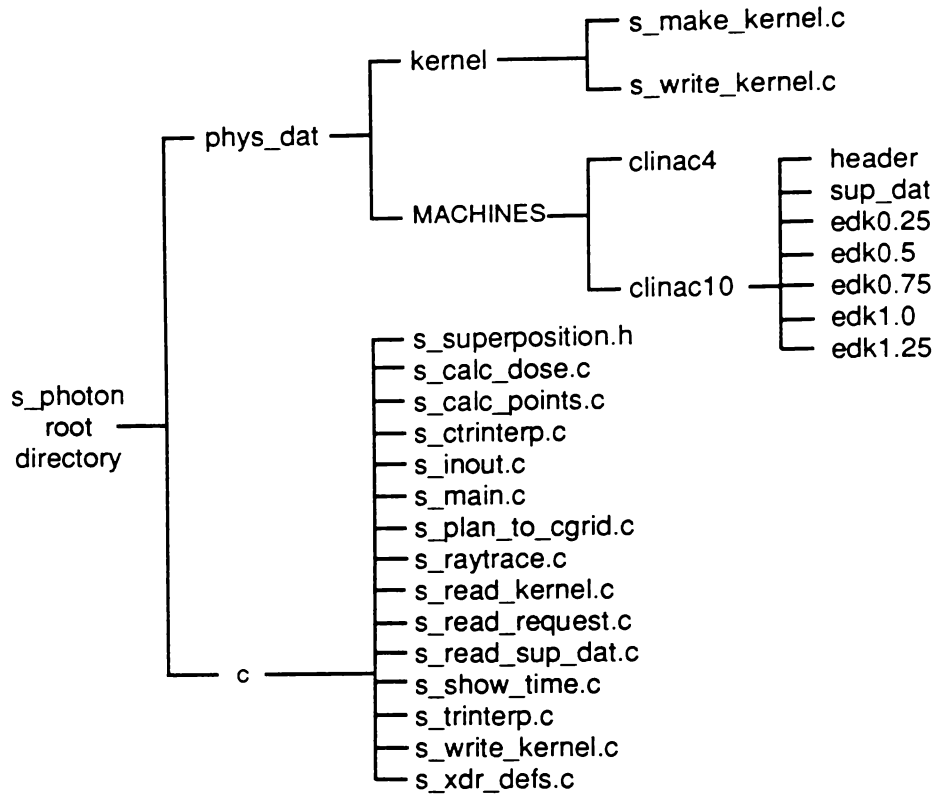

Figure 3.18. Directory structure for superposition code under GRATIS.

increments, assigning energy to the corresponding cartesian voxel (see Hoban[15]).

(d) **s_read_sup_dat()** (Hoban). Reads in data used for normalising superposition dose. This data has previously been created using the **-U** option to **s_photon** (also see (n) below).

(e) **mk_fluence_map()** (Sherouse, UNC). This routine, distributed with the **c_photon** algorithm, uses the beam outline and beam flatness tables to generate a fluence map defined at the treatment machine's SAD.

(f) **s_plan_to_cgrid()** (Murray). Converts GRATIS planning image to a grid of type **CGRID**, where each density value is represented by a byte value between 0 and 255 (corresponding to relative electron densities 0.0–2.55). This routine will skip over CT slices as necessary to ensure that the number of slices read in to the **CGRID** is not greater than the globally defined constant **MAX_SCANS**. It also reduces scan resolution to 128 × 128 points per slice. CT number is converted to electron density using the relationships discussed by Metcalfe *et al.*[20,21]

(g) **s_trinterp()** (Murray). Performs tri-linear interpolation on a three-dimensional grid, given coordinate interpolation fractions and voxel coordinates. Adapted from code

found in GRATIS's **grid_slicer** utility. Used for transforming between patient and beam coordinate systems.

(h) **s_ctrinterp()** (Murray). As with **s_trinterp()** but operating on a grid of single-byte integers (of type **char**), where each single-byte value is converted to an electron density by dividing by 100. Returns a floating point value. Used for interpolating between voxels of a patient electron density grid.

(i) **s_inout()** (Hoban). Determines whether a point in the electron density grid is inside or outside the skin contour. Called from **s_main()** for every point in the electron density grid, assigning a boolean value to **inside_contour**. Required in order to suppress calculation of dose for voxels outside the patient.

(j) **s_raytrace()** (Murray). Uses Siddon's parametric ray tracing algorithm to determine radiological depth, as discussed in Section 3.3.2. Appendix E presents this author's implementation of the algorithm.

(k) **s_calc_dose()** (Hoban). Calculates dose deposited due to terma liberated from a single interaction voxel (the "interaction point of view"). Uses one of three density scaling algorithms (no scaling, z-axis scaling, or full ray trace) as discussed by Hoban.[15] Accesses kernel and electron density grids, and modifies dose grid. Optimisation of **s_calc_dose()** by this author is discussed in Chapter 4.

(l) **s_calc_points()** (Hoban). Calculates dose deposited at a single voxel due to energy liberated from all other points within the patient (the "deposition point of view"). Used for calculating point doses when **s_photon** is called in point mode.

(m) **s_show_time()** (Murray). Displays elapsed real and CPU time (on a Sun workstation) since **s_photon** was invoked, and also since the last call to **s_show_time()**.

(n) **s_write_sup_dat()** (Hoban). When **s_photon** is called with the **-U** option, this routine generates a binary file containing normalisation data for superposition.

(o) **s_xdr_defs** (Murray and Hoban). Routines contained in this module are responsible for reading and writing spherical polar EDKs (**xdr_PEDK()**) and cartesian EDK header information (**xdr_EDKS_header()**) using Sun's XDR (external data representation) protocol. This module also contains other routines, discussed in Chapter 4, used for communicating between Sun workstations using the RPC (remote procedure call) paradigm.

There is also a separate executable image, **s_make_kernel**, used for generating binary spherical-polar kernels from data files generated by the EGS4 user code **RTPEDK** (see Chapter 2, Section 2.6.4). It uses Sun's XDR (external data representation) protocol to write the files in a machine independent format. **s_make_kernel** is automatically executed during the GRATIS installation procedure, but if kernels are subsequently modified then it must be run again by executing "**make install**" in the **kernel** directory.

3.5.4. Data Structures Used in Superposition

The superposition approach requires the definition of several data structures not previously used in GRATIS. They are contained in the header file **s_superposition.h**, located in the **c** subdirectory shown in Figure 3.18.

The most significant new structures are those associated with cartesian kernels. The entire set of kernels is represented using a single structure of type **EDKS**:

```
typedef struct {
    int unit_id;
    char name[NAME_LENGTH];
    int modality;
    int spectral_component_number;
    float component_energy[MAX_SPECTRAL_COMPONENTS];
    float component_weight[MAX_SPECTRAL_COMPONENTS];
    float component_mu_water[MAX_SPECTRAL_COMPONENTS];
    float component_mu_ab_water[MAX_SPECTRAL_COMPONENTS];
    float reference_kerma;
    float reference_terma;
    POINT inc;
    int edk_number;
    float edk_density[MAX_EDK_DENSITIES];
    EDK primary_edk[MAX_EDK_DENSITIES];
    EDK scattered_edk[MAX_EDK_DENSITIES];
} EDKS;
```

Note that this structure contains a complete set of information on the photon spectrum used to generate the kernels. The field **inc** (of type **POINT**) contains the (x, y, z) voxel increments. There are **edk_number** primary kernels and **edk_number** scattered kernels, with densities specified in the **edk_density** array. Each component kernel array, of type **EDK**, represents one quadrant of the entire kernel:

```
typedef struct {
    int x_count;
    int y_count;
    int z_count;
    int origin_depth;
    float *edk;
} EDK;
```

where **x_count**, **y_count** and **z_count** are the number of voxels in the x, y and z directions; **origin_depth** is the number of backscattered voxels in the z direction; and **edk** is a pointer

to a *one-dimensional* array containing the packed *three-dimensional* kernel. Voxel (i, j, k) within a specified kernel can be referenced using macros also defined in `s_superposition.h`. For example, `PRIMARY_EDK_VALUE` is used to reference primary kernels:

```
#define PRIMARY_EDK_VALUE(EDKS_ptr, density_index, i, j, k)\
    (EDKS_ptr)->primary_edk[density_index].edk[(i) +\
     EDKS_ptr)->primary_edk[density_index].x_count *\
     (j) + ((k)+(EDKS_ptr)->primary_edk[density_index].origin_depth) *\
     EDKS_ptr)->primary_edk[density_index].y_count)])
```

There are also some other new structures, notably those used for representing spherical-polar kernels (`PEDKS` and `PEDK`), superposition normalisation data (`SUP_DOSE_PARAMETERS`), and an electron density grid in patient coordinates (`CGRID`).

3.6. Summary

Many existing dose computation algorithms fail to accurately model electron transport, which is especially apparent in regions of heterogeneity such as the lung. Chapter 2 discusses the Monte Carlo approach to modelling particle transport, which has the potential of being arbitrarily accurate, given enough computer time. The superposition approach discussed in this chapter is essentially a Monte Carlo approach with all the non-determinism removed, and with much of the computation done in advance (when generating the energy deposition kernels). Although density scaling a pre-computed kernel must always be less accurate than full Monte Carlo simulation when in a heterogeneous medium, the superposition algorithm *does effectively model electron transport.* Even the more simplistic convolution approach, using much more efficient Fourier space transformation, performs better than conventional algorithms in heterogeneous regions.

The (real space) superposition algorithm developed at Waikato University has been implemented as part of the GRATIS Treatment Planning System. This combination has provided an attractive and functional user-interface to a very powerful dose computation algorithm. As always, the disadvantage with this algorithm is the computation time involved — approximately 5 minutes per typically-sized beam using no kernel density scaling, 7 minutes using a simple z-scaling algorithm, and approximately 3 hours using full point-to-point scaling (on a Sun SPARCstation IPC).

Even with a *simple* kernel scaling algorithm, superposition is nonetheless superior to conventional algorithms. However, the ultimate goal must be a highly sophisticated (and hence physically accurate) kernel scaling algorithm, which will necessarily require large computational effort. Chapter 4 discusses one approach to providing the computing resources needed for such an algorithm.

References

(1) Dean R.D., A scattering kernel for use in true three-dimensional dose calculations. *Med. Phys.* (1980), **7** 429.

(2) Mackie T.R., Scrimger J.W. and Battista J.J., A convolution method of calculating dose for 15–MV x rays. *Med. Phys.* (1985), **12** 188.

(3) Boyer A.L. and Mok E.C., A photon dose distribution model employing convolution calculations. *Med. Phys.* (1985), **12** 169.

(4) Mohan R., Chui C. and Lidofsky L., Differential pencil beam dose computation model for photons. *Med. Phys.* (1986), **13** 64.

(5) Ahnesjö A., Andreo P. and Brahme A., Calculation and application of point spread functions for treatment planning with high energy photon beams. *Acta Oncologica* (1987), **B26** 49.

(6) Boyer A.L. and Mok E.C., Calculation of photon dose distributions in an inhomogeneous medium using convolutions. *Med. Phys.* (1986), **13** 503.

(7) Boyer A.L., Zhu Y., Wang L. and Francois P., Fast Fourier transform convolution calculations of x-ray isodose distributions in homogeneous media. *Med. Phys.* (1989), **16** 248.

(8) Ahnesjö A., Collapsed cone convolution of radiant energy for photon dose calculation in heterogeneous media. *Med. Phys.* (1989), **16** 577.

(9) Zhu Y. and Boyer A., X-ray dose computations in heterogeneous media using 3-dimensional FFT convolution. *Phys. Med. Biol.* (1990), **35** 351.

(10) Murray D.C., Hoban P.W., Metcalfe P.E. and Round W.H., 3-D superposition for radiotherapy treatment planning using fast Fourier transforms. *Australas. Phys. Eng. Sci. Med.* (1989), **12 (3)** 128.

(11) Metcalfe P.E, Hoban P.W., Murray D.C. and Round W.H., Modelling polychromatic high energy photon beams by superposition. *Australas. Phys. Eng. Sci. Med.* (1989), **12 (3)** 138.

(12) Hoban P.W., Murray D.C., Metcalfe P.E. and Round W.H., Superposition dose calculation in lung for 10 MV photons. *Australas. Phys. Eng. Sci. Med.* (1990), **13 (2)** 81.

(13) Metcalfe P.E., Hoban P.W., Murray D.C. and Round W.H., Beam hardening of 10 MV x-rays: Analysis Using a convolution/superposition method. *Phys. Med. Biol.* (1990), **35 (11)** 1533.

(14) Metcalfe P.E., *X-ray Beam Modelling in Radiotherapy: The Effect of Lung Inhomogeneities.* University of Waikato D.Phil. Thesis (1990).

(15) Hoban P.W., *Lateral Electron Disequilibrium in Radiotherapy Treatment Planning.* University of Waikato D.Phil. Thesis (1991).

(16) Goitein M., Limitations of two-dimensional treatment planning programs. *Med. Phys.*. (1982), **9** 580.

(17) Mohan R., Dose calculations for three-dimensional radiation treatment planning. *Austral. Phys. Eng. Sci. Med.* (1989), **12 (4)** 241.

(18) Mohan R., Barest R., Brewster L.J., Chui C.S., Kutcher G.J., Laughlin J.S. and Fuks Z., A comprehensive three-dimensional radiation treatment planning system. *Int. J. Rad. Onc. Biol. Phys.* (1988) **15 (2)** 481.

(19) Mackie T.R., Bielajew A.F., Rogers D.W.O. and Battista J.J., Generation of photon energy deposition kernels using the EGS Monte Carlo code. *Phys. Med. Biol.* (1988), **33** 1.

(20) Metcalfe P.E. and Beckham W.A., Radiotherapy planning accuracy in terms of C.T. numbers and inhomogeneity correction techniques. *Australas. Radiol.* (1988), **32** 371.

(21) Metcalfe P.E, Beckham W.A., Long B.H. and Battista J.J, The effect of patient density variation on radiotherapy dose calculations. *Australas. Phys. Eng. Sci. Med.* (1988), **11 (3)** 107.

(22) Siddon R.L., Fast calculation of the exact radiological path for a three-dimensional CT array. *Med. Phys.* (1985), **12** 252.

(23) Mohan R., Chui C. and Lidofsky L., Energy and angular distributions of photons from medical linear accelerators. *Med. Phys.* (1985), **12** 592.

(24) Johns H.E. and Cunningham J.R., *The Physics of Radiology,* 4th Edition. Charles C. Thomas (1983).

(25) Chui C. and Mohan R., Extraction of pencil beam kernels by the deconvolution method. *Med. Phys.* (1988), **12** 138.

(26) Nelson W.R, Hirayama H. and Rogers D.W.O., *The EGS4 Code System.* Stanford Linear Accelerator Report SLAC–265 (1985).

(27) O'Connor J.E., Variation of scattered x-rays with density in an irradiated body. *Phys. Med. Biol.* (1957), **1** 352.

(28) Mackie T.R., *The study of megavoltage x-ray beams.* University of Alberta Ph.D. Thesis (1985).

(29) Field G.C. and Battista J.J., Photon dose calculations using convolution in real and Fourier space: Assumptions and time estimates. In *The Use of Computers in Radiation Therapy, p103,* edited by Bruinvis I.A.D. *et al.*, Elsevier Science Publishers (North-Holland) (1987).

(30) Agarwal R.C. and Cooley J.W., New algorithms for digital convolution. *Proceedings of the 1977 International Conference on Acoustics, Speech and Signal Processing (Hartford), p360* (1977).

(31) Brigham E.O, *The Fast Fourier Transform.* Prentice-Hall, Englewood Cliffs, N.J. (1974).

(32) Dudgeon D.E. and Mersereau R.M., *Multidimensional Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, N.J. (1984).

(33) Winograd S., A new principle for fast Fourier transformation. *IEEE Trans. Acoust., Speech, Signal Process.* (1976), **ASSP–24** 264.

(34) Nussbaumer H.J., *Fast Fourier Transform and Convolution Algorithms* (2nd Edition). Springer-Verlag, N.Y. (1982).

(35) Elliott D.F. and Rao K.R., *Fast Transforms: Algorithms, Analyses, Applications.* Academic Press, Orlando, FL. (1982).

(36) Murray D.C., Hoban P.W., Graham I.D.G., Round W.H. and De Vel O.Y., Radiotherapy treatment planning using transputers. *N.Z. Jnl Comput.* (1989), **1 (2)** 30.

(37) Murray D.C., Hoban P.W., Round W.H., Graham I.D. and Metcalfe P.E., Superposition on a multi-computer system. *Med. Phys.* (1991), **18** 468.

(38) Newman W.M. and Sproull R.F., *Principles of Interactive Computer Graphics (2nd edition).* McGraw-Hill (1979).

(39) Siddon R.L., Solution to treatment planning problems using coordinate transformations. *Med. Phys.* (1981), **8** 766.

(40) Sherouse G.W., *GRATIS Programmer's Manual.* University of North Carolina, Chapel Hill (1990).

Chapter 4

Dose Computation: A Parallel Approach

4.1. Introduction

In Chapter 3 the superposition dose computation algorithm is discussed. Although it has significant advantages over established algorithms, superposition requires far more computational effort than is necessary with any of the other techniques discussed in Chapter 1 (except Monte Carlo). However, the need for a rapid treatment planning cycle remains. These two competing requirements suggest the use of more powerful computing technology, and parallel processing is one way of obtaining the desired computational power.

There are several arguments against the use of parallelism to accelerate computation. Summarised by Quinn,[1] they are briefly described here:

- **Grosch's law,** that the speed of computers is proportional to their cost. This may be true within a given *class* of computer, but is not necessarily true *across* classes. For example, the cost of a microprocessor-based parallel processing system will generally be less than that of a correspondingly powerful vector processor. Thus multiprocessors may be more cost-effective solutions for many types of problems.

- **Minsky's conjecture,** that the speedup achievable by a parallel computer increases as the logarithm of the number of processing elements. For certain types of algorithm this statement is untrue: linear or near-linear speedup is observed for many problems.

- **Amdahl's law,** that sequential operations limit the speedup of a parallel algorithm. For example, an algorithm which has a 10% component of serial instructions can never run more than ten times as fast on equivalent parallel hardware. If f is the fraction of the computation which must be performed sequentially, then the maximum speedup S on a machine of p processors is limited such that:[1]

$$S \leq \frac{1}{f + (1-f)/p} \quad . \tag{4.1}$$

Sections of this chapter appear in *Medical Physics*, **18 (3)** 468–473 (1991), under the title "Superposition on a multicomputer system," and in the *New Zealand Journal of Computing*, **1 (2)** 30–38 (1989), under the title "Radiotherapy treatment planning using transputers."

Note that when the number of processors p becomes sufficiently large, the factor $(1 - f)/p$ in Equation 4.1 becomes insignificant, and negligible speedup is achieved by further increasing the processor number. Amdahl's law effectively reduces the attraction of parallel processing for many applications, but many algorithms are entirely or almost entirely parallel in nature, in which case large-scale parallelism can usefully be employed.

- **Software inertia,** an argument which raises the difficulty of re-writing many existing applications to run on parallel processors. Although there is much work being done to develop software tools which automatically rewrite code into a parallel form, intuition does play an important role in recognising potential parallelism, and therefore new skills *will* be required of programmers.

Two important conclusions can be drawn from the above discussion. First, the algorithm must be suitable for parallel implementation, or Amdahl's law (and other limitations) will severely limit the potential speedup which can be achieved. Secondly, provided the algorithm is suitable, a parallel machine will *always* run faster than a sequential machine using equivalent technology. In applications where every increase in speed is important, such as radiotherapy dose calculation, this second factor ensures that parallel processors have a future.

4.2. Approaches to Parallelism

4.2.1. Overview

Ever since the development of the computer, scientific applications have consistently demanded more computational power than can be delivered with the prevailing technology of the time. In general these requirements have been satisfied with very powerful single processors, often employing some vector processing capability. However, more recently a wide variety of supercomputers have appeared,[2] and many of these are based on other types of parallel processing architecture. The future of these high-performance computer systems in science applications seems assured.[3] Full discussion of parallel processing systems, and their use in solving various types of problem, can be found in texts by Quinn,[1] Hockney and Jesshope,[4], Krishnamurthy,[5] Desrochers,[6] and others.

Parallel computer architectures have been defined in many ways, but one of the most common approaches is the taxonomy of Flynn.[7,8] He classifies any computer into one of four categories:

- Single-instruction stream single-datastream (SISD). Conventional (non-parallel) computers fall into this category.

- Multiple-instruction stream single-datastream (MISD), where several different instructions act on the same datastream simultaneously. Many authors consider pipelined processors to be MISD machines.

- Single-instruction stream multiple-datastream (SIMD), most commonly *array processors* (or *processor arrays*). In SIMD machines, processors execute identical instructions on different data sets.

- Multiple-instruction stream multiple-datastream (MIMD), where a number of independent processors operate on their own sets of data. Such systems may share memory (multiprocessors) or have separate memory (multicomputers).

Another class of parallel computer, the *vector* processor, is difficult to classify using this taxonomy. These machines can be classified as SISD machines because they have only one processor, or as SIMD machines because they perform an identical sequence of operations on several streams of data (vector elements). However, they usually incorporate *pipelining*, where functional units are arranged such that the output of one functional unit becomes input to the next. This feature implies an MISD classification. Due to this confusion, vector processors are often not classified using Flynn's taxonomy. Other schemes, such as Shore's taxonomy[9] or Krishnamurthy's taxonomy tree,[5] classify parallel architectures more fully.

Some other terminology is also used in describing parallel systems. First, the degree of *coupling* can be defined, where

- *Tightly-coupled* processors share common memory and operate under centralised control.

- *Loosely-coupled* processors have separate memories and operate independently.

Secondly, since a process executing in parallel may need to interact with another process, a method of *synchronisation* is also required. This generally falls into one of two categories:[10]

- *Shared variables* under centralised control, including busy waiting, semaphores, conditional critical regions, monitors, and path expressions.

- *Messages* passed between processes (also known as handshaking).

Thirdly, at the hardware level communication between processes may be either:

- *Synchronous,* where a central clock is used to ensure data movement occurs simultaneously on all processors, or
- *Asynchronous,* where each processor moves data according to its own clock.

Finally, *control* may be:

- *Centralised,* where a single unit with knowledge of the entire topology controls the operation of all processors.
- *Distributed,* where each processor, knowing details of local connections only, controls its own operation.

In general, SIMD machines operate synchronously using centralised control, whereas MIMD processors (multiprocessors and multicomputers) operate asynchronously using distributed control.

The following sections examine in more detail some of the most commonly found parallel architectures. Other architectures (such as systolic processor systems) also exist, but the intention here is to focus on those systems which could be successfully turned to a variety of problems encountered in the radiotherapy field.

4.2.2. Vector Processors

Vector processors normally utilise both MIMD and SIMD features — that is, they employ pipelining as well as vector processing. Ibbett and Topham[11] present many examples of these machines, but the *Cray* and *CDC-Cyber* families are the most widely cited examples of vector computers. In general these machines share two features: the ability to perform operations on a large number of elements simultaneously, and the ability to pipeline instructions. For example, the Cray-2 operates on 64-element vectors with a pipeline of up to nine functional units.

A critical requirement of pipelined vector processors is that they can be programmed in such a way as to fully utilise their capabilities. This can be most efficiently done in the native assembly language of the processor, but since many applications are written in high-level languages, particularly FORTRAN, this approach is often time-consuming and difficult. A more convenient solution is to use a *vectorizing compiler,* designed to automatically exploit any parallelism inherent in the sequential code.

Many problems exhibit a high susceptibility to vectorization. For example, in both the real-space superposition and FFT convolution algorithms discussed in Chapter 3, essentially identical operations are being performed on a large number of elements. On the other hand,

the EGS4 Monte Carlo code is not suitable for vectorization, since each particle history is completely different from all others — this is implicit in the nature of the Monte Carlo technique itself. However, by using an *event-based* algorithm where a vector of particle steps (from different histories) are formed, considerable vectorization can be achieved, as demonstrated by Brown and Martin on a Cyber-205.[12] Miura[13] succeeded in vectorizing the EGS4 code itself, but the effort involved in recoding the algorithm was considerable.

Pipelined vector processors are, in general, expensive and complicated machines, requiring a mainframe processor as a "front-end" to the vector processor. This level of complexity and expense makes the vector approach unsuitable for use in radiotherapy treatment planning workstations.

4.2.3. Array Processors

Array processors (sometimes called *processor arrays*) are a less expensive alternative to pipelined vector processors. Array processors are available for a wide variety of machines, from mainframes through minicomputers to microcomputers such as IBM–PC compatible machines.

Array processors are SIMD machines (see Section 4.2.1). That is, a single set of instructions is executed on many processors, with each processor operating on a different array element. Similarly, array processing languages support *data-level* parallelism, where each instruction is applied to a number of data elements in parallel. There is often some degree of functional unit pipelining within each processor. In general, array processors must also have a *scalar* processor, since some instructions will be sequential in nature.

In a *lockstep array*, each processor receives a set of inputs in time with a common clock, and also produces output in time with that clock. However, in *cyclic arrays* each processor operates independently of the others, with the controlling unit allocating operands as each processor becomes free. Array processors are generally of the lockstep variety.

The memory associated with an array processor can be either *distributed*, so that each processor has its own local memory, or *globally shared*, so that each processor has access to all elements of the array. Globally shared memory is obviously a more versatile approach, but the distributed memory approach has advantages in that the switching network connecting memory to processors can be much simpler, leading to reductions in cost and increases in memory access performance. Distributed memory machines with very large numbers of processors are commonly called *massively parallel* machines — examples include the

Goodyear Aerospace MPP, the *Connection Machine*, and more recently Digital Equipment Corporation's *MasPar* processor.

The performance of an array processor in executing any particular algorithm depends upon the degree to which the algorithm can be made parallel (that is, the ratio of parallel array-type instructions to sequential-type instructions), and also on the way in which the parallel parts of the algorithm can be mapped onto the processing elements.

4.2.4. Multiprocessors

MIMD machines employing shared memory are known as multiprocessors. Unlike array processors, these machines are capable of operating independently on several *streams* of data. Both multiprocessors and multicomputers (see Section 4.2.5) may be *coarse-grained* (a few large processors), *medium-grained* (many smaller processors), or *fine-grained* (very many small processors).

As mentioned in Section 4.2.1, multiprocessors may be *tightly coupled*, where all processors use a central switching mechanism to access shared memory (examples of this type of processor are the *HEP*, *C.mmp*, and *Sequent Balance 8000* machines). The switching mechanism can be implemented using a bus, crossbar switch, packet switched network, or some other interconnection network. When the number of processors in the network becomes very large, the efficiency of communication becomes a dominant factor in the performance of this type of machine.

In contrast, *loosely coupled* multiprocessors achieve shared memory by combining the local memory of each processor in the network. The time required to access a particular memory location depends upon the ease with which that location can be reached from the processor in question, and hence algorithms with strong locality of reference tend to perform well on loosely-coupled multiprocessors. The interconnection network and message passing mechanism used to retrieve memory contents held elsewhere can have many forms. These are described briefly in Section 4.3.1.

4.2.5. Multicomputers

Multicomputers are similar to multiprocessors, except that they have no shared global memory. The most common examples of this type of machine are the *Intel iPSC* (based on the *Intel 80286* processor), and a variety of machines based on the *Inmos transputer*. Reed and Fujimoto[14] identify some key characteristics of multicomputer networks:

- A relatively large number of processors. Since the basic processor element (a microprocessor) is inexpensive, multicomputer networks tend to be medium or fine grained.

- Message-based communication. Since multicomputers are loosely coupled (have no shared memory), shared variable communication is not easily implemented, and hence message passing is always used.

- Medium-grained computation. The amount of computation between synchronisations cannot be too small (communication and context-switching will cause inefficiencies), or too large (uneven load sharing will occur as some processors become idle towards the end of the computation).

- Asynchronous execution. Each processor clock operates independently of all others, with process synchronisation being achieved under software control.

- Program decomposition which is managed by the user. Unlike vector and array processors, where software tools are available to automatically handle division of the computation into sections which can be executed in parallel, multicomputers rely on the user to identify and implement any available parallelism.

Discussions of multicomputers and their characteristics have been provided by Ibbett and Topham,[15] and Athas and Seitz.[16] Reed and Fujimoto[14] examine in detail techniques used in programming multicomputer systems.

4.2.6. Summary

The previous sections discuss the broad categories into which parallel processing systems fall, based on Flynn's taxonomy. Section 4.2.2 discusses vector processors, which are large, expensive machines designed for very rapid processing of vector quantities. Section 4.2.3 discusses array processors (processor arrays), less expensive and (usually) less powerful, and Sections 4.2.4 and 4.2.5 discuss multiprocessors and multicomputers, two very similar types of MIMD machine.

Which type of parallel processing architecture is best for a radiotherapy treatment planning system? Pipelined vector processors are certainly too expensive for treatment planning workstations, as are massively parallel arrays. Conventional array processors are more suitable, but are usually host-specific and can be relatively difficult to program for complex applications. A good solution then seems to be an MIMD machine, a class of parallel processor which offers considerable flexibility and is currently one of the most cost-effective approaches to obtaining large amounts of computational power. Particular issues concerning MIMD processors (multiprocessors and multicomputers) are addressed in the following sections.

4.3. Issues in MIMD Parallelism

4.3.1. Interconnection Networks

The distinct processing elements of multiprocessors and multicomputers can be connected in a variety of ways. Generally, the optimal interconnection network depends upon the particular problem being solved, and several of the more obscure types of network are used in solving specialised classes of problems.

The most flexible type of interconnection network is undoubtedly a *crossbar switch*, where every processor can be connected to every other processor. However, for all but the smallest networks this requires a very large number of interconnections (whose cost is proportional to n^2, where n is the number of processors[15]). Therefore, one of the following less general communication network topologies is usually used:

- A **pipeline**, where processors are connected in a linear array.
- A **ring**, similar to a pipeline except that the ends are connected together.
- A **star**, where a central processor is connected to all other processors. The size of the network is limited by the number of connections available from the central processor.
- A **rectangular mesh**, where each processor is connected to its $2q$ neighbours, q being the *dimension* of the mesh. This type of network is commonly used in processor arrays (where $q = 2$). Sometimes processors on the edge of the mesh are connected to the opposite edge, or even to the next row or column in a toroidal fashion.
- A **tree**, where two or more "child" processors are attached to each upper-level processor in the network. If all the processors on each level of the tree are connected together in a mesh, then the resulting topology is called a **pyramid**.
- A **binary k-cube** (or **hypercube**), where 2^k processors are connected in a k-dimensional cube.

These topologies are illustrated schematically in Figure 4.1. Other approaches such as *shuffle-exchange networks*, *butterfly networks*, and *cube-connected cycles* are also used. Several texts examine the issue of network topology in detail.[1,14,15,17] The specific advantages of certain topologies can be combined by using a *dynamic network*, where the topology can be changed under program control. This approach is commonly used for general-purpose processing systems, where the type of inherent parallelism varies according to the problem being solved.

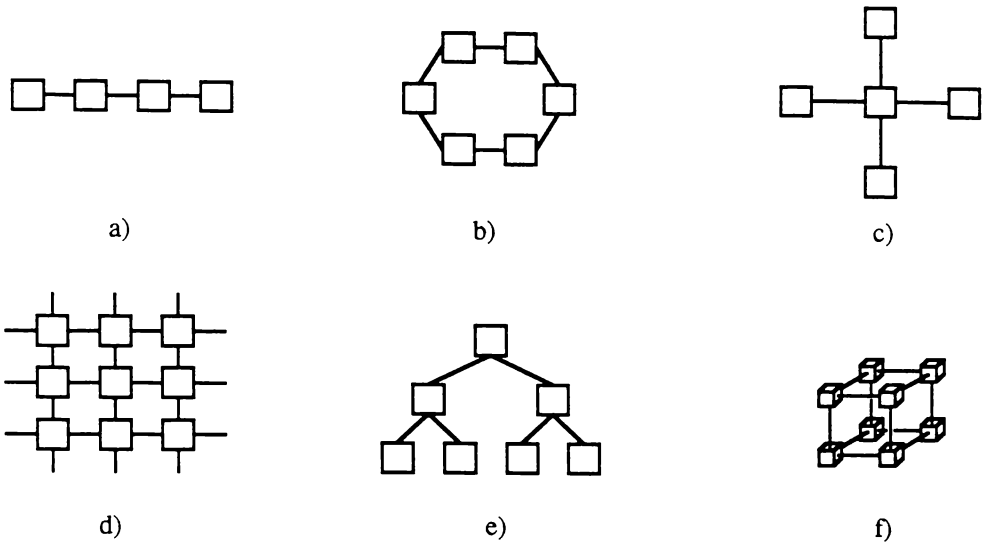

Figure 4.1. Some network topologies used in MIMD systems. (a) Pipeline (or linear array). (b) Ring. (c) Star. (d) Rectangular mesh. (e) Tree. (f) Binary 3-cube.

How can these various topologies be compared? Obviously the effectiveness of any given topology depends upon the details of the problem being solved, but the following can be used as general measures:[5,14]

- **Diameter,** the maximum cost of getting a message from one processor to another.
- **Mean internode distance,** the *average* number of hops required to get a message from one processor to another.
- **Bandwidth,** the number of messages that can be sent/received by a processor per unit time.
- **Message capacity,** the *total* number of messages that can be sent through the network per unit time.

Several other metrics have also been suggested for measuring the communication performance of processor networks.[18] As well as performance, other factors such as *reliability*, *fault tolerance* (especially important in very large systems), and *extensibility* (the ease with which the network can be extended, and the corresponding cost) may be important.

4.3.2. Communication and Synchronisation

In MIMD systems it is necessary that processes executing on different processors can *communicate* (pass data between themselves). This process in turn requires *synchronisation*, either to force a *critical section* of code to be executed as an atomic (indivisible) operation, or to perform *condition synchronisation*, where a process waits until some shared data item

is in a required state. These objectives can be achieved using one of two general schemes, discussed below.

Shared Variables

One method of communication between processes is by the use of *shared variables*. This approach is commonly used in multiprocessors, since shared memory is always available on these machines. However, there is clearly also a need for some form of explicit synchronisation between processes using shared variables. This usually takes one of the following forms:[10]

- **Busy waiting,** where processes repeatedly test the value of a shared variable (or *spin lock*) until its value is such that they can proceed. The major disadvantage of such a technique is that the idle (or *spinning*) processes waste processor power.

- **Semaphores,** where a process issues a *wait* command to request a resource, and a *signal* command to release that resource for use by other processes. This approach is similar to busy waiting, except that semaphores are usually handled by the processor kernel, which automatically deschedules processes that are waiting. Hence this approach is generally more efficient than busy waiting when there is more than one process executing on each processor.

- **Conditional critical regions (CCRs),** where a critical section of code is defined such that only one process can execute the code at any given time. Synchronisation is provided by requiring that an associated boolean variable be true before the CCR can be entered.

- **Monitors,** which are special sections of code that can be accessed by only one process at a time. However, a monitor differs from a CCR in that it contains *everything* associated with a particular resource — variables representing its state, initialising code, and code to perform operations on the resource. Hence the resource can only be accessed from within the monitor. Synchronisation is achieved within the monitor by using semaphores.

- **Path expressions,** which explicitly specify the manner and order in which resources can be used by processes. This approach removes the need for the programmer to specify critical regions, but it can be difficult to predict synchronisation requirements, especially for nondeterministic algorithms.

Message Passing

The second method of communicating between processes is to use *message passing*. Both communication and synchronisation are provided by this approach, since receiving a message implicitly implies that the receiving process has had to wait for that message. Since multicomputers have no shared memory, message passing is always used in these systems. It occurs in the following forms:

- **Direct naming,** where the source and destination processes reference each other explicitly.
- **Mailboxes,** where source processes send messages to an intermediate destination (mailbox), and destination processes receive messages from that mailbox.
- **Ports,** which are like mailboxes except that each has only one *source* of messages (it can have many destinations for those messages). This approach is easier to implement on multicomputers.

The channels across which messages are passed can be named either *statically* (at compile time), or *dynamically* (at run-time). If both the sending and receiving processes cannot continue until the message has been transmitted then this is called *synchronous* message passing, but if the messages are buffered such that the sending process can get ahead of the receiving process then this is called *asynchronous* message passing.

Parallel programming languages which implement concurrency using message passing are known as *message-oriented* languages. Examples in common use include Communicating Sequential Processes (CSP) and occam. As well as procedure-oriented and message-oriented languages, there is a third approach — *operation-oriented* languages (such as Ada), which use remote procedure calls to implement concurrency. However, shared variables, message passing, or a combination of both must be used in the underlying implementation of these languages.

4.3.3. MIMD Algorithms

Given a particular problem to solve, the parallel programmer must decide upon the best way of implementing the algorithm on a parallel machine. For MIMD computers, these algorithms belong to one of three categories:[1]

- **Pipelining,** where the output of one *segment* of the computation is input to the next (which may be on a different processor). In order for this type of algorithm to work

efficiently, each segment must produce output at nearly the same rate, or else processor capability will be wasted while waiting for the next input.

- **Partitioning,** where the computation is *shared* between processors. The results from all processors are then combined to yield the complete solution. This type of algorithm may be *pre-scheduled,* where each process is allocated a share of the work before execution, or *self-scheduled,* where each processor is allocated work during execution. Partitioning algorithms are often used when dealing with arrays or other structures which have a large number of readily decomposable tasks.

- **Relaxation,** where processes work without synchronisation. In this type of algorithm, each processor can proceed without waiting for data or instructions from other processors.

In order to determine which type of algorithm is best for a particular problem, there must be some way of analysing the *performance* of these algorithms on a parallel processing machine. This is usually done using the following two measures:

- **Speedup,** ratio of the time needed to perform the most efficient sequential algorithm to the time needed to perform the computation in parallel.[1] Note that this definition does not require that the two algorithms be of a similar nature. Intuitively, speedup can be no more than *linear* with increasing processor number, but Quinn[1] argues that *superlinear* speedup is possible if the sequential and parallel algorithms are chosen *before* the particular *instance* of the problem is determined.

- **Efficiency,** the speedup of the algorithm divided by the number of processors. It is a measure of the degree of success with which the algorithm has been implemented in parallel. A value much less than 1.0 (or 100%) indicates either an inefficiency in the parallel algorithm or excessive communication overhead.

The aim of any parallel algorithm is to keep all processors as busy as possible. Factors which affect this include:

- **Network architecture** (since this may affect communication overhead).

- The proportion of **sequential** code (Amdahl's law).

- The amount of **contention** for resources (leading to possible *software lockout* or even *deadlock*).

- **Load balancing** between processors. Process workload can be determined *statically* (before execution) or *dynamically* (during execution). Dynamic load balancing is usu-

ally used when the amount of computation required is not precisely known before execution.

- The **granularity** of the algorithm. The process granularity is defined as the ratio of the amount of computation to the number of synchronisation events.[15] *Fine-grained* processes, having many synchronisation events, tend to produce good load balancing, but at the expense of high communication overhead.

4.3.4. Summary

Major issues concerning MIMD parallelism are discussed in the previous sections. Accepting that the superposition algorithm involves calculation over a large dose array, and that *partitioning* is the most suitable method for dividing the workload between processors, two hardware solutions are immediately apparent:

- **A multiprocessor network,** where a shared global memory is used to store input arrays (electron density and kernels) and the output array (dose). In general, memory locations from throughout the array would have to be accessed in the superposition process, so a tightly coupled multiprocessor would be preferable to a loosely coupled (distributed-memory) one.
- **A multicomputer network,** where copies of the input and output arrays are stored in the local memory of each processor. Only when the superposition calculation is complete would the results from all processors be combined to yield the total result.

The multiprocessor approach would obviously require less memory, and therefore might be less expensive. However, for this project a multicomputer network has significant advantages:

- Once arrays have been passed to all processors, communication overhead is very low, since only the voxel indices and terma need to be passed to the processor performing the calculation.
- The low communication overhead permits a relatively simple interconnection network to be used, which simplifies programming of a message-passing multicomputer.
- One particular type of multicomputer processor (the *transputer*) has already been used for a number of years at Waikato University, in a variety of applications.

Hence a multicomputer based on *Inmos T800 transputers*[19] has been chosen for use as a dose calculation engine. The remainder of this chapter describes two variants of a transputer-based multicomputer (one PC-hosted and the other hosted by a Sun workstation), and then

discusses their application to dose computation in the radiotherapy treatment planning process.

4.4. Transputer Arrays

4.4.1. Hardware and Architecture

4.4.1.1. The Transputer Processor

Transputers[19] are 32-bit computing elements designed for parallel processing. Manufactured by the British (and now French-owned) company *Inmos*, 20 MHz versions of the T800 transputer are capable of operating at 10 MIPS (million instructions per second), and have a 64-bit floating point unit capable of 1.5 MFlops (million floating point instructions per second). Newer versions of the T800, operating at 25 MHz, have been used on the Sun-hosted system discussed in Section 4.4.1.3. Transputers address their own local memory, 4 kbytes of which is on-chip static RAM (random access memory). If this fast memory is used as stack and/or code space then significant improvement in performance can occur. Each transputer processor communicates with others via four serial links, each with a speed of 20 Mbits/sec (corresponding to a data transfer rate of approximately 1.75 Mbytes/sec using the transputer's communication protocol). A network of transputers is generally classified as a *multicomputer* — that is, an MIMD machine with local memory. They are usually loosely coupled, message-passing, asynchronous machines with distributed control.

Transputers are commonly mounted on industry-standard printed circuit boards called TRAM (Transputer RAM) modules.[20] TRAM modules have standard pin-outs and pin spacings for mounting in many different types of motherboard — the two motherboard types discussed in this chapter are designed for PC-bus and double-extended Eurocard systems. Using TRAMs allows processing elements to be easily replaced or rearranged within the network. Transputer modules generally contain a single T800 processor and a quantity of external memory, although a variety of special-purpose modules, such as graphics and SCSI (Small Computer Systems Interface) TRAMs are also available.

4.4.1.2. The PC-hosted System

Initial development of the superposition code has been carried out on the PC–AT hosted parallel processing system illustrated in Figure 4.2. The microcomputer is a Commodore–PC40 with 110 Mbytes of disk storage, colour display, tablet, and colour inkjet printer attached.

As shown schematically in Figure 4.2, the transputers are located on two boards housed within the PC. These boards are:

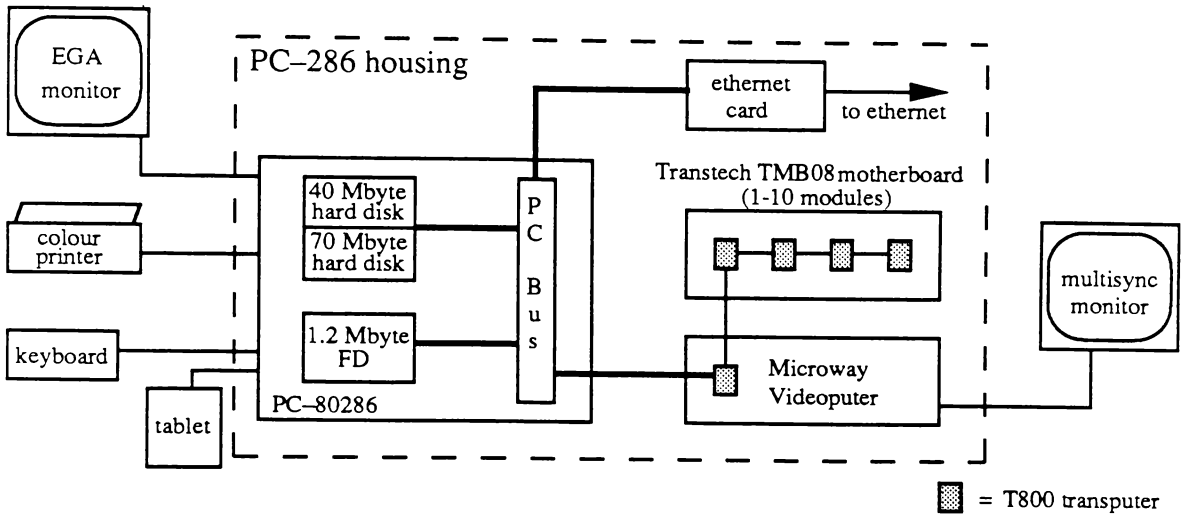

Figure 4.2. Hardware configuration of PC-hosted system. A Microway Videoputer acts as the root processor of a network hosted by a PC–AT compatible microcomputer (dashed box). Up to ten transputer modules on a Transtech TMB-08 (Inmos B004-compatible) motherboard can be connected to the Videoputer in a pipeline configuration. Inmos C004 link multiplexer and connections (for network reconfiguration) are not shown.

- A Microway Videoputer display board[21] (Microway Ltd, Surrey, England). The Videoputer has one T800–20 (20 MHz) transputer with 2 Mbytes of memory and video driver circuitry. The transputer is connected to the PC–AT bus via a host interface operating at 0.02 Mbytes/sec. One megabyte of memory is dual-ported to both the transputer and the video display hardware, while the other megabyte is used for the dose calculation master and worker tasks. A further 3 Mbytes of memory are available as an option. The graphics hardware supports a variety of screen resolutions, with a library of graphics primitives also supplied. The display used is an NEC Multisync monitor, operating in 512 pixel × 512 pixel mode.

- A Transtech TMB-08 ten module motherboard[22] (Transtech Devices Ltd, Buckinghamshire, England). Up to eight modules, each containing one T800–20 transputer and an associated 2 Mbytes of memory, have been installed on this motherboard. Each module forms a node in a linear array (pipeline), and a T212 processor and C004 link multiplexer can be programmed to connect the unused links to form other topologies. A PC interface with full DMA (direct memory access) and interrupt capability is also provided, although this has been left unused for the applications discussed in this chapter. An external D-type connector is available for connection to a larger network.

In the usual configuration, the Videoputer is connected to the PC bus, making it the root node of the network. Module 0 of the TMB-08 motherboard is connected to a link from the

Videoputer processor, with the remaining modules connected in a pipeline from module 0. Figure 4.3 illustrates the connections required for a four-node system. In some situations, such as configuration into a tree network for the timing tests discussed in Section 4.4.3.1, module 0 of the Transtech motherboard has been connected directly to the host processor. This has allowed a network configuration program executing on the host processor to have direct access to the C004 link multiplexer. Appendix I details the link patch, switch, and jumper settings for both network configurations.

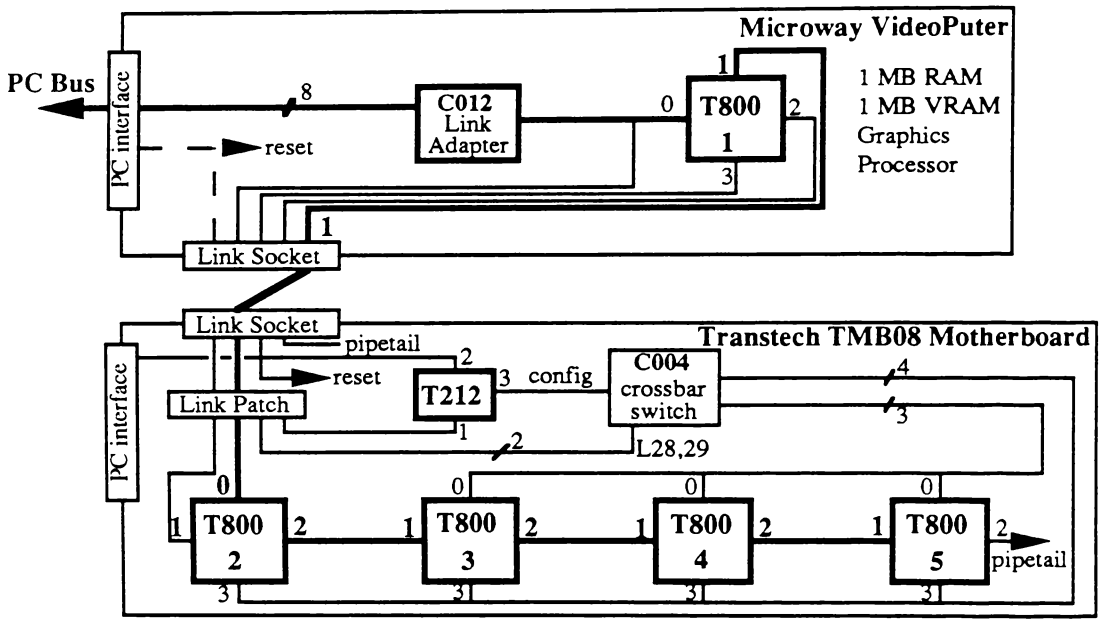

Figure 4.3. Transputer network configuration of PC-hosted system. A Microway Videoputer is connected to the host processor via a C012 link adapter, and transputer modules installed on a Transtech TMB–08 motherboard are connected in a pipeline originating from the Videoputer. A C004 link multiplexer and T212 processor located on the TMB–08 allow reconfiguration of the network.

4.4.1.3. The Sun-hosted System

A transputer network attached to a Sun workstation has been used in more recent development of the superposition algorithm, particularly in conjunction with the GRATIS Treatment Planning System discussed in Chapter 3. The transputer modules used in this system employ 25 MHz versions of the T800, each with 2 Mbytes of local memory.

The complete Sun-hosted system is illustrated schematically in Figure 4.4. The transputer network is interfaced to a Sun SPARCstation IPC workstation by means of an Sbus adaptor based on the Dawn VME Products Sbus development platform (DPS–1). Using a DMA (direct memory access) controller and Inmos C012 link interface, this card supports data transfer rates of 750 kbytes/sec (Sbus to transputer link) and 600 kbytes/sec (transputer

link to Sbus). These transfer rates are approximately half those possible *between* transputer links, and many times higher than the PC to transputer link rate mentioned in the previous section. A UNIX device driver installed on the SPARCstation is used to communicate with the Sbus card by using Posix-standard `open()`, `read()`, `write()`, `ioctl()`, `select()`, and `close()` system calls.

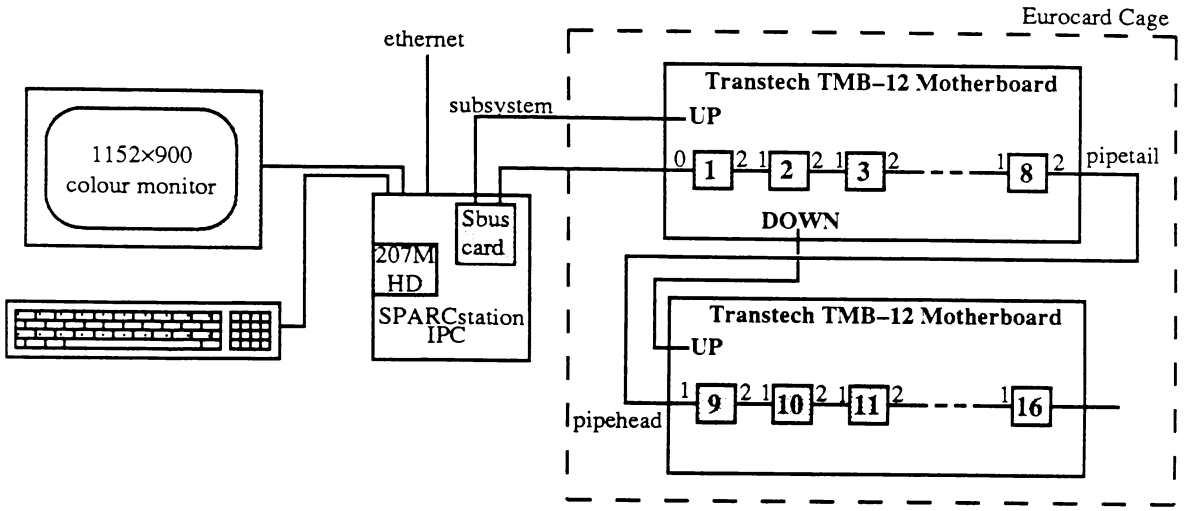

Figure 4.4. Hardware configuration of Sun-hosted system. An Inmos B012-compatible motherboard containing eight 25 MHz transputers is connected to an Sbus card mounted in a Sun SPARCstation IPC workstation. A UNIX device driver is used to communicate with the DMA controller and C012 link interface on the card. A second B012-compatible board can be attached to this system by using the connections illustrated. Not shown are the Inmos C004 crossbar switches and associated T212 processors, which can be used for network reconfiguration.

The transputer modules themselves are mounted on two Transtech TMB–12 motherboards (Transtech Devices Ltd, Buckinghamshire, England). These Inmos B012[23] compatible Eurocard TRAM motherboards are mounted in a 6 unit, 220 mm deep DIN41494 card cage, and each has slots for up to 16 TRAM modules. On each board two C004 link multiplexers are available for network reconfiguration, although these have not been used in any of the applications discussed in this chapter. Normally eight TRAMs connected in a pipeline configuration are mounted on each board, with the two boards daisy-chained together. This arrangement allows transputers to be easily relocated for use in other transputer-based projects. A separate 5V supply built at Waikato University is used to power the transputer boards.

The transputers are arranged in a pipeline by connecting the pipetail of the first motherboard to the pipehead of the second, and the `DOWN` link of the first transputer board must also be connected to the `UP` link of the second, so as to propagate the control (subsystem) signals.

Figure 4.5 illustrates the complete Sun-hosted radiotherapy system, including the Sun IPC workstation (left), 5V power supply (rear), and card cage and installed transputer motherboards (right). Figure 4.6 presents a more detailed view of the TRAM modules installed on a B012-compatible motherboard.

4.4.2. Task Placement and Communication

4.4.2.1. Overview

The transputer-based parallel processing systems used at Waikato University comprise a relatively small number of loosely coupled processors, where each processor has its own memory and own set of instructions (program). Such a machine is known as a *multicomputer*, employing a MIMD (multiple instruction stream, multiple data stream) approach (see Section 4.2.5).

Transputers have no memory management capability. However, it is possible to place several simultaneously executing processes, known as *tasks*, on a single node. Tasks have their own memory areas and communicate via internal serial channels. Channels may also exist between tasks on adjacent processors, in which case they must correspond to a physical link between those processors. There are two levels of task priority — *urgent* tasks, which continue to execute until blocked by an input-output request or some other descheduling event; and *non-urgent* tasks, which are time-sliced with other non-urgent tasks when no urgent tasks are able to execute.

There are many development environments and compilers available for transputers. The applications discussed in this chapter have been developed using a C compiler with parallel extensions, developed by 3L Ltd (Livingston, Scotland).[24] Parallel-C uses message passing as the means of communication and synchronisation, based on the communicating sequential processes (CSP) paradigm.[25] This approach is implemented using procedure calls such as `chan_out_word()` (which sends four bytes of data through a specified channel) and `chan_in_word()` (which waits for four bytes of input from a specified channel). Similar procedure calls exist for dealing with bytes and messages (sequences of bytes).

Each task in Parallel-C may be composed of one or more simultaneously executing (time-sliced) *threads*, which communicate using semaphores. Threads are a kind of "lightweight" task, useful for performing concurrent operations which communicate within a single processor (such as waiting for input from one of several channels), where the complexity of a separate task is unnecessary.

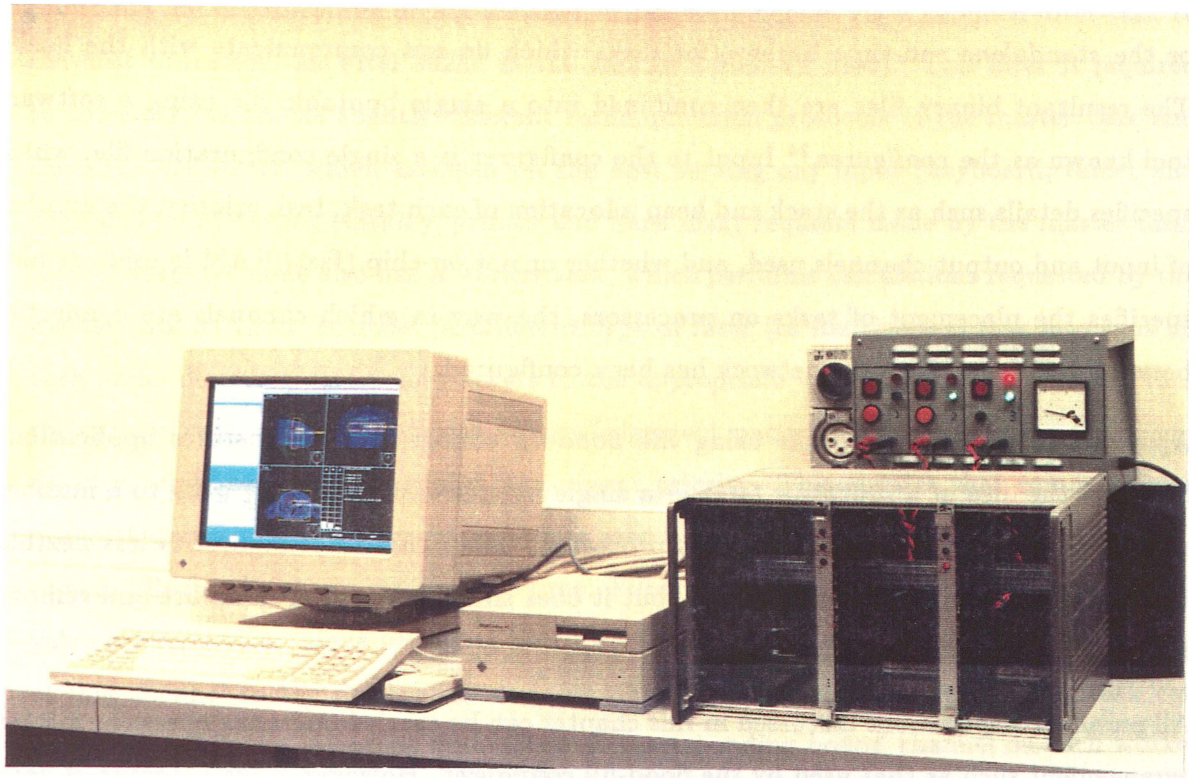

Figure 4.5. Sun SPARCstation IPC with attached transputer network.

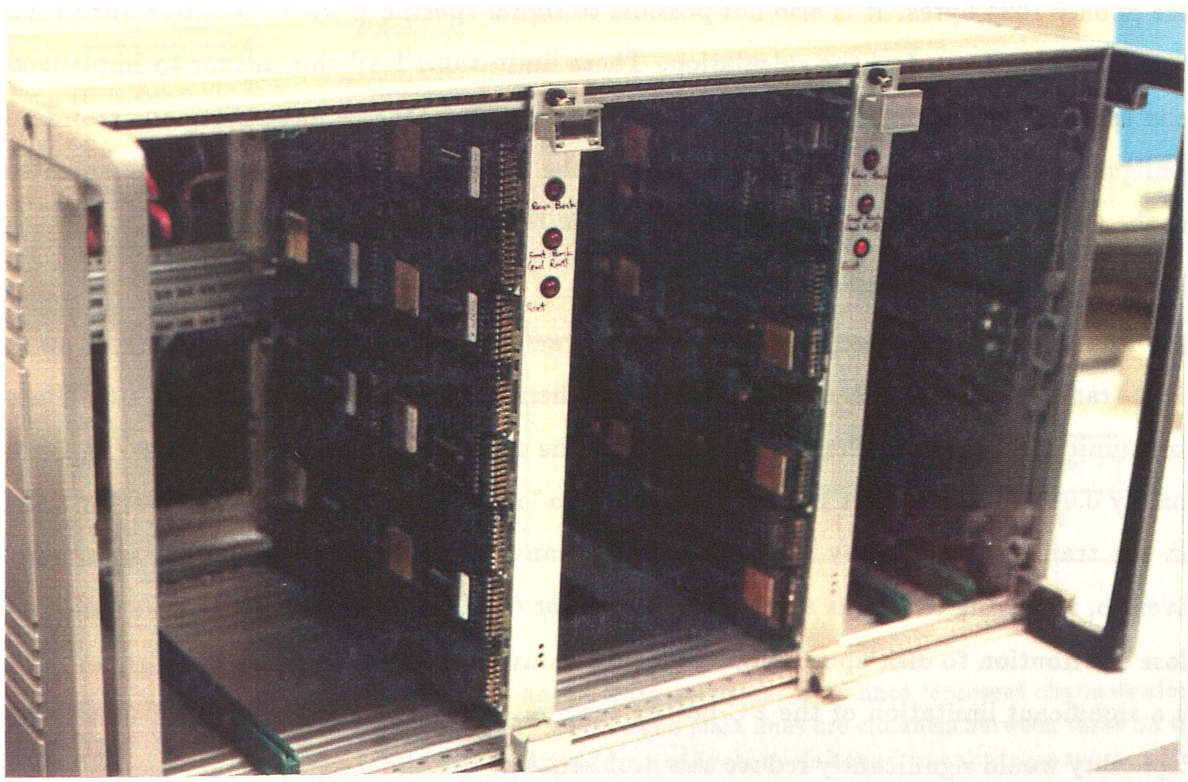

Figure 4.6. Transputer TRAM modules installed on a Transtech TMB–12 motherboard.

Binary code for each task is formed by compiling and linking one or more user-written modules with either the run-time library (for tasks which communicate with the host) or the *standalone* run-time library (for tasks which do not communicate with the host). The resultant binary files are then combined into a single bootable file using a software tool known as the *configurer*.[24] Input to the configurer is a single configuration file, which specifies details such as the stack and heap allocation of each task, task priority, the number of input and output channels used, and whether or not on-chip (fast) RAM is used. It also specifies the placement of tasks on processors, the way in which channels are connected between tasks, and how the network has been configured.

Applications may also be built using the *flood-fill configurer*. It generates applications based on the idea of a *processor farm* — a single *master* task allocating work to a number of *worker* tasks, executing one on each processor. The flood-fill configurer is less flexible than manually controlled configuration, but it does simplify the task structure and remove the need for user-written filter tasks.

All parallel computations discussed in this chapter can be carried out using a master-worker arrangement such as that used by the flood-fill configurer. However, large amounts of data need to be passed at the beginning and end of these computations — a process which is not well suited to the flood-fill configurer, which has a maximum communication packet size of only 1024 bytes. It is also not possible to signal specific worker tasks to return their dose arrays at the end of the calculation. These limitations lead this author to implement master-worker networks by means of specially designed protocols built using the standard configurer. The following two sections outline the details of these protocols, implemented on the two different transputer platforms used in the radiotherapy project.

4.4.2.2. The PC-hosted System

The first transputer network to be used by this author was a PC-hosted network of up to eight transputers, which has been illustrated schematically in Figures 4.2 and 4.3. Since communication between the host system and the root transputer is very slow (approximately 0.02 Mbytes/sec), it has been necessary to locate both the master and worker tasks on the transputers, thereby reducing host to transputer communication to a minimum. Even so, retrieving the input arrays from disk prior to calculation and writing the resultant dose distribution to disk upon completion take many seconds for large array sizes, and this is a significant limitation of the PC-hosted system. Using the Transtech TMB-08's DMA capability would significantly reduce this problem, but the Videoputer could then no longer be the root processor, making display utilities significantly more complex to program.

Figure 4.7 illustrates the placement of tasks on a network of three transputer nodes. The root node has a controlling or *master* task which is connected via a supplied filter task to the host processor (an Intel 80286 in the case of a PC–AT host). This filter is required to correctly match the slightly different communication protocols of the master task and the *alien file server*, which executes on the host serving any input (keyboard, tablet and hard disk) and output (display, printer and hard disk) requests made by the master task. Each transputer node also has a *worker* task, which performs calculations requested by the master task on the root node. A different type of *filter* (or multiplexer) task also runs on each node, directing command packets to the appropriate worker tasks and ensuring that the master task correctly receives acknowledgement packets. Filter tasks run at urgent priority, while worker tasks run at non-urgent priority. This ensures that command and acknowledgement packets are passed on as soon as they are received, so that worker tasks remain idle for the minimum possible time. Filter tasks each have an input channel whose value is bound to a unique processor number by the configurer, and by comparing the value of this channel to the destination node number of an incoming packet, a filter task can correctly pass on or intercept the packet. A corresponding bound channel on each worker task enables workers to place their unique processor number on acknowledgement packets which they originate.

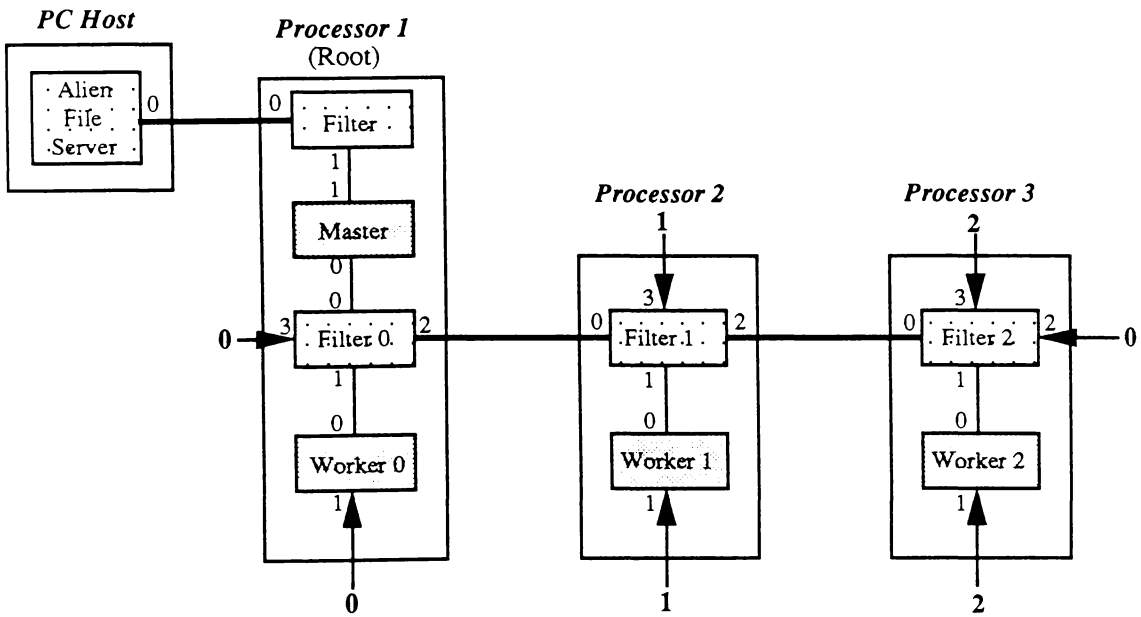

Figure 4.7. Task placements on a PC-hosted linear network. Bold lines represent channels along wires between tasks on two different processors, while plain lines are channels between tasks on the same processor. Single-ended arrows indicate input-only channels bound to an integer representing the identity of the processor (enabling message packets to be correctly routed across the network).

Figure 4.8 illustrates task placements and channel connections for a three-node network of transputers arranged in a *tree* configuration. Note that for the linear array in Figure 4.7 the nodes are numbered consecutively away from the root node, whereas for the tree network in Figure 4.8 the nodes are numbered such that the "left" child node has a smaller number than the "parent," and the "right" child node a larger number (an in-ordered tree). By comparing its own node number against that contained in a communication packet header, a node in the tree network can immediately determine which "branch" of the tree that the packet should be transmitted down in order to reach the destination node.

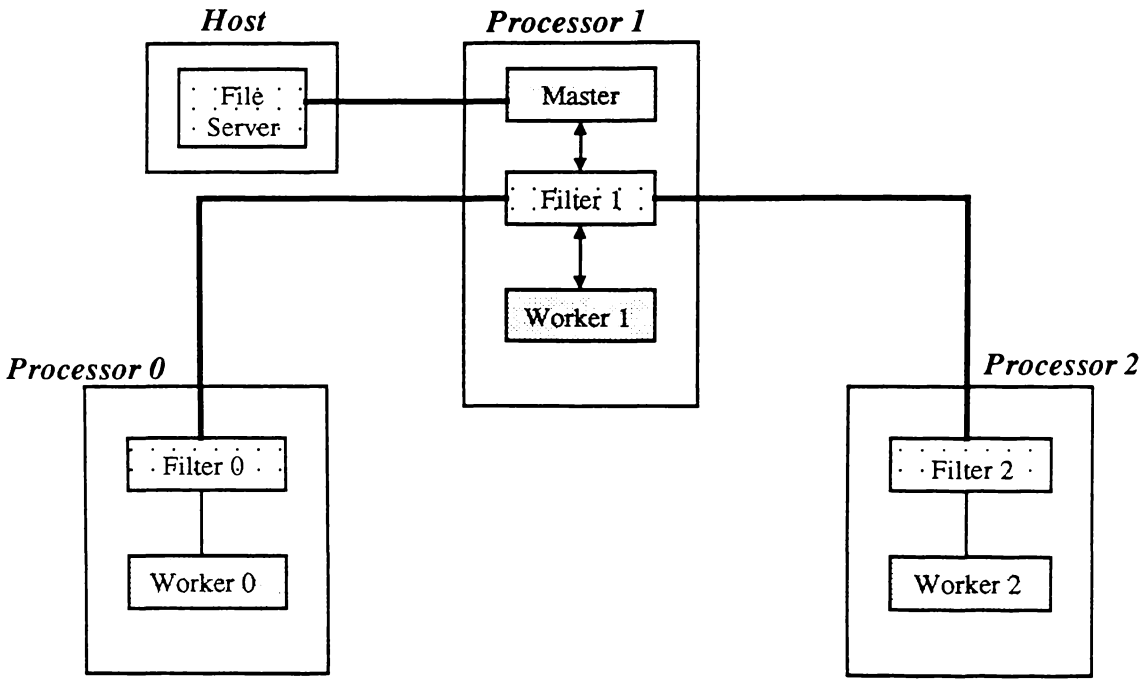

Figure 4.8. Task placements on a PC-hosted tree network. As in Figure 4.7, bold lines represent channels along wires between tasks on two different processors, while plain lines are channels between tasks on the same processor. Tasks are numbered such that all tasks in the left sub-tree have a smaller task number, while those in the right sub-tree have a greater task number. This simplifies the message routing process.

Packets transmitted throughout the network conform to a specific format. The first byte in each communication packet defines the packet *type*. Supported types are `SEND_WORD_TO_WORKER_CMD` and `SEND_WORD_TO_MASTER_CMD`, for transmitting four byte quantities; `SEND_BLOCK_TO_WORKER_CMD` and `SEND_BLOCK_TO_MASTER_CMD`, for transmitting arrays and structures; `SEND_ACK_TO_MASTER`, for sending acknowledgement packets to the master task; and `BOUNCE_CMD`, which is used communicate directly between the sending and receiving threads of a master task. The second byte of each packet contains a node number, which may be the destination node (for packets originated by the master task) or the source node

(for packets originated by a worker task). The next four bytes contain either the value of the data word (for the `WORD` and `BOUNCE` packet types), or an integer specifying the number of bytes in the following block of data (for the `BLOCK` packet types). A packet may also contain one other piece of information — if the most significant bit of the command byte is set then the input and output channels are *locked* until the next packet is received down the same input channel. Transmitting a sequence of locked packets allows large quantities of data to be transmitted without interference from other tasks, although care must be taken to ensure that high priority communications are not locked out for long periods. In general, this locking feature is used only for transmitting large arrays, either prior to the calculation beginning or at the end of the calculation.

Figure 4.9 illustrates the communication protocol in more detail.

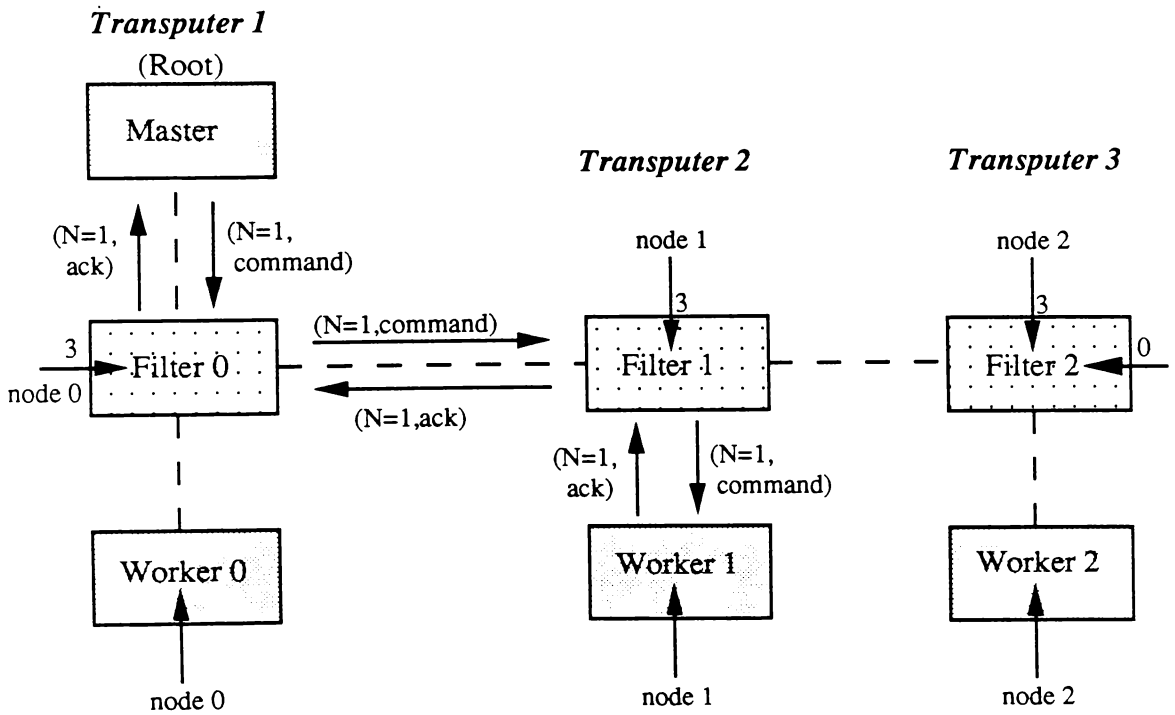

Figure 4.9. Communication between tasks on a PC-hosted network. The master task sends a command packet into the network containing the destination worker node number (N). Each filter task passes the packet on until N matches that bound to input channel 3 of the filter, in which case it is passed down to the worker task. The worker task then unpacks the command packet, performs the requested calculations and sends an acknowledgement packet (containing the worker node number) back to the master task. The master task maintains an array of flags indicating whether each worker is available or busy.

In a typical command/acknowledge sequence, the master task sends a command packet out into the network, containing instructions for a particular worker task to perform a particular calculation. Each filter task examines the contents of the command packet,

passing it on to the next filter task unless the packet's destination node number matches its own node number, in which case the filter task re-directs the packet to its worker. Although the code for each worker task is identical, the input from channel 3 of each filter task can be interrogated to determine the node upon which the worker task is executing. This is done by binding the channel input to the node number before program execution, as discussed previously. When the relevant worker task receives the command packet, it performs the calculation and returns an acknowledgement packet to the master task. Once again, the worker task places its own node identification number (obtained by interrogating input channel 1) into the packet, so that the master task can determine which worker task originated the acknowledgement. A new command packet can then be sent to a worker task as soon as it becomes free.

It is quite possible for two or more pairs of tasks to be attempting to communicate simultaneously, and in general the ordering of these communications is unknown. Hence each filter task must listen simultaneously on three channels (or four in the case of a tree network) — one connected to each of the neighbouring filter tasks and another to the local worker process. This is done by creating one execution thread for each channel. When any one of the threads detects incoming traffic, it waits to obtain control of appropriate semaphores to prevent other threads using the busy channels. In this way a filter process may permit packets to travel simultaneously from master to worker and worker to master, but not from worker to master and filter to master (see Figure 4.10). This prevents collision of incoming acknowledgement packets from two different worker tasks.

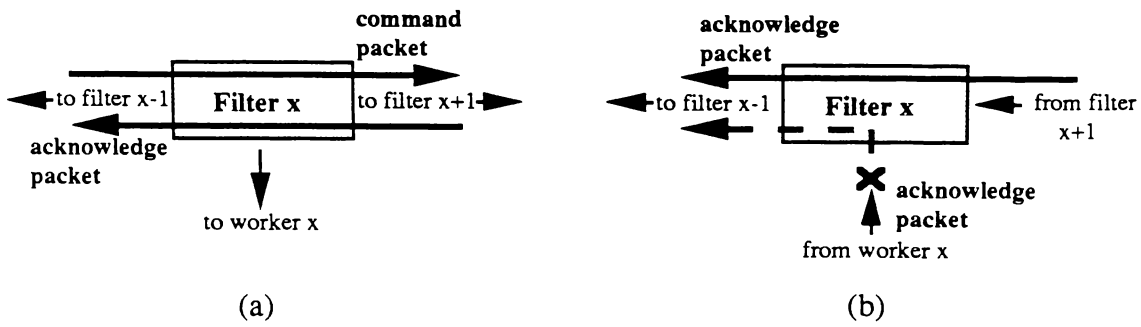

Figure 4.10. Collision prevention on a PC-hosted linear network. Each filter listens simultaneously on three input channels, maintaining semaphores indicating which input and output channels are being used. These semaphores allow simultaneous transmission of packets in opposite directions (a), thereby preventing communication deadlock. However, packets from two distinct worker tasks cannot be simultaneously routed through the same filter task (b). The semaphores are used to temporarily halt transmission from one source.

4.4.2.3. The Sun-hosted System

At a later stage of the radiotherapy project, the GRATIS Treatment Planning System discussed in Chapter 3 was installed on a Sun SPARCstation network, and the superposition code developed at Waikato University was ported across to that platform. Therefore the transputer-based parallel processing system also had to be transferred to the Sun host. A transputer network running under the *Helios* operating system was already running on the Sun, and the 3L Parallel-C environment was also ported to the Sun network with only minor modification (the alien file server had to be modified to conform to UNIX rather than MS-DOS file name and program parameter conventions).

Task placement and communication on the Sun-hosted system could be made essentially identical to that described in the previous section for the PC-hosted system. However, the much faster host to transputer interface in the Sbus adaptor, and the more powerful processor in the host itself, offer another option — a master task based on the host processor (Sun) rather than on the root transputer. This option has two significant advantages. The parallel part of the Sun-hosted superposition code can be incorporated simply as a subroutine of GRATIS in the usual way, and the large arrays required by the worker tasks can be transferred directly, rather than via a transputer-based master task as with the PC-hosted system. Since there is no master task on the root transputer, no additional external memory is needed on the root, and therefore the network can be entirely homogeneous.

This alternative task arrangement used on the Sun-hosted system is illustrated in Figure 4.11. Note that the master task runs on the host processor as a subroutine of the GRATIS code itself, and communicates with the root processor filter using a UNIX device driver (written at Waikato University). Although the tasks are compiled, linked and configured in the same manner as described in the previous section, the resulting bootable code is sent to the transputers simply by pushing it down a link, rather than by invoking the alien file server. The transputer network appears to the master task as a UNIX device, and to the root processor filter the master appears as just another transputer task connected to channel 0. Filters and workers are numbered consecutively outward from the root transputer, but do not have ports bound to the processor number, as is the case on the PC-hosted system. Only filter channel 2 of the last processor is bound to a constant value (zero), to indicate to the filter task that it is at the end of the pipeline. Due to the efficiency of this linear array for radiotherapy applications, a tree network has not been used on the Sun-hosted system.

The configuration file used as input to the 3L configurer is generated automatically, using

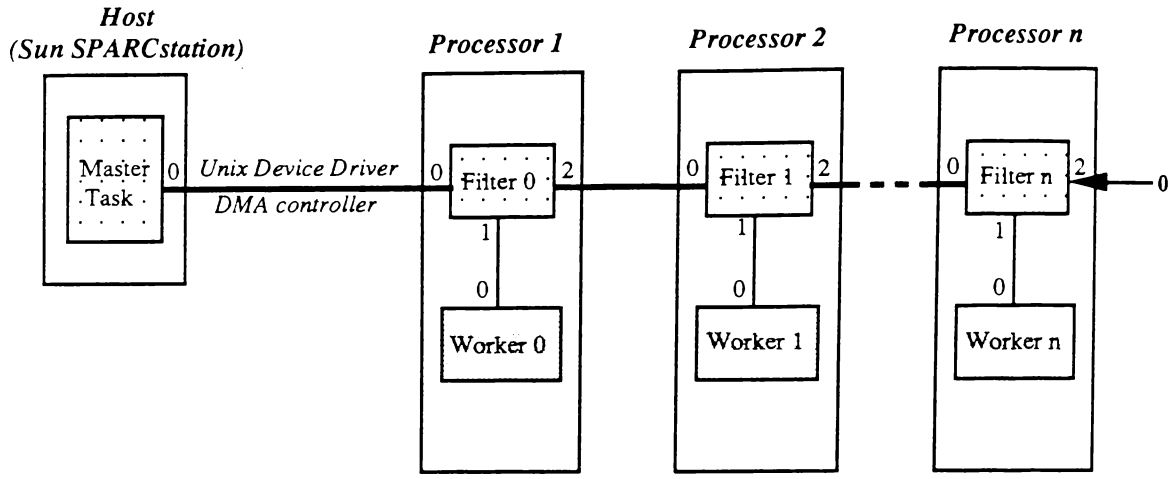

Figure 4.11. Task placements on a Sun-hosted network. Bold lines represent channels along wires between tasks on two different processors, while plain lines are channels between tasks on the same processor.

a shell script which takes a parameter indicating the number of processors in the network. For example, to generate a configuration file for the superposition algorithm (discussed in Section 4.4.3.1) on a linear network of four processors, the command

```
s_config 4 >s_trans_calc.cfg
```

would be entered at the command prompt, where **s_config** is the name of the script and **s_trans_calc.cfg** is the name of the configuration file to be generated.

Appendix J lists the **s_photon** script and also the resultant configuration file when invoked for four processors. Note from the output file listing that the filter tasks are run at urgent priority to reduce the wait time for worker tasks to a minimum, and that the data space required by the filter tasks is 20 kbytes. This value must be changed according to the value of the constant **MAX_MESSAGE_SIZE**. The worker tasks each have 2 kbytes of stack space and the rest of the available memory as heap. The **Opt=Stack** directive causes the stack to be placed in on-chip (fast) RAM, with **Opt=Code** placing the beginning of the worker task code in the remaining 2 kbytes of on-chip RAM.

The communication protocol used in the Sun-hosted system is somewhat different to that described for the PC-hosted system. There are five supported packet types, illustrated in Figure 4.12. The first byte in each packet is the packet type as before, except that the types are now **WORD**, for transmitting four byte quantities to or from the master task; **MATRIX**, for transmitting arrays and structures to or from the master task; and **VECTOR**, for transmitting vectors of voxels from the master task to a worker task. There are also two other communication packet types: **WORD_TO_ALL**, for transmitting a four byte quantity

from the master task to *all* worker tasks; and **MATRIX_TO_ALL**, for transmitting arrays or structures to all worker tasks. Filter tasks receiving these last two packet types pass the packet on to *both* the local worker task and the "downstream" filter task. This greatly simplifies and accelerates the distribution of arrays to worker tasks prior to calculation beginning, since a single copy of each array is propagated to all worker tasks.

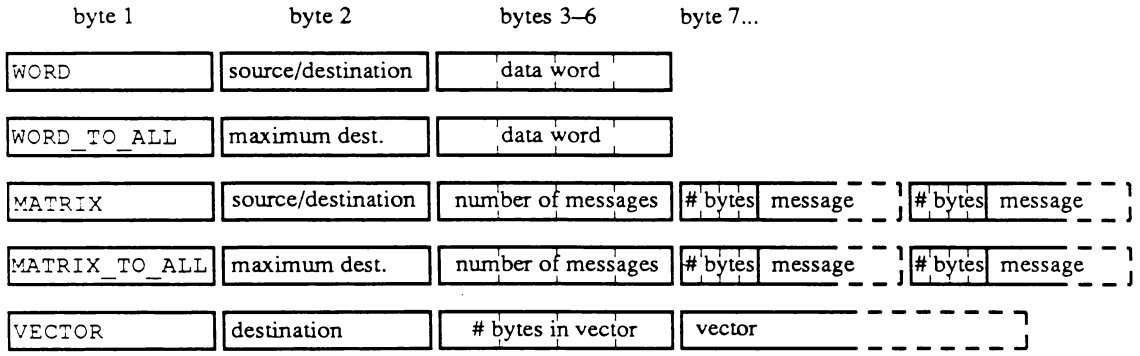

Figure 4.12. Communication packet types on a Sun-hosted network.

The second byte in each packet is the source or destination node number, which is used to route the packet using the method illustrated in Figure 4.13. If the packet is being sent from the master task to node N, then the master task originates the packet with the (destination) node number set to N. The filter on the root processor will then receive the packet and check the destination node number to see if it is zero, and if so pass the packet to its local worker task. If the destination node number is not zero then the filter passes the packet on to the downstream filter, but *decrements* the destination node number before doing so. In this way the packet is passed down the pipeline until the destination node number becomes zero, which will occur at filter N. A similar process occurs when handling acknowledgement packets originated by the worker task, except that the (source) node number is *incremented* as it is passed up the pipeline. Is this way the master can examine the source node number to determine which worker originated the packet, and mark that worker as free. This approach to packet addressing is conceptually simpler and more convenient than that used in the PC-hosted system, since there is no longer any need for channels bound to identifying node numbers.

For packets of type **WORD** and **WORD_TO_ALL**, the next (and final) four bytes of the packet contain the word to be transmitted. For packets of type **VECTOR**, the next four bytes are an integer representing the vector size (in bytes), and following those is the communication vector itself. For packet types **MATRIX** and **MATRIX_TO_ALL**, the next four bytes contain an integer representing the *number of messages* which follow, where each message is part of

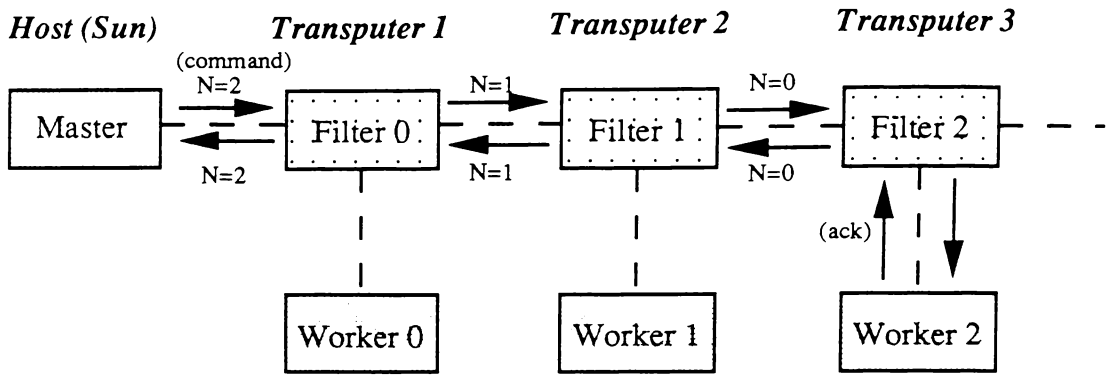

Figure 4.13. Communication between tasks on a Sun-hosted network. The master task sends a packet into the network containing the destination node number N (in this example $N = 2$). Each filter task decrements N and passes the packet downstream, unless $N = 0$, in which case the packet is passed to the worker task. When the requested calculations have been performed the worker task originates an acknowledgement packet with a source node number of zero. The packet is passed up the pipeline, with the source node number incremented at each step, until it reaches the master task.

the array to be transferred. The maximum size of each message is defined by the constant `MAX_MESSAGE_SIZE`. Following that is a sequence of messages, each message preceded by an integer indicating the message length. Note that since an entire array is transmitted in a single packet, there is no need for explicit locking of the communication channels.

Although they differ in details of implementation, the Sun-hosted and PC-hosted systems discussed in this and the previous section are essentially the same kind of parallel processor: local memory multicomputers which communicate by synchronous message passing. The remainder of this chapter presents three radiotherapy applications which have been implemented in parallel using these systems.

4.4.3. Parallel Algorithms

4.4.3.1. Superposition

Over recent years much effort has been put into the development of convolution-based algorithms for radiotherapy dose calculation.[26–28] However, performing convolution in real space with a density-scaled energy deposition kernel (often termed *superposition*) is a computationally intensive task. The Fourier transform approach has been shown to speed up execution relative to the superposition technique[29] (as is seen in Section 4.4.3.2), but at the expense of requiring an invariant kernel. Although correction techniques have been suggested,[30] the Fourier transform method is not completely satisfactory in heterogeneous media, and therefore a density scaled superposition method based on that proposed by Mackie *et al.*[26] has been selected as the method of choice at Waikato University.[31]

Parallel processing has already been used to *streamline* the treatment planning process,[32] and one group has suggested the use of a shared memory multiprocessor as a radiotherapy workstation.[33] The effect of partitioning and granularity on performance is dependent upon the exact nature of both the multi-processor system and the algorithm being implemented,[34] but the aim of this section is to demonstrate the performance characteristics of a *particular* computationally-intensive dose calculation algorithm, superposition, implemented on a transputer-based multicomputer. The applicability of the multicomputer approach to superposition calculation will be determined by answering the following questions:

- Is speedup linear with increasing processor number?
- Is communication overhead dependent upon communication packet (grain) size?
- Does altering network topology affect communication overhead?

The superposition algorithm used here has been discussed in Chapter 3, and also by Murray *et al.*,[35] Metcalfe *et al.*,[36] and Hoban *et al.*[31] The dose $D(i,j,k)$ to a voxel of density $\rho(i,j,k)$ is presented in Equation 3.7, and is restated here:

$$D(i,j,k) \;=\; \frac{1}{\rho(i,j,k)} \sum_{i'} \sum_{j'} \sum_{k'} \rho(i',j',k') \cdot T(i',j',k') \cdot H(i-i', j-j', k-k') \quad , \quad (4.2)$$

where $T(i',j',k')$ is the *terma* (total energy released per unit mass) at the interaction site and $H(i-i', j-j', k-k')$ is the value of the *energy deposition kernel* at the deposition site. The summation is performed over all voxels in the medium which contribute scattered dose to (i,j,k). Noting that in general $H(i-i', j-j', k-k')$ is dependent upon position within and composition of the medium, and that dose must be calculated at all points in the medium, it is apparent that superposition is computationally intensive. It also becomes rapidly more intensive as the array sizes increase — for a terma array of side n the computation required increases approximately as n^6 when no kernel scaling is used, or as n^7 when full kernel scaling is used. Hence the superposition approach is very much in need of the kind of performance improvements which parallel processing is capable of delivering.

The superposition algorithm has been implemented on a multicomputer by means of a master-worker network, described in Section 4.4.2. A single master task executes on either the root transputer (in the PC-hosted system) or on the host itself (in the Sun-hosted system). This master task has direct access to input/output facilities such as hard disk storage and terminal I/O. Several worker tasks execute throughout the network, communicating with the master task by passing messages or *packets* along communication channels linking the tasks. Workers receive *command* packets from the master task and issue *ready* packets

when they have completed the requested work. Routing or *filter* tasks are also present on each node in the network, either to pass the communication packets along the network, or to pass them to and from the worker task executing on their node. This ensures that the master task perceives the network as a number of directly reachable worker tasks, and makes details of the network topology transparent to the master and worker tasks.

A worker task performing a superposition for a single point (i', j', k') must have access to the entire energy deposition kernel (or several kernels when using density scaling), and must also have access to the patient density array, since voxel composition must be taken into account when scaling the energy deposition kernel. The energy deposition kernel(s) and density array are therefore distributed to all worker processes prior to calculation beginning.

Figure 4.14 illustrates schematically the approach used to implement superposition in parallel. After the density array and kernels have been sent to the worker tasks (not shown in Figure 4.14), the master task steps through the terma array in order of increasing (i', j', k'), processing any voxels whose terma exceeds a threshold value. The (i', j', k') voxel coordinates and corresponding termas are appended to a one-dimensional array or *command vector*. When this vector reaches a predetermined size the master task waits until a worker task becomes available, then sends the vector to that task. The worker task then performs the superposition for the interaction voxels contained in the vector, storing the calculated dose in its own local dose array. The worker sends a *ready* packet to the master task to indicate the calculations are complete, and awaits a further command vector. This process continues until all voxels have been processed and all corresponding ready packets have been received by the master task. The master task then prompts all worker tasks to return their copies of the dose array, summing them to form the total dose distribution.

The corresponding pseudocode description of the superposition code is presented in Figure 4.15. The strategy adopted is that of *synchronous message passing*, where tasks are forced to wait until a sent message is received by some other task. Similarly, execution will not continue beyond a receive statement until a message is available to be received. Messages sent from the master task can be routed directly to the destination worker task, since the identity of the worker task is known. However, the master task must be able to receive a ready signal from the *first available* worker task, whose identity cannot be predicted. This is indicated in the pseudocode using the concept of a *mailbox*, to which all worker tasks send their ready packets. Many–to–one communication of this type can also be represented by the *port* notation, as discussed by Andrews and Schneider.[10]

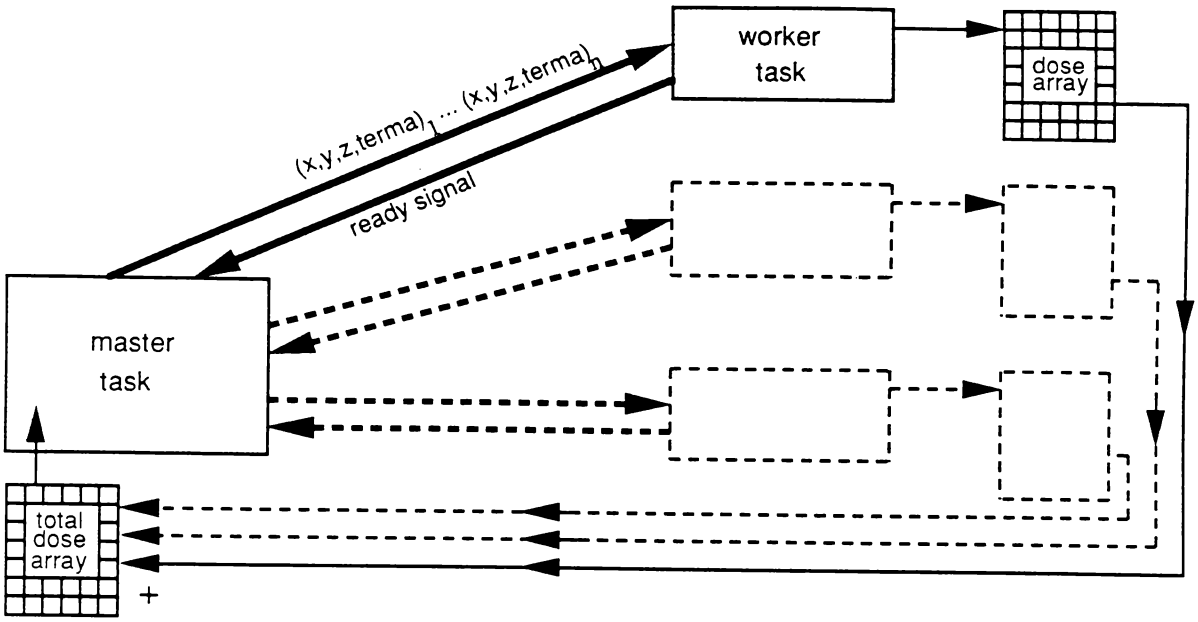

Figure 4.14. Implementation of superposition on a multicomputer network. The master task sends a complete copy of the patient density and energy deposition kernel(s) to each worker task (not shown). It then issues command packets containing interaction voxel coordinates and terma to each worker task as it becomes free. The dose calculated is deposited in a copy of the dose array kept locally by each worker task. Finally, the dose arrays are returned to the master task and summed together to yield the total dose distribution.

Parallel superposition code has been installed on both the PC-hosted and Sun-hosted multicomputer systems, although there are significant differences in the way the algorithm has been implemented on these two systems. The remainder of this section highlights these differences, and presents results which have been obtained with each system for typically-sized superposition calculations.

The PC-hosted System

Superposition on the PC-hosted system has been implemented using a master-worker network, with the master located on the root transputer. The communication protocol used has been described in Section ~~in Section~~ 4.4.2.2. The performance of this system will now be illustrated with a superposition calculation involving the following arrays:

- A $16 \times 16 \times 60$ voxel *terma* array, stored on the root node.
- A $16 \times 16 \times 60$ voxel *dose* array, stored on the root node and for each worker task.
- A $16 \times 16 \times 60$ voxel *electron density* array, stored for each worker task.
- An $8 \times 8 \times 68$ voxel array (representing *one quadrant* of a $16 \times 16 \times 68$ *energy deposition kernel*), stored for each worker task. When scaling for density using the method adapted from Mackie *et al*,[26] several kernels must be stored by each worker.

```
type vector_element =  record
                          i,j,k : integer;
                          terma : real;
                       end;
constant MAX_VECTOR_SIZE, WORKERS;
var inward_messages : mailbox;

process master;
   var   n : integer;
         vector : array[0..MAX_VECTOR_SIZE-1] of vector_element;
   send electron density grid to workers;
   send energy deposition kernel(s) to workers;
   while (still voxels left to process)
      build command vector;
      if (all workers have been allocated vectors)
         receive ready signal from inward_messages;
         determine source node number n of ready signal
      else
         determine node number n of next free worker
      end if
      send vector to worker n;
   end while;
   while (some packets have not yet acknowledged)
      receive ready signal from inward_messages
   end while;
   send dose request to workers;
   do n := 0 . . WORKERS - 1
      receive worker dose array from inward_messages;
      master dose array := master dose array + worker dose array
   end do;
   end;

process worker;
   var   i : integer;
   initialise worker dose array;
   receive energy electron density grid from master;
   receive energy deposition kernel(s) from master;
   loop
      receive vector from master;
      if (vector = dose request)
         goto end_label: ;
      do i := 0 . . number of vector elements - 1
         perform superposition using vector[i];
      send ready signal to inward_messages
   end loop;
   end_label:
      send worker dose array to inward_messages
   end;
```

Figure 4.15. Pseudocode description of the parallel superposition algorithm.

Note that only one quarter of the energy deposition kernel need be stored, since the other three quadrants can be generated by symmetry. Cartesian energy deposition kernels are pre-calculated using the EGS4 Monte Carlo[37] user code `RTPCART`, discussed in Chapter 2, Section 2.6.3. Terma is calculated from radiological depth using an adaption of the ray tracing method described by Siddon.[38] For the array sizes quoted above the master task takes 10.8 seconds to calculate radiological depth, and another 25.0 seconds to calculate terma (using ten spectral components). Terma calculation could easily be performed in parallel, using a method similar to that described for the superposition process itself. This would

result in a total terma calculation time of approximately 9 seconds (7 seconds calculating, and 2 seconds communicating the arrays) using an 8-element transputer network.

Assuming that all array elements are represented in single-precision floating point format, each using four bytes of memory, then the total requirement for the electron density array, dose array and seven kernels (for density scaling) is 132 kbytes. Prior to the calculation beginning, these arrays are distributed to each worker task individually (as required by the communication protocol), a process which takes approximately 3 seconds for a linear array of eight transputers. Code for the worker and filter tasks is small, so 2 Mbytes of local memory is adequate for arrays with 15 times as many voxels as stated above (approximately 2.5 times as many voxels in each dimension), assuming that a larger amount of memory is available on the root node to accommodate the master task and associated arrays.

The superposition calculation specified above was performed on three types of computer system — a DEC MicroVAX-II running VMS and using the VAX C language, a DEC VAX 6330 running VMS (not employing the SMP parallel environment) and also using the VAX C language, and linear arrays of 1–8 Inmos T800–20 transputers. Figure 4.16 shows computation times for this problem using no scaling of the energy deposition kernel and communication vectors of size 40 voxels. A density scaling algorithm developed by Hoban *et al.*[31] (see Chapter 3, Section 3.3.4), based upon the use of multiple energy deposition kernels[26] and simple z-axis ray tracing, results in computation times approximately twice those shown in Figure 4.16. Although the exact details of the algorithm have a significant influence on the efficiency of computation, the times presented in Figure 4.16 are (not surprisingly) superior to those reported by Boyer *et al.*[29] for their VAX-based system.

Computation time should ideally decrease in inverse proportion to the number of processors in the network. Figure 4.17 illustrates how total computation *power*, or *speedup*, varies with increasing processor number in a linear network for two sizes of command vector, where speedup S for N processors is calculated according to the relation:

$$S(N) = \frac{\text{computation time for 1 processor}}{\text{computation time for } N \text{ processors}} \times \frac{100}{1} \quad (\%) \ . \tag{4.3}$$

Using command vectors containing only one voxel each, the performance of the network begins to degrade significantly for larger network sizes, until for eight processors 55% of the power of one processor is lost to communication overhead. However, increasing vector size to 40 voxels results in almost linear speedup, with very little communication overhead for linear arrays of up to eight processors. Thus it appears that process *granularity* (the amount of computation performed between synchronisation events) has a significant effect on the

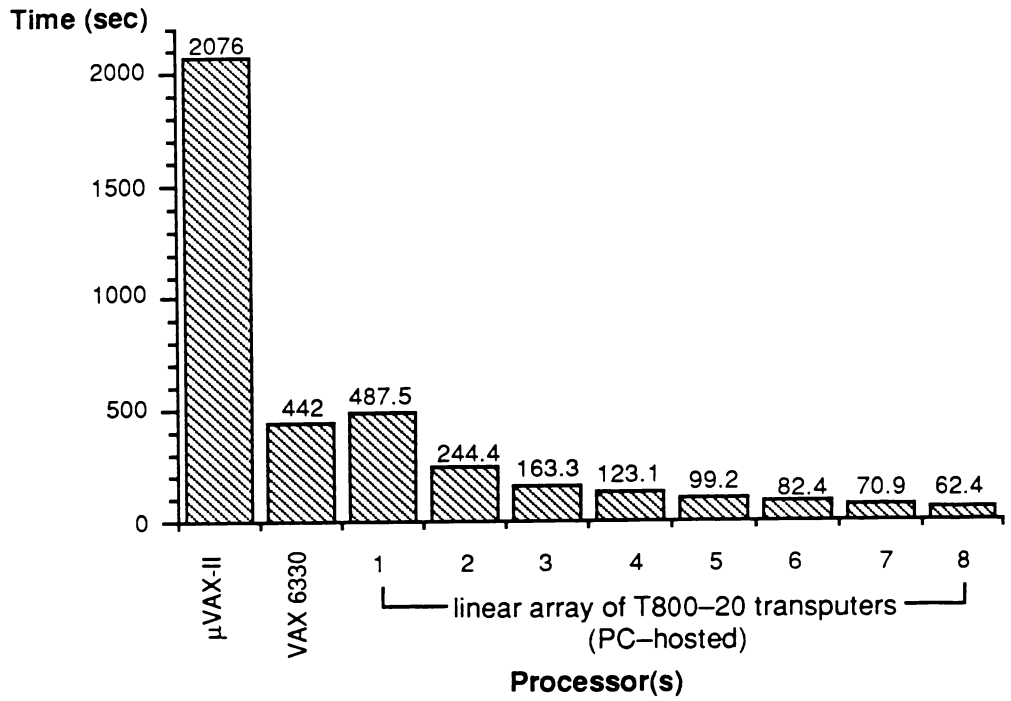

Figure 4.16. Computation times for a typical superposition problem on various processors. Times shown are for superposition of a 16 × 16 × 60 voxel terma array with a 16 × 16 × 68 voxel energy deposition kernel, simulating a 5 cm × 5 cm photon field (voxel size is 0.5 cm on each side). The number of non-zero terma voxels is 9180 (59.7% of the terma array), and the total number of multiplications required is 4.7×10^7. No density scaling of the energy deposition kernel was used in obtaining these results. Communication packet size is 40 voxels.

efficiency of the superposition calculation. For large numbers of processors, performance using 40-voxel vectors might be expected to degrade in a similar way to that observed using on small linear networks when using 1-voxel vectors.

The communication overhead C per processor for a network of N elements and a given vector size can be calculated according to the relation:

$$C(N) = \left(\frac{(\text{computation time for } N \text{ processors}) \times N}{\text{optimal computation time for 1 processor}} - 1 \right) \times \frac{100}{1} \quad (\%) \ , \tag{4.4}$$

where it is assumed that the optimal single-processor algorithm (using a large vector size) has negligible communication overhead. Figure 4.18 demonstrates the communication overhead C for various vector sizes on a linear array of seven processing elements. For small vector sizes the overhead is relatively large, then reduces to a minimum of 1.80% at vector size 40. This improvement is due mainly to two factors:

- Six bytes of "packaging" are required around each command vector, consisting of the command type (one byte), the destination node number (one byte), and an integer specifying the length of the packet (four bytes). These bytes constitute a significant

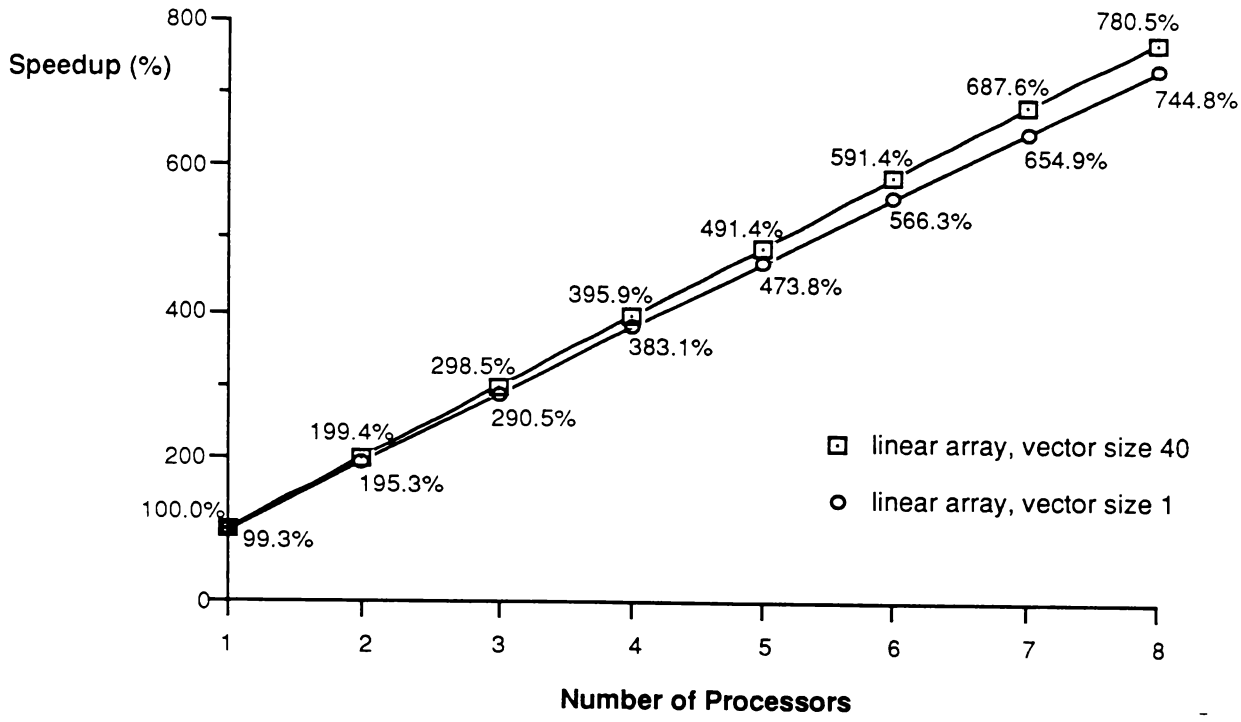

Figure 4.17. Speedup of the superposition algorithm on a PC-hosted linear array of transputers using command vectors of size 1 voxel (lower line) and 40 voxels (upper line). Refer to Equation 4.3.

overhead for small vector sizes, but for large vector sizes the percentage overhead is very much smaller. A large vector size reduces both the time spent communicating and the likelihood of "collisions" between packets travelling throughout the network.

- The processors themselves require a small amount of time to switch between the calculation task (worker) and the communication task (filter). Increasing vector size reduces the number of these context switches.

The increase in overhead apparent for vector sizes larger than 40 voxels occurs due to uneven *load-balancing* between the processors. In the example calculation there are 9290 non-zero terma voxels distributed between seven processors, or 1327 voxels per processor. If the command vector size is 100, then $\frac{100}{1327} = 7.5\%$ of total computation time is spent processing each vector. In the worst possible case this could result in a $0.075 \times 70.9 = 5.3$ second delay in computation (assuming that processing of all other vectors was completed just as the last vector was sent out). However, there is an automatic load-balancing effect built into the calculation itself, because the last voxels processed (near the maximum z-coordinate of the terma array) require less calculation, since much of the energy deposition kernel lies beyond the maximum z-coordinate. Even so, uneven load-balancing does contribute to computation time for command vector sizes over 40 voxels. As the number of processing

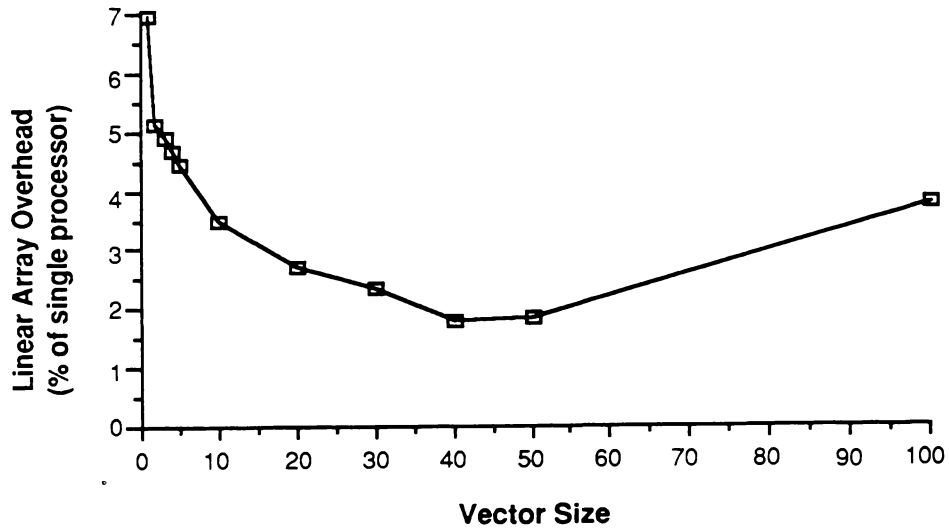

Figure 4.18. Communication overhead per processor as a function of command vector size for a PC-hosted linear array of seven processing elements. Values are calculated assuming that there is zero overhead on a single processor using an optimal command vector size (40 voxels).

elements increases beyond seven, uneven load-balancing would be observed for even smaller command vector sizes, until eventually a more sophisticated approach to load sharing would be required.[39] One possible solution is to reduce the vector size as the superposition nears completion, thereby ensuring all processors finish at nearly the same time. This approach is investigated in conjunction with the Sun-hosted system.

Earlier in this section it was mentioned that one way of reducing communication overhead is alteration of the network topology. To examine this possibility the transputers were reconfigured into a seven-element *binary tree*, with one, two and four elements at each level in the tree (see Figure 4.19). This reduced the maximum number of "hops" from six to two — in general the maximum number of "hops" for a linear array of N elements is $N-1$, and for a binary tree is $\lfloor \log_2 N \rfloor$. Although many different topologies are possible,[1,14,15,17] a tree network is well suited to the one-to-many communications required by master/worker systems, since there is a unique path from the master to each worker task.

Communication overhead for the tree network was recorded as a function of command vector size, and compared with results for the linear array. Figure 4.20 illustrates these results for vector sizes of 1–5 voxels, showing a reduction in overhead from 6.9% (linear array) to 5.6% (tree network) for size-1 vectors. This advantage becomes less pronounced as the vector size increases, until for greater than 5 voxels per vector there is no detectable improvement. For very large networks the ratio $\frac{\log_2 N}{N-1}$ becomes smaller and hence a tree network would become more efficient in reducing communication overhead. Since the specific processors

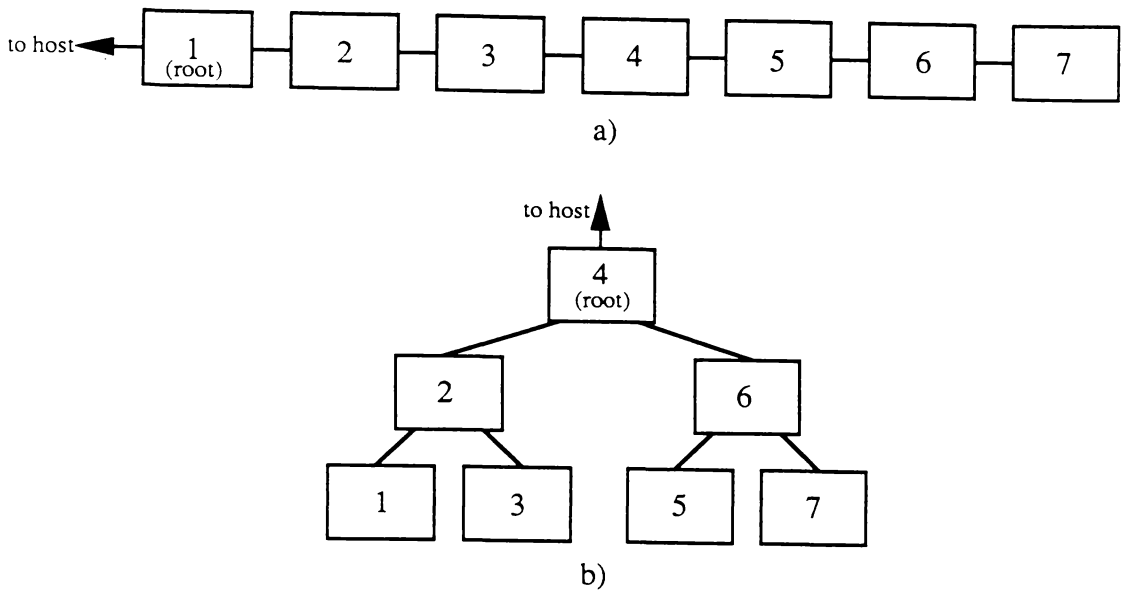

Figure 4.19. Comparison of two network topologies. (a) Linear array (pipeline) configuration. The maximum number of "hops" to a worker task is 6. (b) Tree configuration. The maxiumum number of "hops" to a worker task is 2.

(transputers) used here have four links, it would also be possible to connect them in a ternary tree, reducing the above ratio to $\frac{\log_3 N}{N-1}$.

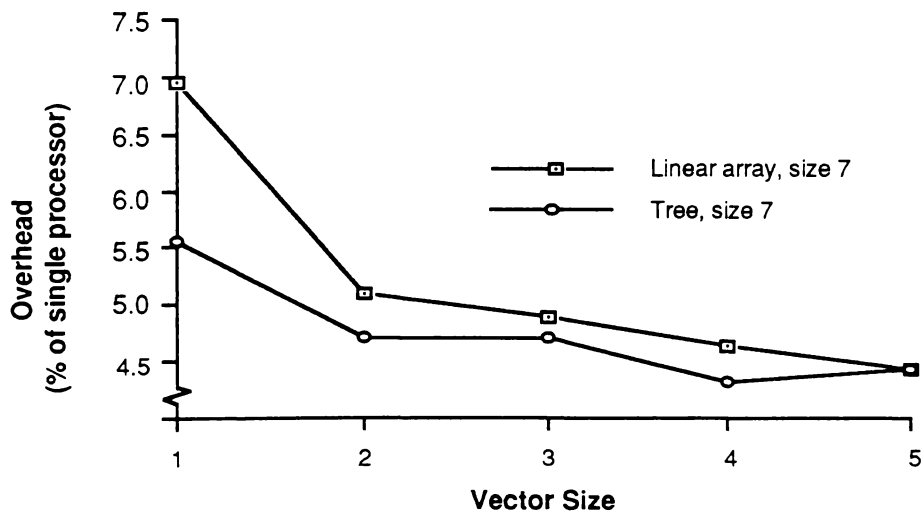

Figure 4.20. Comparison of communication overhead for PC-hosted linear and tree networks of seven processing elements. Values are calculated assuming that there is zero overhead on a single processor using an optimal command vector size (40 voxels). For command vector sizes greater than 5 voxels (not shown) there is no observable difference in communication overhead.

In response to the questions posed at the beginning of this section, results obtained using the PC-hosted transputer network can be summarised as follows:

- Both linear and tree network topologies provide near-linear speedup with increasing processor number. A linear array of 8 processors provides 7.81 times the computing power of a single processor when using an optimal communication vector size.

- Increasing communication packet size from 1 voxel to an optimum of approximately 40 voxels significantly reduces communication overhead per processor. Overhead per processor for a 7-element linear array is 6.9% when using 1-voxel vectors, but only 1.8% when using 40-voxel vectors.

- The topology of the network has some effect on communication overhead. Arranging 7 processors in a 1–2–4 binary tree reduces overhead to 80.1% of that encountered using a 7-element linear array (when using a packet size of 1 voxel).

The Sun-hosted System

When the superposition algorithm was incorporated into the GRATIS Treatment Planning System (see Chapter 3, Sections 3.4 and 3.5), it became necessary to adapt the parallel superposition code to the Sun-hosted transputer network. This system's hardware configuration, task placement and communication protocol have already been described (see Sections 4.4.1 and 4.4.2). This section discusses the software organisation and implementation of the superposition algorithm, then presents results obtained for a typically-sized radiotherapy problem.

The **s_photon** code has four optional parameters used to control parallel execution. These parameters are **-T** *processor_type*, **-N** *processor_number*, **-V** *vector_size*, and **-R** *remote_server_node*, and are relevant only for calculation of dose to a grid (they are ignored when calling **s_photon** in point dose mode). Parallel dose computation is turned on by specifying **transputer** as the *processor_type* — this is done automatically when invoking **s_photon** using the shell script **transbm**. This script also accepts two command-line parameters — the number of processors in the network, and the desired number of voxels in a communication vector. If these parameters are supplied to **transbm** then they are passed directly on to **s_photon** via the **-N** and **-V** program parameters. If they are absent then default values (of **1** and **1**) are supplied to **s_photon**. The final parameter, **-R** *remote_server_node*, specifies the name of a machine on which a remote dose computation server is running.

Figure 4.21 shows the names and locations of source files written to implement the superposition algorithm in parallel. They are additional to those used for the standard (non-

parallel) superposition algorithm, which are shown in Chapter 3, Figure 3.11. The new files are grouped according to their function, in three subdirectories:

- The `c` subdirectory. These files implement the host-based part of the parallel code (the master task), and the client side of the remote dose computation facility.
- The `rpc` subdirectory. These files implement the server side of the remote dose computation facility.
- The `s_parallel` directory. These files implement the transputer-based part of the parallel code (the filter and worker tasks).

The remote dose computation facility, implemented using Sun's RPC (remote procedure call) paradigm,[40] is discussed in Section 4.4.4. The code which implements the master-worker dose computation algorithm will now be explained in detail.

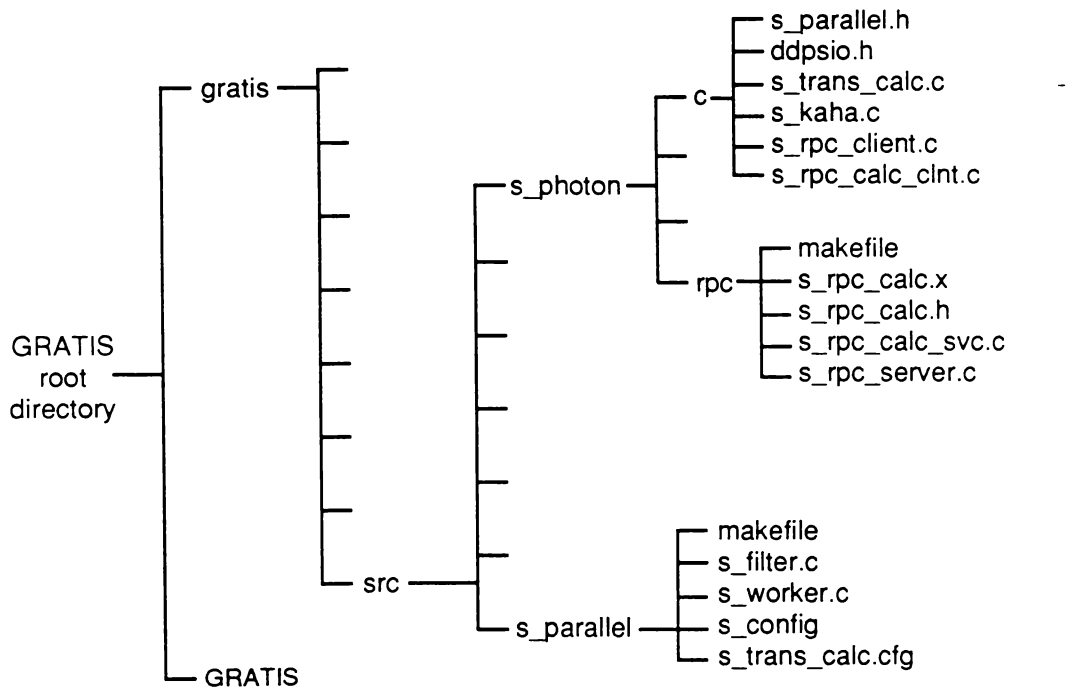

Figure 4.21. Directory structure for parallel superposition code under GRATIS.

As is illustrated in Figure 4.11, the master task executes on the *host* in the Sun-based system. Code for the master task is contained in the module `s_trans_calc.c`, located in the `c` subdirectory — a pseudocode description of `s_trans_calc()` is presented in the first part of Figure 4.22. It begins by initialising the transputer network (using `init_root()` in `kaha.c`), then uses `boot_root()` to boot the network with the filter and worker code generated by the configurer. Immediately upon booting, the master task sends a packet of

type WORD_TO_ALL into the network. Each filter task responds by returning a packet of type WORD, and s_trans_calc() waits to receive all these packets, thereby confirming that the network has been booted correctly. The required parameters and arrays are then sent to the worker tasks using the WORD_TO_ALL and MATRIX_TO_ALL packet types — these are (in order of transmission) the scaling type, dose array sizes, electron density grid, and kernels. The total memory required by all the arrays is then calculated and displayed, so that the user is aware of the transputers' memory usage. If the value of the *vector_size* parameter is non-zero then the communication vector size is set to that value (unless it is larger than the protocol permits), otherwise a more sophisticated vector size "ramping" approach is used (see later this section for an explanation of this technique).

Having booted the network and sent all required data to the worker tasks, the master task can then begin sending command vectors into the network. It does this by looping through the entire terma array, processing interaction voxels whose value exceeds a threshold value (defined by TERMA_THRESHOLD in s_superposition.h). The terma, kerma, and (x, y, z) indices associated with each of these voxels are appended to the command vector, then the byte order of each of these quantities is reversed (Suns have big-endian byte ordering, while transputers have little-endian byte ordering). This reordering must also be done prior to sending out arrays to worker tasks (described above), and upon receiving each worker dose distribution at the end of the calculation. The subroutines send_word(), matrix_to_workers(), and matrix_from_worker() automatically do this.

When the command vector reaches a size determined either by the value of the *vector_size* parameter or by the dynamic vector-size ramping algorithm, then the master task suspends execution until it receives an acknowledgement or "ready" packet from a worker task. However, there is no need to wait at the start of the calculation, since not all worker tasks have been allocated work. The master task sends the command vector out into the network, with the (destination) node number set to that of the free worker. This cycle of vector compilation, waiting, and transmission continues until all interaction voxels have been processed, and all incoming acknowledgement packets have been received.

The final phase of the computation is the summing of all worker dose distributions to yield the complete dose distribution. The master task sends a dose request to all worker tasks (using a packet of type WORD_TO_ALL), which signals the worker tasks to return their dose arrays. The master task then sums these incoming arrays into GRATIS's dose array and returns control back to the main() routine of s_photon.

```
type vector_element =  record
                           terma,kerma : real;
                           x,y,z : integer;
                       end;
constant MAX_VECTOR_SIZE;
var    inward_messages : mailbox;
       processor_number, vector_size : integer;

process s_trans_calc;
   var    n : integer;
          vector : array[0..MAX_VECTOR_SIZE-1] of vector_element;
   initialise and boot transputer system;
   do n := 0 . . processor_number - 1
      receive reply from filter n
   send kernel scaling type to all workers;
   send dose array sizes to all workers;
   send electron density grid to all workers;
   send energy deposition kernels to all workers;
   if (vector_size > 0)
      check vector_size is within protocol limits
   else
      determine number of non-zero terma voxels;
      determine starting communication vector size (for vector-size ramping)
   end if;
   while (still voxels left to process)
      build command vector;
      reorder bytes in command vector;
      if (all workers have been allocated vectors)
         receive ready signal from inward_messages;
         determine source node number n of ready signal
      else
         determine node number n of next free worker
      end if
      send vector to worker n;
      if (dynamic vector size ramping)
         recalculate command vector size
   end while;
   while (some packets have not yet acknowledged)
      receive ready signal from inward_messages
   send dose request to all workers;
   do n := 0 . . processor_number - 1
      receive worker dose array from inward_messages;
      master dose array := master dose array + worker dose array
   end do;
   end;

process s_worker;
   var    i : integer;
   receive kernel scaling type from master;
   receive dose array sizes from all master;
   initialise worker dose array;
   receive electron density grid from master;
   receive energy deposition kernels from master;
   loop
      receive vector from master;
      if (vector = dose request)
         goto end_label: ;
      do i := 0 . . number of vector elements - 1
         perform superposition using vector[i];
      send ready signal to inward_messages
   end loop;
   end_label:
      send worker dose array to inward_messages;
   end;
```

Figure 4.22. Pseudocode description of master and worker tasks (Sun-hosted system).

A pseudocode description of the worker task **s_worker()** is presented in the second part of Figure 4.22. The initial stages are essentially the opposite of the master task — the worker *receives* the scaling type, dose array sizes, electron density grid, and kernels. It then waits for a command vector and calls **s_calc_dose()** for each element in the vector (the **s_calc_dose()** routine is identical to that used in the non-parallel implementation of superposition). If the communication protocol is running efficiently then the wait part of the cycle is several orders of magnitude shorter than the time taken to process a vector. This cycle continues until the worker receives a dose request (of packet type **WORD_TO_ALL**), when it jumps out of the calculation loop and returns its dose array to the master task.

The worker task code (contained in **s_worker.c**) is compiled and linked with object code generated from **s_trans_calc.c** (located in the **c** subdirectory), and also with the stand-alone run-time library. The filter code **s_filter.c** is also compiled and linked with the stand-alone library, then the two task object files (**s_worker.b4** and **s_filter.b4**) are configured using the **s_trans_calc.cfg** configuration file generated by the **s_config** shell script (see Section 4.4.2.3). The resulting bootable code is then installed into the **GRATIS/bin** directory (see Chapter 3, Section 3.4.1). The entire compilation, linking, and configuration process is managed by a UNIX **make** file.

How does the *performance* of the Sun-hosted transputer network compare to that of the PC-hosted system discussed earlier is this section? The absolute performance of the two systems cannot be compared directly, since both the transputers and the external memory run at different speeds on the two systems, and the superposition algorithm is implemented differently. However, linearity of speedup and the influence of command vector size can be evaluated in the same way as for the PC-hosted system.

The superposition problem used for the following performance measurements is a 15 cm $\times$ 15 cm photon field of energy 10 MV, incident on the chest region of a Rando phantom. This field is in fact the left of the two fields illustrated in Chapter 3, Figures 3.13 and 3.14. The dose calculation grid used is that illustrated in Chapter 3, Figure 3.16, where the grid points are indicated by white dots. The transformed terma grid (in beam coordinates) is of size 22 $\times$ 22 $\times$ 69 voxels, each voxel being a cube of side 1.02 cm. The superposition kernels are of size 20 $\times$ 20 $\times$ 31 voxels.

Figure 4.23 shows computation times for the above problem on linear transputer arrays of up to 16 processors. Also shown are the computation times on a Sun SPARCstation IPC (whose processor is about 1.2 times as powerful as a single T800–25 transputer), and an estimate

of the time taken on a microVAX-II. The transputer network timings were originally almost twice as large as those presented in Figure 4.23, but careful re-coding of the innermost loops improved the code's efficiency considerably. Especially beneficial was the pre-calculation of complex array subscripts associated with the dose array and kernels.

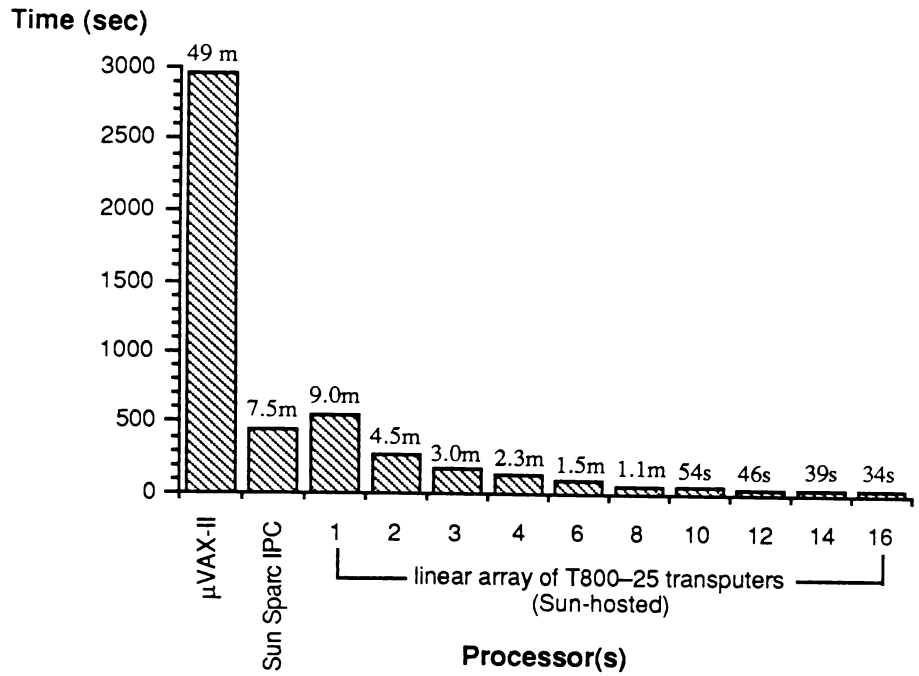

Figure 4.23. Computation times for a typical superposition problem under GRATIS (with the estimated microVAX-II timing provided for comparison). Times shown are for superposition of a 22 × 22 × 69 voxel terma array with a 20 × 20 × 31 voxel energy deposition kernel, simulating a 15 cm × 15 cm photon field (voxel size is 1.02 cm on each side). The number of non-zero terma voxels is 5818 (17.4% of the terma array), and the total number of interaction-deposition voxel pairs is 5.85×10^7 (primary and scattered energy was calculated separately). The z-axis algorithm for density scaling of the energy deposition kernel was used in obtaining these results. Communication packet size is 10 voxels, and total transputer memory usage is 391 kbytes.

As the network size increases the computation time appears to decrease as expected, and for a 16-transputer network the problem specified above can be calculated in approximately 34 seconds. For this problem the precalculations performed on the Sun's processor take over 50 seconds, so for larger network sizes the majority of calculation time is spent in these serial sections of code. However, the timings in Figure 4.23 have been generated using the z-scaling algorithm, which is approximately 40 times less computationally intensive than using a full ray trace — timings for the three methods (on a 16-element network) are 27.05 seconds using no scaling, 34.06 seconds using z-scaling, and 22.9 minutes using a full ray trace. Thus a 16-element transputer network is barely adequate for the example problem when using the most complex of the three density-scaling algorithms.

Linearity of speedup with increasing processor number is illustrated in Figure 4.24. A network of 16 processors provides 15.72 times the power of a single processor when using command vectors of 10 voxels each, which means that only 28% of the power of one processor is lost to communication overhead. For an 8-element network, 7% of the power of one processor is lost, which compares favourably with 19.5% for the PC-hosted system. Note that the command vector size used here is 10 voxels, rather than 40 voxels used for the PC-hosted system. This smaller vector size is required due to a more restrictive load-balancing constraint, imposed by a larger number of processors in the network and a smaller number of non-zero interaction voxels.

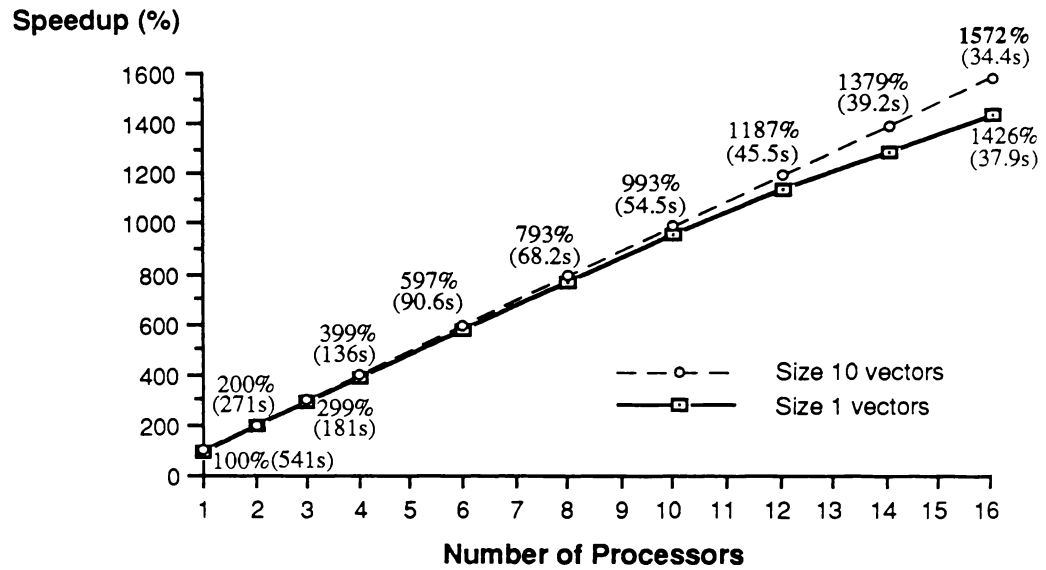

Figure 4.24. Speedup of the superposition algorithm on a Sun-hosted linear array, using command vectors of size 1 voxel (lower line) and 10 voxels (upper line).

Speedup in Figure 4.24 (and in Figure 4.17 for the PC-hosted system) has been calculated using a base figure which is the time taken by the fastest parallel algorithm executing on a single processor. However, a better measure of the value of a parallel implementation is *true speedup*, which measures performance against the fastest *serial* algorithm. Superposition has not been implemented on transputers using a serial algorithm, so this figure is not available directly. However, the timings presented in this section have been produced by having each individual transputer time its own calculation, then return that value to the master task for summation. By making the communication vector so large that only one vector is needed for all non-zero interaction voxels, and delaying the start of the timing until that vector has been received by the transputer, it is possible to obtain computation time *without communication overhead* on a "network" of 1 transputer. For the problem specified above this value is 539.00 $\pm$ 0.005 seconds. Figure 4.25 presents true speedup

calculated using this figure as a baseline, and it shows that the parallel algorithm is still operating efficiently (true speedup for a 16-element network is 1566%, compared with 1572% in Figure 4.24).

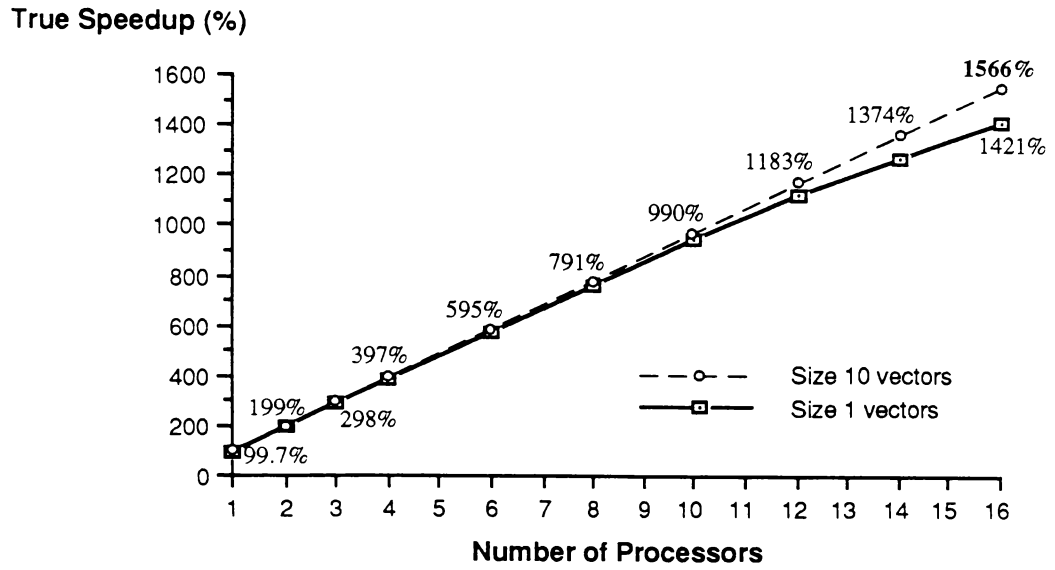

Figure 4.25. True speedup of the superposition algorithm on a Sun-hosted linear array, using command vectors of size 1 voxel (lower line) and 10 voxels (upper line).

A more intuitive way of looking at network performance is to use *true efficiency* E_{true}, where

$$E_{\text{true}} = \frac{\text{true speedup}}{\text{number of processors}} \times \frac{100}{1} \quad (\%) \ . \tag{4.5}$$

Figure 4.26 illustrates true efficiency as a function of network size for command vectors of size 1 and 10 voxels. It is clear from this graph that the true efficiency of a Sun-hosted network when using 10-voxel command vectors is very close to 100%, and drops off almost linearly for networks of up to 16 processors. However, using 1-voxel vectors significantly degrades network performance, which seems to drop off rapidly beyond 10 processors.

How does command vector size affect communication overhead? Figure 4.27 illustrates total computation times for networks of 1, 4, and 16 processors as a function of command vector size. The dependence on vector size is similar to that observed for the PC-hosted system (see Figure 4.18), where very small vector sizes have a large "packaging" and context-switching overhead, and very large vector sizes cause uneven load-balancing. The optimum vector size for most networks is approximately 10 voxels, which was the vector size used in generating Figures 4.24–4.26.

It is significant that the "flat" area at the bottom of the 16-processor curve is less noticeable than for smaller networks, and in fact 10-voxel vectors are not optimal in the case of a 16-

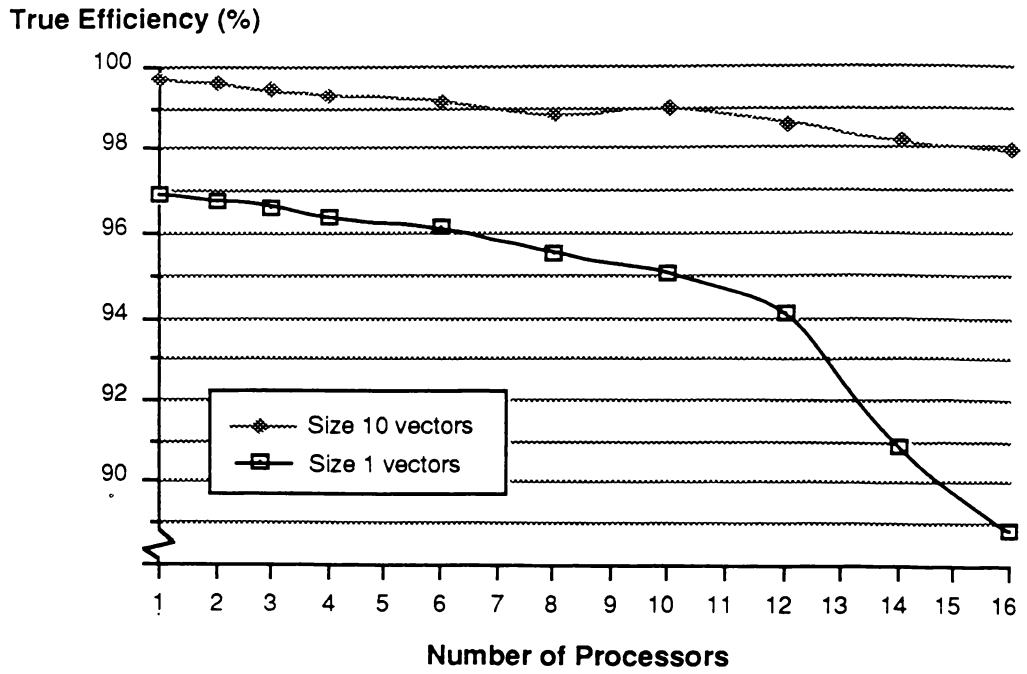

Figure 4.26. True efficiency of the superposition algorithm on a Sun-hosted linear array, using command vectors of size 1 voxel (lower line) and 10 voxels (upper line).

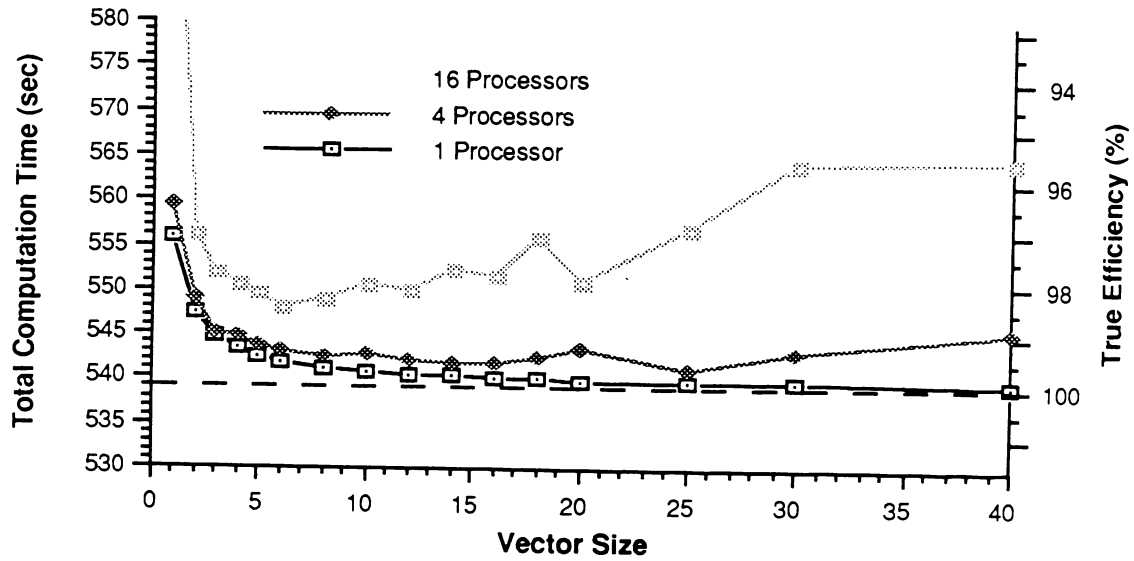

Figure 4.27. Total computation time as a function of command vector size for Sun-hosted linear arrays of various sizes. Right axis shows the true efficiency of the network.

processor network. This difficulty would be even more pronounced for networks of more than 16 processors, since the constraint imposed by the need for load-balancing would push the flat area of the curve towards a small vector size. For very large networks the load-balancing requirement would dictate a vector size of less than 5 voxels, leading to a significant reduction in network efficiency.

One way of lessening the above problem is to reduce command vector size as the calculation proceeds, so that all processors finish at nearly the same time. An approach of this type has been implemented on the Sun-hosted system by *ramping* the command size at the end of the calculation. Initially this was done such that the final vectors were of size ...6 5 4 3 2 1, as illustrated in Figure 4.28a. However, for the worst possible situation, where all processors are waiting as the last vectors are sent out, there is a large potential for uneven load balancing (15 units in the 6-processor network shown in Figure 4.28a). This potential has been reduced by adjusting the rate of ramping so that *two* vectors of each size are sent out by the master task. In the same example all processors are fully utilised when using this technique (see Figure 4.28b). Having all processors waiting is no longer the worst case when using this "slope = 1/2" approach, but the principle is clear: a slower reduction in vector size increases the likelihood of even load balancing. However, the amount of idle processor time is difficult to predict when employing a self-scheduled algorithm such as that used for superposition.

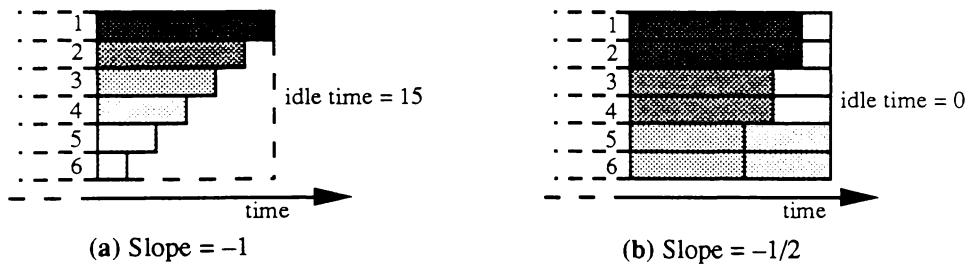

Figure 4.28. Ramping the command vector size to improve load balancing. (a) Worst-case scenario with vector size ramping, slope $= -1$. Unshaded space indicates idle time of processors near end of calculation. (b) Same scenario with ramping slope $= -1/2$. The slower decay in vector size allows all processors to be completely utilised (for this particular example).

Vector-size ramping is turned on by invoking **s_photon** with a *vector_size* of zero for the **-V** parameter. It is implemented within **s_trans_calc()** by determining the total number of non-zero terma voxels T, then finding the corresponding value for the starting vector size V_s such that the last vector is transmitted using a vector size of 1. For ramping with "slope = 1/2" the expression to calculate this value is:

$$V_s = \left\lfloor \frac{\sqrt{4T+1}-1}{2} \right\rfloor \quad . \tag{4.6}$$

If there is a large number of interaction voxels then V_s may well exceed the vector size limit V_{max} enforced by the communication protocol, in which case V_s is set to that limit value. The number of voxels in the ramp is then $V_s(V_s+1)$, and the T non-zero interaction voxels are transmitted as:

(a) One vector of $(T - V_s(V_s + 1))$ *mod* V_{max} voxels.

(b) $(T - V_s(V_s + 1))$ *div* V_{max} vectors of V_{max} voxels.

(c) $2V_s$ vectors of sizes V_s, V_s, $V_s - 1$, $V_s - 1$, $V_s - 2 \ldots 4$, 4, 3, 3, 2, 2, 1, 1.

For example, the problem used previously has 5818 non-zero terma voxels, which are transmitted as one vector of 43 voxels and 150 vectors of sizes 75, 75, 74, 74, 73 ...3, 3, 2, 2, 1, 1. Note that there are no vectors of type (b) in this example, since $V_{max} = 100$ for this protocol, which is greater than V_s.

The effect which vector-size ramping has on total computation time (and also true efficiency) is shown in Figure 4.29. Vector-size ramping with "slope = −1/2" reduces communication overhead to about half that occurring with the optimum constant vector size for each network, and even "slope = −1" vectors are superior to optimally-chosen constant vector sizes. Thus the vector-size ramping technique offers *two* significant advantages: communication overhead is less, and the vector sizes are automatically determined, rather than having to be estimated from a knowledge of the network's past behaviour.

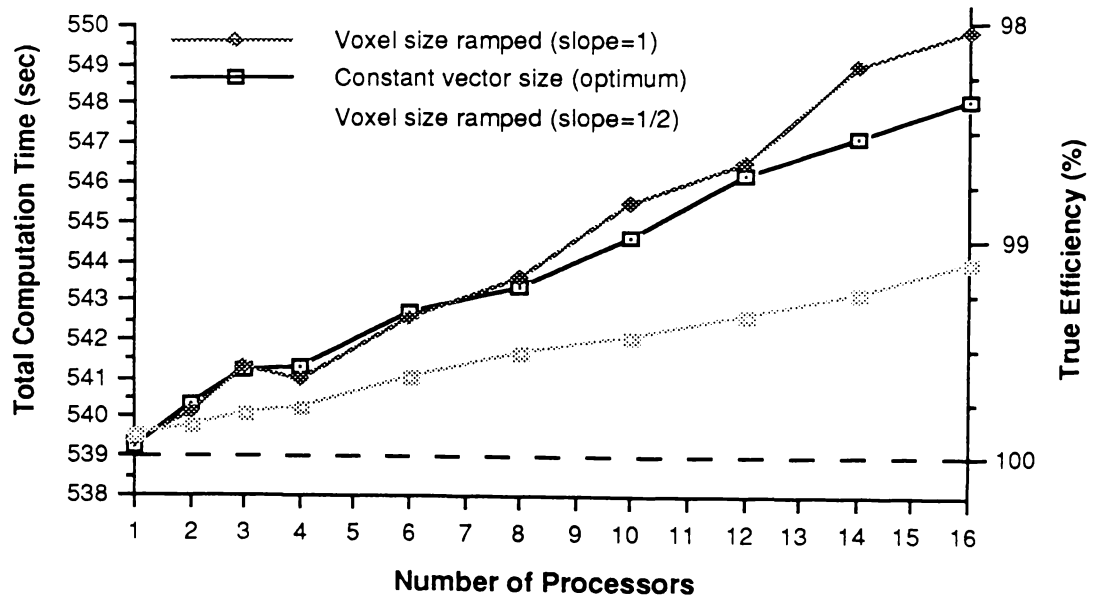

Figure 4.29. Total computation time as a function of Sun-hosted network size, using vector-size ramping. Right axis shows the true efficiency of the network.

The performance of the Sun-hosted transputer network, as demonstrated by the above results, can be summarised as follows:

- Speedup is superior to that obtained using the PC-hosted system. For a typically-sized sample problem a linear array of 16 processors provides 15.66 times the computing

power of a serial algorithm running on a single transputer (when using a 10-voxel vector size).

- Larger communication vector sizes reduce communication overhead, but the requirement of even load-balancing places a lower limit on the maximum vector size when using large arrays of processors.
- A "voxel-ramping" technique is effective in increasing true efficiency to more than 99% for a network of 16 processors, and this approach also removes the need for any *a priori* assumptions about the optimal communication vector size.

Summary

This section has explored the performance characteristics of the superposition algorithm when implemented on two very similar parallel-processing systems. The results obtained suggest that the superposition algorithm is well-suited to implementation in a parallel form, and that one type of parallel system, a multicomputer network based on transputers, has excellent performance characteristics when performing a superposition calculation. It appears that a multicomputer network comprising more powerful, or more numerous, processors should continue to behave in a way which would result in significant speed improvements. The most sophisticated of the three density-scaling algorithms (full ray tracing) is certainly in need of a more powerful system, but use of a coarse calculation grid in considering secondary scatter components,[26] in conjunction with interpolation in regions of low dose gradient,[41] could significantly reduce computation time. However, the 16-element Sun-hosted network is more than adequate for typically-sized superposition calculations when using the z-scaling algorithm.

4.4.3.2. FFT Convolution

Convolution using the fast Fourier transform approach is discussed in Chapter 3, Section 3.3.5, and by Murray *et al.*[35] Parallel computation of FFTs has been extensively documented, particularly with regard to array processors,[42] but in general these approaches rely on a relatively complex decomposition of the problem so that individual one-dimensional FFTs are calculated by many processors.[1] In conjunction with early work on FFT convolution,[35] this author implemented a parallel FFT convolution algorithm on the PC-hosted transputer network, using a simpler approach not involving decomposition of individual one-dimensional transforms.

The FFT calculation technique used was the row-column decomposition method discussed in Chapter 3, Section 3.3.5, where each three-dimensional FFT is calculated by performing

a set of one-dimensional FFTs in each dimension. Using this method an $i \times j \times k$ array requires $\mathcal{F}_{\text{total}}$ one-dimensional FFTs where

$$\mathcal{F}_{\text{total}} = (i \times j) \times \mathcal{F}_k \; + \; (i \times k) \times \mathcal{F}_j \; + \; (j \times k) \times \mathcal{F}_i \tag{4.7}$$

and $\mathcal{F}_n$ is a one-dimensional FFT of length n. To calculate the i-direction FFTs each worker task is passed a packet containing the j and k subscripts, i (the number of points in the FFT) and a sequence of i floating point numbers representing the data to be transformed (see Figure 4.30). The worker task then performs the FFT and sends back a packet identical in format except that the i floating point numbers contain transformed data. The j and k indices are included so that the master task can identify the completed FFT and replace the transformed data correctly into the three-dimensional array. An analogous sequence of operations is then performed for the j and k directions.

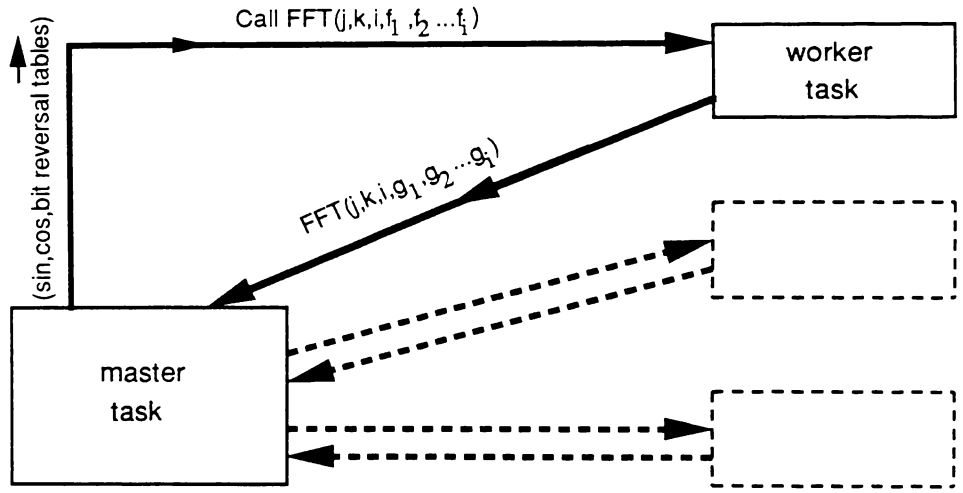

Figure 4.30. Implementation of FFT convolution on a multicomputer network.

To calculate dose using the FFT approach the above sequence of operations is carried out on the terma array and energy deposition kernel in turn. The resulting arrays (which have real and imaginary components) are multiplied together by the master task, then the inverse FFTs are calculated by the worker tasks.

The FFT algorithm used by the worker tasks is a standard radix-2 FFT using lookup tables for sine, cosine and bit reversal operations. Note that since the one-dimensional FFT arrays are typically less than 4 kbytes in length they can be stored in on-chip memory, resulting in relatively fast execution (a single transputer running in this mode almost exactly matches the performance of a VAX 6200-series processor).

FFT convolution is considerably faster than superposition for large array sizes. For example, calculating the superposition of a 16 × 16 × 100 TERMA array with a 16 × 16 × 72 energy deposition kernel requires 1560 seconds on a VAX 6220, compared with just 213 seconds for the same calculation using an FFT approach (see Chapter 3).

Figure 4.31 presents FFT timings for the problem which was described in conjunction with the superposition algorithm (see Figure 4.16). The processing network is a PC-hosted linear array of T800–20 transputers. Note that computation time decreases slowly up to a four-element array, then remains constant for larger networks — in all cases the computation time is actually greater than that obtained by a single transputer using a non-parallel algorithm. The vector-radix decomposition approach has also been implemented on the transputer network, but the overhead in performing butterfly multiplications is sufficiently large that no significant performance advantage can be gained.

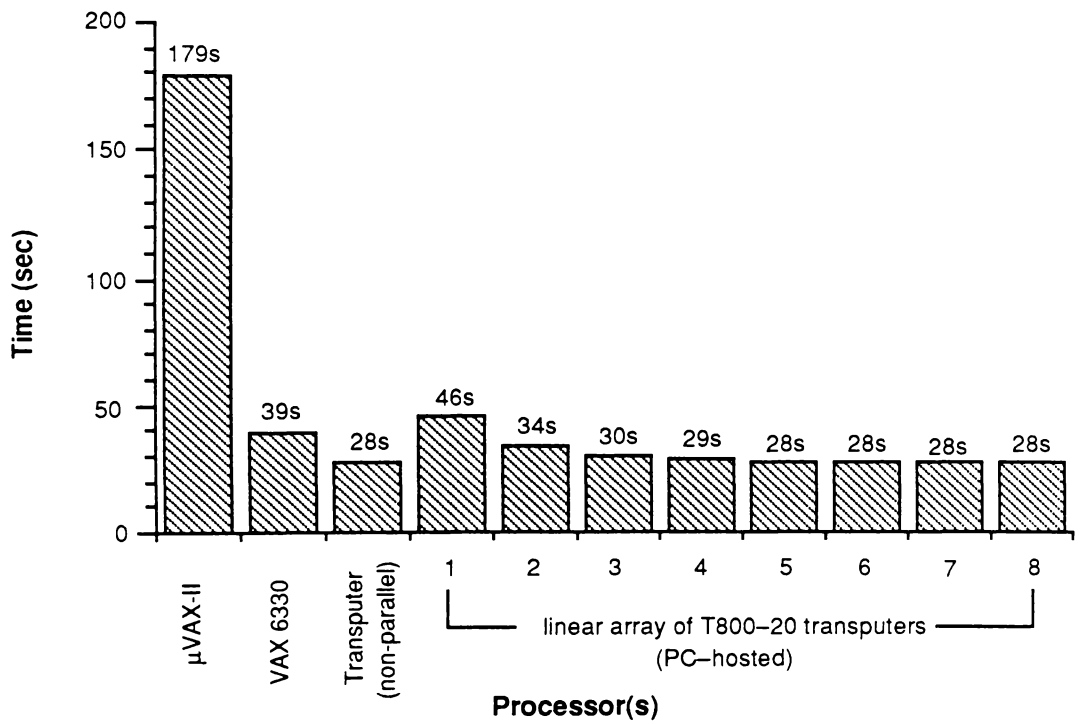

Figure 4.31. Computation times for a typical FFT convolution problem on various processors, using row-column decomposition and radix-2 transforms. Times shown are for convolution of a 16 × 16 × 60 voxel terma array with a 16 × 16 × 68 voxel energy deposition kernel, simulating a 5 cm × 5 cm photon field (voxel size is 0.5 cm on each side). These arrays have been padded out to 32 × 32 × 128 voxels for the FFT calculation. Also shown is the time required by a single transputer running in a non-parallel mode.

The results presented in Figure 4.31 demonstrate that the row-column approach to FFT convolution is an unsatisfactory method of problem decomposition on a multicomputer, even

for small network sizes. This is due primarily to the large amount of processor time spent communicating the one-dimensional arrays (communication time is of the same order of magnitude as FFT computation time when dealing with 1024-element FFTs on a transputer network). A far more successful approach would be to connect the transputers into a butterfly network[1] rather than a linear array, and use a computation algorithm designed to exploit these communication paths. However, since the forced invariance of the convolution kernels is a serious disadvantage of Fourier space convolution, this FFT approach was abandoned in favour of real-space superposition, discussed in Section 4.4.3.1.

4.4.3.3. Electron Pencil Beams

Dose distributions due to *electron* beams are also routinely calculated on treatment planning workstations. The GRATIS Treatment Planning System described in Chapter 3 does not have an explicit electron beam capability, but `xvsim` can easily be used to place electron rather than photon beams, provided an appropriate electron treatment machine has been defined.

An electron pencil-beam dose calculation algorithm (called `p_electron`) has been implemented on the GRATIS system by Hoban.[43] Based on the method of Hogstrom,[44] it first calculates radiological depth for each voxel, and also a corresponding *standard deviation* representing the lateral spread of the electron beam at that depth. It then steps down each column of voxels in the radiological depth array, assigning dose due to that "pencil" on the corresponding plane. In the approximation used with this algorithm, the dose is spread out using a normal distribution, where the standard deviation is obtained from the previously calculated grid. The source code for the pencil beam algorithm is very similar to that developed for the superposition algorithm (see Section 4.4.3.1), with the majority of the modules being either identical or very similar to those in `s_photon`.

The parallel-code modules written by this author are also very similar to those for superposition. The master task `p_trans_calc()` first sends the treatment head jaw positions, the machine SAD, parameters specifying the electron beam, a table of error functions (used for "spreading" dose across each plane), the radiological depth grid, the standard deviation grid, and the dose array sizes. Next it compiles vectors of (i, j) column coordinates and distributes them among the worker tasks, then finally it retrieves the component dose arrays and returns control to the main routine of `p_electron`.

The worker task `p_worker()` is located in the `p_parallel` directory (a subdirectory of `src`). It first receives the arrays transmitted by `p_trans_calc()`, then processes incoming

vectors of (i, j) column coordinates by calling the pencil beam dose calculation routine `p_calc_dose()`. Finally, it transmits the dose array back to `p_trans_calc()` and exits.

Figure 4.32 illustrates the speedup achieved for the electron pencil-beam algorithm on Sun-hosted transputer networks of 1–16 processors. The beam being calculated is a 5 cm × 5 cm electron beam of energy 12 MeV, incident on the chest region of a Rando phantom. The *total* transputer time required by this problem is only a little more than 1 minute, so it is clear that the electron pencil beam algorithm is much less computationally intensive than superposition. A 16-processor network performs the calculation in under 6 seconds, which is negligible when compared to the time spent on serial calculations (approximately 40 seconds). Note that the vector-size ramping technique has been compared with constant-sized vectors of only one (i, j) column. This very small vector size is needed because there are only 784 voxel columns, and when these columns are distributed among 16 processors then a 1-column vector represents more than 2% of a processor's workload.

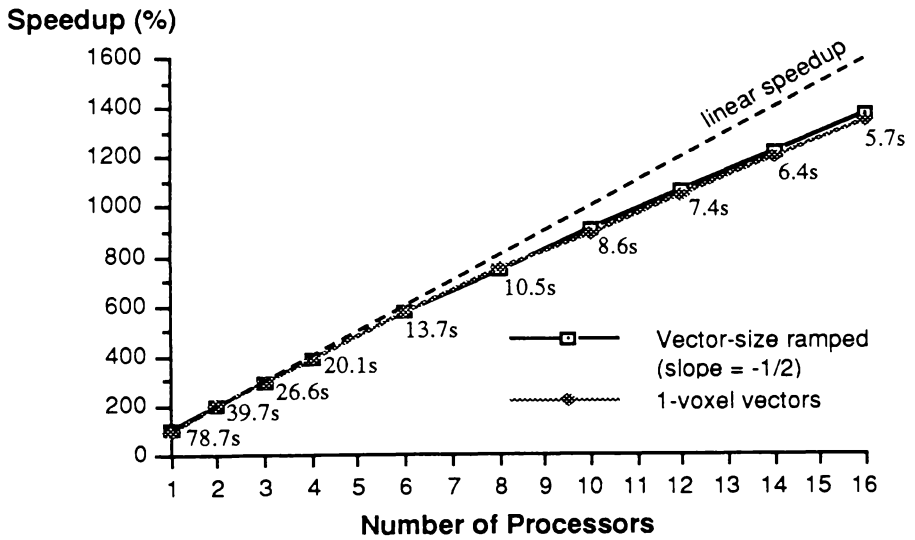

Figure 4.32. Speedup of the electron pencil beam algorithm on a Sun-hosted linear network of transputers. Results shown are for a 5 cm × 5 cm 12 MeV electron beam incident on a Rando phantom. The radiological depth grid (in beam coordinates) is of size 36 × 36 × 89 voxels (voxel size is 0.5 cm on each side). The total number of pencils is 784, and total transputer memory usage is 421 kbytes.

From Figure 4.32 it is apparent that the speedup is much less linear than observed for superposition. Ramping vector size does not significantly improve linearity over 1-column vectors, and 2-column and 3-column vectors (not shown in Figure 4.32) also have very similar speedup. Hence communication overhead is not responsible for the non-linearity. It is in fact due to the significant response time of the master task, which has to compile and

transmit a large number of vectors in a short time interval. As the network size increases the time interval becomes shorter, and the likelihood of acknowledgement packets arriving in rapid succession increases. When this occurs a worker task must wait for another task's vector to be transmitted before its own is compiled, and thus it is idle during that period.

This non-linearity occurs solely because of the small problem size. The amount of computation required by the pencil beam algorithm is in many cases insufficient to warrant a 16-element transputer array, but for much larger problems (such as those involving very fine grid spacing), its efficiency is comparable to that observed for the superposition algorithm.

4.4.4. Remote Dose Computation on a Sun Network

There is a significant limitation of using a transputer-based parallel processor on a *network* of workstations: the dose computation code must be executing on the particular node which contains the Sbus interface. In a clinical situation there might be many separate treatment planning sessions executing simultaneously, and therefore the machine hosting the transputer network could easily become overloaded. In addition, the X-server for each session has to be redirected to another display (using `setenv DISPLAY` or equivalent), thereby increasing network traffic.

Another option is to write the parallel part of the dose calculation algorithm in such a way that it can be executed as a *remote procedure*. This can be done using Sun's implementation of the RPC (remote procedure call) communications paradigm, and associated XDR (external data representation) protocol. These facilities are explained fully in Sun's *Network Programming Guide*,[40] but in this context a brief overview of network programming will be given, illustrated using the dose computation server developed by this author.

RPC implements the *client-server model* of network communication — that is, a locally running application (client) sending requests to a remote procedure (server). It provides mechanisms such that clients and servers can be written independently of the nature of the underlying network. However, network dependent aspects of the protocol can be modified by the programmer if desired. For example, higher levels of RPC assume that communication is based on the UDP transport protocol, but this has disadvantages when dealing with large arrays (no error correction, and an 8 kbyte limit on the size of arguments and results). The dose computation server has therefore been implemented using the TCP protocol, which provides error correction facilities and allows large arguments and results. This approach must be specified using the lowest of the three RPC interface levels.

To simplify the development of network applications, RPC has an associated protocol com-

piler called **rpcgen**, which automatically supplies much of the low-level (implementation-dependent) code. Input to **rpcgen** is a single remote program interface definition, written in a C-like language. For the dose computation server this definition (**s_rpc_calc.x**) takes the following form:

```
program S_RPC_PROGRAM {
    version S_RPC_VERSION {
        s_rpc_result_structure S_RPC_SERVER(s_rpc_call_structure) = 1;
    } = 1;
} = 0x20000099;
```

where **s_rpc_call_structure** is a structure containing pointers to all data required by the dose computation server, and **s_rpc_result_structure** is a structure for returning the result. These structures, defined in **s_parallel.h**, are required because the RPC protocol dictates that there be only one argument to a remote procedure, with the procedure returning one result. A procedure number, version number, and program number must be specified — the important point here is that user-written RPC routines must have (unique) program numbers in the range **0x20000000** – **0x3fffffff**.

The command "**rpcgen s_rpc_calc.x**" then invokes the **rpcgen** protocol compiler with **s_rpc_calc.x** as input. This produces modules containing source for both the client (**s_rpc_calc_clnt.c**) and server (**s_rpc_calc_svc.c**). It also produces a header file, **s_rpc_calc.h**, which contains **define** statements for the procedure, version, and program numbers.

If the argument and result structures had been defined in the RPC definition **s_rpc_calc.x**, then XDR (external data representation) routines would also have been generated automatically by **rpcgen**. The XDR protocol[40] is a Sun-developed communications standard which allows machine-independent data transfer using low-level primitives such as **xdr_float()** (for transferring a four-byte floating-point value between machines) and **xdr_array()** (for transferring arrays of a specified component type). Provided both machines have these primitives, communication can be independent of considerations such as byte-ordering and floating-point representation. The **rpcgen** protocol compiler is capable of building XDR routines for transferring structures across a network, but the very complex nature of the **GRID** and **EDKS** data structures used in the GRATIS system precludes this approach. In particular, the number of matrix elements in these data structures is dependent upon other elements in those structures (**x_count**, **y_count**, and **z_count**), which requires hand-coding of the XDR routines. An XDR routine for the **GRID** structure was supplied with the GRATIS system, a routine for the **EDKS** structure was written by Hoban, and routines for **CGRID**,

s_rpc_call_structure, and **s_rpc_result_structure** were written by this author. These routines are located in **s_xdr_defs.c**.

Figure 4.33 illustrates how the RPC-based client-server model is implemented in the case of superposition dose computation. The client program (**s_photon**) calls the server from the procedure **s_rpc_client()**, using the stub **s_rpc_calc_clnt()** (generated by **rpcgen**) as a "dummy" server module. To do this **s_rpc_client()** first gets the network address of the server host by passing the server host name (obtained from **s_photon**'s **-R** parameter) to **gethostbyname()**. It then obtains a TCP-based transport handle using the RPC routine **clnt_tcp_create()**, which in turn obtains the address of the remote service by querying the remote portmapper. After setting the server timeout to one hour using **clnt_control()**, it builds the server argument (of type **s_rpc_call_structure**). It then calls the remote service routine **s_rpc_server_1()**, which encodes (*serialises*) the argument using the appropriate XDR routine, and waits for the remote service to return a result, *deserialising* it using another XDR routine. Finally, **s_rpc_client()** assigns the result to its dose array parameter, destroys the client transport handle using **clnt_destroy()**, and returns.

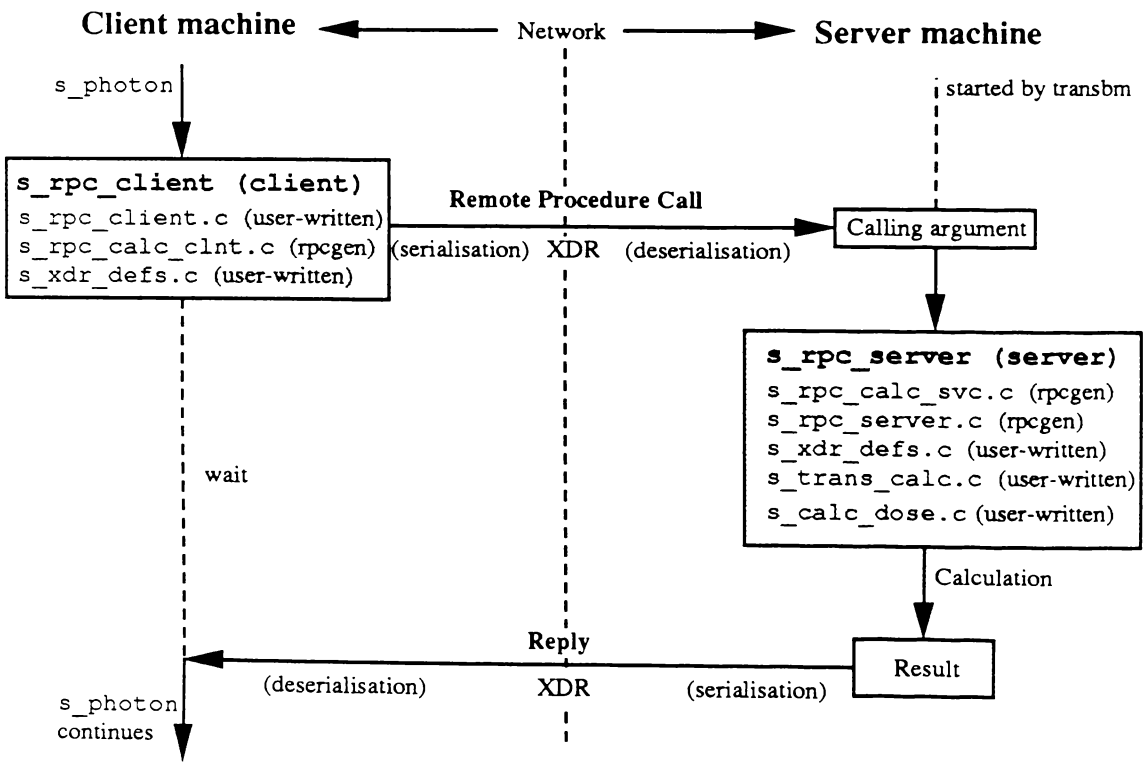

Figure 4.33. Performing superposition over a network using the remote procedure call (RPC) paradigm.

The server side of the application is implemented by the **s_rpc_calc_svc.c** and **s_rpc_server.c** modules. The main routine, located in **s_rpc_calc_svc.c**, is generated

automatically by **rpcgen** — it gets a transport handle using **svctcp_create()**, destroys any trace of previous servers' mappings using **pmap_unset()**, registers the server with the host's portmapper using **svc_register()**, and sets the server procedure running using **svc_run()**. The server procedure listens for incoming client requests, deserialises the incoming structure using an XDR routine, calls the user written service routine **s_rpc_server_1()**, then serialises the result structure and sends it to the client, again using an XDR routine. Normally the server would sit idle until another client request arrives, but for this particular application the server has been modified to automatically exit after servicing a client request.

The user-written service procedure **s_rpc_server_1()** is relatively simple. It allocates space for the resultant dose matrix, then calls **s_trans_calc()** in the usual way, using parameters obtained from its argument (of type **s_rpc_call_structure**).

The above implementation of the remote dose computation facility assumes that the server is already running on the remote machine. One way to ensure this is to use the UNIX Internet services daemon **inetd**, but the approach taken by this author is to start the remote server directly from the **transbm** script using the UNIX remote shell (**rsh**) command. The relevant extract from **transbm** is as follows ("uranus" is the name of the server host):

```
# Use remote server if we are not running on uranus
set SERVERNODE = uranus
set SERVERNAME = s_rpc_server
set HOST = `hostname`
echo Running on $HOST
if ($HOST != $SERVERNODE) then
  set ROPTION = "-R $SERVERNODE"
  echo Starting server on $SERVERNODE...
  rsh $SERVERNODE $SERVERNAME &
else
  set ROPTION =
  echo Calculating dose on this node...
endif
```

The above script excerpt determines the name of the host on which it is running, and if it is not uranus then it sets **ROPTION** to "**-R uranus**" and starts the server on the remote node using **rsh**. If the script is running on uranus then it simply sets **ROPTION** to the null string. Later in the script **s_photon** is invoked with **ROPTION** as one of its parameters.

Setting up a remote service is relatively straightforward, although the programmer must pay attention to detail, especially if XDR routines have to be coded by hand. However, once developed the technique works well. For the superposition problem used in Section 4.4.3.2 the total overhead imposed by using RPC on the (lightly-loaded) Computer Science Sun

network is 7.3 ± 1.1 seconds. This overhead is a small proportion of the time required to compute a typical dose distribution (particularly when using full ray tracing).

RPC could be adapted to other dose computation algorithms with very little modification. It could also be used in other areas of radiotherapy planning, in a manner similar to that described by Neblett and Hogan[45] using a PC-based local area network. Since it is generally possible for client programs to continue processing while waiting for a server, the RPC approach also provides a means of using a (possibly heterogeneous) network of processors as a parallel processing system. In principle a transputer network could be connected directly to an ethernet (using an ethernet TRAM), thereby completely removing the need for a local host processor.

4.5. Summary

Development of more accurate treatment planning algorithms is highly desirable, especially for the higher treatment energies generated by linear accelerators. Monte Carlo and superposition techniques are the only algorithms currently known which take electron ranging into account, and hence are the most suitable on which to conduct further research.

It is however necessary to increase the speed at which both Monte Carlo and superposition calculations can be performed. One way to do this is by developing the algorithm itself, but for any given algorithm/processor combination a significant increase can also be achieved by the use of a parallel processor, such as a multicomputer transputer network. Even so, Monte Carlo simulations are unlikely to be fast enough for routine treatment planning use in the foreseeable future, but this chapter has shown how a parallel processing system can be used to successfully accelerate the photon superposition and electron pencil beam algorithms to a degree where they can be clinically useful. In particular, a 16-element Sun-based transputer network has been used to calculate dose due to a 15 cm × 15 cm 10 MV photon beam incident on a Rando phantom in 34 seconds when using a z-scaling superposition algorithm. The high efficiency of the parallel superposition algorithm described in this chapter should enable the same approach to be used with larger networks, possibly employing more powerful processors. Such networks should enable full ray tracing to be used for kernel density scaling (a process which is approximately 40 times more computationally intensive than the z-scaling algorithm).

References

(1) Quinn M.J., *Designing Efficient Algorithms for Parallel Computers.* McGraw-Hill, New York (1987).

(2) Electronics, Supercomputers: The Proliferation Begins (Special Issue). *Electronics* (1988), **61 (5)** (March 3), 51.

(3) Bell G., The future of high performance computers in science and engineering. *Commun. ACM* (1989), **32 (9)** 1091.

(4) Hockney R.W. and Jesshope C.R., *Parallel Computers: Architecture, Programming and Algorithms.* Adam Hilger Ltd, Bristol (1981).

(5) Krishnamurthy E.V., *Parallel Processing: Principles and Practice.* Addison-Wesley, New York (1989).

(6) Desrochers G.R., *Principles of Parallel and Multi-processing.* McGraw-Hill, New York (1987).

(7) Flynn M.J., Very high speed computing systems. *Proc. IEEE* (1966), **54** 1901.

(8) Flynn M.J., Some computer organizations and their effectiveness. *IEEE Trans. Computers* (1972), **21** 948.

(9) Shore J.E., Second thoughts on parallel processing. *Comput. Elect. Eng.* (1973), **1** 95.

(10) Andrews G.R. and Schneider F.B., Concepts and notations for concurrent programming. *Comput. Surv.* (1983), **15 (1)** 3.

(11) Ibbett R.N. and Topham N.P., *Architecture of High Performance Computers (Volume I): Uniprocessors and Vector Processors.* MacMillan, London (1989).

(12) Brown F.B. and Martin W.R., Monte Carlo methods for radiation transport analysis on vector computers. *Prog. in Nucl. Ener.* (1984), **14** 269.

(13) Miura K., EGS4–V: Vectorization of Monte Carlo cascade shower simulation code EGS4. *Comput. Phys. Comm.* (1987), **45 (1)** 127.

(14) Reed D.A. and Fujimoto R.M., *Multicomputer Networks: Message-Based Parallel Processing.* MIT Press, Cambridge, Massachusetts (1987).

(15) Ibbett R.N. and Topham N.P., *Architecture of High Performance Computers (Volume II): Array Processors and Multiprocessor Systems.* MacMillan, London (1989).

(16) Athas W.C. and Seitz C.L., Multicomputers: Message-passing concurrent computers. *Comput.* (1988), **21 (8)** 9.

(17) Ullman J.D., *Computational Aspects of VLSI.* Computer Science Press, Rockville, Maryland (1984).

(18) Levitan S.P., Evaluation criteria for communication structures in parallel architectures. In *Proceedings of the International Conference on Parallel Processing, 1985.* IEEE, New York (1985).

(19) Inmos, *The Transputer Reference Manual.* Prentice-Hall, U.K. (1988).

(20) Inmos, *The Transputer Applications Notebook.* Inmos, Bristol, U.K. (1989).

(21) Microway, *Videoputer User Manual.* Microway Limited, Surrey, U.K. (1988).

(22) Transtech, *TMB–08 Installation and User Guide.* Transtech Devices Limited, Buckinghamshire, U.K. (1988).

(23) Inmos, *IMS–B012 User Guide and Reference Manual.* Inmos, Bristol, U.K. (1988).

(24) 3L, *Parallel C User Guide.* 3L Limited, Livingston, Scotland (1988).

(25) Hoare C.A.R., *Communicating Sequential Processes.* Prentice-Hall, Englewood-Cliffs, New Jersey (1985).

(26) Mackie T.R., Scrimger J.W. and Battista J.J., A convolution method of calculating dose for 15-MV x rays. *Med. Phys.* (1985), **12** 188.

(27) Boyer A.L. and Mok E.C., A photon dose distribution model employing convolution calculations. *Med. Phys.* (1985), **12** 169.

(28) Mohan R., Chui C. and Lidofsky L., Differential pencil beam dose computation model for photons. *Med. Phys.* (1986) **13** 64.

(29) Boyer A.L., Wackwitz R. and Mok E., A comparison of the speeds of three convolution algorithms. *Med. Phys.* (1988), **15** 224.

(30) Boyer A.L. and Mok E.C., Calculation of photon dose distributions in an inhomogeneous medium using convolutions. *Med. Phys.* (1986), **13** 503.

(31) Hoban P.W., Murray D.C., Metcalfe P.E. and Round W.H., Superposition dose calculation in lung for 10 MV photons. *Australas. Phys. Eng. Sci. Med.* (1990), **13 (2)** 81.

(32) Kalet I.J. and Jacky J.P., Radiation therapy treatment planning using concurrent programming. *Comput. Programs Biomed.* (1988), **26** 115.

(33) Werner T, Herrman G., Schlegel W. and Lorenz W.J., Parallel processing in radiotherapy treatment planning. In *The Use of Computers in Radiation Therapy, p21*, edited by Bruinvis I.A.D. *et al.*, Elsevier Science Publishers (North Holland) (1987).

(34) Cvetanovic Z., The effect of problem partitioning, allocation, and granularity on the performance of multiple-processor systems. *IEEE Trans. Computers* (1987), **36** 421.

(35) Murray D.C., Hoban P.W., Metcalfe P.E. and Round W.H., 3-D superposition for radiotherapy treatment planning using fast Fourier transforms. *Australas. Phys. Eng. Sci. Med.* (1989), **12 (3)** 128.

(36) Metcalfe P.E., Hoban P.W., Murray D.C. and Round W.H., Modelling polychromatic high energy photon beams by superposition. *Australas. Phys. Eng. Sci. Med.* (1989), **12 (3)** 138.

(37) Nelson W.R., Hirayama H. and Rogers D.W.O., *The EGS4 Code System.* Stanford Linear Accelerator Center Report SLAC–265 (1985).

(38) Siddon R.L., Fast calculation of the exact radiological path for a three-dimensional CT array. *Med. Phys.* (1985), **12** 252.

(39) Kruatrachue B. and Lewis T., Grain size determination for parallel processing. *IEEE Softw.* (1988), **5 (1)** 23.

(40) Sun Microsystems, *Network Programming Guide.* Sun Microsystems, Mountain View, California (1990).

(41) Niemierko A. and Goitein M., The use of variable grid spacing to accelerate dose calculations. *Med. Phys.* (1989), **16** 357.

(42) Jamieson L.H., Mueller P.T. and Siegel H.J., FFT algorithms for SIMD parallel processing systems. *J. Parallel Dist. Comput.* (1986), **3** 48.

(43) Hoban P.W., *Lateral Electron Disequilibrium in Radiotherapy Treatment Planning.* University of Waikato D.Phil. Thesis (1991).

(44) Hogstrom K.R., Mills M.D. and Almond P.R., Electron beam dose calculations. *Phys. Med. Biol.* (1981), **26 (3)** 445.

(45) Neblett D.L. and Hogan S.E., Local area networks as a multiprocessor treatment planning system. In *The Use of Computers in Radiation Therapy, p567*, edited by Bruinvis I.A.D. *et al.*, Elsevier Science Publishers (North Holland) (1987).

Chapter 5

Future Trends in Radiotherapy Dose Computation

5.1. Calculation Algorithms

The simplest methods of radiotherapy dose calculation involve simply summing the influence of beams whose dose distribution is assumed to be the same as that in a water phantom. The next level of sophistication involves correcting for the effective depth of each point within the patient. Following that there have been a number of techniques developed which account for scattered radiation, and the more advanced of these algorithms are used in today's commercially available radiotherapy treatment planning systems.

All these correction techniques make the approximation of *local energy deposition*, where charged particles are assumed to have negligible range. This is a good approximation in the case of cobalt-60 beams, but for higher energy x-rays, such as those produced by linear accelerators, electrons may range several centimetres, depositing energy along the entire length of their track. This phenomenon introduces the possibility of *electronic disequilibrium*, where the number of electrons scattering into a region is not equal to the number scattering out. Commonly this occurs in the build-up region, in the beam penumbra, and also in regions of heterogeneity. The effect on dose distribution in the build-up and penumbral regions is well known, but in regions of heterogeneity all the algorithms mentioned above completely fail to model electron ranging.

One method which does account for electron ranging is the *Monte Carlo* technique,[1] which seeks to predict energy deposition by directly modelling particle transport. The Monte Carlo method is used in many areas of science, and has been used to model particle interactions from as early as 1954, when Hayward and Hubbell followed the histories of 67 incident photons using a desk calculator.[2] However, the accuracy required for most radiotherapy applications dictates that many millions of histories be followed in a typical photon beam simulation.

In the face of such a serious limitation, why is the Monte Carlo technique so desirable for radiotherapy computations? Here are some advantages:

- Modelling energy deposition by simulating physically occurring processes is conceptually appealing to the physicist.
- A rigorous Monte Carlo algorithm is theoretically capable of modelling *all* possible physically occurring effects, and hence is truly a general-purpose method.
- Monte Carlo can generate results which are incapable of being measured experimentally, such as the energy deposited in very small regions which would be disturbed by the presence of a dosimeter.

- Given enough particle histories Monte Carlo is capable of arbitrarily high accuracy, provided the physical processes are correctly modelled.

In the particular case of radiotherapy treatment planning, the first and last of the above points are probably the most influential — medical radiation physicists see Monte Carlo as a conceptually simple (although implementationally complex) technique with potential for very high accuracy. It also has applicability to both electron and photon beams.

However, the Monte Carlo approach to radiotherapy dose computation has two major disadvantages. First, it requires knowledge of the incident beam spectrum, which is difficult to obtain for high-energy photon beams. Generally this problem is solved by using Monte Carlo modelling to simulate the treatment machine itself,[3] or alternatively the spectrum can be inferred by deconvolution.[4]

The second and more serious limitation, mentioned above, is that the computational effort required by Monte Carlo currently prevents its use in routine treatment planning. However, taking one step back from Monte Carlo, it is possible to deterministically calculate the primary energy *liberated* from each point in the medium. If the deposition of energy due to liberation of particles from a point is *pre-calculated* using Monte Carlo, then it is possible to use this distribution (or *kernel*) to account for the *deposition* of dose. In a homogeneous medium there is no loss of accuracy, provided the particle spectrum is modelled correctly. This *convolution* technique reduces computation time enormously, especially if a Fourier space approach is used.

In many clinical situations the region of interest is not homogeneous, hence the distribution of energy liberated from a point depends upon the location of that point within the medium. The accuracy of convolution is immediately compromised, since the kernel must now be *scaled* to account for inhomogeneities. The *superposition* technique does this, but the now non-invariant kernel necessitates real-space calculation. In addition, rigorous treatment of kernel scaling requires determination of the average density between *each pair* of interaction and deposition points.

What will be the future method of choice for radiotherapy dose computation? Clearly it must be Monte Carlo. If a lack of computational power makes this technique impractical, then *superposition* can most closely approach the physically correct dose distribution which Monte Carlo delivers. However, both techniques are computationally intensive, and thus there is a need for more powerful computer hardware which is capable of accelerating these algorithms.

5.2. Computer Hardware

Both the superposition and Monte Carlo techniques require a large amount of computation to deliver accurate results. The very fastest of today's serial processors are probably capable of performing rudimentary superposition in clinically useful times, but both

techniques are really in need of yet another level of performance improvement. A parallel processing architecture is required, and this thesis has shown how one such architecture, a microprocessor-based multicomputer, can be used to provide a clinically useful superposition-based treatment planning system. However, full ray-tracing of the superposition kernel requires perhaps an order of magnitude performance improvement, and the Monte Carlo technique requires several orders of magnitude performance improvement.

What architectures are most suitable for these applications? There are three possibilities:

- *Vector processors.* It is not possible to accelerate computation of a Monte Carlo algorithm by vectorising individual particle histories, since each history is essentially serial in nature. However, Miura has succeeded in vectorising the widely-used EGS4 Monte Carlo code by developing each particle history as early as possible, then using an *event-based* algorithm.[5] Superposition is more suitable for vectorisation, since each interaction voxel is processed in essentially the same way.

- *SIMD machines.* Once again, the Monte Carlo technique is not easily implemented on architectures which have only a single instruction stream, such as array processors or massively-parallel machines. In contrast, superposition is ideally suited to this type of architecture.

- *MIMD machines.* The multiple instruction streams of multiprocessors and multicomputers make them ideal for implementation of the Monte Carlo technique, since individual particle histories can be assigned to individual processors.[6,7] This requires very little modification of existing Monte Carlo codes. Superposition is also readily implemented using a data-parallel approach, where processors run essentially the same set of instructions, but operate on different subsets (partitions) of the problem.

Of the above alternatives, vector processors and massively parallel machines are currently too expensive to be considered for use in a radiotherapy workstation. Array processors are less expensive, but implementation of the Monte Carlo algorithm is difficult on these machines. Clearly MIMD architectures offer the most promise for Monte Carlo calculations, and work in this area is already underway.[8] They also have enough flexibility to be applied the superposition algorithm. This thesis has examined one variety of MIMD system, based on a multicomputer architecture and implemented using *Inmos T800 transputers,* which has been used to implement the superposition algorithm in parallel.

Are there other more powerful microprocessors suitable for use in such a system? One possibility is the *Intel i860,* but of particular interest to this author is the next generation of transputer, the *Inmos T9000.*[9] Designed to be code-compatible with the T800, the T9000 is an extension of the transputer concept. It features faster links (100 Mbits/sec), an on-chip cache, a superscalar (pipelined) processor with "instruction grouper," a "programmable memory interface," and a "virtual channel processor." The T9000's peak performance is estimated to be 200 MIPS and 25 MFlops. Thus a network of these processors should be

capable of running the superposition code presented in this thesis at more than ten times the speed of a corresponding T800 network, without any significant modification to the code. This would make full ray-trace superposition a clinically viable dose computation algorithm when using a network of modest size, and a much larger network should be capable of performing Monte Carlo calculations for use in routine planning. Nahum[10] estimates that Monte Carlo simulation of a 10 MV beam to within 2% per voxel (of dimensions 5 mm on each side) would take approximately 400 hours on a microVAX-II — this corresponds to about 6 minutes on a network of 20 T9000 transputers.

This thesis has investigated in detail the superposition and Monte Carlo dose calculation algorithms. It has also shown how superposition can be implemented in parallel on a multicomputer, thereby producing dose distributions which are more accurate than those generated by currently available treatment planning systems. In the future lies the refinement of the superposition algorithm, and its incorporation into multicomputer- or multiprocessor-based commercial treatment planning systems. The ultimate goal must be an affordable Monte Carlo based treatment planning system, capable of delivering accurate dose distributions in clinically useful times.

References

(1) Jenkins T.M., Nelson W.R. and Rindi A. (editors), *Monte Carlo Transport of Electrons and Photons.* Plenum Press, New York (1988).

(2) Hayward E. and Hubbell J., The albedo of various materials for 1-MeV photons. *Phys. Rev.* (1954), **93 (5)** 955.

(3) Mohan R., Chui C. and Lidofsky L., Energy and angular distributions of photons from medical linear accelerators. *Med. Phys.* (1985), **12** 592.

(4) Chui C. and Mohan R., Extraction of pencil beam kernels by the deconvolution method. *Med. Phys.* (1988), **12** 138.

(5) Miura K., EGS4-V: Vectorization of the Monte Carlo cascade shower simulation code EGS4. *Comput. Phys. Comm.* (1987), **45** 127.

(6) Babb R.G., Storc L. and Hiromoto R., Developing a parallel Monte Carlo transport algorithm using large-grain data flow. *Parallel Comput.* (1988), **7** 187.

(7) Miura K. and Babb R.G., Tradeoffs in granularity and parallelization for a Monte Carlo shower simulation code. *Parallel Comput.* (1988), **8** 91.

(8) Mackie T.R., Kubsad S.S., Rogers D.W.O. and Bielajew A.F., The OMEGA Project: Electron dose planning using Monte Carlo simulation. Paper BB1 of the 32nd annual meeting of the AAPM, 1990, listed in the Addendum to Scientific Paper Abstracts and Sessions, *Med. Phys.* (1990), **17** 510. See also abstracts F20 and P25 of the 33rd annual meeting of the AAPM, *Med. Phys.* (1991), **18 (3)**.

(9) Inmos, *The T9000 Transputer Products Overview Manual.* Inmos, Bristol, U.K. (1991).

(10) Nahum A.E., Overview of photon and electron Monte Carlo. In *Monte Carlo Transport of Electrons and Photons, p3*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Appendix A

Installing EGS4 on a PC-based System

EGS4/PC comes on three 1.2 MByte flexible disks. It is designed for a 80386 based machine running Lahey FORTRAN under OS/386. In an environment other than this some modifications may have to be made. Also refer to Appendix 5 of the EGS4 User Manual during installation and testing.

Step 1. Installing the Files

Create a directory under which you wish EGS4 to reside. The directory `D:\MCARLO\` is used in the following installation notes. Copy the file **`install.bat`** from disk 1 into that directory. Edit **`install.bat`** and add a line appropriate to that directory: in this case

```
if "%1" == "D:\MCARLO\" goto have_drive
```

If another directory name is chosen then substitute it for `D:\MCARLO\` throughout this document. Under MS-DOS the command **`@echo off`** will also have to be changed to **`echo off`** in this file and all other command files. Now type

```
D> install D:\MCARLO\
```

and the system will be installed (change floppy disks as requested). Further information can be found in `D:\MCARLO\readme.doc`, as well as `D:\MCARLO\EGS4\APPENDIX\appndx8.ap8`.

Step 2. Building Mortran3

Edit the file `D:\MCARLO\MORTRAN3\makemor3.bat`. This file compiles the Mortran3 processor. Remove the **`@`** as in step 1 (if necessary) and modify the root directory line to read

```
EGS4_home = D:\MCARLO\
```

If using a compiler other than Lahey FORTRAN modify the **`f7713`** commands to the appropriate command for invoking the compiler (**`t8f %1`** in the case of 3L FORTRAN on a transputer system). Also modify the link command **`132`** (to linkt %1.bin+D:\TF2V0\frtlt8.bin+D:\TF2V0\t8harn.bin,%1.b4 for a transputer system). This should result in an **`mortran3.exe`** (or similar) file being created.

Step 3. Creating the Hex Input File for Mortran3

Edit the file `D:\MCARLO\MORTRAN3\rawtohex.bat`. Remove the `@` if necessary and modify the `EGS4_home` command as before. If not using OS/386 then change the `up` and `sold` commands to the equivalent command for executing Mortran3 (`afserver -:b mortran3.b4` for a transputer system). Then invoke `rawtohex.bat`, which should run the Mortran3 preprocessor, and produce a hex data file `mortran3.dat`. Edit `mortran3.dat` and delete the first line of the file.

Step 4. Modifying the EGS4 Batch Files

Firstly, edit `D:\MCARLO\EGS4\egs4bcom.bat`. This file is used to build an executable EGS4 code from an input user code and the EGS4 system files provided. It may be necessary to modify the `@echo off, EGS4_home, f7713, 132, up` and `sold` commands as before. If the operating system does not support the `call` statement for nesting command files then it will be necessary to include the file `D:\MCARLO\EGS4\stdconf.bat` into `egs4bcom.bat` at this point. References to `C:\` will have to be changed to `D:\MCARLO\` in `stdconf.bat` in either case.

Secondly, edit `D:\MCARLO\EGS4\egs4brun.bat`. This file runs the EGS4 code once `egs4bcom.bat` has been used to build it. The `@echo off, EGS4_home, up` and `sold` commands may need to be modified.

Step 5. Building PEGS4

The preprocessor for EGS4 (PEGS4) is distributed as both a FORTRAN and Mortran source code. Edit the file `D:\MCARLO\PEGS4\makepegs.bat`, modifying `@echo off, f7713, 132, up` and `sold` if necessary. Running this file will build the PEGS4 executable module. Alternatively, PEGS4 can be built directly from the FORTRAN source also provided. In either case, some modifications to the FORTRAN may be needed for different compilers - for example, transputer FORTRAN will not compile the `NAMELIST` and `REAL*8` FORTRAN features. Next edit `D:\MCARLO\PEGS4\pegs4b.bat`, modifying `@echo off, up` and `sold` if necessary. This command file can be used to create material data sets from user-specified input files described in appendix 3 of the EGS4 User Manual.

Step 6. Testing the System

The EGS4 system should now be capable of compiling and running an EGS4 application. For example, to build the application `D:\MCARLO\EGS4\TUTOR\tutor1.mor` the command

```
D> egs4bcom tutor1
```

should be issued when in the **TUTOR** directory (it may be necessary to modify the MS-DOS **path** statement to point to the EGS4 command files). This should produce a **tutor1.exe** (or equivalent) file, which can be executed with the command

D> egs4brun tutor1 *inputfile pegsfile*

where **tutor1** is the executable file, *inputfile* is the parameter input file ("**dummy**" if no input file is required), and *pegsfile* is the PEGS4 input data file (a comprehensive data set is available in **D:\MCARLO\PEGS4\DAT\egs4.dat**). The results which the tutor codes should produce are documented in Chapter 2, Section 3 of the EGS4 User Manual.

Appendix B

EGS4 User Code RTPCART

```
"*****************************************************************************"
"                                                                             "
"                            ****************                                 "
"                            *              *                                 "
"                            * RTPCART.MOR  *                                 "
"                            *              *                                 "
"                            ****************                                 "
"                                                                             "
"     DAVID C. MURRAY, University of Waikato, Hamilton N.Z.                   "
"     This code is based on the INHOM(P) code originally developed by         "
"     D.W.O. Rogers (see Health Physics 46 (1984) 891-914 or Nucl. Inst. and  "
"     Methods A227 (1984) 535-548)                                            "
"     The code is written in MORTRAN3 for EGS4                                "
"                                                                             "
"     This code simulates a square photon or electron beam incident on a      "
"     phantom composed of 1 or 2 materials (usually water-like and            "
"     bone-like). Phantom consists of $XMAX by $YMAX by $ZMAX Cartesian        "
"     voxels of arbitrary density. Resulting dose distribution is scored in   "
"     a similar array.                                                        "
"                                                                             "
"     Code originally written by David C. Murray, October 1989.              "
"     Subsequent areas of modification are as follows:                        "
"     16/11/89 - Columns>7 facility added to GRIDLP                           "
"     16/11/89 - Card 11 (output options) added                               "
"     27/11/89 - Added machine-dependent macros for transputer                "
"     29/11/89 - Added dose spread array and raw output                       "
"     11/12/89 - Added Fibonacci RNG for transputer                           "
"     04/03/90 - Added NCASES<10 option                                       "
"     30/03/90 - Added GMFP-weighting for DSA kernels                         "
"     10/04/90 - Added TVSTEP<VSTEP correction in LCA                         "
"     11/04/90 - Added smearing options (card 8C) for DSAs                    "
"     17/05/90 - Added total/primary/scattered dose option for DSAs           "
"     31/10/90 - Patched $FLAG1 to eliminate reference to LOCMED              "
"     13/11/90 - Patched AUSGAB to ensure photoelectrons flagged as primary   "
"     26/11/90 - Changed brem to scattered dose                               "
"                                                                             "
"     At compile time this code requires NRC4MACP.MOR from unit 8 and         "
"     NRCCAUXP.MOR from unit 10 (EGS4.MOR unit switches should be             "
"     %U2,%U10,%U3,%U8,%U4). Note that NRCCAUX.MOR contains many extra        "
"     geometry routines for cylinders which may cause linker warnings         "
;
REPLACE {$MXDATA} WITH {1}
"Default for sigma routine in EGS4.MOR at NRCC"
!INDENT F 2;
%C80   use 80 columns of input
"                                                                             "
"   Unit assignments (run time)                                               "
"           unit 1    output listing file                                     "
"           unit 5    input file (or keyboard)                                "
"           unit 6    prompts for input and echos input                       "
"           unit 8    echos media input from pegs(assign to null file)        "
"           unit 12   input file with crossection data from pegs              "
"           unit 13   output file for DOTPLOT data                            "
"           unit 14   output file for RAWDATA data                            "
"                                                                             "
"*****************************************************************************"
"   The code can be run interactively, or batched with the following input file"
;
"Card 1  title          80A1                                                  "
"Card 2  material 1     24A1                                                  "
"Card 3  material 2     24A1                                                  "
"Card 4  XSIZE,YSIZE,ZSIZE sizes of voxels in cm
;
"Card 5A XSIZE,YSIZE,ZSIZE<0                                                  "
"        MED(all voxels),RHOR(all voxels)                                     "
"        Indicates that all voxels are identical in medium and density.       "
"        This is a single input line indicating material index and density    "
"        Equivalent voxels averaged.                                          "
"Card 5B ZSIZE<0 only                                                         "
"        MED(x,y,Z),RHOR(x,y,Z)                                               "
"        Indicates that all LAYERS (same z value) are identical voxels        "
"        Following is a sequence of $ZNUMBER cards with material index (1 or 2)"
"        and density in g/cm3. Equivalent voxels averaged.                    "
"Card 5C XSIZE,YSIZE,ZSIZE>0                                                  "
"        MED(x,y,z),RHOR(x,y,z)                                               "
"        Indicates that all voxels are potentially different regions (have    "
"        different density and material index). Following are                 "
```

```
 78 "        $XMAX*$YMAX*$ZMAX cards each with material index and density                "
 79 "        MED(X,Y,Z),RHOR(X,Y,Z). Equivalent voxels NOT averaged.                     "
 80 "Card 5D XSIZE,YSIZE<0,ZSIZE>0                                                       "
 81 "        MED(phantom),RHOR(phantom)                                                  "
 82 "        MED(block),RHOR(block)                                                      "
 83 "        IXLOW,IXHIGH,IYLOW,IYHIGH,IZLOW,IZHIGH                                      "
 84 "        Indicates that a rectangular block in an otherwise homogeneous medium      "
 85 "        is to be modelled. IXLOW etc indicate voxel indices of block bounds        "
 86 ;                                                                                    "
 87 "Card 6  E(kinetic energy),ECUTIN,PCUTIN,SMAX,ESTEPE,ESAVE                           "
 88 "               E       KINETIC energy of incident beam                             "
 89 "                       for E<=0, input a spectrum on cards 6A (below)              "
 90 "               ECUTIN  electron cut off in MeV(total energy)                       "
 91 "               PCUTIN  photon cut off in MeV                                       "
 92 "               SMAX    max step size in cm - this acts as well as SMAXIR           "
 93 "               ESTEPE  fractional energy loss per electron step                    "
 94 "                       if left 0, EGS uses its own default step size               "
 95 "                       if non-zero, step size is chosen to reduce E by this       "
 96 "                       fraction                                                    "
 97 "               ESAVE   electrons below this energy if they cannot escape the      "
 98 "                       current region.                                             "
 99 "               Note: SMAX, ESTEPE, ESAVE not required for PRESTA                   "
100 "                     ESTEPE,ESAVE MUST NOT BE SUPPLIED for PRESTA                  "
101 "Card 6A if E<=0.0   input an energy spectrum                                       "
102 "        First card   EIMIN   lowest KINETIC energy in spectrum                     "
103 "        Repeat EBIN(I),EPHI(I)                                                     "
104 "                  EBIN(I)  top of energy bin I                                     "
105 "                  EPHI(I)  probability of energy being in this bin                 "
106 "                           (not necessarily normalized)                            "
107 "        End with EBIN(I) < EBIN(I-1)   or zero                                     "
108 ;                                                                                   "
109 "Card 7  ISOURC                                                                     "
110 "               =0        parallel beam source                                     "
111 "               =1        point source on axis                                     "
112 "               =2        dose spread array                                        "
113 "Card 7A (if ISOURC=0)                                                             "
114 "        XCENTRE,YCENTRE,BEAMWIDTH   beam centre and width (cm)                     "
115 "Card 7B (if ISOURC=1)                                                             "
116 "        XCENTRE,YCENTRE,BEAMWIDTH,DISTZ                                            "
117 "               XCENTRE,YCENTRE,BEAMWIDTH as above                                  "
118 "               DISTZ        distance of point source to left of phantom           "
119 "Card 7C(if ISOURC=2)                                                              "
120 "        XCENTRE,YCENTRE,ORIGINDEPTH,ISMEAR,IDOSETYPE                               "
121 "               XCENTRE,YCENTRE    x and y coords of interaction point (cm)        "
122 "               ORIGINDEPTH        z coord of interaction point (cm)               "
123 "               ISMEAR = 0 for no smearing of interaction site                     "
124 "                      = 1 for 2D (surface) smearing (electron pencil beam)        "
125 "                      = 2 for 3D smearing throughout interaction voxel            "
126 "               IDOSETYPE = 0 for scoring of total dose                            "
127 "                         = 1 for scoring of primary dose only                     "
128 "                         = 2 for scoring of scattered/brem dose only              "
129 ;                                                                                   "
130 "Card 8  NCASE,IQIN,IWATCH,TIMMAX,INSEED,IDORAY,NOMSCT,NOPLCI,ICSDA                 "
131 "               NCASE   # histories to run                                         "
132 "               IQIN    charge on incident beam                                    "
133 "               IWATCH = 0 for normal output                                       "
134 "                      = 1 to print all interactions                               "
135 "                      = 2 to print every electron step                            "
136 "               TIMMAX  maximum cpu time allowed for job in hours                  "
137 "                       job terminates with as many batches as possible within"
138 "                       this time limit. Default=1 hr.                             "
139 "               INSEED  initial seed to random number generator- 0 is ok           "
140 "               IDORAY  =1, include Rayleigh (coherent scatter) = 0, don't         "
141 "               NOMSCT  if non-zero, no multiple scattering is done, nor           "
142 "                       is a path length correction used                           "
143 "               NOPLCI  if non-zero, path-length corrections are turned off        "
144 "                       should also set IPLC = -1 in PRESTA inputs                 "
145 "               ICSDA   when =1, if material 1 (2) are for the same material      "
146 "                       and have IUNRST=2(5), then a proper CSDA calculation       "
147 "                       is done - i.e. all the brem escapes                        "
148 ;                                                                                   "
149 "Card 9  RRZ,RRCUT,CEXPTR                                                         "
150 "               For Russian roulette - as any photon crosses the Z=RRZ plane       "
151 "               Russian roulette is played with a probability of survival =        "
152 "               RRCUT - weight increases by 1/RRCUT if it survives                 "
153 "               If blank or both zero, no Russian roulette is played               "
154 "               CEXPTR is the parameter for pathlength biasing:                    "
155 "                  <0 for shortening                                               "
156 "                  if 0.0, no biasing done                                         "
157 ;                                                                                   "
158 "Card 10  (PRESTA inputs)                                                          "
159 "   IPLC,IBCA,ILCA,IOLDTM,BLCMIN                                                   "
160 "       All zeros is fine for a default PRESTA run - see the PRESTA write up       "
161 "       for details on other values possible. To obtain a pure EGS4 run,           "
162 "       insert a line   1,1,1,1. If the 4th 1 is changed to zero, the improved    "
163 "       FIXTMX algorithm is used. If NOPLCI is non-zero, i.e. then IPLC            "
164 "       should be -1 to signal no PLC is to be used.  NOPLCI will over-ride        "
165 "       and set IPLC to -1                                                         "
166 ;                                                                                   "
167 "Card 11 (Output options)                                                          "
168 "   PLOTIT,MASSFLAG,IDOTPLOT                                                       "
169 "               PLOTIT=0  dose table turned off                                    "
170 "                     =1  dose table turned on                                     "
171 "               MASSFLAG=0  region masses table turned off                         "
```

```
172 "                          =1  region masses table turned on                    "
173 "                  IDOTPLOT    1 = output IQ,X,Z to unit IDOTFILE               "
174 "                              2 = output IQ,X,Z,U,W,USTEP to unit IDOTFILE     "
175 "                  IRAWDATA    1 = output raw dose data to unit IRAWFILE,       "
176 "                                with $XMAX,$YMAX,$ZMAX,XSIZE,YSIZE,ZSIZE,      "
177 "                                origin voxel z index (starting at 0), one      "
178 "                                dose per line, (X1,Y1,Z1),(X2,Y1,Z1)...        "
179 "                                (Xn,Yn,Zm) order                               "
180 "                              2 = output central axis (x,z) plane only         "
181 "*******************************************************************************"
182 " Some COMMON variables:                                                        "
183 "                                                                               "
184 "COMMON/SCORE/                                                                  "
185 "        MXNP                  maximum value of stack pointer-checked in AUSGAB "
186 "        DOSEIS($XMAX,$YMAX,$ZMAX,IS) the energy in MeV deposited in each       "
187 "                                region for batch IS.                           "
188 "        IWATCH            =0 normal output                                     "
189 "                          =1 prints detail of every discrete interaction       "
190 "                          =2 print detail of every step taken                  "
191 "        IS                current batch number                                 "
192 "        XSIZE             size of x voxel increment in cm                      "
193 "        YSIZE             size of y voxel increment in cm                      "
194 "        ZSIZE             size of z voxel increment in cm                      "
195 "        ISTEPA            # electron steps taken - not including discards     "
196 "        IQIN              charge of incident particle                          "
197 "        IDOSETYPE         total/primary/scattered dose scoring for DSAs        "
198 ;
199 "COMMON/GEOM/                                                                   "
200 "        XSIZE,YSIZE,ZSIZE voxel dimensions                                     "
201 "        NREG              number of regions+2                                  "
202 "                                                                               "
203 " Some non-COMMON variables:                                                   "
204 "        AINFLU            incident fluence                                     "
205 "                          parallel beam  = NCASE/area of beam                  "
206 "                          point source =NCASE/area of sphere -i.e. larger      "
207 "                          dose spread array = NCASE                            "
208 "        AMASS($ZMAX,$XMAX)  mass of scoring region (z,x) in plotting plane     "
209 "        DOSEIT($ZMAX,$XMAX) dose deposited in plotting plane                   "
210 "        DOSE($XMAX,$YMAX,$ZMAX)-average energy dep. per batch in each zone     "
211 "                          scored as MeV per batch but converted to rad per     "
212 "                          unit incident energy fluence                         "
213 "        UNCER($XMAX,$YMAX,$ZMAX)  fractional uncertainty on dose-calculated    "
214 "                          as uncertainty on mean from 10 batches               "
215 "        PLOTIT            flag for not plotting/plotting doses                 "
216 "        MASSFLAG          flag for not plotting/plotting masses                "
217 "        IDOTPLOT          flag for not plotting/plotting interactions          "
218 "        IDOTFILE          unit number for DOTPLOT output                       "
219 "        IRAWDATA          flag for outputting raw dose data                    "
220 "        IRAWFILE          unit number for raw data output                      "
221 "        IVOXAVG           flag for controlling voxel averaging                 "
222 "        LNGTH             source to surface (oblique) dist for square field    "
223 "        XCENTRE           x coordinate of beam centre                          "
224 "        YCENTRE           y coordinate of beam centre                          "
225 "        ORIGINDEPTH       depth of origin below phantom surface for DSA        "
226 "        TOTALMEV          total incident K.E.                                  "
227 "        PHANTOMMEV        total dose scored in phantom                         "
228 "        ISMEAR            flag for controlling interaction voxel smearing      "
229 ;
230 "*******************************************************************************"
231 " Installation specific routines:                                              "
232 "        This code was developed on a Vax system using Vax Fortran 77.          "
233 "        Non-standard features used include variable-format I/O, time, date and"
234 "        timing routines. Comment out lines with reference to:                  "
235 "        TIMEN, DATEN, SECNDS, or CPUTIME                                       "
236 "        T=SECNDS(TO)      returns time in seconds since TO                     "
237 "        CALL CPUTIME(IT) returns IT(INTEGER*4)=task cpu time in 10 millisec    "
238 "        Note that the code assumes that the NRCC4 macros are used and that     "
239 "        the routines WATCH and FIXTMX are linked with it                       "
240 "        PRESTA override of PRESTA-MSCAT-1 also creates a linker warning        "
241 ;
242 "*******************************************************************************"
243 " User set parameter values                                                     "
244 %I;
245 PARAMETER         $MXMED = 2 ;       "#media"
246 PARAMETER         $STAT = 5 ;        "# bins for uncertainty analysis"
247 PARAMETER         $MXSTACK = 20 ;    "maximum stack size"
248 PARAMETER         $XMAX = 28;        "x,y,z scoring region voxel number limits"
249 PARAMETER         $YMAX = 25;
250 PARAMETER         $ZMAX = 16;
251 PARAMETER         $MXREG = {COMPUTE $XMAX*$YMAX*$ZMAX+2};
252                                      "1 per voxel + 1 front and back"
253 PARAMETER         $COLNO = 6 ;       "number of columns for GRIDLP"
254                                      "must also change <> value in GRIDLP formats"
255 PARAMETER         $SMALLDIST = 0.00001 ;  "round-off limit for geometry"
256                                           "calculations in HOWFAR"
257 %I4
258 ;
259 "*******************************************************************************"
260 " Common block definitions and macro replacements                               "
261 "IDOTPLOT,IDOTFILE added to flag DOTPLOT listing"
262 "IDOSETYPE added for total/primary/scattered DSA scoring"
263 REPLACE {;COMIN/SCORE/;} WITH
264 {;COMMON/SCORE/MXNP,DOSEIS($XMAX,$YMAX,$ZMAX,$STAT),IWATCH,
```

```
265 IS,ISTEPA,IQIN,ICSDA,IDOTPLOT,IDOTFILE,IDOSETYPE;}
266
267 REPLACE {;COMIN/GEOM/;} WITH {;COMMON/GEOM/XSIZE,YSIZE,ZSIZE,NREG;}
268
269 REPLACE {;COMIN/ESPECT/;} WITH
270 {;COMMON/ESPECT/ECPD(50),EBIN(50),EPHI(50),EIMIN;}
271 "Added to input an energy spectrum for the source. ECPD(I) is the probability  "
272 "of the source energy being less than EBIN(I)  - i.e. cumulative probability   "
273 "distribution.
274
275 REPLACE {;COMIN/RUSROU/;} WITH
276 {;LOGICAL RUSROU;COMMON/RUSROU/RRZ,RRCUT,RUSROU;}
277 "RUSROU is the common for the Russian roulette in AUSGAB/HOWFAR"
278
279 "The following macro is needed for the PRESTA implementation in this geometry. "
280 "It is only called in ELECTR and it assumes COMMON GEOM is available and IRL   "
281 "has been set to IR(NP).                                                       "
282 REPLACE {$CALL-HOWNEAR(*);} WITH
283 {;IF(IRL > 1)[
284   XTEMP=AMOD(X(NP),XSIZE);
285   YTEMP=AMOD(Y(NP),YSIZE);
286   ZTEMP=AMOD(Z(NP),ZSIZE);
287   {P1} = AMIN1(XTEMP,XSIZE-XTEMP,YTEMP,YSIZE-YTEMP,ZTEMP,ZSIZE-ZTEMP);];}
288
289 "The following macro replaces NRCC version. It allows forcing at the origin    "
290 "(for dose spread arrays) if ISOURC=2, with pathlength biasing -CEXPTR          "
291 "It does an exponential transformation of the photon pathlength for             "
292 "forward-going photons.                                                        "
293 REPLACE {$SELECT-PHOTON-MFP;} WITH
294 {$RANDOMSET RNNO35;  IF(RNNO35 = 0.0)[RNNO35=1.E-30;]
295 DPMFP=-ALOG(RNNO35);
296 IF ((ISOURC.EQ.2).AND.(U(NP).EQ.0.0).AND.(V(NP).EQ.0.0).AND.(W(NP).EQ.1.0))
297    [DPMFP=0.0;]
298 ELSEIF(CEXPTR.NE.0.0)[IF(W(NP)>0.0)[TEMP=CEXPTR*W(NP);BEXPTR=1./(1.-TEMP);
299    DPMFP=DPMFP*BEXPTR;WT(NP)=WT(NP)*BEXPTR*EXP(-DPMFP*TEMP);]];}
300
301 "PRESTA LCA sometimes fails due to TVSTEP<VSTEP (rounding error). This         "
302 "macro replaces the one in NRC4MACP.MOR, setting TVSTEP=VSTEP if necessary.    "
303 "Modified by David Murray 1990.                                                "
304 REPLACE {$PRESTA-LCA;} WITH
305    {;IF((ILCA.EQ.0).AND.(ISHRTN.NE.0))[
306       "Modify VSTEP if necessary"
307       IF(TVSTEP<VSTEP)[VSTEP=TVSTEP;]
308       TVSTP2=TVSTEP**2;VSTEP2=VSTEP**2;
309       DELR1=0.5*TVSTEP*SINTHE;
310       VTEST2=VSTEP2+DELR1**2;
311       IF(VTEST2.GT.TVSTP2)DELR1=SQRT(TVSTP2-VSTEP2);
312       X(NP)=X(NP)+DELR1*UTRANS;
313       Y(NP)=Y(NP)+DELR1*VTRANS;
314       Z(NP)=Z(NP)+DELR1*WTRANS;];}
315
316 "The following macros control machine-dependent code. Opts VAX and TRANSPUTER  "
317 "Edit the following lines, re GENERATE and NOGENERATE commands"
318 REPLACE {$VAX;} WITH {GENERATE;}
319 REPLACE {$TRANSPUTER;} WITH {NOGENERATE;}
320
321 "VAX-only macros"
322 $VAX;
323   REPLACE {$MACHINE} WITH {'VAX'}
324   REPLACE {$TPUTER-RNG-INIT;} WITH {CONTINUE;}
325 ENDGENERATE;
326
327 "TRANSPUTER-only macros"
328 $TRANSPUTER;
329   REPLACE {CALL CPUTIME(*);} WITH {CALL ICLOCK({P1});{P1}={P1}*100;}
330   REPLACE {CALL TIME(*);} WITH {CONTINUE;}
331   REPLACE {CALL DATE(*);} WITH {CONTINUE;}
332   REPLACE {SECNDS(*)} WITH {0;}
333   REPLACE {LOGICAL*1} WITH {LOGICAL}
334   REPLACE {INTEGER*2} WITH {INTEGER}
335   REPLACE {INTEGER*4} WITH {INTEGER}
336   REPLACE {REAL*4} WITH {REAL}
337   REPLACE {REAL*8} WITH {DOUBLE PRECISION}
338   REPLACE {CALL EXIT;} WITH {STOP;}
339   REPLACE {EXIT;} WITH {STOP;}
340   "Reduce total static memory size if necessary"
341   PARAMETER $MAXIT = 3;
342   "Replace RNG with Fibonacci sequence"
343   REPLACE {$TPUTER-RNG-INIT;} WITH {$FIBONACCI-RNG-INIT;}
344   REPLACE {$RANDOMSET#;} WITH {$FIBONACCI-GENERATOR{P1};}
345   REPLACE {;COMIN/RANDOM/;} WITH
346           {;COMMON/RANDOM/URNG(97),CRNG,CDRNG,CMRNG,IRNG,JRNG,IXX;}
347   REPLACE {;COMMON/RANDOM/IXX;} WITH
348           {;COMMON/RANDOM/URNG(97),CRNG,CDRNG,CMRNG,IRNG,JRNG,IXX;}
349   "Deactivate graph plotting options"
350   REPLACE {CALL PLOTA2(*)} WITH {OUTPUT;(' PLOTA2 unavailable on transputer');}
351   REPLACE {CALL PLOTA4(*)} WITH {OUTPUT;(' PLOTA4 unavailable on transputer');}
352   REPLACE {,T<*>,} WITH {,T1,}
353   REPLACE {<IWIDTH>} WITH {1}
354   "Avoid linker warnings by replacing with known variables"
355   REPLACE {CYRAD2(*)} WITH {X(NP)}
356   REPLACE {ZPLANE(*)} WITH {X(NP)}
357   "Remove carriage control formats on output statements"
358   REPLACE {OUTPUT;('$#);} WITH {OUTPUT;('{P1});}
```

```
359    REPLACE {OUTPUT;('O#');} WITH {OUTPUT;(' ');OUTPUT;('{P1}');}
360    REPLACE {OUTPUT;(' #');} WITH {OUTPUT;('{P1}');}
361    "Indicate machine"
362    REPLACE {$MACHINE} WITH {'TRANSPUTER'}
363  ENDGENERATE;
364
365  "Fibonacci RNG initialising routine. IRNGSEED is set to IXX, while the "
366  "other three seeds are set to the values below"
367  REPLACE {$FIBONACCI-RNG-INIT;} WITH
368    {
369    "Initialise transputer RNG"
370    "IRNGSEED,JRNGSEED,KRNGSEED,LRNGSEED are RNG seeds"
371    JRNGSEED=30;KRNGSEED=61;LRNGSEED=121;
372    "Check IXX is in range 1..168"
373    IF (IXX.GT.168) [IXX = MOD(IXX,168)+1;]
374    IF (IXX.LT.1)   [IXX = MOD(-IXX,168)+1;]
375    IRNGSEED=IXX;
376    DO II=1,97[
377      SRNG=0.0;TRNG=0.5;
378      DO JJ=1,24[
379        MRNG=MOD(MOD(IRNGSEED*JRNGSEED,179)*KRNGSEED,179);
380        IRNGSEED=JRNGSEED;JRNGSEED=KRNGSEED;KRNGSEED=MRNG;
381        LRNGSEED=MOD(53*LRNGSEED+1,169);
382        IF (MOD(LRNGSEED*MRNG,64).GE.32) [SRNG=SRNG+TRNG;]
383        TRNG=TRNG*0.5;
384      ]
385      URNG(II)=SRNG;
386    ]
387    CRNG=362436.0/16777216.0;
388    CDRNG=7654321.0/16777216.0;
389    CMRNG=16777213.0/16777216.0;
390    "Set IRNG,JRNG to 97,33 for first call to RNG"
391    IRNG = 97;JRNG = 33;
392    }
393  ;
394  "Fibonacci RNG. After generation of the random number IXX is set to that "
395  "value so that the user will see different 'seeds'"
396  "This generator requires 4 initial seeds to generate a further 97 seeds. "
397  "It has a very long cycle"
398  REPLACE {$FIBONACCI-GENERATOR#;} WITH
399    {
400    {P1} = URNG(IRNG)-URNG(JRNG);
401    IF ({P1}.LT.0.0)[{P1}={P1}+1.0;]
402    URNG(IRNG) = {P1};
403    IRNG = IRNG - 1;
404    IF (IRNG.EQ.0)[IRNG=97;]
405    JRNG = JRNG - 1;
406    IF (JRNG.EQ.0)[JRNG=97;]
407    CRNG=CRNG-CDRNG;
408    IF (CRNG.LT.0.0)[CRNG=CRNG+CMRNG;]
409    {P1} = {P1} - CRNG;
410    IF ({P1}.LT.0.0)[{P1}={P1}+1.0;]
411    "Debugging option"
412    "IF (({P1}.LT.0.0) .OR. ({P1}.GT.1.0))[error - log values          "
413    "  OUTPUT {P1},CRNG,CDRNG,CMRNG,IRNG,JRNG,URNG(IRNG+1),URNG(JRNG+1);"
414    "  (' RN,CRNG,CDRNG,CMRNG,IRNG,JRNG,URNG(IRNG+1),URNG(JRNG+1) = ',  "
415    "  F12.8,F12.8,F12.8,F12.8,I12,I12,F12.8,F12.8);]                    "
416    IXX = INT({P1}*10000.0);  "update seed for printout"
417    }
418  ;
419  "*************************************************************************"
420  " Data declarations"
421  LOGICAL*1  DATEN(9),TIMEN(8),TITLE(80); "Vax specific variables"
422  INTEGER*4 CPUTO,CPUT1,CPUT2,CPUT3,CPUT4;"CPU time counters"
423  REAL*4 UNCER($XMAX,$YMAX,$ZMAX),AMASS($ZMAX,$XMAX),
424  DOSE($XMAX,$YMAX,$ZMAX),DOSEIT($ZMAX,$XMAX),UNCERT($ZMAX,$XMAX);
425  REAL*8 TOTALMEV;     "Total incident K.E."
426  REAL*8 PHANTOMMEV;   "Total medium K.E. (for DSAs)"
427  REAL*4 ORIGINDEPTH;  "DSA forced interaction depth"
428  REAL*4 BEAMWIDTH;    "Width of square beam"
429  REAL*4 XCENTRE;      "x position (cm) of beam centre"
430  REAL*4 YCENTRE;      "y position (cm) of beam centre"
431  REAL*4 DISTZ;        "Distance from point source to phantom"
432  INTEGER*4 IVOXAVG;   "Flag for controlling voxel averaging"
433  INTEGER*4 PLOTIT,MASSFLAG,IRAWDATA,IRAWFILE; "Flags for controlling output"
434  INTEGER*4 ISMEAR;    "Flag for interaction voxel smearing"
435  INTEGER*4 IPARTNO;   "Flag for particle splitting near beam axis"
436
437  "Note comin PHOTIN added to allow access to GMFP for DSA calcs"
438  COMIN/EPCONT,MEDIA,GEOM,SCORE,THRESH,BOUNDS,MISC,ELECIN,RANDOM,
439  STACK,MULTS,RUSROU,USER,USEFUL,ESPECT,PHOTIN/;
440  ;
441  "*************************************************************************"
442  " MAIN code starts here"
443  CALL CPUTIME(CPUTO);"initial cpu time taken by this task so far"
444  OUTPUT; (' EGS4 Monte Carlo Simulation using code RTPCART');
445  OUTPUT; (' University of Waikato, Hamilton, New Zealand.');
446  OUTPUT $MACHINE ; (' Running on a ',A,'...');  "indicate machine"
447
448  "Open files for transputer only - VAX files are preconnected"
449  $TRANSPUTER;
450    OPEN(UNIT=1,FILE='FOR001.DAT',STATUS='UNKNOWN'); "program output"
451    OPEN(UNIT=3,FILE='FOR003.DAT',STATUS='UNKNOWN'); "plot file output"
452    OPEN(UNIT=5,FILE='FOR005.INP',STATUS='OLD');     "parameter input"
```

```
453   OPEN(UNIT=7,STATUS='SCRATCH');                    "echos media input"
454   OPEN(UNIT=8,STATUS='SCRATCH');                    "echos media input"
455   OPEN(UNIT=12,FILE='FORO12.INP',STATUS='OLD');     "PEGS input data"
456   OPEN(UNIT=13,FILE='FORO13.DAT',STATUS='UNKNOWN');"DOTPLOT output"
457   OPEN(UNIT=14,FILE='FORO14.DAT',STATUS='UNKNOWN');"RAWDATA output"
458 ENDGENERATE;
459
460 " Read inputs"
461 "Card 1"
462 OUTPUT;(/'$ Title: ');
463 READ(5,100,END=:END:)TITLE;OUTPUT TITLE;(1X,80A1);
464
465 "Card 2"
466 OUTPUT;(/'$ Material 1: ');READ(5,120,END=:END:)(MEDIA(J,1),J=1,24);
467 OUTPUT(MEDIA(J,1),J=1,24);('+',24A1);
468
469 "Card 3"
470 OUTPUT;(/'$ Material 2: ');READ(5,120,END=:END:)(MEDIA(J,2),J=1,24);
471 OUTPUT(MEDIA(J,2),J=1,24);('+',24A1);
472
473 "Card 4"
474 OUTPUT;(/'$ x voxel size, y voxel size, z voxel size (cm): ');
475 READ(5,105,END=:END:)XSIZE,YSIZE,ZSIZE;
476 OUTPUT XSIZE,YSIZE,ZSIZE;('+',3F9.4);
477 NMED=1;"assume only one material needed unless second asked for on card 5   "
478
479 "Card 5A XSIZE,YSIZE,ZSIZE<0"
480 IF(XSIZE<0.0 & YSIZE<0.0 & ZSIZE<0.0)[
481   IVOXAVG = 1; "Perform voxel averaging"
482   OUTPUT;('  ...VOXEL AVERAGING will be performed');
483   OUTPUT;('$ MED,RHOR for all voxels: ');
484   INPUT IMED,XRHOR;(I5,F12.0);
485   IF(IMED = 0)[IMED=1;]
486   IF(IMED= 2)[NMED=2;"WILL NEED 2ND MATERIAL"]
487   IF(XRHOR<=0.0)[OUTPUT IZ;('0*****DENSITY<=0 READ ERROR -LAYER',I5);]
488   OUTPUT IMED,XRHOR;('+',I5,F9.4);
489   DO IZ=1,$ZMAX[
490     DO IX=1,$XMAX[
491       DO IY=1,$YMAX[
492         "Note: Add 1 for region to left of plate"
493         MED((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = IMED;
494         RHOR((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = XRHOR;
495 ] ] ] ]
496
497 "Card 5B ZSIZE<0 only"
498 IF(XSIZE>0.0 & YSIZE>0.0 & ZSIZE<0.0)[
499   IVOXAVG = 1; "Perform voxel averaging"
500   OUTPUT;('  ...VOXEL AVERAGING will be performed');
501   OUTPUT;('  MED,RHOR for each z layer of voxels: ');
502   DO IZ=1,$ZMAX[
503     INPUT IMED,XRHOR;(I5,F12.0);
504     IF(IMED = 0)[IMED=1;]
505     IF(IMED= 2)[NMED=2;"WILL NEED 2ND MATERIAL"]
506     IF(XRHOR<=0.0)[OUTPUT IZ;('0*****DENSITY<=0 READ ERROR -LAYER',I5);]
507     "Note: Add 1 for region to left of plate"
508     DO IX=1,$XMAX[
509       DO IY=1,$YMAX[
510         MED((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = IMED;
511         RHOR((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = XRHOR;
512 ] ] ] ]
513
514 "Card 5C ZSIZE>0"
515 IF(XSIZE>0.0 & YSIZE>0.0 & ZSIZE>0.0)[
516   IVOXAVG = 0; "DON'T perform voxel averaging"
517   OUTPUT;('  ...NO VOXEL AVERAGING will be performed');
518   OUTPUT;('  MED,RHOR for each voxel(x varying fastest): ');
519   DO IZ=1,$ZMAX[
520     DO IY=1,$YMAX[
521       DO IX=1,$XMAX[
522         INPUT IMED,XRHOR;(I5,F12.0);
523         IF(IMED = 0)[IMED=1;]
524         IF(IMED= 2)[NMED=2;"WILL NEED 2ND MATERIAL"]
525         IF(XRHOR<=0.0)[OUTPUT IZ;('0*****DENSITY<=0 READ ERROR -LAYER',I5);]
526         "Note: Add 1 for region to left of plate"
527         MED((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = IMED;
528         RHOR((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = XRHOR;
529 ] ] ] ]
530
531 "Card 5D XSIZE,YSIZE<0,ZSIZE>0 - block simulation"
532 IF(XSIZE<0.0 & YSIZE<0.0 & ZSIZE>0.0)[
533   IVOXAVG = 0; "DON'T perform voxel averaging"
534   OUTPUT;('  ...NO VOXEL AVERAGING will be performed');
535   OUTPUT;('  Block simulation');
536   OUTPUT;('$ MED,RHOR for all phantom voxels: ');
537   INPUT IMED,XRHOR;(I5,F12.0);
538   IF(IMED = 0)[IMED=1;]
539   IF(IMED= 2)[NMED=2;"WILL NEED 2ND MATERIAL"]
540   IF(XRHOR<=0.0)[OUTPUT IZ;('0*****DENSITY<=0 READ ERROR -LAYER',I5);]
541   OUTPUT IMED,XRHOR;('+',I5,F9.4);
542   DO IZ=1,$ZMAX[
543     DO IY=1,$YMAX[
544       DO IX=1,$XMAX[
545         "Note: Add 1 for region to left of plate"
```

```
546         MED((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = IMED;
547         RHOR((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = XRHOR;
548   ] ] ]
549   "Block is in region IX=IXLOW,IXHIGH,IY=IYLOW,IYHIGH,IZ=IZLOW,IZHIGH"
550   OUTPUT;('$ MED,RHOR for all voxels in block: ');
551   INPUT IMED,XRHOR;(I5,F12.0);
552   IF(IMED = 0)[IMED=1;]
553   IF(IMED= 2)[NMED=2;"WILL NEED 2ND MATERIAL"]
554   IF(XRHOR<=0.0)[OUTPUT IZ;('0*****DENSITY<=0 READ ERROR -LAYER',I5);]
555   OUTPUT IMED,XRHOR;('+',I5,F9.4);
556   OUTPUT;(' Block low x index,high x,low y,high y,low z,high z: ');
557   INPUT IXLOW,IXHIGH,IYLOW,IYHIGH,IZLOW,IZHIGH;(6I3);
558   OUTPUT IXLOW,IXHIGH,IYLOW,IYHIGH,IZLOW,IZHIGH;(6I3);
559   DO IZ=IZLOW,IZHIGH[
560     DO IY=IYLOW,IYHIGH[
561       DO IX=IXLOW,IXHIGH[
562         "Note: Add 1 for region to left of plate"
563         MED((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = IMED;
564         RHOR((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1) = XRHOR;
565 ] ] ] ]
566 "All Card 5's - set XSIZE, YSIZE, ZSIZE positive"
567 IF (XSIZE<0.0)[XSIZE = -XSIZE;]
568 IF (YSIZE<0.0)[YSIZE = -YSIZE;]
569 IF (ZSIZE<0.0)[ZSIZE = -ZSIZE;]
570 "All Card 5's - set number of regions to $MXREG"
571 NREG=$MXREG;
572
573 "Card 6"
574 OUTPUT;(/'$ Beam energy,ECUT,PCUT,SMAX,ESTEPE,ESAVE: ');
575 READ(5,110) EIN,ECUTIN,PCUTIN,SMAX,ESTEPE,ESAVIN;
576 IF(SMAX <= 0.0)["set SMAX default to bin width"
577   DO IZ=2,$MXREG-1[
578     IF(SMAXIR(IZ) =  0.0)[SMAXIR(IZ)=AMIN1(XSIZE,YSIZE,ZSIZE);]
579   ]
580   SMAX=1.E10;
581 ]
582 ELSE ["set SMAXIR values to SMAX if it is more restrictive"
583   DO IZ=2,$MXREG-1[
584     IF(SMAXIR(IZ)=0.0) [SMAXIR(IZ) = SMAX;]
585     ELSE [SMAXIR(IZ)=AMIN1(SMAXIR(IZ),SMAX);]
586   ]
587 ]
588 OUTPUT EIN,ECUTIN,PCUTIN,SMAX,ESTEPE,ESAVIN;(3F10.3,1PE12.3,0PF10.4,F10.3);
589 "Set default energy cutoffs where needed"
590 ECUTMN = 1.E10;   "need to find minimum ECUT used for PRESTA"
591 DO I=2,$MXREG[
592   IF(ECUT(I) = 0.0)[ECUT(I)=ECUTIN;]
593   ECUTMN = MIN(ECUTMN,ECUT(I));
594   IF(PCUT(I) = 0.0)[PCUT(I)=PCUTIN;]
595   IF(ESAVE(I) = 0.0)[ESAVE(I)=ESAVIN;]
596 ]
597
598 "Card 6A   only if EIN<=0.0 which means input spectrum"
599 IF(EIN<=0.0)[
600   OUTPUT;(/'  Input tops of energy bins and bin probabilities,'/
601   '  one pair per line, lowest energy first - first line EIMIN only,'/
602   '  ending with bin top energy lower than previous');
603   I=0; INPUT EIMIN;(F10.0);"lowest energy in spectrum"
604   LOOP [I=I+1;
605     INPUT EBIN(I),EPHI(I);(2F10.0);
606     IF(I.NE.1 .AND. EBIN(I).LE.EBIN(I-1))[EXIT;"end of input spectrum"]
607     ELSEIF (I.EQ.1 .AND. EBIN(I).LE.EIMIN) [EXIT;"error probably"]
608   ]
609   NEBIN=I-1; "number of energy bins read in"
610   TOTPHI=0.0;DO I=1,NEBIN [TOTPHI=TOTPHI+EPHI(I);]"spectrum total"
611   ECPD(1) = EPHI(1)/TOTPHI;
612   DO I=2,NEBIN [ ECPD(I) = ECPD(I-1) + EPHI(I)/TOTPHI;]
613   OUTPUT (I,EBIN(I),EPHI(I),ECPD(I),I=1,NEBIN);
614   ('  Bin#   Energy top    Probability   Cumulative prob'/(I4,F10.3,2F16.4));
615 ]"end of input energy spectrum"
616
617 "Card 7"
618 OUTPUT;
619 (/'$ Parallel beam(0),Point source on axis(1),Dose spread array(2): ');
620 READ(5,130) ISOURC; OUTPUT ISOURC;('+',I2);
621
622 "Card 7A"
623 IF(ISOURC = 0)["parallel beam input"
624   OUTPUT;(/'$ X,Y centre of beam, Beam width: ');
625   READ(5,110)XCENTRE,YCENTRE,BEAMWIDTH;
626   OUTPUT XCENTRE,YCENTRE,BEAMWIDTH;('+',3F12.3);
627 ]
628 "Card 7B"
629 IF(ISOURC = 1)["source on axis"
630   OUTPUT;
631   (/'$ X,Y centre of beam, Beam width, Distance from top of phantom: ');
632   READ(5,110)XCENTRE,YCENTRE,BEAMWIDTH,DISTZ;
633   OUTPUT XCENTRE,YCENTRE,BEAMWIDTH,DISTZ;(4F12.3);
634 ]
635 "Card 7C"
636 IF(ISOURC = 2)["dose spread array input"
637   OUTPUT;
638   (/'$ X,Y int. centre, Int. depth, smear flag (0,1,2), Dose type (0,1,2):');
```

```
639   READ(5,111,END=:END:)XCENTRE,YCENTRE,ORIGINDEPTH,ISMEAR,IDOSETYPE;
640   OUTPUT XCENTRE,YCENTRE,ORIGINDEPTH,ISMEAR,IDOSETYPE;(3F12.3,2I6);
641 ]
642
643 "Card 8"
644 OUTPUT;
645 (/'$ #Hists,IQIN,IWATCH,TIMMAX(HR),INSEED,IDORAY,NOMSCI,NOPLCI,ICSDA');
646 READ(5,130,END=:END:)NCASE,IQIN,IWATCH,TIMMAX,INSEED,
647 IDORAY,NOMSCI,NOPLCI,ICSDA;
648 IF(TIMMAX = 0.0)[TIMMAX=1.0;"default time limit is 1 hour"]
649 IF(INSEED ~= 0)[IXX=INSEED;"IXX from COMIN RANDOM- also init. in HATCH"]
650 OUTPUT NCASE,IQIN,IWATCH,TIMMAX,INSEED,IDORAY,NOMSCI,NOPLCI,ICSDA;
651 (1X,I8,2I6,F8.2,' HRS',I12,8I6);
652 IF(IDORAY ~= 0)[DO J=1,$MXREG[IRAYLR(J)=1;]]
653 IF(IWATCH = 0)["don't call AUSGAB unless needed"
654   DO J=6,25[IAUSFL(J)=0;]
655 ]
656 "The call WATCH(-99,IWATCH) changes these below for IWATCH non-zero"
657 IF(IQIN = 0)["incident photons, we need to call AUSGAB before"
658              "P.E. and Pair and after Compton events (IARG=19,18,15)"
659   /IAUSFL(20),IAUSFL(19),IAUSFL(16)/=1;
660 ]
661 ELSEIF(IQIN = -1) ["for incident electrons we want to call AUSGAB"
662                    "after all brem and moller events(IARG=7&9)"
663   /IAUSFL(8),IAUSFL(10)/=1;
664 ]
665 IF(NOMSCI ~= 0) ["turn off multiple scattering and PLC"
666   "changed slightly to handle PRESTA"
667   DO J=1,$MXREG [/NOMSCT(J),NOPLC(J)/=-1; "we turn off PLC too"]
668 ]
669 ELSE [DO J=1,$MXREG [ NOMSCT(J)=0;]]
670 IF(NOPLCI ~= 0)[DO J=1,$MXREG[NOPLC(J) = -1;"-1 flags PRESTA for no PLC"]]
671 "note the patch below $PRESTA-INPUTS which ensures this def'n holds"
672
673 "Card 9"
674 OUTPUT;(/'$ Russian roulette: Z plane,Survival prob, Pathlength param: ');
675 READ(5,110,END=:END:) RRZ,RRCUT,CEXPTR;
676 OUTPUT RRZ,RRCUT,CEXPTR;(3F12.4);
677 RUSROU=.FALSE.;IF(RRZ+RRCUT ~= 0.0)[RUSROU=.TRUE.;]
678
679 "Card 10    PRESTA inputs"
680 IF(IQIN = 0)[EI=EIN;]ELSE[EI=EIN+0.511;]"get total energy"
681 "PRESTA requires the max K.E. of electron"
682 IF(EIN <= 0.0) [EKO = EBIN(NEBIN);] ELSE [EKO = EIN;]
683 "Initialise media and call HATCH"
684 /MED(1),MED($MXREG)/=0; "Vacuum outside phantom"
685 DUNIT=1; "units are cm"
686 IF(ICSDA.EQ.1) [NMED=2;"radiative stopping power also needed, so set NMED=2"]
687 CALL TIME(TIMEN);"time of day (VMS)"
688 OUTPUT TIMEN;(/'  Call to HATCH at ',8A1,'...');
689 CALL HATCH;
690 CALL TIME(TIMEN);OUTPUT TIMEN;('  ...HATCH completed at ',8A1);
691 "Check media data covers energy range requested"
692 DO I=1,NMED[
693   IF(EIN > UP(I)   |  EIN > UE(I)-0.511)[
694     OUTPUT I,EIN,UP(I),UE(I);(//1X,80('*')//' FOR MEDIUM',
695     I3,' INCIDENT ENERGY=',F10.3,' MEV'/' IS GREATER THAN',
696     ' COVERED BY DATA FILE WHERE UP,UE=',2F10.3,' MEV'//
697     1X,80('*')//);
698   ]
699   "Check cross sections are available for rayleigh scattering if requested"
700   IF(IDORAY ~= 0  &  IRAYLM(I) ~= 1)["data not there"
701     OUTPUT I;(//1X,80('*')/'ORAYLEIGH SCATTERING DATA NOT THERE FOR',
702     ' MATERIAL',I3,'  GIVE UP');
703 ] ] "end loop over media"
704 $PRESTA-INPUTS;
705 "Warn user if PRESTA selected and ESTEPE values also specified"
706 OUTPUT ESTEPE;(' ESTEPE = ',F12.4);
707 IF(IPLC=0 & ESTEPE~=1.0)[
708   OUTPUT;(' WARNING - both PRESTA and ESTEPE specified - resetting ESTEPE');
709   ESTEPE = 1.0;
710 ]
711 "In case IPLC was not set to -1 when the user requested NOPLC,redo here"
712 IF(NOPLCI ~= 0) [IPLC = -1;DO I = 1,NREG [NOPLC(I) = -1;]]
713 "Change electron step sizes if requested"
714 IF(ESTEPE > 0.0)["Replace EGS4 values of TMXS with ETRAN like constant "
715   "fractional energy loss steps -note 200*TEFF0 restrictions still active"
716   DO I=1,NMED[CALL FIXTMX(ESTEPE,I);"do for all media being used"]
717 ]
718
719 "Card 11       Output options                                          "
720 OUTPUT;('$ PLOTIT,MASSFLAG,IDOTPLOT,IRAWDATA: ');
721 READ(5,115,END=:END:) PLOTIT,MASSFLAG,IDOTPLOT,IRAWDATA;
722 OUTPUT PLOTIT,MASSFLAG,IDOTPLOT,IRAWDATA;('+',I3,I3,I3,I3);
723 ;
724 "*************************************************************************"
725 " Initialize variables"
726 IOUT=1; "unit for output listing"
727 IDOTFILE=13;"output listing unit for DOTPLOT file"
728 IRAWFILE=14;"output listing unit for RAWDATA file"
729 TOTALMEV=0.0; "initialise total incident kinetic energy"
730 IRIN=1; "source is in region 1"
731 WTIN=1.;"initial weight always 1"
```

```
732 IPARTWO=1;"default is no particle splitting"
733 MXSP=0; "initialise current maximum value of stack pointer"
734 "Ensure JCASE>0 and flag small NCASE value"
735 JSMALLRUN=0;
736 IF(NCASE<$STAT)[JSMALLRUN=NCASE;NCASE=$STAT;]
737 JCASE = NCASE/$STAT;NCASE=JCASE*$STAT;ISTEPA=0;
738 "Set up initial particle parameters"
739 IF(ISOURC=0)[UIN=0.0;VIN=0.0;WIN=1.0;ZIN=-0.0001;]
740 ELSEIF(ISOURC=1)[ZIN=-0.00001;]
741 ELSEIF(ISOURC=2)[
742   UIN=0.0;VIN=0.0;WIN=1.0;
743   XIN=XCENTRE;YIN=YCENTRE;ZIN=ORIGINDEPTH;
744   IXBASE=INT(XIN/XSIZE)+1;IYBASE=INT(YIN/YSIZE)+1;IZBASE=INT(ZIN/ZSIZE)+1;
745   IRIN=(IXBASE-1)*$YMAX*$ZMAX+(IYBASE-1)*$ZMAX+IZBASE+1;
746 ]
747 "Initialise DOSEIS array"
748 DO IX=1,$XMAX[
749   DO IY=1,$YMAX[
750     DO IZ=1,$ZMAX[
751       DO IS=1,$STAT[
752         DOSEIS(IX,IY,IZ,IS)=0.0;
753 ] ] ] ]
754 "Fibonacci RNG may be initialised here"
755 $TPUTER-RNG-INIT;
756 IF(IWATCH ~= 0)["Set up to call ausgab for all interactions"
757   CALL WATCH(-99,IWATCH);
758 ]
759 "Calculate incident fluence"
760 IF(ISOURC = 0) [AINFLU=FLOAT(NCASE)/(BEAMWIDTH**2);]
761 ELSEIF(ISOURC = 1)[AINFLU=FLOAT(NCASE)/(BEAMWIDTH**2);]
762 ELSEIF(ISOURC = 2)[AINFLU=FLOAT(NCASE);]
763 ;
764 "*****************************************************************************"
765 " Summarise input"
766 CALL DATE(DATEN);CALL TIME(TIMEN);WRITE(IOUT,140)TITLE,DATEN,TIMEN;
767 WRITE(IOUT,150) EIN,IQIN,NCASE;
768 IF(EIN<=0.0)[ WRITE(IOUT,151)(I,EBIN(I),EPHI(I)/TOTPHI,ECPD(I),I=1,NEBIN);]
769 IF(IDORAY ~= 0)[WRITE(IOUT,152);]ELSE[WRITE(IOUT,153);];
770 WRITE(IOUT,155) ECUTIN,PCUTIN;
771 DO I=1,NMED[
772 WRITE(IOUT,160) I,(MEDIA(J,I),J=1,24),RHO(I),AE(I),AP(I),200*TEFFO(I);]
773 IF(ICSDA ~= 0)[WRITE(IOUT,244);]
774 IF(ESTEPE ~= 0.0)[WRITE(IOUT,165) ESTEPE;]
775 IF(NOPLCI ~= 0) [WRITE(IOUT,245);]
776 IF(NOMSCI ~= 0) [WRITE(IOUT,240);]
777 "Print MED,RHOR for central voxel on each layer"
778 WRITE(IOUT,161);
779 IX=INT(XCENTRE/XSIZE)+1;
780 IY=INT(YCENTRE/YSIZE)+1;
781 DO IZ=1,$ZMAX[
782   IMED=MED((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1);
783   XRHOR=RHOR((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1);
784   WRITE(IOUT,162) IZ,IMED,XRHOR;
785 ]
786 IF(ISOURC = 0)[WRITE(IOUT,190) XCENTRE,YCENTRE,BEAMWIDTH,AINFLU;]
787 ELSEIF(ISOURC = 1)[WRITE(IOUT,200) XCENTRE,YCENTRE,DISTZ,BEAMWIDTH,AINFLU;]
788 ELSEIF(ISOURC = 2)[
789   WRITE(IOUT,195) XCENTRE,YCENTRE,ORIGINDEPTH,ISMEAR,IDOSETYPE;
790   IF(ISMEAR=0)[WRITE(IOUT,351);]
791   ELSEIF(ISMEAR=1)[WRITE(IOUT,352);]
792   ELSEIF(ISMEAR=2)[WRITE(IOUT,353);]
793   IF(IDOSETYPE=0)[WRITE(IOUT,354);]
794   ELSEIF(IDOSETYPE=1)[WRITE(IOUT,355);]
795   ELSEIF(IDOSETYPE=2)[WRITE(IOUT,356);]
796 ]
797 ISEED=IXX;"save original IXX value"
798 WRITE(IOUT,205) INSEED;
799 OUTPUT INSEED,ISEED;(/'  Input R.N. seed, initial seed = ',2I12);
800 IF(RUSROU)[WRITE(IOUT,305) RRZ,RRCUT;]
801 IF(CEXPTR ~= 0.0)[WRITE(IOUT,315) CEXPTR,1./(1.-CEXPTR);]
802 $PRESTA-INPUT-SUMMARY; "Print PRESTA input summary"
803 ;
804 "*****************************************************************************"
805 " Perform Simulation"
806 CALL CPUTIME(CPUT3);TZERO=SECNDS(0.0);
807 "Process each batch in turn"
808 DO IS=1,$STAT[
809   TIMEB=SECNDS(TZERO);CALL CPUTIME(CPUT4);TIMCPU=(CPUT4-CPUT3)*0.01+.0001;
810   OUTPUT IS,TIMEB,TIMCPU,TIMEB/TIMCPU,IXX;
811   (' START BATCH',I3,' TIMES-ELAPSED,CPU,RATIO,RNG',2F8.1,F8.1,I12);
812   IF(IS ~= 1)["check there is time left"
813     TIMJOB=CPUT4*0.01;"time since job started"
814     BATCHT=TIMCPU/FLOAT(IS-1);"time per batch so far"
815     IF(TIMJOB + 1.1*BATCHT  >  TIMMAX*3600.)[ "not enough time"
816       ISTAT=IS-1;WRITE(6,220) TIMMAX,ISTAT;
817       WRITE(IOUT,220) TIMMAX,ISTAT;
818       IABORT=1;GO TO :END-SIM:
819     ]
820   ]
821   "Check that a small number of NCASEs is not being requested"
822   IF (JSMALLRUN~=0 & IS>JSMALLRUN)[
823     OUTPUT JSMALLRUN;(' RUN STOPPED DUE TO JSMALLRUN = ',I3);
824     ISTAT=JSMALLRUN;
825     IABORT=1;
```

```
826       GO TO :END-SIM:
827     ]
828     "Begin each history"
829     DO ICASE=1,JCASE[
830       IF(ISOURC = 0)["Parallel square beam"
831         $RANDOMSET XIN;$RANDOMSET YIN;
832         XIN=(BEAMWIDTH*XIN) + XCENTRE - (BEAMWIDTH/2.0);
833         YIN=(BEAMWIDTH*YIN) + YCENTRE - (BEAMWIDTH/2.0);
834       ]
835       ELSEIF(ISOURC = 1)["Point source on axis"
836         $RANDOMSET XPROJ;$RANDOMSET YPROJ; "Projection onto surface"
837         XPROJ=(BEAMWIDTH*XPROJ)-BEAMWIDTH/2.0;
838         YPROJ=(BEAMWIDTH*YPROJ)-BEAMWIDTH/2.0;
839         XIN=XPROJ+XCENTRE;YIN=YPROJ+YCENTRE;
840         LNGTH = SQRT(XPROJ**2 + YPROJ**2 + DISTZ**2);
841         UIN = XPROJ/LNGTH;VIN = YPROJ/LNGTH;WIN = DISTZ/LNGTH;
842         "Option to perform particle splitting near central axis - commented out"
843         "IF(ABS(XPROJ)<2.0 & ABS(YPROJ)<2.0)[IPARTNO=5;WTIN=0.2;]"
844         "ELSE[IPARTNO=1;WTIN=1.0;]"
845       ]
846       ELSEIF(ISOURC = 2)["Dose spread array"
847         IF(ISMEAR=1)["2D interaction voxel smearing - for electrons"
848           $RANDOMSET XIN;$RANDOMSET YIN;
849           XIN=(IXBASE-1+XIN)*XSIZE;
850           YIN=(IYBASE-1+YIN)*YSIZE;
851         ]
852         ELSEIF(ISMEAR=2)["3D interaction voxel smearing"
853           $RANDOMSET XIN;$RANDOMSET YIN;$RANDOMSET ZIN;
854           XIN=(IXBASE-1+XIN)*XSIZE;
855           YIN=(IYBASE-1+YIN)*YSIZE;
856           ZIN=(IZBASE-1+ZIN)*ZSIZE;
857         ]
858       ]
859     IF(EIN<=0.0)[ "Select energy from distribution"
860       CALL ESPEC(EI);EINN=EI; "Returns kinetic energy - need total"
861       IF(IQIN.NE.0) [EI = EI+0.511;]
862     ]
863     ELSE [EINN = EIN;]
864     "If a DSA is being generated we need to weight the photon according to its"
865     "mean free path, and add the weighted K.E to DSA total.                    "
866     IF (ISOURC=2)[
867       IF(IQIN=0)["photon"
868         MEDIUMOLD=MEDIUM;
869         MEDIUM=MED(2);
870         IF(EINN<PCUT(2))[GAMMALOG=ALOG(PCUT(2))]
871         ELSE [GAMMALOG=ALOG(EINN);]
872         $SET INTERVAL GAMMALOG,GE;
873         $EVALUATE GAMMAMFP USING GMFP(GAMMALOG);
874         IF(GAMMAMFP<0.001)[
875           OUTPUT EINN,MEDIUM,GAMMALOG,LGAMMALOG,GAMMAMFP,WTIN;
876           (' EINN,MED,GLOG,LGLOG,GMFP,WTIN= ',F10.3,I3,F10.3,I6,2F10.3);
877         ]
878         WTIN=1.0/GAMMAMFP;
879         MEDIUM=MEDIUMOLD;
880       ]
881       ELSE["electron"
882         WTIN=1.0;
883       ]
884     ]
885     "Add energy to total incident energy"
886     TOTALMEV=TOTALMEV+EINN*WTIN;
887     "If WATCH is active, output current status"
888     IF(IWATCH > 0) [OUTPUT 1,EINN,IQIN,IRIN,XIN,YIN,ZIN,UIN,VIN,WIN,LATCHI,WTIN;
889     (' INITIAL SHOWER VALUES',T36,':',I2,F9.3,I4,I7,3F8.3,3F7.3,I10,1PE10.3);]
890     "Simulate the particle!"
891     "If particle splitting option has been selected, then split particle"
892     IF (IPARTNO>1)[
893       DO I=1,IPARTNO[
894         CALL SHOWER(IQIN,EI,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN);
895         "OUTPUT XIN,YIN,WTIN; ('Split - X,Y,WT = ',3F12.6);"
896       ]
897     ]
898     ELSE[
899       CALL SHOWER(IQIN,EI,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN);
900       "OUTPUT XIN,YIN,WTIN; (' Not split - X,Y,WT = ',3F12.6);"
901     ]
902     "To signal end of history"
903     IF(IWATCH > 0) CALL WATCH(-1,IWATCH);]
904   ] "end of IS loop"
905   ISTAT=$STAT;"number of batches completed - full set if code reaches here"
906   ;
907   "*****************************************************************************"
908   "Analyse Results"
909   :END-SIM:
910   TIMEB=SECNDS(TZERO);CALL CPUTIME(CPUT4);TIMCPU=(CPUT4-CPUT3)*0.01;
911   OUTPUT TIMEB,TIMCPU,TIMEB/TIMCPU;
912   (//'  FINISHED SIMULATIONS- times elapsed,cpu,ratio=',2F8.1,F8.2);
913   WRITE(IOUT,295) TIMEB,TIMCPU,TIMEB/TIMCPU,MINP;
914   WRITE(IOUT,310) ISTEPA,NOSCAT;
915   LASTIX=IXX; $RANDOMSET XSI;
916   WRITE(IOUT,280) LASTIX,XSI;
917   STAT=FLOAT(ISTAT);
918   IF(ISTAT ^= 1)[SDENOM=STAT*(STAT-1.);]ELSE[SDENOM=0.00001;]
919   AINFLU=AINFLU*STAT/FLOAT($STAT);"Correct fluence in case of aborted run"
```

```
NCASE = ISTAT*JCASE;"Correct total number of histories if aborted run"

IF(IVOXAVG=1)["calculate dose with equivalent voxel averaging"
  ICENTRE=INT(ICENTRE/XSIZE)+1;
  JCENTRE=INT(YCENTRE/YSIZE)+1;
  OUTPUT ICENTRE,JCENTRE; (/' Layer voxels averaged about (',I3,',',I3,')');
  DO IS=1,ISTAT[
    DO IZ=1,$ZMAX[
      DO IX=ICENTRE,$XMAX[
        DO IY=JCENTRE,$YMAX[
          "Calculate average dose"
          IMISSING=0;
          DOSESUM = DOSEIS(IX,IY,IZ,IS);
          INEWX = ICENTRE-(IX-ICENTRE);
          INEWY = JCENTRE-(IY-JCENTRE);
          IF(INEWX>0)[DOSESUM=DOSESUM+DOSEIS(INEWX,IY,IZ,IS);]
          ELSE[IMISSING=IMISSING+1;]
          IF(INEWY>0)[DOSESUM=DOSESUM+DOSEIS(IX,INEWY,IZ,IS);]
          ELSE[IMISSING=IMISSING+1;]
          IF(INEWX>0 & INEWY>0)[DOSESUM=DOSESUM+DOSEIS(INEWX,INEWY,IZ,IS);]
          ELSE[IMISSING=IMISSING+1;]
          "Debugging lines"
          "OUTPUT IX,IY,IZ,INEWX,INEWY,DOSE;"
          "(' IX,IY,IZ,INEWX,INEWY,DOSE = ',5I4,F10.4);"
          "Assign dose to voxels in positive corner"
          DOSEIS(IX,IY,IZ,IS) = DOSESUM/FLOAT(4-IMISSING);
    ] ] ]
    "If XSIZE = YSIZE then we can average over 2 voxels in the same quadrant"
    IF(XSIZE=YSIZE)[
      DO IZ=1,$ZMAX[
        DO IX=ICENTRE,$XMAX[
          DO IY=JCENTRE,$YMAX[
            IF(IX-ICENTRE+YCENTRE<=$YMAX & IY-YCENTRE+ICENTRE<=$XMAX)[
              DOSEIS(IX,IY,IZ,IS)=(DOSEIS(IX,IY,IZ,IS) +
                        DOSEIS(IY-YCENTRE+ICENTRE,IX-ICENTRE+YCENTRE,IZ,IS))/2.0;
              DOSEIS(IY-YCENTRE+XCENTRE,IX-ICENTRE+YCENTRE,IZ,IS) =
                      DOSEIS(IX,IY,IZ,IS);
] ] ] ] ] ] ]

"Get total dose for each region"
PHANTOMMEV=0.0; "initialise total phantom dose"
DO IX=1,$XMAX[
  DO IY=1,$YMAX[
    DO IZ=1,$ZMAX[
      DOSE(IX,IY,IZ)=0.0;
      DO IS=1,ISTAT[
        DOSE(IX,IY,IZ) = DOSE(IX,IY,IZ) + DOSEIS(IX,IY,IZ,IS);
      ]
      PHANTOMMEV=PHANTOMMEV+DOSE(IX,IY,IZ);
      DOSE(IX,IY,IZ) = DOSE(IX,IY,IZ)/STAT;"divide by no. of batches"
] ] ]

"Now calculate uncertainties on dose"
DO IX=1,$XMAX[
  DO IY=1,$YMAX[
    DO IZ=1,$ZMAX[
      UNCER(IX,IY,IZ)=0.0;
      DO IS=1,ISTAT[
        UNCER(IX,IY,IZ)=UNCER(IX,IY,IZ)+(DOSEIS(IX,IY,IZ,IS)-
                         DOSE(IX,IY,IZ))**2;
      ]
      IF(DOSE(IX,IY,IZ) ~= 0.0)[
        UNCER(IX,IY,IZ)=SQRT(UNCER(IX,IY,IZ)/SDENOM)/DOSE(IX,IY,IZ);
] ] ] ]

"Convert dose from MeV per region per batch to Gy/unit incident energy fluence"
"i.e. Gy.cm**2 /MeV"
IF(ISOURC<2)[
  DO IX=1,$XMAX[
    DO IY=1,$YMAX[
      DO IZ=1,$ZMAX[
        DOSE(IX,IY,IZ)=DOSE(IX,IY,IZ)*STAT*1.602E-10/AINFLU/
                       (RHOR((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1)*
                       XSIZE*YSIZE*ZSIZE)/SNGL(TOTALMEV/FLOAT(NCASE));
        "Recall 1 Mev = 1.602e-06 ERGS and dose is per batch-hence *STAT "
] ] ] ]
ELSEIF(ISOURC=2)["divide by total incident energy"
  DO IX=1,$XMAX[
    DO IY=1,$YMAX[
      DO IZ=1,$ZMAX[
        DOSE(IX,IY,IZ)=DOSE(IX,IY,IZ)*STAT/TOTALMEV;
] ] ] ]
;
"*****************************************************************************"
" Print Results"
WRITE(IOUT,250) TITLE,DATEN,TIMEN,NCASE,IQIN,EIN,
  ECUTIN,PCUTIN,SMAX,CEXPTR,TIMCPU;
IF(ISOURC = 0)[WRITE(IOUT,190) XCENTRE,YCENTRE,BEAMWIDTH,AINFLU;]
ELSEIF(ISOURC = 1)[WRITE(IOUT,200) XCENTRE,YCENTRE,DISTZ,BEAMWIDTH,AINFLU;]
ELSEIF(ISOURC = 2) [
  WRITE(IOUT,195) XCENTRE,YCENTRE,ORIGINDEPTH,ISMEAR,IDOSETYPE;
]

"Draw table of doses"
IF(PLOTIT~=0)["draw table if flag is non-zero"
```

```
1015   IF (ISOURC ~=2)[WRITE(IOUT,260);]
1016   ELSE [
1017     IF(IDOSETYPE=0)[WRITE(IOUT,265);]
1018     ELSEIF(IDOSETYPE=1)[WRITE(IOUT,266);]
1019     ELSEIF(IDOSETYPE=2)[WRITE(IOUT,267);]
1020   ]
1021   IF(IVOXAVG=0)["plot entire array - use y layer next to YCENTRE"
1022     DO IX=1,$XMAX[
1023       DO IZ=1,$ZMAX[
1024         DOSEIT(IZ,IX)=DOSE(IX,INT(YCENTRE/YSIZE)+1,IZ);
1025         UNCERT(IZ,IX)=UNCER(IX,INT(YCENTRE/YSIZE)+1,IZ);
1026       ]
1027     ]
1028     WRITE(IOUT,197) INT(YCENTRE/YSIZE)+1; "indicate y layer"
1029     CALL GRIDLP(DOSEIT,UNCERT,$ZMAX,$XMAX,IOUT);
1030   ]
1031   ELSEIF(IVOXAVG=1) ["plot averaged quadrant only"
1032     ICENTRE=INT(XCENTRE/XSIZE)+1;
1033     DO IX=ICENTRE,$XMAX[
1034       DO IZ=1,$ZMAX[
1035         DOSEIT(IZ,IX-ICENTRE+1)=DOSE(IX,INT(YCENTRE/YSIZE)+1,IZ);
1036         UNCERT(IZ,IX-ICENTRE+1)=UNCER(IX,INT(YCENTRE/YSIZE)+1,IZ);
1037       ]
1038     ]
1039     WRITE(IOUT,197) INT(YCENTRE/YSIZE)+1; "indicate y layer"
1040     CALL GRIDLP(DOSEIT,UNCERT,$ZMAX,$XMAX-ICENTRE+1,IOUT);
1041   ]
1042 ]
1043
1044 "If IRAWDATA is set to 1, write all raw data to file"
1045 IF (IRAWDATA = 1)[
1046   IF (IVOXAVG=1)[
1047     OUTPUT $XMAX-ICENTRE+1,$YMAX-JCENTRE+1,$ZMAX;
1048     (' IRAWDATA SIZE IS (',3I5,')');
1049     WRITE(IRAWFILE,302) $XMAX-ICENTRE+1, $YMAX-JCENTRE+1, $ZMAX;
1050     WRITE(IRAWFILE,303) XSIZE, YSIZE, ZSIZE;
1051     WRITE(IRAWFILE,304) INT(ORIGINDEPTH/ZSIZE);
1052     DO IZ=1,$ZMAX[
1053       DO IY=JCENTRE,$YMAX[
1054         DO IX=ICENTRE,$XMAX[
1055           WRITE(IRAWFILE,300) DOSE(IX,IY,IZ);
1056       ] ]
1057       WRITE(IRAWFILE,301);
1058   ] ]
1059   ELSE[
1060     OUTPUT $XMAX,$YMAX,$ZMAX;(' IRAWDATA SIZE IS (',3I5,')');
1061     WRITE(IRAWFILE,302) $XMAX, $YMAX, $ZMAX;
1062     WRITE(IRAWFILE,303) XSIZE, YSIZE, ZSIZE;
1063     WRITE(IRAWFILE,304) INT(ORIGINDEPTH/ZSIZE);
1064     DO IZ=1,$ZMAX[
1065       DO IY=1,$YMAX[
1066         DO IX=1,$XMAX[
1067           WRITE(IRAWFILE,300) DOSE(IX,IY,IZ);
1068       ] ]
1069       WRITE(IRAWFILE,301);
1070 ] ] ]
1071
1072 "If IRAWDATA is set to 2, write central axis (x,z) plane only to output file"
1073 IF (IRAWDATA = 2)[
1074   IF (IVOXAVG=1)[
1075     OUTPUT $XMAX-ICENTRE+1,$ZMAX;
1076     (' IRAWDATA SIZE IS (',2I5,')');
1077     IY = JCENTRE;
1078     OUTPUT IY; (' Y voxel plane is ',I3);
1079     DO IZ=1,$ZMAX[
1080       DO IX=ICENTRE,$XMAX[
1081         WRITE(IRAWFILE,300) DOSE(IX,IY,IZ);
1082       ]
1083       WRITE(IRAWFILE,301);
1084   ] ]
1085   ELSE[
1086     OUTPUT $XMAX,$ZMAX;(' IRAWDATA SIZE IS (',2I5,')');
1087     IY = INT(YCENTRE/YSIZE)+1;
1088     OUTPUT IY; (' Y voxel plane is ',I3);
1089     DO IZ=1,$ZMAX[
1090       DO IX=1,$XMAX[
1091         WRITE(IRAWFILE,300) DOSE(IX,IY,IZ);
1092       ]
1093     WRITE(IRAWFILE,301);
1094 ] ] ]
1095
1096 "Write brief summary of inputs"
1097 WRITE(IOUT,250) TITLE,DATEN,TIMEN,NCASE,IQIN,EIN,
1098   ECUTIN,PCUTIN,SMAX,CEXPTR,TIMCPU;
1099 IF(ISOURC = 0)[WRITE(IOUT,190) XCENTRE,YCENTRE,BEAMWIDTH,AINFLU;]
1100 ELSEIF(ISOURC = 1)[WRITE(IOUT,200) XCENTRE,YCENTRE,DISTZ,BEAMWIDTH,AINFLU;]
1101 ELSEIF(ISOURC = 2)[WRITE(IOUT,195) XCENTRE,YCENTRE,ORIGINDEPTH,ISMEAR,IDOSETYPE;
1102                     WRITE(IOUT,196) TOTALMEV,PHANTOMMEV;]
1103 "Print voxel masses if requested"
1104 IF(MASSFLAG = 0)[GO TO :SKIP-MASSES:;]
1105 WRITE(IOUT,250) TITLE,DATEN,TIMEN,NCASE,IQIN,EIN,
1106 ECUIN,PCUIN,SMAX,CEXPTR,TIMCPU;
1107 WRITE(IOUT,180);
1108 "Calculate AMASSes for designated layer"
1109 IY=INT(YCENTRE/YSIZE)+1;
```

```
1110 DO IX=1,$XMAX[
1111   DO IZ=1,$ZMAX[
1112     AMASS(IZ,IX)=RHOR((IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1);
1113 ] ]
1114 CALL GRIDLP(AMASS,AMASS,$ZMAX,$XMAX,IOUT); "plot masses"
1115 :SKIP-MASSES:
1116 IF(IABORT=1)[GO TO :END:]
1117 ;
1118 "*******************************************************************************"
1119 "End of simulation"
1120 :END:
1121 CALL DATE(DATEN);CALL TIME(TIMEN);
1122 CALL CPUTIME(CPUT4);TIMCPU=(CPUT4-CPUT0)*0.01;
1123 OUTPUT DATEN,TIMEN;(/'  ...end of file read',10X,9A1,1X,8A1);
1124 WRITE(IOUT,210) DATEN,TIMEN,TIMCPU,TIMCPU/3600.;
1125 STOP;
1126 ;
1127 "*******************************************************************************"
1128 " Fortran output formats"
1129 %I   need as switch to FORTRAN with %F
1130 %F   switch to FORTRAN
1131 100   FORMAT(80A1)
1132 105   FORMAT(3F12.0)
1133 110   FORMAT(7F12.0)
1134 111   FORMAT(3F12.0,2I3)
1135 115   FORMAT(7I3)
1136 120   FORMAT(24A1)
1137 130   FORMAT(3I10,F10.0,I14,8I10)
1138 140   FORMAT(/' ',80A1//' University of Waikato RTPCART (EGS4/PRESTA)',
1139      1' 1990', 6X,9A1,1X,8A1)
1140 150   FORMAT(/' Incident energy=',F8.3,' MeV,   charge=',I3,
1141      1',    #histories=',I8)
1142 151   FORMAT(/T20,' Bin   Top E   Prob.     Cumulative prob.'/
1143      1 (T20,I3,0PF9.3,2(1PE12.3)))
1144 152   FORMAT(/'  Rayleigh (coherent) scattering INCLUDED'/)
1145 153   FORMAT(/'  Rayleigh (coherent) scattering NOT INCLUDED')
1146 155   FORMAT(/'  Global user cutoffs ECUT,PCUT=',2F12.3,' MeV',
1147      1 //'  Material',T25,'Density',T46,'AE',T62,'AP',
1148      2 T70,'200*TEFF0')
1149 160   FORMAT(I3,2X,24A1,T20,F10.3,T38,F12.3,T54,F12.3,T65,F12.4)
1150 161   FORMAT(/'  Z voxel,  medium,  density (for central axis voxels)')
1151 162   FORMAT('    ',I3,'       ',I3,'              ',F9.4)
1152 165   FORMAT('  Electron step size has been modified to',
1153      1 ' reduce energy by',F8.3,' in each step')
1154 180   FORMAT(/'  Masses of material in each scoring zone (g)'/)
1155 190   FORMAT(/'  Parallel beam with XY centre=(',F6.2,F6.2,
1156      1 '), Width=',F6.2,' cm',/'  Fluence=',1PE10.2,'/cm**2')
1157 195   FORMAT(/'  Dose spread array with X,Y centre=(',F6.2,F6.2,
1158      1'),  Origin depth=',F7.2,'/cm**2',/'  Smear flag = ',I2,
1159      2'  Dose type = ',I2)
1160 196   FORMAT(/'  Total incident Energy = ',F12.4,
1161      1' MeV, Phantom energy = ',F12.4,' MeV')
1162 197   FORMAT(T20'  Y voxel level used for GRIDLP = ',I3)
1163 200   FORMAT(/'  Point source beam with centre=(',F6.2,F6.2,
1164      1'), Distance from surface=',F7.2,/'  Width=',F7.2,
1165      2' cm,  Fluence=',1PE10.2,'/cm**2')
1166 205   FORMAT(/'  Initial random number seed=',I12)
1167 210   FORMAT(/'  ...end of file read',5X,9A1,1X,8A1,5X,
1168      1'CPU=',F10.1,' sec=',F8.3,' hrs')
1169 220   FORMAT(/'  *****NOT ENOUGH TIME TO FINISH WITHIN LIMIT OF',
1170      1 F6.2,' HOURS',I10,' BATCHES USED********'/////)
1171 240   FORMAT(/' ',10('='),'  Multiple scattering turned off',10('=')//)
1172 244   FORMAT(/'  ============CSDA Calculation============='/
1173      1 '  needs IUNRST=2 and 5 data sets'//)
1174 245   FORMAT(/'  ',10('='),'Path-length correction turned off'
1175      1 10('=')//)
1176 250   FORMAT(/2X,80A1,T60,9A1,1X,8A1/I8,' particles of charge'
1177      1,I2,', energy=',F8.3,' MeV on phantom',
1178      2 //'  RTPCART (ECUT,PCUT)=',
1179      3'[',F7.3,',',F6.3,'], SMAX=',1PE9.2,/'  CEXPTR=',0PF6.3,
1180      4'   CPU=',0PF8.1,' sec')
1181 260   FORMAT(/T20,' TOTAL DOSE: Gray per incident unit ENERGY fluence')
1182 265   FORMAT(/T20,' TOTAL ENERGY (normalised to incident energy)')
1183 266   FORMAT(/T20,' PRIMARY ENERGY (normalised to incident energy)')
1184 267   FORMAT(/T20,' MULTIPLE-SCATTERED ENERGY ',
1185      1'(normalised to incident energy)')
1186 275   FORMAT(/T10,' TOTAL, e OTHER MED,PART.THRU OTHER MED,STOPPERS:',
1187      2' (Gray/INCIDENT ENERGY FLUENCE)')
1188 280   FORMAT(/'  Final random no. seed=',I12,'  Final random no.=',
1189      1  0PF15.8)
1190 295   FORMAT(/'  Simulation times - elapsed,CPU,ratio=',2F8.1,F7.2/
1191      1 '  Maximum stack=',I5)
1192 305   FORMAT(/'  Russian roulette for photons crossing Z=',
1193      1 F8.3,' cm with survival=',F7.4)
1194 310   FORMAT(/'  Total number of charged particle steps taken=',I10
1195      1'  NOSCAT=',I10)
1196 315   FORMAT(/'  Pathlength exponential transformation variable=',F10.3,
1197      1 '  for forward going photons only'/'  Shortens initial ',
1198      2 'pathlength by:',F10.3)
1199 300   FORMAT(2X,E10.4)
1200 301   FORMAT(2X)
```

```
1201 302   FORMAT(3I5)
1202 303   FORMAT(3F12.4)
1203 304   FORMAT(I5)
1204 351   FORMAT('  NO voxel smearing performed')
1205 352   FORMAT('  2D voxel smearing performed')
1206 353   FORMAT('  3D voxel smearing performed')
1207 354   FORMAT('  TOTAL DOSE is scored')
1208 355   FORMAT('  PRIMARY DOSE ONLY is scored')
1209 356   FORMAT('  MULTIPLE-SCATTERED DOSE ONLY is scored')
1210 %M
1211 %I4
1212 END;" end of RTPCART main routine"
1213 ;
1214 "*****************************************************************************"
1215 REAL FUNCTION TAN(X);                                                        "
1216 "   Given x in radians this function calculates the tangent of X             "
1217 A=COS(X);
1218 IF(ABS(A) > 1.E-10) [TAN = SIN(X)/A;] ELSE [TAN = 1.E10;]
1219 RETURN;
1220 END;"end of TAN function"
1221 ;
1222 "*****************************************************************************"
1223 SUBROUTINE ESPEC(EI);
1224 "   This routine returns an initial kinetic energy given the cumulative      "
1225 "   probability distribution for the source spectrum stored in ECPD, the     "
1226 "   energy bin tops in EBIN and the minimum E in EIMIN  - all in common ESPECT "
1227 "   Initially written in Tampere Finland, D.W.O. Rogers,Aug 1985             "
1228
1229 ;COMIN/ESPECT,RANDOM/;
1230 $RANDOMSET RNN040; $RANDOMSET RNN041;  "pick two random numbers"
1231 "Use the first to select which energy bin we are in"
1232 I=0; LOOP [I=I+1;] UNTIL ECPD(I)>RNN040;
1233 "Energy is between EBIN(I-1) and EBIN(I)"
1234 IF(I.NE.1)[ELOW=EBIN(I-1);] ELSE [ELOW = EIMIN;]
1235 EI = ELOW + RNN041* (EBIN(I)-ELOW);  "select randomly in this bin"
1236 RETURN; END;
1237 ;
1238 "*****************************************************************************"
1239 SUBROUTINE AUSGAB(IARG);
1240 "                                                                            "
1241 "        An AUSGAB routine to be used with RTPCART.MOR                       "
1242 "        This routine scores the dose in a Cartesian geometry within         "
1243 "        an array ($XMAX,$YMAX,$ZMAX) of voxels.                             "
1244 "        For DSAs (ISOURC=2), IDOSETYPE controls total/primary/sec dose      "
1245
1246 "COMIN USER added to AUSGAB to allow access to ISOURC"
1247 "COMIN GEOM added to AUSGAB to allow access to XSIZE,YSIZE,ZSIZE"
1248 ;COMIN/ELECIN,EPCONT,MISC,RUSROU,RANDOM,SCORE,STACK,USER,GEOM/;
1249  "Initialize for first pass -will be wrong for first pass in each history"
1250 DATA IRLAST/1/;
1251
1252 "  The following macros set and pickup a flag based on LATCH. This feature   "
1253 "  is currently used in RTPCART to signal primary/scattered dose. FLAG1 is   "
1254 "  the units digit of LATCH. Usage is $SET-FLAG1(NP); and IF($FLAG1)[...     "
1255 "Macro to define flag"
1256 REPLACE {$FLAG1} WITH {ABS(MOD(LATCH(NP),100))}
1257 "Macro to set flag"
1258 REPLACE {$SET-FLAG1(#)} WITH
1259 {;IF(LATCH({P1}) >= 0)[LATCH({P1})=LATCH({P1})+1;]
1260 ELSE [LATCH({P1})=LATCH({P1})-1;];}
1261
1262 "Monitor stack size"
1263 MXNP=MAX(MXNP,NP);"keep track of how deep stack is"
1264 IF(NP >= $MXSTACK)["stack as deep as allowed"
1265 OUTPUT NP,$MXSTACK;(' IN AUSGAB, NP=',I3,' >= MAXIMUM STACK ALLOWED=',
1266 I3,/1X,80('*')/);IF(MXNP > $MXSTACK)[STOP;] ]
1267
1268 "Write particle tracks to DOTPLOT file if requested"
1269 IF(IDOTPLOT=1)["If DOTPLOT=1, write particle IQ,X,Z to unit 13"
1270   IF(USTEP<100.0)[
1271     WRITE(IDOTFILE,7) IQ(NP),X(NP),Z(NP);
1272 ] ]
1273 ELSEIF (IDOTPLOT=2)["If DOTPLOT=2, write particle IQ,X,Z,U,W,USTEP to unit 13"
1274   IF(USTEP<100.0)[
1275     WRITE(IDOTFILE,9) IQ(NP),X(NP),Z(NP),U(NP),W(NP),USTEP;
1276 ] ]
1277
1278 "Ensure particle has been flagged as primary or scattered"
1279 IF(ISOURC=2 & $FLAG1=0 & NP>1)[
1280   DO I=1,NP[
1281     IF(IQ(I)=0)[$SET-FLAG1(I);$SET-FLAG1(I);] "set flag to 2 for multiple-scat"
1282     ELSE[$SET-FLAG1(I);] "set flag to 1 for primary"
1283   ]
1284 ]
1285 "Following line patched 12/11/90 - DCM"
1286 "Ensure photoelectrons are scored as primary dose"
1287 IF(ISOURC=2 & $FLAG1=0 & NP=1 & IQ(1)~=0)[$SET-FLAG1(1);]
1288 "Assign brem to scattered dose - altered 26/11/90 - DCM"
1289 "If photon and $FLAG1=1 then brem, so assign to secondary dose"
1290 IF($FLAG1=1 & IQ(NP)=0) [$SET-FLAG1(NP);]
1291 "Count electron step or check Russian roulette"
1292 IF(IARG = 0)["about to transport a particle"
```

```
1293   IF(IQ(NP) ^= 0)[ISTEPA=ISTEPA+1;]"electron - count charged particle steps"
1294   ELSE ["Photon step - play Russian roulette?"
1295     IF(RUSROU  &  W(NP) >  0.0)["yes play if crosses boundary going forward"
1296       IF(Z(NP) <= RRZ  &  Z(NP)+USTEP*W(NP)  >=  RRZ)["crossed the plane"
1297         $RANDOMSET XSI;
1298         IF(XSI < RRCUT)["particle survives"
1299           WT(NP)=WT(NP)/RRCUT; "modify particle weight"
1300         ]
1301         ELSE ["particle to be destroyed"
1302           WT(NP)=0.0;"the particle will be discarded in HOWFAR"
1303 ] ] ] ] ]
1304
1305 "Check particle needs to be scored"
1306 IF(IWATCH > 0) CALL WATCH(IARG,IWATCH); "monitor history if requested"
1307 IF(IARG >=5)[RETURN;] "only score in usual cases"
1308 IF(EDEP = 0.0)[RETURN;] "photons don't usually deposit energy"
1309 "Find indices of current voxel"
1310 IX=INT(X(NP)/XSIZE)+1;IY=INT(Y(NP)/YSIZE)+1;IZ=INT(Z(NP)/ZSIZE)+1;
1311 "Check we are within scoring phantom and particle not discarded"
1312 IF (IX>$XMAX | IX<1 | IY>$YMAX | IY<1 | IZ>$ZMAX | IZ<1 | IDISC=2)[RETURN;]
1313
1314 "Check for CSDA, otherwise score energy"
1315 IF(IARG = 0 & ICSDA = 1)["Check for CSDA option"
1316   "Perform CSDA calculation by subtracting radiative stopping power from       "
1317   "EDEP (brem photons are assumed to escape). Assume IUNRST=2&5 for meds 1&2   "
1318   MEDIUM=2;
1319   $SET INTERVAL ELKE,EKE;$EVALUATE DEDXRD USING EDEDX(ELKE);
1320   EDEPL = EDEP - TVSTEP*DEDXRD;
1321   MEDIUM=1;
1322   "Note ELECTR calls SET INTERVAL just after call to AUSGAB"
1323   "so we don't have to reset for medium 1 here"
1324 ]
1325 ELSE [EDEPL = EDEP;  "deposit all the energy usually"
1326   "Debugging line..."
1327   "OUTPUT IX,IY,IZ,EDEPL;(' Score, xyz,edepl = ',3I6,F10.4);"
1328   IF(ISOURC"=2)["not a dose spread array"
1329     DOSEIS(IX,IY,IZ,IS)=DOSEIS(IX,IY,IZ,IS) + WT(NP)*EDEPL;
1330   ]
1331   ELSE["dose spread array - check IDOSETYPE and score appropriate dose"
1332     IF(IDOSETYPE=0 | (IDOSETYPE=1 & $FLAG1=1) | (IDOSETYPE=2 & $FLAG1=2))[
1333       DOSEIS(IX,IY,IZ,IS)=DOSEIS(IX,IY,IZ,IS) + WT(NP)*EDEPL;
1334 ] ] ]
1335
1336 RETURN;
1337
1338 "RAWDATA formats"
1339 %I
1340 %F
1341 7     FORMAT(I3,2F10.4)
1342 9     FORMAT(I3,5F10.4)
1343 %M
1344 %I4
1345 END;"end of AUSGAB"
1346 ;
1347 "*******************************************************************************"
1348 SUBROUTINE HOWFAR;                                                              "
1349 "                                                                               "
1350 "        A HOWFAR routine to be used with RTPCART.MOR                           "
1351 "                                                                               "
1352 "        REGION 1                    Z<0                                        "
1353 "        REGION (IX*$YMAX*$ZMAX,IY*$ZMAX,IZ+1) = voxel (IX,IY,IZ)               "
1354 "        REGION $XMAX*$YMAX*$ZMAX+2 = region beyond phantom                     "
1355
1356 ;COMIN/EPCONT,STACK,GEOM/;
1357 IRL=IR(NP);
1358 "Terminate if particle has been reflected from phantom or escapes rear"
1359 IF((IRL <= 1  &  W(NP) <= 0.0) | IRL >= NREG) [IDISC=2;RETURN;]
1360
1361 "Discard zero weight particle (used in Russian roulette)"
1362 IF(WT(NP) = 0.0)[IDISC=2;RETURN;]
1363
1364 "Transport particle from region 1 into phantom"
1365 IF(IRL = 1)[" in source region to left of plate-note W(NP)>0 only here"
1366   TVAL = -Z(NP)/W(NP);"distance to plane at origin"
1367   IF(TVAL<USTEP)[
1368     USTEP=TVAL;
1369     IX=INT(X(NP)/XSIZE)+1;
1370     IY=INT(Y(NP)/YSIZE)+1;
1371     IZ=1;
1372     IRNEW=(IX-1)*$YMAX*$ZMAX+(IY-1)*$ZMAX+IZ+1;
1373   ]
1374   "Check particle is within phantom"
1375   IF(X(NP)<0.0|X(NP)>FLOAT($XMAX)*XSIZE|Y(NP)<0.0|Y(NP)>FLOAT($YMAX)*YSIZE)[
1376     IDISC=2;
1377   ]
1378   RETURN;
1379 ] "end of region 1 case"
1380
1381 "All other regions"
1382 "Check if outside region of interest"
1383 IF((Z(NP)<=0.0 & W(NP)<0.0) | Z(NP)>=FLOAT($ZMAX)*ZSIZE |
1384    X(NP)<=0.0 | X(NP)>=FLOAT($XMAX)*XSIZE |
```

```
1385    Y(NP)<=0.0 | Y(NP)>=FLOAT($YMAX)*YSIZE)[
1386   IDISC=2;
1387   RETURN;
1388 ]
1389
1390 "Transport particle within phantom, setting direction flag"
1391 "Note: If particle is very close (<$SMALLDIST) to boundary then consider"
1392 "it to have already been transported to new region"
1393 TVAL=10000.0; "set large in case W(NP)=0.0"
1394 IF(W(NP)>0.0)[                                  "z forward direction"
1395   TVAL=(ZSIZE-AMOD(Z(NP),ZSIZE))/W(NP);
1396   IF(TVAL<$SMALLDIST)[TVAL=ZSIZE/W(NP);]
1397   IDIRN=1;]
1398 ELSEIF(W(NP)<0.0)[                              "z backward direction"
1399   TVAL=(AMOD(Z(NP),ZSIZE))/-W(NP);
1400   IF(TVAL<$SMALLDIST)[TVAL=ZSIZE/-W(NP);]
1401   IDIRN=2;]
1402 IF(U(NP)>0.0)[                                  "x forward direction"
1403   TEMP=(XSIZE-AMOD(X(NP),XSIZE))/U(NP);
1404   IF(TEMP<$SMALLDIST)[TEMP=XSIZE/U(NP);]
1405   IF(TEMP<TVAL)[TVAL=TEMP;IDIRN=3;]]
1406 ELSEIF(U(NP)<0.0)[                              "x backward direction"
1407   TEMP=(AMOD(X(NP),XSIZE))/-U(NP);
1408   IF(TEMP<$SMALLDIST)[TEMP=XSIZE/-U(NP);]
1409   IF(TEMP<TVAL)[TVAL=TEMP;IDIRN=4;]]
1410 IF(V(NP)>0.0)[                                  "y forward direction"
1411   TEMP=(YSIZE-AMOD(Y(NP),YSIZE))/V(NP);
1412   IF(TEMP<$SMALLDIST)[TEMP=YSIZE/V(NP);]
1413   IF(TEMP<TVAL)[TVAL=TEMP;IDIRN=5;]]
1414 ELSEIF(V(NP)<0.0)[                              "y backward direction"
1415   TEMP=(AMOD(Y(NP),YSIZE))/-V(NP);
1416   IF(TEMP<$SMALLDIST)[TEMP=YSIZE/-V(NP);]
1417   IF(TEMP<TVAL)[TVAL=TEMP;IDIRN=6;]
1418 ]
1419
1420 "Update IRNEW if necessary"
1421 IF(TVAL<=USTEP)[
1422   USTEP=TVAL;
1423   IF(IDIRN=1)[IRNEW=IRL+1;]                     "increment z"
1424   ELSEIF(IDIRN=2)[IRNEW=IRL-1;]                 "decrement z"
1425   ELSEIF(IDIRN=3)[IRNEW=IRL+$YMAX*$ZMAX;]       "increment x"
1426   ELSEIF(IDIRN=4)[IRNEW=IRL-$YMAX*$ZMAX;]       "decrement x"
1427   ELSEIF(IDIRN=5)[IRNEW=IRL+$ZMAX;]             "increment y"
1428   ELSEIF(IDIRN=6)[IRNEW=IRL-$ZMAX;]             "decrement y"
1429   "The following are debug lines for this routine"
1430   "OUTPUT IRNEW,USTEP,X(NP),Y(NP),Z(NP),U(NP),V(NP),W(NP),IDIRN;"
1431   "(' Irnew=',I6,' ustep=',F10.7,' xyz=',3F5.2,' uvw=',3F5.2,' idirn=',I2);"
1432   "Check for bad region, which may occur if particle is about to be discarded"
1433   IF(IRNEW<1 | IRNEW>$MXREG)[IRNEW=1;IDISC=2;]
1434 ]
1435 RETURN;
1436
1437 END; "end of HOWFAR"
1438 ;
1439 "*************************************************************************"
1440 SUBROUTINE GRIDLP(DATA,UNCERT,NROW,NCOL,IOUT);
1441 "                                                                         "
1442 "        Routine to print an array of values and their uncertainties      "
1443 "        in a labeled grid                                                "
1444 "        DATA(IROW,ICOL)         data to be printed                       "
1445 "        UNCERT(IROW,ICOL)       fractional uncertainty in data (converted to  "
1446 "                                integer %age. If same as data, not printed.   "
1447 "        NROW                    # rows in grid down page (no limit).     "
1448 "        NCOL                    # columns in grid across page (20 + 15/col)   "
1449 "                                set to 7 for 132 column write            "
1450
1451 "Note COMIN GEOM added to allow access to XSIZE and YSIZE"
1452 ;COMIN/SCORE,GEOM/;
1453
1454 LOGICAL SAME;
1455 INTEGER MED(1);
1456 REAL DATA(NROW,NCOL),UNCERT(NROW,NCOL);
1457 INTEGER COLBASE,COLTOP; "base and top vars for multi-page printout"
1458
1459 "Check if data and uncert are the same and if they are, don't print uncert"
1460 SAME=.FALSE.;
1461 DO ICOL=1,NCOL[
1462   DO IROW=1,NROW[
1463     IF(DATA(IROW,ICOL) ^= UNCERT(IROW,ICOL))[
1464       GO TO :EXITLOOPS:;"leave SAME=FALSE"
1465 ] ] ]
1466 SAME=.TRUE.;"If control has fallen through, arrays are the same"
1467
1468 :EXITLOOPS:
1469 DO COLBASE=1,NCOL,$COLNO["split columns into groups of $COLNO"
1470   COLTOP = COLBASE+$COLNO-1; "find highest numbered column on current page"
1471   IF (NCOL<COLTOP)[COLTOP=NCOL;] "reduce COLTOP if printing past end"
1472   WRITE(IOUT,50); "blank line"
1473   WRITE(IOUT,100) (FLOAT(ICOL-1)*XSIZE,ICOL=COLBASE,COLTOP+1); "x voxel limits"
1474   DO IROW=1,NROW["write all rows to bottom of grid"
1475     WRITE(IOUT,200) FLOAT(IROW-1)*ZSIZE; "z voxel limit"
1476     WRITE(IOUT,210)(DATA(IROW,ICOL),ICOL=COLBASE,COLTOP); "row data"
1477     IF(.NOT.SAME)["row uncertainties"
```

```
1478        WRITE(IOUT,220) (AMIN1(UNCERT(IROW,ICOL)*100.,99.),
1479                         ICOL=COLBASE,COLTOP);
1480      ]
1481    ]
1482    WRITE(IOUT,200) (FLOAT(NROW)*ZSIZE); "last z voxel limit"
1483  ]
1484  ;
1485  "Output line formats"
1486  %F
1487  50    FORMAT(5X)
1488  100   FORMAT(T4,7F15.3)
1489  200   FORMAT(1X,F12.3,2X,6('|',14('-')),'|')
1490  210   FORMAT(12X,2X,1X,'|',6(1PE12.3,2X,'|'))
1491  220   FORMAT(1X,13X,1X,'|',6(3X,'+/-',F4.1,'%',3X,'|'))
1492  %M
1493  RETURN;
1494  END;"end of GRIDLP"
1495  ;
1496  "End of RTPCART.MOR"
1497  "*****************************************************************************"
```

Appendix C

Output File from RTPCART

```
$!  LOGIN.COM - Dave Murray 1991
$
$ if f$mode() .nes. "BATCH" then goto cont1
$ sys/exquota
$       write sys$output "Executing on a "+f$getsyi("hw_name")
Executing on a VAX 6380
$!  set process/name = "DaveBatch"
$ exit
$ SET DEFAULT phys$:[1219.egs]
$! sys/exquota
$ sys/z
$ assign sys$output for001
$ assign NL: for003
$ assign 10mv.cart for005
$ assign sys$output for006
$ assign NL: for007
$ assign NL: for008
$ assign phys$:[egs4.data]water_15mv_1.dat for012
$ assign dotplot.dat for013
$ assign rawdata.dat for014
$ run phys$:[egs4.rtpcart]rtpcart
EGS4 Monte Carlo Simulation using code RTPCART
University of Waikato, Hamilton, New Zealand.
Running on a VAX...

 Title:
EDK in H2O (10 MeV)

 Material 1:
WATER

 Material 2:
DUMMY

 x voxel size, y voxel size, z voxel size (cm):
  -0.5000  -0.5000  -0.5000
 ...VOXEL AVERAGING will be performed
 MED,RHOR for all voxels:
    1   1.0000

 Beam energy,ECUT,PCUT,SMAX,ESTEPE,ESAVE:
   -1.000      0.561      0.050   1.000E+10     0.0000      0.000

 Input tops of energy bins and bin probabilities,
 one pair per line, lowest energy first - first line EIMIN only,
 ending with bin top energy lower than previous
 Bin#    Energy top    Probability   Cumulative prob
  1      1.000          17.5000          0.1750
  2      2.000          22.0000          0.3950
  3      3.000          20.4000          0.5990
  4      4.000          12.9000          0.7280
  5      5.000           9.2000          0.8200
  6      6.000           7.1000          0.8910
  7      7.000           4.9000          0.9400
  8      8.000           3.2000          0.9720
  9      9.000           1.9000          0.9910
 10     10.000           0.9000          1.0000

 Parallel beam(0),Point source on axis(1),Dose spread array(2):
 2

 X,Y int. centre, Int. depth, smear flag (0,1,2), Dose type (0,1,2):
      5.250       5.250       1.250      2      0

 #Hists,IQIN,IWATCH,TIMMAX(HR),INSEED,IDORAY,NOMSCI,NOPLCI,ICSDA
  200000      0      0   12.00 HRS    123456789      0      0      0      0

 Russian roulette: Z plane,Survival prob, Pathlength param:
    20.0000       0.2000       0.0000

 Call to HATCH at 18:46:25...
EGS SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.
 ...HATCH completed at 18:46:29
*** PRESTA INPUTS ***
IPLC,IBCA,ILCA,IOLDTM,BLCMIN:
          0           0           0           0   4.554
 ---------------------------------------------------------------

PRESTA CALCULATED MINIMUM STEP SIZES FOR MAXIMUM ENERGY ELECTRONS
```

```
                    MEDIUM NO.          t,prime,min for E=Emax
                         1                    0.126E-01 cm
 ------------------------------------------------------------

 ESTEPE =       1.0000
  PLOTIT,MASSFLAG,IDOTPLOT,IRAWDATA:
    1  0  0  2

 EDK in H20 (10 MeV)

 University of Waikato RTPCART (EGS4/PRESTA) 1990      14-AUG-91 18:46:30

 Incident energy=  -1.000 MeV,   charge=  0,   #histories=  200000

                     Bin    Top E    Prob.      Cumulative prob.
                      1     1.000   1.750E-01    1.750E-01
                      2     2.000   2.200E-01    3.950E-01
                      3     3.000   2.040E-01    5.990E-01
                      4     4.000   1.290E-01    7.280E-01
                      5     5.000   9.200E-02    8.200E-01
                      6     6.000   7.100E-02    8.910E-01
                      7     7.000   4.900E-02    9.400E-01
                      8     8.000   3.200E-02    9.720E-01
                      9     9.000   1.900E-02    9.910E-01
                     10    10.000   9.000E-03    1.000E+00

  Rayleigh (coherent) scattering NOT INCLUDED

  Global user cutoffs ECUT,PCUT=       0.561       0.050 MeV

  Material              Density              AE              AP      200*TEFFO
  1  WATER              1.000               0.561           0.05      0.5318
  Electron step size has been modified to reduce energy by   1.000 in each step

  Z voxel,  medium,  density (for central axis voxels)
     1        1              1.0000
     2        1              1.0000
     3        1              1.0000
     4        1              1.0000
     5        1              1.0000
     6        1              1.0000
     7        1              1.0000
     8        1              1.0000
     9        1              1.0000
    10        1              1.0000
    11        1              1.0000
    12        1              1.0000
    13        1              1.0000
    14        1              1.0000
    15        1              1.0000
    16        1              1.0000

  Dose spread array with X,Y centre=(  5.25  5.25),  Origin depth=   1.25/cm**2
  Smear flag =  2  Dose type =  0
  3D voxel smearing performed
  TOTAL DOSE is scored

  Initial random number seed=   123456789

  Input R.N. seed, initial seed =    123456789   123456789

  Russian roulette for photons crossing Z=  20.000 cm with survival= 0.2000
 ------------------------------------------------------------------------------
                    ***** PRESTA INPUTS *****
                              NEW PRESTA PATH-LENGTH CORRECTION USED
                              BOUNDARY CROSSING ALGORITHM INVOKED
                              BLCMIN=    4.554
                              LATERAL CORRELATION ALGORITHM INVOKED
                              NEW TMXS CALCULATION USED

 PRESTA CALCULATED MINIMUM STEP SIZES FOR MAXIMUM ENERGY ELECTRONS
                      MEDIUM NO.          t,prime,min for E=Emax
                           1                    0.126E-01 cm
                    ***** END OF PRESTA INPUTS *****
 ------------------------------------------------------------------------------

 START BATCH  1 TIMES-ELAPSED,CPU,RATIO,RNG      0.0      0.0      0.0   123456789
 START BATCH  2 TIMES-ELAPSED,CPU,RATIO,RNG   2083.2   1954.3      1.1  -714847863
 START BATCH  3 TIMES-ELAPSED,CPU,RATIO,RNG   4109.8   3884.0      1.1  2048606629
 START BATCH  4 TIMES-ELAPSED,CPU,RATIO,RNG   6478.5   5856.1      1.1 -1919513771
 START BATCH  5 TIMES-ELAPSED,CPU,RATIO,RNG   9454.2   7841.8      1.2  -561803307


  FINISHED SIMULATIONS- times elapsed,cpu,ratio= 12675.6  9850.0    1.29

  Simulation times - elapsed,CPU,ratio= 12675.6  9850.0   1.29
  Maximum stack=    8

  Total number of charged particle steps taken=   8915324  NOSCAT=     42436

  Final random no. seed= -1597119315  Final random no.=     0.82131028

  Layer voxels averaged about ( 11, 11)

  EDK in H20 (10 MeV)                                   14-AUG-91 18:46:30
  200000 particles of charge 0, energy=  -1.000 MeV on phantom
```

RTPCART (ECUT,PCUT)=[0.561, 0.050], SMAX= 1.00E+10
CEXPTR= 0.000 CPU= 9850.0 sec

Dose spread array with X,Y centre=(5.25 5.25), Origin depth= 1.25/cm**2
Smear flag = 2 Dose type = 0

TOTAL ENERGY (normalised to incident energy)
Y voxel level used for GRIDLP = 11

	0.000	0.500	1.000	1.500	2.000	2.500	3.000
0.000	2.075E-05 +/-20.1%	1.819E-05 +/- 4.1%	1.222E-05 +/- 5.7%	7.413E-06 +/-12.1%	5.537E-06 +/- 7.0%	3.564E-06 +/-10.8%	
0.500	2.764E-04 +/- 4.6%	1.136E-04 +/- 2.7%	3.092E-05 +/- 6.7%	1.325E-05 +/- 8.0%	7.424E-06 +/-17.2%	5.555E-06 +/- 8.7%	
1.000	1.581E-01 +/- 0.3%	9.155E-03 +/- 0.7%	1.968E-04 +/- 2.6%	2.946E-05 +/- 5.9%	1.223E-05 +/-12.8%	5.212E-06 +/-12.7%	
1.500	1.115E-01 +/- 0.2%	1.735E-02 +/- 0.4%	7.434E-04 +/- 1.7%	7.677E-05 +/- 7.4%	1.986E-05 +/- 9.0%	8.712E-06 +/- 6.7%	
2.000	2.728E-02 +/- 0.6%	9.500E-03 +/- 0.6%	1.147E-03 +/- 1.6%	1.511E-04 +/- 3.9%	3.395E-05 +/- 7.1%	1.058E-05 +/- 7.4%	
2.500	8.148E-03 +/- 0.5%	4.322E-03 +/- 1.0%	1.010E-03 +/- 2.2%	2.066E-04 +/- 1.7%	4.401E-05 +/- 6.7%	1.738E-05 +/- 8.9%	
3.000	2.748E-03 +/- 1.9%	1.816E-03 +/- 1.5%	6.538E-04 +/- 1.5%	1.858E-04 +/- 2.6%	5.091E-05 +/- 3.0%	1.738E-05 +/- 4.5%	
3.500	9.592E-04 +/- 1.4%	7.617E-04 +/- 1.3%	3.675E-04 +/- 2.6%	1.159E-04 +/- 5.3%	4.368E-05 +/- 4.1%	1.964E-05 +/- 9.5%	
4.000	3.967E-04 +/- 1.7%	3.416E-04 +/- 1.2%	1.965E-04 +/- 2.3%	8.950E-05 +/- 3.5%	3.912E-05 +/- 5.6%	1.727E-05 +/-14.7%	
4.500	2.217E-04 +/- 6.7%	1.640E-04 +/- 5.6%	1.026E-04 +/- 4.2%	6.173E-05 +/- 3.6%	2.885E-05 +/- 6.7%	1.591E-05 +/- 8.3%	
5.000	1.351E-04 +/- 7.5%	9.172E-05 +/- 6.5%	6.480E-05 +/- 4.8%	4.306E-05 +/-11.8%	2.081E-05 +/-13.9%	1.687E-05 +/-14.4%	
5.500	7.985E-05 +/- 8.0%	6.993E-05 +/- 7.7%	6.191E-05 +/- 9.5%	3.212E-05 +/- 5.9%	1.721E-05 +/-16.4%	1.756E-05 +/-19.1%	
6.000	6.453E-05 +/-14.1%	5.338E-05 +/- 6.2%	4.509E-05 +/- 8.3%	3.387E-05 +/- 7.7%	2.610E-05 +/- 9.5%	1.676E-05 +/- 9.2%	
6.500	4.035E-05 +/-20.0%	4.625E-05 +/- 6.5%	4.103E-05 +/- 5.7%	3.150E-05 +/- 4.5%	2.225E-05 +/-13.6%	1.620E-05 +/- 9.9%	
7.000	5.105E-05 +/-17.7%	4.143E-05 +/- 6.0%	3.468E-05 +/- 6.8%	2.276E-05 +/- 8.9%	1.489E-05 +/-15.4%	1.040E-05 +/-12.4%	
7.500	3.198E-05 +/-21.6%	3.375E-05 +/-12.1%	2.584E-05 +/-14.3%	2.170E-05 +/- 9.0%	1.862E-05 +/-19.4%	1.147E-05 +/-14.2%	
8.000							

	3.000	3.500	4.000	4.500	5.000	5.500	6.000
0.000	3.053E-06 +/-15.1%	2.078E-06 +/-17.3%	1.458E-06 +/-33.2%	9.211E-07 +/-10.2%	1.125E-06 +/-31.8%	1.173E-06 +/-58.9%	
0.500	4.006E-06 +/-14.8%	3.194E-06 +/-12.3%	1.284E-06 +/-11.7%	1.576E-06 +/-17.9%	1.078E-06 +/-16.1%	4.246E-07 +/-40.0%	
1.000	3.250E-06 +/-14.1%	2.665E-06 +/-21.4%	1.651E-06 +/-24.4%	1.796E-06 +/-23.0%	1.906E-06 +/-24.6%	1.768E-06 +/-19.5%	
1.500	6.318E-06 +/-23.8%	3.207E-06 +/-18.6%	2.876E-06 +/-27.5%	2.284E-06 +/-17.1%	1.985E-06 +/-22.4%	1.584E-06 +/-41.4%	
2.000	7.412E-06 +/-10.3%	3.772E-06 +/-16.0%	3.429E-06 +/-15.1%	2.246E-06 +/-16.4%	1.210E-06 +/-35.4%	7.582E-07 +/-34.1%	
2.500	9.863E-06 +/-12.8%	5.235E-06 +/-18.5%	3.139E-06 +/-22.8%	1.642E-06 +/-23.5%	1.535E-06 +/-30.4%	9.705E-07 +/-36.0%	
3.000	9.910E-06 +/- 7.9%	5.289E-06 +/-15.7%	3.670E-06 +/-22.3%	2.178E-06 +/-29.9%	2.141E-06 +/-29.5%	1.973E-06 +/-22.6%	
3.500	1.133E-05 +/- 9.7%	6.099E-06 +/-13.1%	4.706E-06 +/-16.3%	2.945E-06 +/-23.1%	3.012E-06 +/-20.4%	1.662E-06 +/-43.8%	
4.000	9.934E-06 +/-12.0%	7.992E-06 +/- 8.1%	4.628E-06 +/-12.3%	3.857E-06 +/-15.9%	2.662E-06 +/-26.0%	1.829E-06 +/-37.9%	
4.500	9.275E-06 +/-16.1%	7.675E-06 +/- 9.5%	4.495E-06 +/-18.4%	2.612E-06 +/-17.9%	3.008E-06 +/-18.4%	2.764E-06 +/-29.1%	
5.000	9.752E-06 +/-12.3%	5.858E-06 +/-13.2%	3.577E-06 +/-35.0%	4.070E-06 +/-18.1%	1.864E-06 +/-26.4%	2.411E-06 +/-32.2%	
5.500	9.394E-06	7.711E-06	3.963E-06	3.560E-06	2.654E-06	3.058E-06	

	+/-15.3%	+/-17.7%	+/-32.8%	+/-20.9%	+/-24.0%	+/-39.7%
6.000	1.213E-05 +/-12.9%	7.886E-06 +/- 4.6%	3.659E-06 +/-36.0%	3.887E-06 +/-15.4%	2.077E-06 +/-24.6%	2.661E-06 +/-32.7%
6.500	9.725E-06 +/- 9.0%	1.071E-05 +/-16.7%	5.287E-06 +/-21.9%	4.529E-06 +/-11.4%	2.458E-06 +/-33.4%	3.093E-06 +/-29.8%
7.000	1.241E-05 +/-26.4%	7.990E-06 +/-17.4%	5.148E-06 +/-23.4%	4.950E-06 +/-33.0%	3.375E-06 +/-23.0%	1.913E-06 +/-44.8%
7.500	8.806E-06 +/- 9.7%	4.947E-06 +/-10.0%	6.388E-06 +/-21.7%	3.007E-06 +/-25.9%	3.540E-06 +/-21.9%	2.555E-06 +/-33.2%
8.000						

	6.000	6.500	7.000	7.500	8.000	8.500	9.000
0.000	5.779E-07 +/-68.4%	7.437E-07 +/-66.7%	3.116E-07 +/-25.9%	1.269E-06 +/-43.7%	7.072E-08 +/-99.0%	3.072E-07 +/-51.2%	
0.500	4.005E-07 +/-30.1%	6.249E-07 +/-47.0%	2.286E-07 +/-53.0%	8.613E-10 +/-99.0%	9.961E-07 +/-46.2%	2.202E-07 +/-83.4%	
1.000	5.018E-07 +/-35.1%	1.224E-06 +/-13.7%	3.758E-07 +/-91.1%	8.479E-07 +/-30.6%	0.000E+00 +/- 0.0%	5.459E-07 +/-65.6%	
1.500	7.148E-07 +/-57.9%	3.421E-07 +/-81.7%	5.173E-07 +/-90.6%	7.324E-07 +/-73.5%	5.875E-07 +/-74.7%	8.956E-08 +/-61.3%	
2.000	7.801E-07 +/-34.3%	8.348E-07 +/-36.9%	9.201E-07 +/-34.3%	9.393E-07 +/-52.2%	3.845E-07 +/-93.8%	6.621E-07 +/-60.8%	
2.500	1.370E-06 +/-25.2%	6.019E-07 +/-25.6%	1.140E-06 +/-69.4%	1.601E-06 +/-48.0%	1.720E-06 +/-52.4%	4.967E-07 +/-68.9%	
3.000	9.237E-07 +/-43.6%	6.854E-07 +/-38.2%	1.319E-06 +/-34.4%	1.116E-07 +/-64.8%	1.398E-07 +/-99.0%	2.281E-07 +/-99.0%	
3.500	1.465E-06 +/-23.2%	9.951E-07 +/-50.1%	1.794E-06 +/-20.5%	2.764E-07 +/-57.9%	9.613E-07 +/-49.1%	1.125E-07 +/-67.6%	
4.000	1.814E-06 +/-61.6%	9.502E-07 +/-40.4%	1.213E-06 +/-58.5%	1.453E-06 +/-56.1%	1.494E-06 +/-48.8%	4.413E-07 +/-72.4%	
4.500	2.867E-06 +/-29.3%	2.149E-06 +/-39.9%	1.119E-06 +/-35.5%	2.027E-06 +/-54.1%	2.201E-06 +/-73.2%	3.538E-07 +/-99.0%	
5.000	1.629E-06 +/-56.9%	2.476E-06 +/-48.1%	9.296E-07 +/-50.1%	2.638E-07 +/-80.2%	1.399E-07 +/-99.0%	4.912E-07 +/-99.0%	
5.500	2.545E-06 +/-46.5%	1.310E-06 +/-54.6%	1.491E-06 +/-17.5%	6.099E-07 +/-78.3%	4.555E-07 +/-61.5%	1.319E-06 +/-32.8%	
6.000	1.482E-06 +/-27.3%	3.679E-06 +/-27.3%	6.071E-07 +/-54.4%	4.152E-07 +/-46.4%	3.822E-08 +/-61.7%	1.308E-06 +/-73.2%	
6.500	2.675E-06 +/-32.0%	2.751E-06 +/-46.0%	8.478E-07 +/-51.2%	0.000E+00 +/- 0.0%	1.729E-06 +/-49.0%	1.869E-06 +/-32.8%	
7.000	1.699E-06 +/-67.0%	1.437E-06 +/-51.4%	2.437E-06 +/-18.7%	3.823E-07 +/-67.6%	2.213E-06 +/-44.4%	2.935E-07 +/-58.5%	
7.500	1.240E-06 +/-38.1%	2.777E-06 +/-56.1%	1.220E-06 +/-46.4%	1.112E-06 +/-64.6%	3.191E-07 +/-65.9%	1.277E-06 +/-67.2%	
8.000							

```
IRAWDATA SIZE IS (   18   16)
Y voxel plane is  11

 EDK in H20 (10 MeV)                                    14-AUG-91 18:46:30
 200000 particles of charge 0, energy=  -1.000 MeV on phantom

 RTPCART (ECUT,PCUT)=[  0.561, 0.050], SMAX= 1.00E+10
 CEXPTR= 0.000   CPU=  9850.0 sec

 Dose spread array with X,Y centre=(  5.25  5.25),  Origin depth=   1.25/cm**2
 Smear flag =  2  Dose type =  0

 Total incident Energy =   21976.6955 MeV, Phantom energy =   14293.3727 MeV

 ...end of file read          14-AUG-91 22:17:53

 ...end of file read     14-AUG-91 22:17:53     CPU=    9858.8 sec=   2.739 hrs
FORTRAN STOP
$ if f$mode().eqs."INTERACTIVE" then goto LAST
$ rename phys$:[1219]rtpcart.log phys$:[1219.egs]rtpcart.log
$LAST:
$ EXIT
  PHYS1219     job terminated at 14-AUG-1991 22:17:56.28
  Accounting information:
  Buffered I/O count:            85       Peak working set size:  2114
  Direct I/O count:             695       Peak page file size:    4965
  Page faults:                 2393       Mounted volumes:           0
  Charged CPU time:     0 02:44:20.79   Elapsed time:     0 03:31:36.69
```

Appendix D

EGS4 User Code RTPEDK

```
"     RTPEDK.MOR                                                          "
"                                                                         "
"     This is a modified version of the PRESTA-based INHOMP               "
"     code. Modified by David C. Murray, University of                    "
"     Waikato, Hamilton, New Zealand                                      "
"     This code simulates a SPHERICAL scoring geometry                    "
"     It is essentially a cut-down version of RTPCYL                      "
"     (ISOURC=6), with modifications to region and scoring                "
"     boundaries.                                                         "
"                                                                         "
"     This is a PRESTA code                                               "
"     It requires NRC4MACP.MOR from unit 8 and                            "
"     NRCCAUXP.MOR from unit 10 (EGS4.MOR unit switches                   "
"     should be %U2,%U10,%U3,%U8,%U4)                                     "
"     Note, NRCCAUX.MOR contains many extra geometry routines             "
"     for cylinders which one may wish to edit out to avoid               "
"     lots of linker warnings                                             "

;REPLACE {$MXDATA} WITH {1}
"Default for sigma routine in EGS4.MOR at NRCC"
!INDENT F 2;
%E     "RTPEDK.MOR"
%C80    use 80 col of input
"*************************************************************************"
"                                                                         "
"                              ****************                           "
"                              *              *                           "
"                              *  RTPEDK.MOR  *                           "
"                              *              *                           "
"                              ****************                           "
"                                                                         "
"        This code simulates a point source of photons forced to          "
"        interact at the origin of an infinite medium. Energy deposited is"
"        scored in a spherical geometry, with constant R and THETA increments."
"        For incident photons it calculates:                              "
"        -the total dose (IT=1)                                           "
"        -primary dose (IT=2) - DCM                                       "
"        -secondary dose (IT=3) - DCM                                     "
"                                                                         "
"        INHOMP was originally written by                                 "
"        David Rogers NRCC Ottawa K1A OR6  (613)993-2715                  "
"        Code is written in MORTRAN3 and is a user code for EGS4 (see     "
"        SLAC report 265, Nelson, Hirayama and Rogers)                    "
"        Use of this code or its  close relatives is described in         "
"        Health Physics 46 (1984) 891-914 and                             "
"        Nuclear Instruments and Methods A227 (1984) 535-548              "
"                                                                         "
"   Unit assignments                                                      "
"           unit 1    output listing file                                 "
"           unit 5    input file (or keyboard)                            "
"           unit 6    prompts for input and echos input                   "
"           unit 8    echos media input from pegs(assign to null file)    "
"           unit 12   input file with crossection data from pegs          "
"           unit 13   output file for DOTPLOT data - D.C.M.               "
"           unit 14   output file for RAWDATA data - D.C.M.               "
"                                                                         "
"    The code can be run interactively, or in batch with the following    "
"    input file:                                                          "
;
"     Input file 'P1'.INP                                                 "
"     ===================                                                 "
"                                                                         "
"Card 1                                                                   "
"        title        80A1                                                "
"Card 2                                                                   "
"        material 1     24A1                                             "
"Card 3                                                                   "
"        RHOR                                                             "
"                RHOR    medium density                                   "
"Card 4                                                                   "
"        NRADIALS,DELTAR,NTHETAS,NRADSCAT,DELRSCAT,NTHSCAT                "
"                NRADIALS  number of radial scoring regions               "
"                DELTAR    distance (in cm) between radial scoring boundaries"
"                NTHETAS   number of angular scoring regions (in 180 degrees)"
"                For scattered dose (if different from primary):          "
"                NRADSCAT  number of radial scoring regions               "
"                DELRSCAT  distance (in cm) between radial scoring boundaries"
"                NTHSCAT   number of angular scoring regions (in 180 degrees)"
"Card 5                                                                   "
"        E(kinetic energy),ECUT,PCUT,SMAX,ESTEPE,ESAVE                    "
```

```
"                E       KINETIC energy of incident beam                          "
"                ECUT    electron cut off in Mev-total energy                     "
"                PCUT    photon cut off in Mev                                    "
"                        for E<=0, input a spectrum on cards 6A (below)           "
"                SMAX    max step size in cm                                      "
"                ESTEPE  fractional energy loss per electron step                 "
"                        if left 0, EGS uses its own default step size            "
"                        if non-zero, step size is chosen to reduce E by this     "
"                        fraction                                                 "
"                ESAVE   electrons below this energy will be thrown out if they"
"                        cannot escape from their current region.                 "
"                                                                                 "
"Card 6A   if E<=0.0   input an energy spectrum                                   "
"       first card   EIMIN   lowest KINETIC energy in spectrum                    "
"       EBIN(I),EPHI(I)                                                           "
"                EBIN(I)  top of energy bin I                                     "
"                EPHI(I)  probability of energy being in this bin                 "
"                           (not necessarily normalized)                          "
"                end with EBIN(I) < EBIN(I-1)   or zero                           "
"                                                                                 "
"Card 7                                                                           "
"     NCASE,IWATCH,TIMMAX,IDORAY,INSEED                                           "
"                NCASE   # histories to run                                       "
"                IWATCH = 0 for normal output                                     "
"                       = 1 to print all interactions                             "
"                       = 2 to print every electron step                          "
"                TIMMAX  maximum cpu time allowed for job in hours                "
"                        job terminates with as many batches as possible within"
"                        this time limit                                          "
"                        default = 1 hour                                         "
"                IDORAY  =1, include Rayleigh (coherent scatter) = 0, dont        "
"                INSEED  initial seed to random number generator- 0 is ok         "
;
"Card 8                                                                           "
"        RRR,RRCUT,CEXPTR                                                         "
"                for Russian roulette - as any photon crosses the R=RRR sphere "
"                russian roulette is played with a probability of survival =      "
"                RRCUT - weight increases by 1/RRCUT if it survives               "
"                if blank or both zero, no Russian roulette is played             "
"                CEXPTR parameter for pathlength biasing                          "
"                   <0 for shortening                                             "
"                    if 0.0, no biasing done                                      "
"                                                                                 "
"Card 9   (PRESTA inputs)                                                         "
"   IPLC,IBCA,ILCA,IOLDTM,BLCMIN                                                  "
"       all zeros is fine for a default PRESTA run - see the PRESTA write up      "
"       for details on other values possible                                      "
"       to obtain a pure EGS4 run, insert a line   1,1,1,1                        "
"       if the 4th 1 is changed to zero, the improved FIXTMX algorithm is         "
"            used                                                                 "
"       if NOPLCI is non-zero, i.e. then IPLC should be -1 to signal              "
"       no PLC is to be used.  NOPLCI will over-ride and set IPLC to -1           "
"                                                                                 "
"        Definitions of regions                                                   "
"                                                                                 "
"Card 10 (Output options) - D.C.M.                                                "
"       ITOT,IPRI,ISCAT,IDOTPLOT                                                  "
"         ITOT,IPRI,ISCAT                                                         "
"             =0          not output to raw data file                           "
"             =1          output to raw data file                               "
"          IDOTPLOT       0 = no DOTPLOT output                                 "
"                         1 = output IQ,X,Z to unit IDOTFILE                    "
"                         2 = output IQ,X,Z,U,W,USTEP to unit IDOTFILE          "
"                         3 = if primary, output IQ,X,Z to unit IDOTFILE        "
"                         4 = if primary, output IQ,R,Z to unit IDOTFILE        "
"                                                                                 "
"        Some variables                                                           "
"        ==============                                                           "
"                                                                                 "
"COMMON/SCORE/                                                                    "
"        MXNP                  maximum value of stack pointer-checked in AUSGAB  "
"        DOSEIS(IRADIAL,ITHETA,IT,IS) the energy in mev deposited in zone       "
"                         (IRADIAL,ITHETA) for batch IS.                          "
"                         IT=1        total energy deposited                      "
"                         IT=2        primary energy                              "
"                         IT=3        scattered energy                            "
"        IWATCH           =0 normal output                                        "
"                         =1 prints detail of every discrete interaction          "
"                         =2 print detail of every step taken                     "
"        IS               current batch number                                    "
"        IQIN             charge of incident particle  (=0)                       "
"        ISTEPA           number of charged particle steps taken                  "
;"
"COMMON/GEOM/                                                                     "
"        R                radius of current particle                             "
"        TH               angle of current particle                              "
"        NRADIALS         number of radial regions                               "
"        DELTAR           distance (in cm) between radial regions                "
"        NTHETAS          number of angular regions (in 180 degrees)             "
"        NRADSCAT         number of radial regions - scattered E                 "
"        DELRSCAT         distance (in cm) between radial regions - scattered E "
"        NTHSCAT          number of angular regions (in 180 deg) - scattered E   "
"        RMAXNEEDED       used in HOWFAR for discard test                        "
"                                                                                 "
```

```
"Non-common variables                                                          "
"        AINFLU          incident fluence                                      "
"                        parallel beam  = NCASE/area of beam                   "
"                        point source =NCASE/area of sphere -i.e. larger       "
"        DOSE(IRADIAL,ITHETA,IT)-average energy deposited per batch -each zone "
"        DOSEIT(IRADIAL,ITHETA)  value of dose for a given IT-used for output  "
"        UNCER(IRADIAL,ITHETA,IT)  fractional uncertainty on dose-calculated   "
"                        as uncertainty on mean from 10 batches                "
"        ITOT,IPRI,ISCAT flags for outputting energy categories                "
"        RBOUND,THETABOUND  array for storing boundary coords (used by GRIDLP) "
"        IDOTPLOT        flag for not plotting/plotting interactions           "
"        IDOTFILE        unit number for DOTPLOT output - D.C.M.               "
"        IRAWFILE        unit number for raw data output - D.C.M.              "
"        TOTALMEV        total incident K.E. for DSA - DCM                     "
"                                                                              "
"        Installation specific routines                                        "
"        ==============================                                        "
"                                                                              "
"        This code was developed on a Vax system using Vax fortran 77          "
"        This means expressions are used in output lists plus a few other      "
"          non-standard fortran 4 features -                                   "
"        Several other features related to timing routines, time and date      "
"        routines and a routine to set up an la120 printer have also been      "
"        used.  Just comment out all lines with references to:                 "
"        TIMEN, DATEN, SECNDS, CPUTIME or LA120                                "
"        They do not affect the simulation at all                              "
"        T=SECNDS(T0)      returns time in seconds since T0                    "
"        CALL CPUTIME(IT) returns IT(INTEGER*4)=task cpu time in 10 millisec   "
"        CALL LA120        sends an escape sequence to set up la120 printer    "
"                                                                              "
"       The code assumes that the NRCC4 macros are used and that the routines  "
"        WATCH and FIXTMX are linked with it                                   "
"                                                                              "
;
"******************************************************************************"
"                                                                              "
;"User set values"

%I
;
PARAMETER        $MXMED = 1 ;      "$media"
PARAMETER        $MXREG = 1 ;      "#regions"
PARAMETER        $STAT = 10  ;      "# bins for uncertainty analysis"
PARAMETER        $MXSTACK = 20 ;  "maximum stack"
PARAMETER        $MAXR = 200 ;     "max # radial energy scoring zones"
PARAMETER        $MAXIR = 201 ;    "$MAXR+1 max no. of boundaries allowed"
PARAMETER        $MAXTHETA = 36 ; "max # angular energy scoring zones"
PARAMETER        $MXITHETA = 37 ; "$MAXTHETA+1"
PARAMETER        $MAXIT = 3 ;      "max number of scoring categories"
PARAMETER        $COLNO = 7 ;      "number of cols across page in GRIDLP"
%I4

"Common block definitions"

REPLACE {;COMIN/SCORE/;} WITH
"               ============="
{;COMMON/SCORE/MXNP,DOSEIS($MAXR,$MAXTHETA,$MAXIT,$STAT),IWATCH,IS,ISTEPA,
IQIN,IDOTPLOT,IDOTFILE;}

REPLACE {;COMIN/GEOM/;} WITH
"               ============="
{;COMMON/GEOM/NRADIALS,DELTAR,NTHETAS,NRADSCAT,DELRSCAT,
NTHSCAT,R,TH,RMAXNEEDED;}

REPLACE {;COMIN/ESPECT/;} WITH
"               ==============="
{;COMMON/ESPECT/ECPD(50),EBIN(50),EPHI(50),EIMIN;}
" above to input an energy spectrum for the source"
"ECPD(I) is the probability of the source energy being"
"less than EBIN(I)  - i.e. cumulative probability distribution"

REPLACE {;COMIN/RUSROU/;} WITH
"               ==============="
{;LOGICAL RUSROU;COMMON/RUSROU/RRR,RRCUT,RUSROU;}
"RUSROU is the common for the Russian roulette in AUSGAB/HOWFAR"

"The following macro is needed for the PRESTA implementation          "
"in this geometry.  It is only called in ELECTR and it assumes        "
"COMMON GEOM is available.

REPLACE {$CALL-HOWNEAR(#);} WITH
"               ================"
"There is a small problem here with rounding due to the conversion  "
"from radians to degrees, manifesting as TVSTP2<VSTEP2 (prohibited)."
"This is fixed by subtracting a small constant from P1 - DCM, 1991  "
"Note  DNEAR={P1}. Try making"
{;IF ($FLAG1=1)["primary"
    RFRAC=AMOD(R,DELTAR);
    THFRAC=AMOD(TH,180.0/NTHETAS);
    {P1}=MIN(MIN(RFRAC,DELTAR-RFRAC),
           R*SIN(MIN(THFRAC,(180.0/NTHETAS)-THFRAC)*
           (3.14159265358979/180.0))) - 1.0E-6;]
  ELSEIF ($FLAG1>1) ["scattered"
```

```
268     RFRAC=AMOD(R,DELRSCAT);
269     THFRAC=AMOD(TH,180.0/NTHSCAT);
270     {P1}=MIN(MIN(RFRAC,DELRSCAT-RFRAC),
271             R*SIN(MIN(THFRAC,(180.0/NTHSCAT)-THFRAC)*
272             (3.14159265358979/180.0))) - 1.0E-6;]
273 ;}
274
275 "This macro replaces NRCC version. It performs forcing at the origin,      "
276 "with pathlength biasing -CEXPTR. It does an exponential transformation    "
277 "of the photon pathlength for forward-going photons
278 REPLACE {$SELECT-PHOTON-MFP;} WITH
279 "         ========================="
280 {$RANDOMSET RNNO35;  IF(RNNO35 = 0.0)[RNNO35=1.E-30;]
281 DPMFP=-ALOG(RNNO35);
282 IF ((X(NP).EQ.0.0).AND.(Y(NP).EQ.0.0).AND.(Z(NP).EQ.0.0)
283   .AND.(W(NP).EQ.1.0))
284   [DPMFP=0.0;]
285 ELSEIF(CEXPTR.NE.0.0)[IF(W(NP)>0.0)[TEMP=CEXPTR*W(NP);BEXPTR=1./(1.-TEMP);
286    DPMFP=DPMFP*BEXPTR;WT(NP)=WT(NP)*BEXPTR*EXP(-DPMFP*TEMP);]];}
287
288 "The following macros control machine-dependent code. Options are VAX and "
289 "TRANSPUTER - D.C.M.                                                       "
290 "-------------------------------------------------------------------------"
291 REPLACE {$VAX;} WITH {NOGENERATE;}
292 REPLACE {$TRANSPUTER;} WITH {GENERATE;}
293 "-------------------------------------------------------------------------"
294
295 "VAX commands"
296 $VAX;
297   REPLACE {CALL LA120(*);} WITH {CONTINUE;}
298   REPLACE {$MACHINE} WITH {'VAX'}
299   REPLACE {$TPUTER-RNG-INIT;} WITH {CONTINUE;}
300 ENDGENERATE;
301
302 "TRANSPUTER commands"
303 $TRANSPUTER;
304   "Note that PRESTA-MSCAT-1 macro override creates two linker warnings - DCM"
305   REPLACE {CALL CPUTIME(*);} WITH {CALL ICLOCK({P1});{P1}={P1}*100;}
306   REPLACE {CALL LA120(*);} WITH {CONTINUE;}
307   REPLACE {CALL TIME(*);} WITH {CONTINUE;}
308   REPLACE {CALL DATE(*);} WITH {CONTINUE;}
309   REPLACE {SECNDS(*)} WITH {0;}
310   REPLACE {LOGICAL*1} WITH {LOGICAL}
311   REPLACE {INTEGER*2} WITH {INTEGER}
312   REPLACE {INTEGER*4} WITH {INTEGER}
313   REPLACE {REAL*4} WITH {REAL}
314   REPLACE {REAL*8} WITH {DOUBLE PRECISION}
315   REPLACE {CALL EXIT;} WITH {STOP;}
316   REPLACE {EXIT;} WITH {STOP;}
317   "Replace RNG with Fibonacci sequence"
318   REPLACE {$TPUTER-RNG-INIT;} WITH {$FIBONACCI-RNG-INIT;}
319   REPLACE {$RANDOMSET*;} WITH {$FIBONACCI-GENERATOR{P1};}
320   REPLACE {;COMIN/RANDOM/;} WITH
321           {;COMMON/RANDOM/URNG(97),CRNG,CDRNG,CMRNG,IRNG,JRNG,IXX;}
322   REPLACE {;COMMON/RANDOM/IXX;} WITH
323           {;COMMON/RANDOM/URNG(97),CRNG,CDRNG,CMRNG,IRNG,JRNG,IXX;}
324   "Deactivate graph plotting options"
325   "Avoid linker warnings by replacing with known variables"
326   REPLACE {CYRAD2(*)} WITH {X(NP)}
327   REPLACE {ZPLANE(*)} WITH {X(NP)}
328   "Remove carriage control formats on output statements"
329   REPLACE {OUTPUT;('$*);} WITH {OUTPUT;('{P1}');}
330   REPLACE {OUTPUT;('0*);} WITH {OUTPUT;(' ');OUTPUT;('{P1}');}
331   REPLACE {OUTPUT;(' *);} WITH {OUTPUT;('{P1}');}
332   "Indicate machine"
333   REPLACE {$MACHINE} WITH {'TRANSPUTER'}
334 ENDGENERATE;
335
336 "Fibonacci RNG initialising routine. IRNGSEED is set to IXX, while the "
337 "other three seeds are set to the values below - D.C.M."
338 REPLACE {$FIBONACCI-RNG-INIT;} WITH
339   {
340   "Initialise transputer RNG - D.C.M."
341   "IRNGSEED,JRNGSEED,KRNGSEED,LRNGSEED are RNG seeds - D.C.M."
342   JRNGSEED=30;
343   KRNGSEED=61;
344   LRNGSEED=121;
345   "Check IXX is in range 1..168"
346   IF (IXX.GT.168) [IXX = MOD(IXX,168)+1;]
347   IF (IXX.LT.1)   [IXX = MOD(-IXX,168)+1;]
348   IRNGSEED=IXX;
349   DO II=1,97[
350     SRNG=0.0;
351     TRNG=0.5;
352     DO JJ=1,24[
353       MRNG=MOD(MOD(IRNGSEED*JRNGSEED,179)*KRNGSEED,179);
354       IRNGSEED=JRNGSEED;
355       JRNGSEED=KRNGSEED;
356       KRNGSEED=MRNG;
357       LRNGSEED=MOD(53*LRNGSEED+1,169);
358       IF (MOD(LRNGSEED*MRNG,64).GE.32) [SRNG=SRNG+TRNG;]
359       TRNG=TRNG*0.5;
360     ]
361     URNG(II)=SRNG;
362   ]
363   CRNG=362436.0/16777216.0;
```

```
364   CDRNG=7654321.0/16777216.0;
365   CMRNG=16777213.0/16777216.0;
366   "Set IRNG,JRNG to 97,33 for first call to RNG"
367   IRNG = 97;
368   JRNG = 33;
369   }
370 ;
371 "Fibonacci RNG. After generation of the random number IXX is set to that "
372 "value so that the user will see different 'seeds' - D.C.M."
373 "This generator requires 4 initial seeds to generate a further 97 seeds. "
374 "It has a very long cycle - D.C.M."
375 REPLACE {$FIBONACCI-GENERATOR#;} WITH
376   {
377   {P1} = URNG(IRNG)-URNG(JRNG);
378   IF ({P1}.LT.0.0)[{P1}={P1}+1.0;]
379   URNG(IRNG) = {P1};
380   IRNG = IRNG - 1;
381   IF (IRNG.EQ.0)[IRNG=97;]
382   JRNG = JRNG - 1;
383   IF (JRNG.EQ.0)[JRNG=97;]
384   CRNG=CRNG-CDRNG;
385   IF (CRNG.LT.0.0)[CRNG=CRNG+CMRNG;]
386   {P1} = {P1} - CRNG;
387   IF ({P1}.LT.0.0)[{P1}={P1}+1.0;]
388   "Debugging option"
389   "IF (({P1}.LT.0.0) .OR. ({P1}.GT.1.0))[error - log values          "
390   "  OUTPUT {P1},CRNG,CDRNG,CMRNG,IRNG,JRNG,URNG(IRNG+1),URNG(JRNG+1);"
391   "  (' RN,CRNG,CDRNG,CMRNG,IRNG,JRNG,URNG(IRNG+1),URNG(JRNG+1) = ',  "
392   "  F12.8,F12.8,F12.8,F12.8,I12,I12,F12.8,F12.8);]                   "
393   IXX = INT({P1}*10000.0);  "update seed for printout"
394   }
395
396 "Data declarations"
397 ;LOGICAL*1  DATEN(9),TIMEN(8),TITLE(80); "Vax specific"
398 INTEGER*4 CPUTO,CPUT1,CPUT2,CPUT3,CPUT4;
399 REAL*4 UNCER($MAXR,$MAXTHETA,$MAXIT),DOSE($MAXR,$MAXTHETA,$MAXIT),
400 DOSEIT($MAXR,$MAXTHETA),UNCERT($MAXR,$MAXTHETA),
401 RBOUND($MAXIR),THETABOUND($MXITHETA);
402 "Variable for total incident K.E."
403 REAL*8 TOTALMEV;
404 "Flags for controlling output -DCM."
405 INTEGER*4 ITOT,IPRI,ISCAT,IRAWFILE;
406
407 "Note comin PHOTIN added to allow access to GMFP for DSA calcs - DCM"
408 ;COMIN/EPCONT,MEDIA,GEOM,SCORE,THRESH,BOUNDS,MISC,ELECIN,RANDOM,
409 STACK,MULTS,RUSROU,USER,USEFUL,ESPECT,PHOTIN/;
410 IOUT=1;"OUTPUT LISTING UNIT"
411 IDOTFILE=13;"Output listing unit for DOTPLOT file - D.C.M."
412 IRAWFILE=14;"Output listing unit for RAWDATA file - D.C.M."
413 TOTALMEV = 0.0;  "initialise total incident energy/weighted E counter -DCM"
414 CALL CPUTIME(CPUTO);"initial cpu time taken by this task so far"
415
416 OUTPUT $MACHINE ; (' Running on a ',A);
417 $TRANSPUTER;
418   "Opening of input files for transputer - D.C.M."
419   OPEN(UNIT=1,FILE='FORO01.DAT',STATUS='UNKNOWN'); "program output"
420   OPEN(UNIT=5,FILE='FORO05.INP',STATUS='OLD');     "parameter input"
421   OPEN(UNIT=7,STATUS='SCRATCH');                   "echos media input"
422   OPEN(UNIT=8,STATUS='SCRATCH');                   "echos media input"
423   OPEN(UNIT=12,FILE='FORO12.INP',STATUS='OLD');    "PEGS input data"
424   OPEN(UNIT=13,FILE='FORO13.DAT',STATUS='UNKNOWN');"DOTPLOT output"
425   OPEN(UNIT=14,FILE='FORO14.DAT',STATUS='UNKNOWN');"RAWDATA output"
426 ENDGENERATE;
427
428 "                         Read inputs           "
429 "                         ===========           "
430
431 "Card 1"
432 ;OUTPUT;(//'OTitle:');
433 READ(5,100,END=:END:)TITLE;OUTPUT TITLE;(1X,80A1);
434
435 "Card 2"
436 OUTPUT;('$ Material: ');READ(5,120,END=:END:)(MEDIA(J,1),J=1,24);
437 OUTPUT(MEDIA(J,1),J=1,24);(1X,24A1);
438
439 "Card 3"
440 ;OUTPUT;('$RHOR (density): ');
441 READ(5,110,END=:END:)RHOR; OUTPUT RHOR;('+'F9.4);
442
443 "Card 4"
444 ;OUTPUT;('$NRADIALS,DELTAR,NTHETAS,NRADSCAT,DELRSCAT,NTHSCAT: ');
445 READ(5,105,END=:END:)NRADIALS,DELTAR,NTHETAS,NRADSCAT,DELRSCAT,NTHSCAT;
446 IF(NRADIALS > $MAXR)[NRADIALS=$MAXR;]
447 IF(NTHETAS > $MAXTHETA)[NTHETAS=$MAXTHETA;]
448 IF(NRADSCAT=0.0)[NRADSCAT=NRADIALS;]
449 IF(DELRSCAT=0.0)[DELRSCAT=DELTAR;]
450 IF(NTHSCAT=0.0)[NTHSCAT=NTHETAS;]
451 IF(NRADSCAT > $MAXR)[NRADSCAT=$MAXR;]
452 IF(NTHSCAT > $MAXTHETA)[NTHSCAT=$MAXTHETA;]
453 OUTPUT NRADIALS,DELTAR,NTHETAS,NRADSCAT,DELRSCAT,NTHSCAT;
454 (I6,F9.4,I6,I6,F9.4,I6);
455
456 "Card 5"
457 OUTPUT;('$Beam energy,ECUT,PCUT,SMAX,ESTEPE,ESAVE: ');
```

```
458 READ(5,110) EIN,ECUTIN,PCUTIN,SMAX,ESTEPE,ESAVIN;
459 IF(SMAX <= 0.0)[
460   IF(SMAXIR(1)=0.0)[SMAXIR(1)=DELTAR;SMAX=1.E10;]
461   ]
462 ELSE[
463   IF(SMAXIR(1)=0.0)[SMAXIR(1)=SMAX;]
464   ELSE[SMAXIR(1)=AMIN1(SMAXIR(1),SMAX);]
465   ]
466 OUTPUT EIN,ECUTIN,PCUTIN,SMAX,ESTEPE,ESAVIN;(3F10.3,1PE12.3,0PF10.4,F10.3);
467 "Set default energy cutoffs where needed"
468 ECUTMN = 1.E10;   "need to find minimum ECUT used for PRESTA"
469 IF(ECUT(1) = 0.0)[ECUT(1)=ECUTIN;]
470 ECUTMN=ECUTIN;
471 ECUTMN = MIN(ECUTMN,ECUT(1));
472 IF(PCUT(1) = 0.0)[PCUT(1)=PCUTIN;]
473 IF(ESAVE(1) = 0.0)[ESAVE(1)=ESAVIN;]
474 ESTEPR(1) = 1.0; "Set ESTEPE scaling to 1.0 (default)"
475
476 "Card 6A   only if EIN<=0.0 which means input spectrum"
477 IF(EIN<=0.0)[
478 OUTPUT;('0Input tops of energy bins and bin probabilities'/
479 ' one pair per line, lowest energy first - first line EIMIN only'/
480 ' end with bin top energy lower than previous'/);
481 I=0; INPUT EIMIN;(F10.0);"lowest energy in spectrum"
482 LOOP [I=I+1;
483 INPUT EBIN(I),EPHI(I);(2F10.0);
484 IF(I.NE.1 .AND. EBIN(I).LE.EBIN(I-1))[EXIT;"end of input spectrum"]
485 ELSEIF (I.EQ.1 .AND. EBIN(I).LE.EIMIN) [EXIT;"error probably"]
486 ] "end of loop"
487 NEBIN=I-1; "number of energy bins read in"
488 TOTPHI=0.0;DO I=1,NEBIN [TOTPHI=TOTPHI+EPHI(I);]"spectrum total"
489 ECPD(1) = EPHI(1)/TOTPHI;
490 DO I=2,NEBIN [ ECPD(I) = ECPD(I-1) + EPHI(I)/TOTPHI;]
491 OUTPUT (I,EBIN(I),EPHI(I),ECPD(I),I=1,NEBIN);
492 (' Bin #   Energy top    Probability   Cumulative prob'/(I4,F10.3,2F16.4));
493 ]"end of input energy spectrum"
494
495 "Card 7"
496 OUTPUT;
497 ('$#Histories,IWATCH,TIMMAX(HR),IDORAY,INSEED');
498 READ(5,130,END=:END:)NCASE,IWATCH,TIMMAX,IDORAY,INSEED;
499 IF(TIMMAX = 0.0)[TIMMAX=1.0;"default time limit is 1 hour"]
500 IF(INSEED ^= 0)[IXX=INSEED;"IXX from COMIN RANDOM"
501 "IXX initialized in HATCH if not here"]
502 OUTPUT NCASE,IWATCH,TIMMAX,IDORAY,INSEED;
503 (1X,I8,I6,F8.2,' HRS',I6,I12);
504 IF(IDORAY ^= 0)[DO J=1,$MXREG[IRAYLR(J)=1;]]
505 IF(IWATCH = 0)["don't call AUSGAB unless needed"
506 DO J=6,25[IAUSFL(J)=0;]]
507 "Call AUSGAB for P.E., pair and Compton events (IARG=19,18,15)"
508 /IAUSFL(20),IAUSFL(19),IAUSFL(16)/=1;
509 "The call WATCH(-99,IWATCH) changes these for IWATCH non-zero"
510
511 "Card 8"
512 OUTPUT;(/'$Russian roulette: R plane,Survival prob, Pathlength param: ');
513 READ(5,110,END=:END:) RRR,RRCUT,CEXPTR;
514 OUTPUT RRR,RRCUT,CEXPTR;(3F12.4);
515 RUSROU=.FALSE.;IF(RRR+RRCUT ^= 0.0)[RUSROU=.TRUE.;]
516
517 "Card 9    PRESTA inputs"
518 EI=EIN; "get total energy"
519 IF(EIN <= 0.0)[EKO = EBIN(NEBIN);] ELSE [EKO = EIN;]
520 "PRESTA requires the max K.E. of electron"
521
522 "                Init media and call HATCH            "
523 "                =========================            "
524 ;MED(1)=1;
525 DUNIT=1;
526 CALL TIME(TIMEN);"a DEC specific routine returning the time of day"
527 OUTPUT TIMEN;('0CALL TO HATCH AT  ',8A1);
528 CALL HATCH;
529 CALL TIME(TIMEN);OUTPUT TIMEN;('0HATCH COMPLETED AT ',8A1);
530
531 "        Check media data covers energy range requested"
532 IF(EIN>UP(1)   |  EIN>UE(1)-0.511)[
533   OUTPUT 1,EIN,UP(1),UE(1);(//1X,80('*')//' FOR MEDIUM',
534   I3,' INCIDENT ENERGY=',F10.3,' MEV'/' IS GREATER THAN',
535   ' COVERED BY DATA FILE WHERE UP,UE=',2F10.3,' MEV'//
536   1X,80('*')//);
537 ]
538 "Check cross sections are available for Rayleigh scattering if requested"
539 IF(IDORAY ^= 0  &  IRAYLM(1) ^= 1)["data not there"
540   OUTPUT 1;(//1X,80('*')/'ORAYLEIGH SCATTERING DATA NOT THERE FOR',
541   ' MATERIAL',I3,'  GIVE UP');
542 ]
543
544 "CARD 9    PRESTA Inputs"
545 $PRESTA-INPUTS;
546
547 "        Change electron step sizes if requested                    "
548 "                                                                   "
549 IF(ESTEPE > 0.0)["Replace EGS4 values of TMXS with ETRAN like"
550   "constant fractional energy loss steps -note 200*TEFFO restrictions are"
551   "still active"
```

```
552   CALL FIXTMX(ESTEPE,1);
553 ]
554
555 "Card 10       Output options                                          "
556 OUTPUT;(/'$ITOT,IPRI,ISCAT,IDOTPLOT: ');
557 READ(5,115,END=:END:) ITOT,IPRI,ISCAT,IDOTPLOT;
558 OUTPUT ITOT,IPRI,ISCAT,IDOTPLOT;(4I3);
559 RUSROU=.FALSE.;IF(RRR+RRCUT ~= 0.0)[RUSROU=.TRUE.;]
560
561 "                Initialize scoring arrays and set some constants   "
562 "                ================================================   "
563 IRIN=1; "source is in region 1"
564 WTIN=1.;"initial weight always 1"
565 XIN=0.0;YIN=0.0;ZIN=0.0;UIN=0.0;VIN=0.0;WIN=1.0;
566 IQIN=0; "incident particles are photons"
567 JCASE = NCASE/$STAT;NCASE=JCASE*$STAT;ISTEPA=0;
568 RMAXNEEDED = MAX(NRADIALS*DELTAR,NRADSCAT*DELRSCAT);
569
570 MXNP=0;
571 DO IS=1,$STAT[
572   DO IT=1,$MAXIT[
573     DO ITHETA=1,$MAXTHETA[
574       DO IRADIAL=1,$MAXR[
575         DOSEIS(IRADIAL,ITHETA,IT,IS)=0.0;
576 ]]]]
577
578 "Calculate incident fluence"
579 AINFLU=FLOAT(NCASE);
580
581 "                                                  "
582 "                Print summary of input            "
583 "                ======================            "
584 CALL DATE(DATEN);CALL TIME(TIMEN);WRITE(IOUT,140)TITLE,DATEN,TIMEN;
585 WRITE(IOUT,150) EIN,IQIN,NCASE;
586 IF(EIN<=0.0)[ WRITE(IOUT,151)(I,EBIN(I),EPHI(I)/TOTPHI,ECPD(I),I=1,NEBIN);]
587 IF(IDORAY ~= 0)[WRITE(IOUT,152);]ELSE[WRITE(IOUT,153);];
588 WRITE(IOUT,155) ECUTIN,PCUTIN;
589 WRITE(IOUT,160) 1,(MEDIA(J,1),J=1,24),RHO(1),AE(1),AP(1),200*TEFFO(1);
590 IF(ESTEPE ~= 0.0)[WRITE(IOUT,165) ESTEPE;]
591 WRITE(IOUT,168) (ECUT(1),PCUT(1),MED(1),
592 RHOR(1),ESTEPR(1),SMAXIR(1),ESAVE(1));
593 WRITE(IOUT,202) AINFLU;
594 ISEED=IXX;"save original IXX value"
595 WRITE(IOUT,205) INSEED;
596 OUTPUT INSEED,ISEED;
597 ('0INPUT R.N. SEED, INITIAL SEED',2I12/);
598 IF(RUSROU)[WRITE(IOUT,305) RRR,RRCUT;]
599 IF(CEXPTR ~= 0.0)[WRITE(IOUT,315) CEXPTR,1./(1.-CEXPTR);]
600
601 "Fibonacci RNG may be initialised here - D.C.M."
602 $TPUTER-RNG-INIT;
603
604 $PRESTA-INPUT-SUMMARY;
605
606 IF(IWATCH ~= 0)[
607 "Set up to call AUSGAB for all interactions" CALL WATCH(-99,IWATCH);]
608
609 "                           *****************                      "
610 "                           *               *                      "
611 "                           * Do simulation *                      "
612 "                           *               *                      "
613 "                           *****************                      "
614 CALL CPUTIME(CPUT3);TZERO=SECNDS(0.0);
615
616 "Begin batch"
617 DO IS=1,$STAT[
618 TIMEB=SECNDS(TZERO);CALL CPUTIME(CPUT4);TIMCPU=(CPUT4-CPUT3)*0.01+.0001;
619 OUTPUT IS,TIMEB,TIMCPU,TIMEB/TIMCPU,IXX;
620 (' START BATCH',I3,' TIMES-ELAPSED,CPU,RATIO,RNG',2F8.1,F8.1,I12);
621
622 IF(IS ~= 1)["check there is time left"
623 TIMJOB=CPUT4*0.01;"time since job started"
624 BATCHT=TIMCPU/FLOAT(IS-1);"time per batch so far"
625 IF(TIMJOB + 1.1*BATCHT  >  TIMMAX*3600.)[ "not enough time"
626 ISTAT=IS-1;WRITE(6,220) TIMMAX,ISTAT;
627 WRITE(IOUT,220) TIMMAX,ISTAT;
628 IABORT=1;GO TO :END-SIM:]]
629
630 "Begin each history"
631 DO ICASE=1,JCASE[
632 IRIN=1;  "particle region (allow for backscatter)"
633 XIN=0.0;YIN=0.0;ZIN=0.0;UIN=0.0;VIN=0.0;WIN=1.0;
634 LATCHI=0;  "initialize LATCH to zero - actually redundant"
635 IF(EIN<=0.0)[ "select energy from distribution"
636   CALL ESPEC(EI);EINN=EI; "returns kinetic energy - need total"
637 ]
638 ELSE [EINN = EIN;]
639
640 "Weight photon according to mean free path, and add weighted energy to "
641 "kernel total - DCM"
642 IF (EIN<=0)[
643   MEDIUMOLD=MEDIUM;
644   MEDIUM=MED(1);
645   IF(EINN<PCUT(1))[GAMMALOG=ALOG(PCUT(1));]
```

```
   ELSE [GAMMALOG=ALOG(EINN);]
   $SET INTERVAL GAMMALOG,GE;
   $EVALUATE GAMMAMFP USING GMFP(GAMMALOG);
   WTIN=1.0/GAMMAMFP;
   TOTALMEV=TOTALMEV+EINN*WTIN;
   "Debugging lines - DCM"
   "OUTPUT EINN,MEDIUM,GAMMALOG,LGAMMALOG,GAMMAMFP,WTIN;"
   "(' EINN,MED,GLOG,LGLOG,GMFP,WTIN= ',F10.3,I3,F10.3,I6,2F10.3);"
   MEDIUM=MEDIUMOLD;
 ]
ELSE["only 1 spectral component"
   WTIN=1.0;
   TOTALMEV=TOTALMEV+EINN;
]

IF(IWATCH > 0) [OUTPUT 1,EINN,IQIN,IRIN,XIN,YIN,ZIN,UIN,VIN,WIN,LATCHI,WTIN;
(' INITIAL SHOWER VALUES',T36,':',I2,F9.3,I5,I6,3F8.3,3F7.3,I10,1PE10.3);]

"Swap calling condition between start and end of step to result"
"in average dose position in middle of step - DCM"
IF (IAUSFL(1)=1) [IAUSFL(1)=0;IAUSFL(6)=1;]
ELSE [IAUSFL(1)=1;IAUSFL(6)=0;]

CALL SHOWER(IQIN,EI,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN);

IF(IWATCH > 0) CALL WATCH(-1,IWATCH);"to signal end of history"
]"end of ICASE loop"
]"end of $STAT loop"
ISTAT=$STAT;"number of batches completed - full set here"
:END-SIM:
"                       **********                                  "
"                       *        *                                  "
"                       * End of simulation *                       "
"                       *        *                                  "
"                       **********                                  "
TIMEB=SECNDS(TZERO);CALL CPUTIME(CPUT4);TIMCPU=(CPUT4-CPUT3)*0.01;
OUTPUT TIMEB,TIMCPU,TIMEB/TIMCPU;
(//'OFINISHED SIMULATIONS- TIMES ELAPSED,CPU,RATIO=',2F8.1,F8.2);
WRITE(IOUT,295) TIMEB,TIMCPU,TIMEB/TIMCPU,MXNP;
WRITE(IOUT,310) ISTEPA,NOSCAT;
LASTIX=IXX; $RANDOMSET XSI;
WRITE(IOUT,280) LASTIX,XSI;
"                                                                   "
"                         Analyse results                           "
"                         ===============                           "
STAT=FLOAT(ISTAT);
IF(ISTAT ^= 1)[SDENOM=STAT*(STAT-1.);]ELSE[SDENOM=0.00001;]
AINFLU=AINFLU*STAT/FLOAT($STAT);"correct fluence in case of aborted run"
NCASE = ISTAT*JCASE;"correct total number of histories if aborted run"

"Get total energy for total and primary dose"
DO IT=1,2[
  DO ITHETA=1,NTHETAS[
    DO IRADIAL=1,NRADIALS[
      DOSE(IRADIAL,ITHETA,IT)=0.0;
      DO IS=1,ISTAT[
        DOSE(IRADIAL,ITHETA,IT)=DOSE(IRADIAL,ITHETA,IT)+
                                DOSEIS(IRADIAL,ITHETA,IT,IS);
      ]
      DOSE(IRADIAL,ITHETA,IT)=DOSE(IRADIAL,ITHETA,IT)/STAT;
      "DOSE is the average energy deposited in the region per batch"
]]]
"Get total energy for scattered dose"
DO ITHETA=1,NTHSCAT[
  DO IRADIAL=1,NRADSCAT[
    DOSE(IRADIAL,ITHETA,3)=0.0;
    DO IS=1,ISTAT[
      DOSE(IRADIAL,ITHETA,3)=DOSE(IRADIAL,ITHETA,3)+
                             DOSEIS(IRADIAL,ITHETA,3,IS);
    ]
    DOSE(IRADIAL,ITHETA,3)=DOSE(IRADIAL,ITHETA,3)/STAT;
    "DOSE is the average energy deposited in the region per batch"
]]

"Calculate uncertainties on total and primary dose"
DO IT=1,2[
  DO ITHETA=1,NTHETAS[
    DO IRADIAL=1,NRADIALS[
      UNCER(IRADIAL,ITHETA,IT)=0.0;
      DO IS=1,ISTAT[
        UNCER(IRADIAL,ITHETA,IT)=UNCER(IRADIAL,ITHETA,IT)+
        (DOSEIS(IRADIAL,ITHETA,IT,IS)-DOSE(IRADIAL,ITHETA,IT))**2;
      ]
      IF(DOSE(IRADIAL,ITHETA,IT) ^= 0.0)[
        UNCER(IRADIAL,ITHETA,IT)=
        SQRT(UNCER(IRADIAL,ITHETA,IT)/SDENOM)/DOSE(IRADIAL,ITHETA,IT);
      ]
]]]
"Calculate uncertainties on scattered dose"
DO ITHETA=1,NTHSCAT[
  DO IRADIAL=1,NRADSCAT[
    UNCER(IRADIAL,ITHETA,3)=0.0;
    DO IS=1,ISTAT[
      UNCER(IRADIAL,ITHETA,3)=UNCER(IRADIAL,ITHETA,3)+
      (DOSEIS(IRADIAL,ITHETA,3,IS)-DOSE(IRADIAL,ITHETA,3))**2;
```

```
741     ]
742     IF(DOSE(IRADIAL,ITHETA,3) ^= 0.0)[
743       UNCER(IRADIAL,ITHETA,3)=
744       SQRT(UNCER(IRADIAL,ITHETA,3)/SDENOM)/DOSE(IRADIAL,ITHETA,3);
745     ]
746 ]]
747
748 "Divide by total incident MeV - total and primary energy"
749 DO IT=1,2[
750   DO ITHETA=1,NTHETAS[
751     DO IRADIAL=1,NRADIALS[
752       DOSE(IRADIAL,ITHETA,IT)=DOSE(IRADIAL,ITHETA,IT)*STAT/TOTALMEV;
753 ]]]
754 "Divide by total incident MeV - scattered energy"
755 DO ITHETA=1,NTHSCAT[
756   DO IRADIAL=1,NRADSCAT[
757     DOSE(IRADIAL,ITHETA,3)=DOSE(IRADIAL,ITHETA,3)*STAT/TOTALMEV;
758 ]]
759
760 "        Print results   "
761 "        ==============  "
762
763 "Write total and primary energy"
764 DO IT=1,2[
765   IF((IT=1.AND.ITOT^=0).OR.(IT=2.AND.IPRI^=0))[
766     WRITE(IOUT,250) TITLE,DATEN,TIMEN,NCASE,IQIN,EIN,
767     ECUTIN,PCUTIN,SMAX,CEXPTR,TIMCPU;
768     IF(IT=1)[WRITE(IOUT,251);]
769     ELSEIF(IT=2)[WRITE(IOUT,252);]
770     "Load two dimensional array for GRIDLP"
771     DO IRADIAL=1,NRADIALS[
772       DO ITHETA=1,NTHETAS[
773         DOSEIT(IRADIAL,ITHETA)=DOSE(IRADIAL,ITHETA,IT);
774         UNCERT(IRADIAL,ITHETA)=UNCER(IRADIAL,ITHETA,IT);
775     ]]
776     DO I=1,$MAXR+1[
777       RBOUND(I)=(I-1)*DELTAR;
778     ]
779     DO I=1,$MAXTHETA+1[
780       THETABOUND(I)=(I-1)*180.0/NTHETAS;
781     ]
782     CALL GRIDLP(DOSEIT,UNCERT,NRADIALS,7,$MAXR,RBOUND,THETABOUND,IOUT);
783
784     "Write to raw data file"
785     IF(IT=1)[WRITE(IRAWFILE,302);]
786     ELSEIF(IT=2)[WRITE(IRAWFILE,303);]
787     "Write RHOR,DELTAR,DELTATHETA,NRADIALS to raw data file"
788     DELTATHETA=180.0/NTHETAS;
789     WRITE(IRAWFILE,314);
790     WRITE(IRAWFILE,316) RHOR,DELTAR,DELTATHETA,NRADIALS,NTHETAS;
791     DO IRADIAL=1,NRADIALS[
792       DO ITHETA=1,NTHETAS[
793         WRITE(IRAWFILE,300) DOSEIT(IRADIAL,ITHETA);
794       ]
795       WRITE(IRAWFILE,301);
796 ]]]
797
798 "Write scattered energy"
799 IF(ISCAT^=0)[
800   WRITE(IOUT,250) TITLE,DATEN,TIMEN,NCASE,IQIN,EIN,
801   ECUTIN,PCUTIN,SMAX,CEXPTR,TIMCPU;
802   WRITE(IOUT,253);
803   "Load two dimensional array for GRIDLP"
804   DO IRADIAL=1,NRADSCAT[
805     DO ITHETA=1,NTHSCAT[
806       DOSEIT(IRADIAL,ITHETA)=DOSE(IRADIAL,ITHETA,3);
807       UNCERT(IRADIAL,ITHETA)=UNCER(IRADIAL,ITHETA,3);
808   ]]
809   DO I=1,$MAXR+1[
810     RBOUND(I)=(I-1)*DELRSCAT;
811   ]
812   DO I=1,$MAXTHETA+1[
813     THETABOUND(I)=(I-1)*180.0/NTHSCAT;
814   ]
815   CALL GRIDLP(DOSEIT,UNCERT,NRADSCAT,7,$MAXR,RBOUND,THETABOUND,IOUT);
816
817   "Write to raw data file"
818   WRITE(IRAWFILE,304);
819   "Write RHOR,DELRSCAT,DELTATHETA,NRADSCAT to raw data file"
820   DELTATHETA=180.0/NTHSCAT;
821   WRITE(IRAWFILE,314);
822   WRITE(IRAWFILE,316) RHOR,DELRSCAT,DELTATHETA,NRADSCAT,NTHSCAT;
823   DO IRADIAL=1,NRADSCAT[
824     DO ITHETA=1,NTHSCAT[
825       WRITE(IRAWFILE,300) DOSEIT(IRADIAL,ITHETA);
826     ]
827     WRITE(IRAWFILE,301);
828 ]]
829
830 "Write out total energy to grid file and standard output"
831 WRITE(IOUT,313) TOTALMEV/FLOAT(NCASE);
832 OUTPUT TOTALMEV/FLOAT(NCASE);
833 (' Mean incident weighted particle energy = ',F10.5);
834
835 :END:
```

```
836 CALL DATE(DATE$);CALL TIME(TIME$);
837 CALL CPUTIME(CPUT4);TIMCPU=(CPUT4-CPUT0)*0.01;
838 OUTPUT DATE$,TIME$;('0END OF FILE READ',10X,9A1,1X,8A1//);
839 WRITE(IOUT,210) DATE$,TIME$,TIMCPU,TIMCPU/3600.;
840 STOP;
841                                                                          "
842 "                          Formats                                       "
843 "                          =======
844 %I   need as switch to FORTRAN with %F
845 %F
846 100   FORMAT(80A1)
847 105   FORMAT(I6,F12.6,I6,I6,F12.6,I6)
848 110   FORMAT(7F12.0)
849 115   FORMAT(4I3)
850 120   FORMAT(24A1)
851 130   FORMAT(2I10,F10.0,I10,I14)
852 140   FORMAT(' ',80A1/'0EDK CALCULATION USING RTPEDK',
853      1' 1991', 10X,9A1,1X,8A1)
854 150   FORMAT(//' Incident energy=',F10.3,' MEV Charge=',I3,
855      1'   #Histories=',I8/21X,10('='),8X,3('='),15X,8('=')//)
856 151   FORMAT(/T20,' BIN  TOP E   PROB     CUMULATIVE PROB.'/
857      1 (T20,I3,0PF9.3,2(1PE12.3)))
858 152   FORMAT('0Rayleigh (coherent) scattering included'/)
859 153   FORMAT('0Rayleigh (coherent) scattering NOT included')
860 155   FORMAT(T10,'Global user cutoffs ECUT,PCUT=',2F12.3,' MeV',
861      1 //' Material',T39,'Density',T57,'AE',T73,'AP',
862      2 T84,'200*TEFF0')
863 160   FORMAT(I3,2X,24A1,T37,F10.3,T50,F12.3,T65,F12.3,T80,F12.4)
864 165   FORMAT('0Electron step size has been modified to',
865      1 ' reduce energy by',F8.3,' in each step'/1X,81('*'))
866 168   FORMAT('0ECUT=',F6.3,' PCUT=',F6.3,' MED=',I2,' RHOR=',0PF8.4,
867      1' ESTEPR=',F8.2,' SMAXIR=',1PE12.2,' ESAVE=',0PF10.3)
868 202   FORMAT('0Incident flux=',F12.3,' photons')
869 205   FORMAT('0Initial random no. seed=',I12)
870 210   FORMAT('0End of file read',10X,9A1,1X,8A1,5X,
871      1'Total cputime=',F10.1,' sec=',F8.3,' HOURS')
872 220   FORMAT('0*****NOT ENOUGH TIME TO FINISH WITHIN LIMIT OF',
873      1 F6.2,' HOURS',I10,' BATCHES USED********'/////)
874 250   FORMAT(1X,80A1,T79,9A1,1X,8A1//I8,' Particles of charge'
875      1,I2,' Energy=',F10.3,' MeV,'/T3,'RTPEDK',T15,
876      2'[',F7.3,',',F6.3,'] SMAX=',1PE10.2,'  CEXPTR=',0PF8.3,
877      3'   CPU time=',0PF8.1,' sec'/)
878 251   FORMAT(/T15,' EDK - TOTAL NORMALISED ENERGY (POLAR COORDS)'/)
879 252   FORMAT(/T15,' EDK - PRIMARY NORMALISED ENERGY (POLAR COORDS)'/)
880 253   FORMAT(/T15,' EDK - SCATTERED NORMALISED ENERGY (POLAR COORDS)'/)
881 280   FORMAT('0Final random no. seed=',I12,'  Final random no.=',
882      1  0PF15.8)
883 295   FORMAT('0Simulation times - ELAPSED,CPU,RATIO=',2F8.1,F7.2/
884      1 '0Maximum stack=',I5,5X)
885 305   FORMAT('0Russian roulette played for photons crossing R=',
886      1 F10.3,' cm with survival probability=',F7.4)
887 310   FORMAT('0Total number of charged particle steps taken=',I10
888      1'  NOSCAT=',I10)
889 315   FORMAT('0Pathlength exponential transformation variable=',F10.3,
890      1 '  for forward going photons only'/' shortens initial ',
891      2 'pathlength by:',F10.3)
892 300   FORMAT(1X,E10.4)
893 301   FORMAT(1X)
894 302   FORMAT(' TOTAL')
895 303   FORMAT(' PRIMARY')
896 304   FORMAT(' SCATTERED')
897 306   FORMAT(' Total primary energy=',F10.3,' MEV')
898 313   FORMAT(' Mean incident weighted particle energy = ',F10.5,' MEV')
899 314   FORMAT(' RHOR,DELTAR,DELTATHETA,NRADIALS,NTHETAS=')
900 316   FORMAT(1X,3F12.6,2I5)
901 %M
902 %I4
903 END;" end of RTPEDK main routine"
904 "*****************************************************************************"
905 REAL FUNCTION TAN(X);
906 "                                                                            "
907 "                                                                            "
908 "        Given x in radians this calculates the tangent of x                 "
909 "                                                                            "
910 "*****************************************************************************"
911 A=COS(X);
912 IF(ABS(A) > 1.E-10) [TAN = SIN(X)/A;] ELSE [TAN = 1.E10;]
913 RETURN;END;"end of TAN"
914 %E     "RTPEDK.MOR"
915 "*****************************************************************************"
916 "                                                                            "
917 SUBROUTINE ESPEC(EI);
918 "                                                                            "
919 "   Note that this routine is renamed from ESPECT to avoid transputer        "
920 "   compiler error - D.C.M.                                                  "
921 "   This routine returns an initial kinetic energy given the                 "
922 "   cumulative probability distribution for the source spectrum              "
923 "   stored in ECPD, the energy bin tops in EBIN and the minimum              "
924 "   energy in EIMIN  - all in common ESPECT                                  "
925 "                                                                            "
926 "   Initially written in Tampere Finland, D.R. Aug 1985                      "
927 "                                                                            "
928 "*****************************************************************************"
```

```
929
930 ;COMIN/ESPECT,RANDOM/;
931 $RANDOMSET RNNO40; $RANDOMSET RNNO41;  "pick two random numbers"
932
933 "Use the first to select which energy bin we are in"
934 I=0; LOOP [I=I+1;] UNTIL ECPD(I)>RNNO40;
935 "ENERGY IS BETWEEN EBIN(I-1) AND EBIN(I)"
936 IF(I.NE.1)[ELOW=EBIN(I-1);] ELSE [ELOW = EIMIN;]
937
938 EI = ELOW + RNNO41* (EBIN(I)-ELOW);  "select randomly in this bin"
939 RETURN; END;
940 %E     "RTPEDK.MOR"
941
942 "*************************************************************************"
943 "                                                                         "
944 SUBROUTINE AUSGAB(IARG);
945 "                                                                         "
946 "        An ausgab routine to be used with RTPEDK.MOR                     "
947 "                                                                         "
948 "        The routine scores various dose components:                      "
949 "        IT=1 Total energy                                                "
950 "        IT=2 Primary energy                                              "
951 "        IT=3 Scattered energy                                            "
952 "        The logic is:                                                    "
953 "        Score IT=2 dose whenever FLAG1 is 1 - DCM                        "
954 "        Score IT=3 dose whenever FLAG1 is >1 - DCM                       "
955 "                                                                         "
956 "*************************************************************************"
957 ;COMIN/ELECIN,EPCONT,MISC,RUSROU,RANDOM,SCORE,STACK,GEOM/;
958
959 "        Define some macros for setting and picking up several flags"
960 "        based on the variable LATCH"
961
962 "  FLAG1  is the units digit of LATCH             "
963 REPLACE {$SET-FLAG1(#)} WITH
964 "        ============"
965 {;IF(LATCH({P1}) >= 0)[LATCH({P1})=LATCH({P1})+1;]
966 ELSE [LATCH({P1})=LATCH({P1})-1;];}
967
968 REPLACE {$FLAG1} WITH {LATCH(NP)}
969
970 ;MXNP=MAX(MXNP,NP);"keep track of how deep stack is"
971 IF(NP >= $MXSTACK)["stack as deep as allowed"
972 OUTPUT NP,$MXSTACK;(' IN AUSGAB, NP=',I3,' >= MAXIMUM STACK ALLOWED=',
973 I3,/1X,80('*')/);IF(MXNP > $MXSTACK)[STOP;] ]
974
975 "Calculate particle R,TH"
976 R=SQRT(X(NP)**2+Y(NP)**2+Z(NP)**2);     "R in cm"
977 IF(R<0.000001)[R=0.000001;]"avoid division by zero in next line"
978 TH=ASIN(SQRT(X(NP)**2+Y(NP)**2)/R)*180/3.1415926; "Theta in degrees"
979 IF(Z(NP)<0.0)[TH=180-TH;]
980
981 IF(IDOTPLOT=1)["If DOTPLOT=1, write particle IQ,X,Z to unit 13"
982   IF(USTEP<100.0)[
983     WRITE(IDOTFILE,7) IQ(NP),X(NP),Z(NP);
984   ]
985 ]
986 ELSEIF (IDOTPLOT=2)["If DOTPLOT=2, write particle IQ,X,Z,U,W,USTEP to unit 13"
987   IF(USTEP<100.0)[
988     WRITE(IDOTFILE,9) IQ(NP),X(NP),Z(NP),U(NP),W(NP),USTEP;
989   ]
990 ]
991 ELSEIF (IDOTPLOT=3)["If DOTPLOT=3 and primary, write IQ,X,Z to unit 13"
992   IF(USTEP<100.0 & $FLAG1<2)[
993     WRITE(IDOTFILE,7) IQ(NP),X(NP),Z(NP);
994   ]
995 ]
996 ELSEIF (IDOTPLOT=4)["If DOTPLOT=4 and primary, write IQ,R,Z to unit 13"
997   IF(USTEP<100.0 & $FLAG1<2)[
998     WRITE(IDOTFILE,7) IQ(NP),SQRT(X(NP)*X(NP)+Y(NP)*Y(NP)),Z(NP);
999   ]
1000 ]
1001
1002 IF(IARG = 0 | IARG = 5)["about to transport a particle"
1003   IF(IQ(NP) ^= 0)[ISTEPA=ISTEPA+1;"count charged particle steps taken"]
1004   ELSE ["photon step - play Russian roulette?"
1005     IF(RUSROU & R>(RRR-USTEP) & R<=RRR)["possible Russian Roulette"
1006       NEWR=SQRT((X(NP)+USTEP*U(NP))**2 +
1007                 (Y(NP)+USTEP*V(NP))**2 +
1008                 (Z(NP)+USTEP*W(NP))**2);    "New R in cm"
1009       IF(NEWR>=RRR)["play Russian Roulette"
1010         $RANDOMSET XSI;
1011         IF(XSI < RRCUT)["particle survives"
1012           WT(NP)=WT(NP)/RRCUT; "increase particle weight"]
1013         ELSE["discard particle"
1014           WT(NP)=0.0; "particle will be discarded in HOWFAR"
1015 ] ] ] ] ]
1016                                                                         "
1017 "        Set flags for photon interactions
1018 IF($FLAG1=0 & NP>1)[
1019   DO I=1,NP[
1020     IF(IQ(I)=0)[$SET-FLAG1(I);$SET-FLAG1(I);] "photon is secondary"
1021     ELSE[$SET-FLAG1(I);] "electron is primary"
```

```
  ]
]
"If photon and $FLAG1=1 then brem, so assign to secondary dose"
IF($FLAG1=1 & IQ(NP)=0) [$SET-FLAG1(NP);]
"Ensure photoelectrons are scored as primary dose - corrected 13/11/90 - DCM"
IF($FLAG1=0 & NP=1 & IQ(1)~=0)[$SET-FLAG1(1);]

IF(IWATCH > 0) CALL WATCH(IARG,IWATCH);
IF(IARG > 5) [RETURN;]

"Score energy deposited for 0<=IARG<=5"
IF(EDEP = 0.0)[RETURN;"photons don't usually deposit energy"]

IRADIAL=INT(R/DELTAR)+1;
ITHETA=INT(TH*NTHETAS/180.0)+1;
IF(IRADIAL<=NRADIALS & ITHETA<=NTHETAS)[
  "Debugging line"
  "OUTPUT R,TH,IRADIAL,ITHETA;"
  "(' R=',F8.4,' TH=',F8.4,' IRADIAL=',I3,'ITHETA=',I3);"
  "Score total energy deposited"
  DOSEIS(IRADIAL,ITHETA,1,IS)=DOSEIS(IRADIAL,ITHETA,1,IS) + WT(NP)*EDEP;
  IF($FLAG1 = 1)["primary energy - deposit in IT=2 - DCM"
    DOSEIS(IRADIAL,ITHETA,2,IS)=DOSEIS(IRADIAL,ITHETA,2,IS)+WT(NP)*EDEP;
  ]
]
IF($FLAG1 > 1)["scattered energy - deposit in IT=3 - DCM"
  IRADSCAT=INT(R/DELRSCAT)+1;
  ITHSCAT=INT(TH*NTHSCAT/180.0)+1;
  "Debugging line"
  "OUTPUT R,TH,IRADSCAT,ITHSCAT;"
  "(' R=',F8.4,' TH=',F8.4,' IRADSCAT=',I3,'ITHSCAT=',I3);"
  IF(IRADSCAT<=NRADSCAT & ITHSCAT<=NTHSCAT)[
    DOSEIS(IRADSCAT,ITHSCAT,3,IS)=DOSEIS(IRADSCAT,ITHSCAT,3,IS)+WT(NP)*EDEP;
  ]
]
RETURN;
%I
%F
7     FORMAT(I3,2F10.4)
9     FORMAT(I3,5F10.4)
%M
%I4
END;"end of AUSGAB"
%E     "RTPEDK.MOR"
"*************************************************************************"
"                                                                         "
SUBROUTINE HOWFAR;
"                                                                         "
"        A routine to use with RTPEDK and the EGS system                  "
"                                                                         "
"*************************************************************************"
;COMIN/EPCONT,STACK,GEOM/;

"Since there is only one medium here, HOWFAR simply returns. Note that if not  "
"using PRESTA, ESTEPE will have to be set small enough to reduce errors in     "
"scoring due to crossing of scoring boundaries (this is also helped by swapping"
"energy deposition from start to end alternately - see immediately before      "
"SHOWER call.                                                                  "
"If using PRESTA, CALL-HOWNEAR carefully checks for boundaries of scoring      "
"regions --- so PRESTA should be accurate.                                     "

"Calculate particle R,TH in case this has not been done in AUSGAB"
R=SQRT(X(NP)**2+Y(NP)**2+Z(NP)**2);     "R in cm"
IF(R<0.000001)[R=0.000001;]"avoid division by zero in next line"
TH=ASIN(SQRT(X(NP)**2+Y(NP)**2)/R)*180/3.1415926; "Theta in degrees"
IF(Z(NP)<0.0)[TH=180.0-TH;] "ensure 0<=TH<180"

"Check if particle should be discarded"
IF (WT(NP)=0.0)[IDISC=1;RETURN;]
IF (IQ(NP)=0 & R>(RMAXNEEDED+30.0))
  [IDISC=1;RETURN;]
IF (IQ(NP)~=0 & R>(RMAXNEEDED+5.0))
  [IDISC=1;RETURN;]
RETURN;

END; "END OF HOWFAR"
%E     "RTPEDK.MOR"
"*************************************************************************"
"                                                                         "
SUBROUTINE GRIDLP(DATA,UNCERT,NROW,NCOL,MAXROW,ROW,COL,IOUT);
"                                                                         "
"                                                                         "
"        Routine to print an array of values and their uncertainties      "
"          in a labeled grid                                              "
"                                                                         "
"        DATA(IROW,ICOL)                                                  "
"        UNCERT(IROW,ICOL)        if same as data, then not printed       "
"                                 the fractional uncertainty in data - it is "
"                                 converted to integer percentage!!!      "
"        NROW                     # rows in grid  down page essentially    "
"                                 without limit - about 18 fit on a page  "
"        NCOL                     # columns<=5 on 8.5 in page 13/char per inch "
"                                 uses 15 characters per column + 20 initial "
"                                 no software limit on value              "
```

```
1116 "                                  but NCOL<=7 for standard 133 col fortrn write "
1117 "        MAXROW                    first dimension of calling array in MAIN      "
1118 "        ROW(NROW+1)               an array with bounds for rows in grid         "
1119 "                                  only used for labels                          "
1120 "        COL(NCOL+1)                  ditto                                      "
1121 "                                                                                 "
1122 "        This uses the variable format feature of FORTRAN 77                      "
1123 "        ================================================                         "
1124 "                                                                                 "
1125 "*********************************************************************************"
1126 ;COMIN/SCORE/;
1127
1128 LOGICAL SAME;
1129 REAL DATA(MAXROW,NCOL),UNCERT(MAXROW,NCOL),ROW(1),COL(1);
1130 INTEGER*4 COLBASE,COLTOP; "base and top vars for multi-grid printout -D.C.M."
1131
1132 "Check if DATA and UNCERT are the same and if they are, don't print uncert"
1133 SAME=.FALSE.;
1134 DO ICOL=1,NCOL[
1135 DO IROW=1,NROW[
1136 IF(DATA(IROW,ICOL) ^= UNCERT(IROW,ICOL))[
1137 GO TO :EXITLOOPS:;"leave SAME=FALSE"]]]
1138 "If we fall through, arrays are the same" SAME=.TRUE.;
1139 :EXITLOOPS:
1140
1141 ."Split columns into groups of $COLNO - D.C.M."
1142 DO COLBASE=1,NCOL,$COLNO[
1143 COLTOP = COLBASE+$COLNO-1;
1144 IF (NCOL<COLTOP)[COLTOP=NCOL;]
1145 WRITE(IOUT,50);
1146
1147 WRITE(IOUT,100) (COL(ICOL),ICOL=COLBASE,COLTOP+1);
1148
1149 DO IROW=1,NROW[
1150 WRITE(IOUT,200) ROW(IROW);
1151 WRITE(IOUT,210) (DATA(IROW,ICOL),ICOL=COLBASE,COLTOP);
1152 IF(.NOT.SAME)[
1153 WRITE(IOUT,220) (AMIN1(UNCERT(IROW,ICOL)*100.,99.),
1154 ICOL=COLBASE,COLTOP);]
1155 ]"END IROW LOOP"
1156 WRITE(IOUT,200) ROW(NROW+1);
1157 ]
1158 %F
1159 50    FORMAT(5X)
1160 100   FORMAT(T4,8F15.3)
1161 200   FORMAT(1X,F12.3,2X,7('|',14('-')),'|')
1162 210   FORMAT(15X,'|',7(1PE12.3,2X,'|'))
1163 220   FORMAT(15X,'|',7(3X,'+/-',F4.1,'%',3X,'|'))
1164 %M
1165 RETURN;END;"end of GRIDLP"
1166
1167 "       last statement in RTPEDK.MOR"
```

Appendix E

Implementation of Siddon's Ray Tracing Algorithm

```
/* S_RAYTRACE.C ***************************************************************/
#include <stdio.h>
#include <math.h>
#include "gen.h"
#include "s_superposition.h"

void s_raytrace(electron_density_grid, radiological_depth_grid, point1, point2)
GRID *electron_density_grid,
     *radiological_depth_grid;
POINT point1, point2;
/*
-------------------------------------------------------------------------------
   NAME
        s_raytrace

   SYNOPSIS
        point1 and point2 are the end points of the ray.

   DESCRIPTION
        This function traces the ray from point x to point y (in real
        coords), assigning depth to any voxels which that ray crosses. The
        technique used is that described by Siddon R.L. (Med Phys 12 (2),
        1985). This routine will not be understood without thorough reading
        of that paper! Point1 and point2 are the start and end points of the
        ray, respectively. External structures of type GRID are assumed to
        exist, where electron_density_grid are the electron densities, and
        radiological_depth_grid is the output grid for these calculations.
        Voxels in radiological_depth_grid are initially set -ve prior to
        calling this function.

   AUTHOR
        Written by David C. Murray
                   University of Waikato
                   Private Bag 3105
                   Hamilton
                   New Zealand
        and Copyright (1991) to
                   David C. Murray and Peter W. Hoban,
                   Cancer Society of New Zealand Inc., and
                   University of Waikato.
-------------------------------------------------------------------------------
*/

{
/* Variable descriptions:
   x1,x2,y1,y2,z1,z2 are the coordinates of the ray end points
     (i.e. (x1,y1,z1)=source, (x2,y2,z2)=point beyond phantom)
   xp1,yp1,zp1 are the real coords of voxel region origin (in cm)
   Nx,Ny,Nz are (no. of voxels + 1) in each direction
   dx,dy,dz are the widths in cm of the voxels
*/
float         x1,y1,z1,
              x2,y2,z2,
              xp1,yp1,zp1,
              dx,dy,dz;
int           Nx,Ny,Nz;

/*General ray-trace algorithm variables*/
float xpN, ypN, zpN;                          /*real coords in cm of region limits*/
float alpha_x_min,alpha_y_min,alpha_z_min,alpha_x_max,alpha_y_max,alpha_z_max;
                                              /*limits of alpha x,y,z parameters*/
float alpha_min, alpha_max;                   /*limits of alpha parameter*/
int i_min,i_max,j_min,j_max,k_min,k_max;/*limits of indices for x,y,z dirns*/
float *alpha_x,*alpha_y,*alpha_z;             /*hold sets of x,y,z alpha values*/
float *alpha;                                 /*holds merged set of alpha values*/
int i_index,j_index,k_index;                  /*loop indices for merging alphas*/
int a;                                        /*loop counter*/
int max_index;                                /*max index of merged alpha array*/
float d12;                                    /*distance between ray end points*/
float alpha_mid;                              /*mid-point of intersection length*/
float length;                                 /*intersection length*/
int i,j,k;                                    /*indices of voxel with int. length*/
float rpl = 0;                                /*radiological path length in cm*/
float voxel_density;                          /*temporary voxel density*/

/* Assign variables */
/******************************************************************************/
```

```
 78 x1 = point1.x;
 79 y1 = point1.y;
 80 z1 = point1.z;
 81 x2 = point2.x;
 82 y2 = point2.y;
 83 z2 = point2.z;
 84 xp1 = electron_density_grid->start.x;
 85 yp1 = electron_density_grid->start.y;
 86 zp1 = electron_density_grid->start.z;
 87 Nx = electron_density_grid->x_count + 1;
 88 Ny = electron_density_grid->y_count + 1;
 89 Nz = electron_density_grid->z_count + 1;
 90 dx = electron_density_grid->inc.x;
 91 dy = electron_density_grid->inc.y;
 92 dz = electron_density_grid->inc.z;
 93
 94 /* Calculate xpN,ypN,zpN */
 95 /*************************************************************************/
 96 xpN = xp1 + (Nx-1)*dx;
 97 ypN = yp1 + (Ny-1)*dy;
 98 zpN = zp1 + (Nz-1)*dz;
 99
100 /*Calculate alpha_min and alpha_max*/
101 /*************************************************************************/
102 /*Avoid division by zero*/
103 if (x1==x2)
104   x2 += 0.00001;
105 if (y1==y2)
106   y2 += 0.00001;
107 if (z1==z2)
108   z2 += 0.00001;
109 if ((fabs(x1-x2)<dx) && (fabs(y1-y2)<dy) && (fabs(z1-z2)<dz))
110   {
111   fprintf(stderr,"Error - ray trace region too small\n");
112   exit (1);
113   }
114 alpha_x_min = (((xp1-x1)/(x2-x1))<((xpN-x1)/(x2-x1))) ? ((xp1-x1)/(x2-x1))
115                                                       : ((xpN-x1)/(x2-x1));
116 alpha_y_min = (((yp1-y1)/(y2-y1))<((ypN-y1)/(y2-y1))) ? ((yp1-y1)/(y2-y1))
117                                                       : ((ypN-y1)/(y2-y1));
118 alpha_z_min = (((zp1-z1)/(z2-z1))<((zpN-z1)/(z2-z1))) ? ((zp1-z1)/(z2-z1))
119                                                       : ((zpN-z1)/(z2-z1));
120 alpha_x_max = (((xp1-x1)/(x2-x1))>((xpN-x1)/(x2-x1))) ? ((xp1-x1)/(x2-x1))
121                                                       : ((xpN-x1)/(x2-x1));
122 alpha_y_max = (((yp1-y1)/(y2-y1))>((ypN-y1)/(y2-y1))) ? ((yp1-y1)/(y2-y1))
123                                                       : ((ypN-y1)/(y2-y1));
124 alpha_z_max = (((zp1-z1)/(z2-z1))>((zpN-z1)/(z2-z1))) ? ((zp1-z1)/(z2-z1))
125                                                       : ((zpN-z1)/(z2-z1));
126 alpha_min = (alpha_x_min>alpha_y_min) ? alpha_x_min : alpha_y_min;
127 if (alpha_z_min>alpha_min)
128   alpha_min = alpha_z_min;
129 if (alpha_min<0)
130   alpha_min = 0;
131 alpha_max = (alpha_x_max<alpha_y_max) ? alpha_x_max : alpha_y_max;
132 if (alpha_z_max<alpha_max)
133   alpha_max = alpha_z_max;
134 if (alpha_max>1)
135   alpha_max = 1;
136 /*Monitor lines...
137 fprintf(stdout,"    alpha_x,y,z_min: %7.4f %7.4f %7.4f\n",
138                alpha_x_min,alpha_y_min,alpha_z_min);
139 fprintf(stdout,"    alpha_x,y,z_max: %7.4f %7.4f %7.4f\n",
140                alpha_x_max,alpha_y_max,alpha_z_max);
141 fprintf(stdout,"    alpha_min,alpha_max: %7.4f %7.4f\n",alpha_min,alpha_max);
142 printf("Nx, xpN, x2,x1, dx = %d %5.2f %5.2f %5.2f %5.2f\n",Nx,xpN,x2,x1,dx);
143 */
144 /*Determine the ranges of i,j,k indices*/
145 /*************************************************************************/
146 /*The following assignments require conversion from float to integer types*/
147 /*The value 0.001 is added/subtracted to ensure that the ceiling and floor*/
148 /*functions convert to the correct value. Note that the range of these*/
149 /*variables is from 1 to Nx,Ny,Nz, NOT 0 to Nx-1,Ny-1,Nz-1*/
150 i_min = (x2>x1) ? (int) ceil((float) Nx - (xpN-alpha_min*(x2-x1)-x1)/dx-0.001)
151                 : (int) ceil((float) Nx - (xpN-alpha_max*(x2-x1)-x1)/dx-0.001);
152 i_max = (x2>x1) ? (int) floor(1.0000 + (x1+alpha_max*(x2-x1)-xp1)/dx+0.001)
153                 : (int) floor(1.0000 + (x1+alpha_min*(x2-x1)-xp1)/dx+0.001);
154 j_min = (y2>y1) ? (int) ceil((float) Ny - (ypN-alpha_min*(y2-y1)-y1)/dy-0.001)
155                 : (int) ceil((float) Ny - (ypN-alpha_max*(y2-y1)-y1)/dy-0.001);
156 j_max = (y2>y1) ? (int) floor(1.0000 + (y1+alpha_max*(y2-y1)-yp1)/dy+0.001)
157                 : (int) floor(1.0000 + (y1+alpha_min*(y2-y1)-yp1)/dy+0.001);
158 k_min = (z2>z1) ? (int) ceil((float) Nz - (zpN-alpha_min*(z2-z1)-z1)/dz-0.001)
159                 : (int) ceil((float) Nz - (zpN-alpha_max*(z2-z1)-z1)/dz-0.001);
160 k_max = (z2>z1) ? (int) floor(1.0000 + (z1+alpha_max*(z2-z1)-zp1)/dz+0.001)
161                 : (int) floor(1.0000 + (z1+alpha_min*(z2-z1)-zp1)/dz+0.001);
162 /*Monitor lines...
163 fprintf(stdout,"    i,j,k_min: %3d %3d %3d\n",i_min,j_min,k_min);
164 fprintf(stdout,"    i,j,k_max: %3d %3d %3d\n",i_max,j_max,k_max);
165 */
```

```
/*Generate sets of alpha values,reversing order if necessary*/
/*****************************************************************************/
/*allocate array space on stack*/
if ((alpha_x = (float*) calloc(Nx+1,sizeof(float))) == NULL)
  {
  fprintf(stderr,"Error - insufficient heap for alpha_x allocation\n");
  exit (1);
  }
if ((alpha_y = (float*) calloc(Ny+1,sizeof(float))) == NULL)
  {
  fprintf(stderr,"Error - insufficient heap for alpha_y allocation\n");
  exit (1);
  }
if ((alpha_z = (float*) calloc(Nz+1,sizeof(float))) == NULL)
  {
  fprintf(stderr,"Error - insufficient heap for alpha_z allocation\n");
  exit (1);
  }
if (i_min <= i_max)
  if (x2>x1)
    {
    alpha_x[0] = ((xp1+(i_min-1)*dx)-x1)/(x2-x1);
    for (a=1;a<=i_max-i_min;a++)
      alpha_x[a] = alpha_x[a-1]+dx/(x2-x1);
    }
  else
    {
    alpha_x[i_max-i_min] = ((xp1+(i_min-1)*dx)-x1)/(x2-x1);
    for (a=i_max-i_min-1;a>=0;a--)
      alpha_x[a] = alpha_x[a+1]+(dx/(x2-x1));
    }
alpha_x[i_max-i_min+1] = 10000.0;
if (j_min <= j_max)
  if (y2>y1)
    {
    alpha_y[0] = ((yp1+(j_min-1)*dy)-y1)/(y2-y1);
    for (a=1;a<=j_max-j_min;a++)
      alpha_y[a] = alpha_y[a-1]+dy/(y2-y1);
    }
  else
    {
    alpha_y[j_max-j_min] = ((yp1+(j_min-1)*dy)-y1)/(y2-y1);
    for (a=j_max-j_min-1;a>=0;a--)
      alpha_y[a] = alpha_y[a+1]+(dy/(y2-y1));
    }
alpha_y[j_max-j_min+1] = 10001.0;
if (k_min <= k_max)
  if (z2>z1)
    {
    alpha_z[0] = ((zp1+(k_min-1)*dz)-z1)/(z2-z1);
    for (a=1;a<=k_max-k_min;a++)
      alpha_z[a] = alpha_z[a-1]+(dz/(z2-z1));
    }
  else
    {
    alpha_z[k_max-k_min] = ((zp1+(k_min-1)*dz)-z1)/(z2-z1);
    for (a=k_max-k_min-1;a>=0;a--)
      alpha_z[a] = alpha_z[a+1]+(dz/(z2-z1));
    }
alpha_z[k_max-k_min+1] = 10002.0;

/*Monitor lines...
if (i_max<i_min)
  fprintf(stdout,"    No alpha_x values\n");
else
  fprintf(stdout,"    First & last alpha_x values: %7.4f %7.4f\n",
                 alpha_x[0],alpha_x[i_max-i_min]);
if (j_max<j_min)
  fprintf(stdout,"    No alpha_y values\n");
else
  fprintf(stdout,"    First & last alpha_y values: %7.4f %7.4f\n",
                 alpha_y[0],alpha_y[j_max-j_min]);
if (k_max<k_min)
  fprintf(stdout,"    No alpha_z values\n");
else
  fprintf(stdout,"    First & last alpha_z values: %7.4f %7.4f\n",
                 alpha_z[0],alpha_z[k_max-k_min]);
*/
/*Generate merged set of alpha values*/
/*****************************************************************************/
if ((alpha = (float*) calloc(Nx+Ny+Nz+3,sizeof(float))) == NULL)
  {
  fprintf(stderr,"Error - insufficient heap for alpha allocation\n");
  exit (1);
  }
max_index = (i_max-i_min+1)+(j_max-j_min+1)+(k_max-k_min+1)+1;
alpha[0] = alpha_min;
i_index = 0;
j_index = 0;
k_index = 0;
```

```
256 for (a=1;a<=max_index-1;a++)
257   if (alpha_x[i_index]<alpha_y[j_index])
258     if (alpha_x[i_index]<alpha_z[k_index])
259       {
260       alpha[a] = alpha_x[i_index];
261       i_index += 1;
262       }
263     else
264       {
265       alpha[a] = alpha_z[k_index];
266       k_index += 1;
267       }
268   else
269     if (alpha_y[j_index]<alpha_z[k_index])
270       {
271       alpha[a] = alpha_y[j_index];
272       j_index += 1;
273       }
274     else
275       {
276       alpha[a] = alpha_z[k_index];
277       k_index += 1;
278       }
279 alpha[max_index] = alpha_max;
280 cfree(alpha_x);                          /*deallocate temp array storage*/
281 cfree(alpha_y);
282 cfree(alpha_z);
283 /*Monitor lines...
284 fprintf(stdout,"    Number of elements in merged set = %4d\n",max_index+1);
285 for (a=0;a<=max_index;a++)
286   fprintf(stdout,"      Element %3d = %7.5f\n",a,alpha[a]);
287 */
288 /*Calculate voxel lengths and indices, and assign radiological depth*/
289 /*************************************************************************/
290 d12 = sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)+(z2-z1)*(z2-z1));
291                                          /*d12 is distance between ray end pts*/
292 for (a=1;a<=max_index;a++)
293   {
294   length = d12*(alpha[a]-alpha[a-1]);    /*length is voxel intersection length*/
295   if (fabs(length)>0.01)                 /*do not process unless > 0.01 cm*/
296     {
297     alpha_mid = (alpha[a]+alpha[a-1])/2.0;
298                                          /*alpha_mid is middle of int. length*/
299     i = (int) floor((x1 + alpha_mid*(x2-x1) - xp1)/dx);
300     j = (int) floor((y1 + alpha_mid*(y2-y1) - yp1)/dy);
301     k = (int) floor((z1 + alpha_mid*(z2-z1) - zp1)/dz);
302                                          /*i,j,k are indices of voxel*/
303     /*Monitor line...
304     fprintf(stdout," Voxel indices: %3d %3d %3d\n",i,j,k);
305     */
306     /* Remember that this function traces only a single ray.
307        rpl has been set to zero during initialisation.
308     */
309     voxel_density = GRID_VALUE(electron_density_grid,i,j,k);
310     rpl += length * voxel_density/2.0;  /*add first half of int. length*/
311     /*Store path length in voxel only if top and bottom surfaces are */
312     /*intersected, i.e. voxel path length is greater than voxel z dimension*/
313     if (length>=dz && GRID_VALUE(radiological_depth_grid,i,j,k)<0.0)
314       GRID_VALUE(radiological_depth_grid, i, j, k) = rpl;
315     rpl += length * voxel_density/2.0;  /*add second half of int. length*/
316     }
317   }
318 cfree(alpha); /*deallocate remaining array storage*/
319 }
320 /*End of s_raytrace routine*/
```

Appendix F

Fast Fourier Transform Code

```
1  /*************************************************************************/
2  void fft(xreal,ximag,n,nu)
3  float xreal[],ximag[];
4  int n,nu;
5
6  /* Fast Fourier Transform Routine                                        */
7  /* This function calculates the fast fourier transform of one complex    */
8  /* or two real valued function(s). xreal and ximag are pointers to       */
9  /* the arrays, n is the number of points in the array and nu satisfies   */
10 /* the equation n = 2^nu. If this function is called with a different    */
11 /* value of nu then a bit reversal table is also created.                */
12 {
13 int k,n2,nu1,i,l,kn2,p,arg;
14 float treal,timag,c,s,multiplier;
15 static int nu_flag=0;                  /* stores previous value of nu          */
16 static int bitrev[XYZ_MAX];            /* bit reversal lookup table            */
17 static int power_2[12];                /* powers of 2 lookup table             */
18 static float sine[6300];               /* sine lookup table (radians*1000)     */
19 static float cosine[6300];             /* cosine lookup table (radians*1000)   */
20 float half = 0.5;                      /* defn for rapid expression evaluation */
21
22 /* Create sine,cosine and power function tables if called for the first time  */
23 if (nu_flag==0)
24   {
25   /* Create powers of 2 table */
26   power_2[0] = 1;
27   for (i=1;i<=11;i++)
28     power_2[i] = power_2[i-1]*2;
29   /* Create sine and cosine lookup tables */
30   for (i=0;i<=6299;i++)
31     {
32     sine[i] = sin((float) i /1000.0);
33     cosine[i] = cos((float) i /1000.0);
34     }
35   }
36
37 /* Create bit reversal table if called with n changed */
38 if (nu_flag != nu)
39   {
40   nu_flag = nu;
41   for (i=0;i<n;i++)
42     {
43     /* Use variables nu1,l,kn2,p for i,j1,j2,k */
44     p = 0;
45     l = i;
46     for (nu1=1;nu1<=nu;nu1++)
47       {
48       kn2 = l/2;
49       p = p * 2 + (l - 2*kn2);
50       l = kn2;
51       }
52     bitrev[i] = p;
53     }
54   }
55
56 /* Now perform FFT */
57 k = 0;
58 n2 = n/2;
59 nu1 = nu - 1;
60 multiplier = 6283.185 / (float) n;
61 for (l=1;l<=nu;l++)
62   {
63   while (k<n)
64     {
65     for (i=1;i<=n2;i++)
66       {
67       p = bitrev[k/power_2[nu1]];
68       kn2 = k + n2;
69       arg = (int) (multiplier * (float) p + half);
70       c = cosine[arg]; s = sine[arg];
71       treal = xreal[kn2]*c + ximag[kn2]*s;
72       timag = ximag[kn2]*c - xreal[kn2]*s;
73       xreal[kn2] = xreal[k] - treal;
74       ximag[kn2] = ximag[k] - timag;
75       xreal[k] += treal;
76       ximag[k] += timag;
77       k ++;
```

```
78        }
79      k += n2;
80      }
81    k = 0;
82    nu1--;
83    n2 /= 2;
84    }
85  for (k=0;k<n;k++)
86    {
87    i = bitrev[k];
88    if (i>k)
89      {
90      treal = xreal[k];
91      timag = ximag[k];
92      xreal[k] = xreal[i];
93      ximag[k] = ximag[i];
94      xreal[i] = treal;
95      ximag[i] = timag;
96      }
97    }
98  }
99  /*end of fft routine*/
```

Appendix G

Installing GRATIS on a Sun Workstation

1. Unpacking the distribution tape

The distribution saveset **gratis.tar** must first be restored into an appropriate directory. In the case of installation on the Computer Science Sun network, the home directory is **/home/venus/phys1219**. Thus, the unpack command:

```
%tar xfv gratis.tar
```

produced a directory structure in **/home/venus/phys1219/Release_3**. The **Release_3** directory was then renamed:

```
%mv -f Release_3 gratis
```

and henceforth the GRATIS main directory, **/home/venus/phys1219/gratis**, will be designated *SourceRoot*.

2. Configuring build_image

Before the GRATIS system can be compiled, **build_image** must be configured as per the manual and the instructions in *SourceRoot*/**src/plan_im/build_image/INSTALL**. If Siemens CT scanners are to be used as input devices to GRATIS, then the contents of the **siemens_common** subdirectory must be obtained from UNC Chapel Hill.

3. Editing the makefiles

SourceRoot/**Install/ModSun**, was created, a file similar to **BSDify4.3** except that the include and library paths have been modified within the **$CFLAGS** variable. This was done so that the support libraries could be installed within the GRATIS directory structure (and not in **usr/include** and **/usr/local/lib**). Additionally, any **install** commands within the makefiles needed to have the **-g plan** option removed since the user group **plan** is not defined on the Waikato network. The file **ModSun** is listed below:

```
#! /bin/csh -f
#
# David Murray, 12 December 1990
#
foreach i (`find $argv[1] -name \[Mm\]akefile -print`)
  echo doing $i...
  rm -f $i.bak
  chmod a+w $i
  mv -f $i $i.bak
  sed -e '/^CFLAGS/s/=/= -I\/home\/venus\/phys1219\/gratis\/src\/include/' \
      -e '/^CFLAGS/s/=/= -L\/home\/venus\/phys1219\/gratis\/lib/' \
      -e '/^INSOPTS/s/-g plan//' \
      < $i.bak > $i
```

```
  14    diff $i.bak $i
  15  end
  16  #
  17  echo Edit plan_config.h and top level makefile before building
```

Having created this file, the makefiles were edited by issuing the following commands:

```
%cd SourceRoot/Install
%save_makefiles ../src
%ModSun ../src
```

A library directory and two include subdirectories also had to be created, since on the Waikato system the support libraries are not installed in the usual place (`/usr/local/lib`):

```
%mkdir SourceRoot/lib
%mkdir SourceRoot/src/include/X11
%mkdir SourceRoot/src/include/XMT
```

4. Re-creating the X10 Compatibility Library

The version of *OpenWindows* distributed with the Sun system did not have the X10 header file `X10.h` and associated X10 compatibility library `liboldX.a`. The files `X10.h` and `X10.c`, listed below, were re-created from specifications found in X-windows reference texts:

```
   1  /*
   2   * X10.h - Header definition and support file for the C subroutine
   3   * interface library for V10 support routines.
   4   */
   5  #ifndef _X10_H_
   6  #define _X10_H_
   7
   8  #ifdef USG
   9  #ifndef __TYPES__
  10  #include <sys/types.h>
  11  #define __TYPES__
  12  #endif
  13  #else
  14  #include <sys/types.h>
  15  #endif
  16
  17  #include <X11/Xlib.h>
  18  #include <stdio.h>
  19
  20  /*
  21   * XAssoc - Associations used in the XAssocTable data structure.  The
  22   * associations are used as circular queue entries in the association table
  23   * which is contains an array of circular queues (buckets).
  24   */
  25  typedef struct _XAssoc {
  26  struct _XAssoc *next; /* Next object in this bucket. */
  27  struct _XAssoc *prev; /* Previous object in this bucket. */
  28  Display *display; /* Display which owns the id. */
  29  XID x_id; /* X Window System id. */
  30  char *data; /* Pointer to untyped memory. */
  31  } XAssoc;
  32
  33  /*
  34   * XAssocTable - X Window System id to data structure pointer association
  35   * table.  An XAssocTable is a hash table whose buckets are circular
  36   * queues of XAssoc's.  The XAssocTable is constructed from an array of
  37   * XAssoc's which are the circular queue headers (bucket headers).
  38   * An XAssocTable consists an XAssoc pointer that points to the first
  39   * bucket in the bucket array and an integer that indicates the number
  40   * of buckets in the array.
  41   */
  42  typedef struct {
  43      XAssoc *buckets;        /* Pointer to first bucket in bucket array.*/
  44      int size; /* Table size (number of buckets). */
  45  } XAssocTable;
  46
  47  #ifndef NeedFunctionPrototypes
  48  #if defined(FUNCPROTO)||defined(__STDC__)||defined(__cplusplus)||defined(c_plusplus)
  49  #define NeedFunctionPrototypes 1
  50  #else
```

```
#define NeedFunctionPrototypes 0
#endif /* __STDC__ */
#endif /* NeedFunctionPrototypes */

/*Function Prototypes*/
XAssocTable *XCreateAssocTable(
#if NeedFunctionPrototypes
    int                 /*Number of buckets*/
#endif
);

caddr_t XLookUpAssoc(
#if NeedFunctionPrototypes
    Display*            /*Display*/,
    XAssocTable*        /*Assoc table*/,
    XID                 /*Resource ID*/
#endif
);

XMakeAssoc(
#if NeedFunctionPrototypes
    Display*            /*Display*/,
    XAssocTable*        /*Assoc table*/,
    XID                 /*Resource ID*/,
    caddr_t             /*data*/
#endif
);

XDestroyAssocTable(
#if NeedFunctionPrototypes
    XAssocTable*        /*Assoc table*/
#endif
);

XDeleteAssoc(
#if NeedFunctionPrototypes
    Display*            /*Display*/,
    XAssocTable*        /*Assoc table*/,
    XID                 /*Resource ID*/
#endif
);

#endif /* _X10_H_ */
```

```
/*
 * X10.c - An attempt to duplicate liboldx.a
 *      Dave Murray, 1991
 *
 *      Note that the 'prev' field of XAssoc is not used in this
 *      implementation! Also, the first XAssoc in each bucket is skipped
 *      (this makes implementation of XMakeAssoc and XDestroyAssoc easier).
 */
#include "X10.h"

/*Functions*/

/* Create Association Table */
XAssocTable *XCreateAssocTable(size)
int size;  /*Number of buckets*/
{
XAssocTable *ptr;
int i;
/* Create association table */
ptr = (XAssocTable*) malloc(sizeof(XAssocTable));
if (ptr == NULL)
  return ptr;
/* Add size to structure */
ptr->size = size;
/* Create array of pointers to XAssoc */
ptr->buckets = (XAssoc*) calloc(size,sizeof(XAssoc));
if (ptr->buckets == NULL)
  {
  ptr = NULL;
  return ptr;
  }
/* Set all buckets to null*/
for (i=0;i<size;i++)
  ptr->buckets[i].next = NULL;
return ptr;
}

/*Make Association*/
XMakeAssoc(display,table,x_id,data)
Display *display;
XAssocTable *table;
XID x_id;
```

```
 43 caddr_t data;
 44 {
 45 XAssoc *ptr;
 46 int i;
 47 /* Find bucket by finding remainder */
 48 i = (int) x_id % table->size;
 49 /*Look through bucket, returning if key is found*/
 50 ptr = table->buckets[i].next;
 51 while (ptr != NULL)
 52   if (ptr->x_id==x_id)
 53     return;
 54   else
 55     ptr = ptr->next;
 56 /* Match not found - make new Xassoc */
 57 ptr = (XAssoc*) malloc(sizeof(XAssoc));
 58 if (ptr == NULL)
 59   {
 60   printf(stderr,"Error - bad allocation in XMakeAssoc\n");
 61   exit(1);
 62   }
 63 ptr->display = display;
 64 ptr->x_id = x_id;
 65 ptr->data = data;
 66 /* Add new Assoc to next to head of list */
 67 ptr->next = table->buckets[i].next;
 68 table->buckets[i].next = ptr;
 69 }
 70
 71 /* Look up association */
 72 caddr_t XLookUpAssoc(display,table,x_id)
 73 Display *display;
 74 XAssocTable *table;
 75 XID x_id;
 76 {
 77 XAssoc *ptr;
 78 int i;
 79 caddr_t return_value;
 80 /* Find bucket by finding remainder */
 81 i = (int) x_id % table->size;
 82 /*Look through bucket, returning if key is found*/
 83 ptr = table->buckets[i].next;
 84 while (ptr != NULL)
 85   if (ptr->x_id==x_id)
 86     return (ptr->data);
 87   else
 88     ptr = ptr->next;
 89 /* Key not found - return null value */
 90 return_value = NULL;
 91 return return_value;
 92 }
 93
 94 XDestroyAssocTable(table)
 95 XAssocTable *table;
 96 {
 97 int i;
 98 XAssoc *ptr,*ptr2;
 99 /* Free all assocs */
100 for (i=0;i<table->size;i++)
101   {
102   ptr = &(table->buckets[i]);
103   while (ptr != NULL)
104     {
105     ptr2 = ptr->next;
106     free(ptr);
107     ptr = ptr2;
108     }
109   }
110 /* Free buckets */
111 cfree(table->buckets);
112 /* Free assoc table */
113 free(table);
114 }
115
116 XDeleteAssoc(display,table,x_id)
117 Display *display;
118 XAssocTable *table;
119 XID x_id;
120 {
121 int i;
122 XAssoc *ptr,*prev_assoc;
123 /* Find bucket by finding remainder */
124 i = (int) x_id % table->size;
125 /*Look through bucket, deleting if key is found*/
126 prev_assoc = &(table->buckets[i]);
```

```
127 ptr = table->buckets[i].next;
128 while (ptr != NULL)
129   if (ptr->x_id==x_id)
130     {
131     prev_assoc->next = ptr->next;
132     free(ptr);
133     return;
134     }
135   else
136     {
137     prev_assoc = ptr;
138     ptr = ptr->next;
139     }
140 }
141 /* end of X10.c */
```

Note that only the association table functions are required by GRATIS, and hence they are the only functions implemented. These files were placed in a directory **mods/X10** in the user's home directory, and compiled using **%make** with the following makefile:

```
 1 LIBDIR = /home/venus/phys1219/gratis/lib
 2 INCDIR = /home/venus/phys1219/gratis/src/include/X11
 3 CFLAGS = -O -fsingle -DHAVE_BCOPY
 4
 5 OBJS = X10.o
 6
 7 liboldX.a: $(OBJS)
 8         rm -f liboldX.a
 9         ar crvu liboldX.a $(OBJS)
10         -ranlib liboldX.a
11
12 $(OBJS): makefile
13
14 install: liboldX.a
15         cp liboldX.a $(LIBDIR)/liboldX.a
16         -ranlib $(LIBDIR)/liboldX.a
17         cp X10.h $(INCDIR)/X10.h
18
19 clean:
20         rm -f core a.out *.o *~
21 }
```

The header and library were then installed in the appropriate places:

```
%make install
%make -k clean
```

Note that `liboldX` and other libraries (such as `libXMT` and `XMenu11`) require a symbolic link between `/usr/include/X11` and `usr/openwin/include/X11`, and also between `/usr/lib` and `/usr/openwin/lib`.

5. Building the Support Libraries

The support libraries found in *SourceRoot*`/Support` were then built:

```
%cd SourceRoot/Support/lib3d
%make
%cp *.h ../../src/include
%cp lib3d.a ../../lib
%make clean
%cd SourceRoot/Support/libc_cpde
%make
%cp *.h ../../src/include
%cp libc_cpde.a ../../lib
%make clean
%cd SourceRoot/Support/libcpde
%make
%cp *.h ../../src/include
```

```
%cp libcpde.a ../../lib
%make clean
%cd SourceRoot/Support/libntab
%make
%cp *.h ../../src/include
%cp libntab.a ../../lib
%make clean
%cd SourceRoot/Support/libplot/hp7550
%make
%cp *.h ../../../src/include
%cp libhpgl.a ../../../lib
%make clean
%cd SourceRoot/Support/libXMT
%make
%cp *.h ../../src/include/XMT
%cp libXMT.a ../../lib
%make clean
```

For some of the support libraries there are test programs to validate compilation of the libraries.

To build the **XMenu11** support library, *SourceRoot*/**Support/XMenu11/Makefile** had to be edited so that the **INCLUDES=** command became:

```
INCLUDES = $(TOP) -ISourceRoot/src/include
```

to enable the library **liboldX** to be successfully included. The library was then built in the usual way:

```
%cd SourceRoot/Support/XMenu11
%make
%cp *.h ../../src/include
%cp Xmenu.h ../../src/include/X11
%cp libXMenu11.a ../../lib
%make clean
```

To build the **libzbuf** support library, *SourceRoot*/**Support/libzbuf/Makefile** had to be edited so that the dependency on the last line became:

```
zbuf.o:  /usr/include/stdio.h /usr/include/sys/file.h zbuf.h
```

and in **zbuf.c**, **<zbuf.h>** must be changed to **"zbuf.h"**. Once again the library was built in the usual way:

```
%cd SourceRoot/Support/libzbuf
%make
%cp *.h ../../src/include
%cp libzbuf.a ../../lib
%make clean
```

6. Making dependencies

The source code dependencies were then made using:

```
%cd SourceRoot/src
%make -k clean
%make -k depend >&depend_report &
```

(which runs in the background), then the file **depend_report** was examined for errors. One modification required was the adding of a blank line at the end of *SourceRoot*/**src/usr.image_interface/reduce_con/Makefile**, so that the **ex** editor commands work correctly. Also, *SourceRoot*/**src/plot_plan/defines.h** had to be copied to *SourceRoot*/**src/plot_beam_outline** to enable correct compilation. Warnings still present in the listing included references to absent RCS (Revision Control System) directories, and also files absent in the **build_image** directories (particularly **siemens_common** and **"defs.h"**).

7. Editing plan_config.h and top-level makefile

The header file *SourceRoot*/**src/include/plan_config.h** must be edited to reflect the type of UNIX, and also whether the XDR data format is supported. In this case (Sun UNIX and XDR supported), no change to the file was necessary.

The top-level makefile *SourceRoot*/**src/Makefile** was edited so that the variable **UP** pointed to the target directory for binaries and data:

```
UP = /home/venus/phys1219/GRATIS
```

This directory will henceforth be known as *BinaryRoot*.

8. Building the source code

The source code was then built using:

```
%cd SourceRoot/src
%make -k clean
%make -k >&build_report &
```

(which runs in the background), then the file **build_report** was examined for errors. In the file *SourceRoot*/**src/display/plot_beam_outline**, the **CFLAGS** line had to be changed to include = **-I../../../src/display/plot_plan**, and references to **$UP** had to be removed from the **LIBS** line. It was also necessary to shift the **usr_image** library:

```
%cp SourceRoot/src/usr.image_interface/libim/libim.a SourceRoot/lib
```

and then remake **plan_to_im**. Error messages remaining in the **build_report** included an illegal pointer warning in *SourceRoot*/**src/plan_im/build_image/di/BI_utils.c**, **build_image** errors due to the absent **siemens_common** directory, and various errors due to the absence of **usr/image** software on the Waikato system.

9. Installing the binaries and data

The binaries were then installed in *BinaryRoot*:

```
make -k clean
%cd SourceRoot/src
%make -k install >&install_report &
```

and `install_report` examined for errors. `c_brachy` warns that the installer must be a superuser to install properly (since ownership of the file is changed to `root`), but this is not a problem provided GRATIS is run from the installer's account. An error when making *SourceRoot*`/src/brachy/phys_dat/makefile` causes execution to terminate, but by modifying the install line to read:

```
-(cd $(INSTALL_DIR) ; make_seed_dat ; make_ls_dat)
```

(i.e. adding the preceding hyphen), execution seems to proceed normally. The codes could then be installed manually:

```
%make -k install UP=BinaryRoot
```

Similarly, a hyphen must be added in *SourceRoot*`/src/photon/phys_dat/unit/Makefile`:

```
-cat ../MACHINES/*/unit | $(INSDIR)/$(BINARY)
```

which was then be installed as above.

The path *BinaryRoot*`/bin` was then added to the path statement in `.cshrc`, and access to *OpenWindows* and GRATIS manual pages was enabled by adding the following line to `.login`:

```
setenv MANPATH /usr/man:/usr/openwin/man:SourceRoot/pman
```

10. Testing

The GRATIS virtual simulator software was then tested by issuing the commands:

```
%cd SourceRoot
%simulate XDR_test_data/rando
```

from inside the *OpenWindows* window manager. `xvsim` ran correctly, except that text data (such as beam and point names) could not be entered correctly. This was initially remedied by explicitly setting the Xwindows keyboard focus within the `EnterNotify` case of *SourceRoot*`/Support/libXMT/XMT_button_event`:

```
XSetInputFocus(widget->display,widget->self,RevertToParent,CurrentTime);
```

and then re-compiling the library and dependent source code. However, the problem arises because the *OpenLook* window manager initialises the `input` field of the `XWMHints` structure

as **False**, whereas the passive keyboard input model used by **xvsim** and other applications requires that it be **True**. This was remedied by adding the following lines to *SourceRoot*/Support/libXMT/XMT_make_widget.c, just after it sets the standard X properties:

```
XWMHints wmhints;
wmhints.flags = InputHint;
wmhints.input = True;
XSetWMHints(widget->display,widget->self,&wmhints);
```

After defining appropriate beams using **xvsim**, the script **extbm** was run to specify the grid and beam weighting, then calculate the dose distribution (using the dSAR algorithm supplied).

11. Defining physics data for the Waikato Clinac 18/10

In order to generate plans for the Waikato Hospital Clinac 18/10, appropriate physics data, located in the directory *SourceRoot*/src/photon/phys_dat/MACHINES/Clinac18, had to be supplied. Three files, **unit**, **sar**, and **time_calc**, were required (the formats for these files were obtained from the other treatment machines present in the **MACHINES** directory). In addition, the files **tray** and **filter** can be added to define blocking trays and filters, respectively. The makefile in *SourceRoot*/src/photon/phys_dat/unit must then be re-executed:

```
%cd SourceRoot/src/photon/phys_dat/unit
%make -k UP=BinaryRoot
%make -k install UP=BinaryRoot
%make -k clean
```

in order to make the new machine available to **c_photon**.

Appendix H

Manual Page for s_photon

S_PHOTON (1P) GRATIS S_PHOTON(1P)

NAME

s_photon – calculate doses due to external beams using superposition

SYNOPSIS

s_photon -a skin_anastruct **-b** beam_file **-i** image_file [**-g** grid_description **-o** output_grid] or [**-p** point_description **-O** output_point_dose] or [**-U** unit_id] [options]

DESCRIPTION

S_photon is a 3D implementation of the superposition algorithm developed at Waikato University. Superposition is performed in real space using primary and scattered kernels of arbitrary dimensions and resolutions. It computes dose to points using the "deposition point of view," and to grids using the "interaction point of view."

Options:

-S scaling_type

defines the kernel scaling type (*none*, *zscaling* or *raytrace*, default = *none*)

-T processor_type

defines the host processor (*sun* or *transputer*, default = *sun*)

-N processor_number

defines the number of processing elements (default = *1*)

-V vector_size

defines the communication vector size for a transputer network (default = *1*)

-R remote_server_node

defines a machine on which a remote dose-computation server is running (default = null)

The skin anastruct has to be named "skin".

FILES

UP/phys_dat/*machine*{header,sup_dat,edk#.##}

SEE ALSO

D.C. Murray's and P.W. Hoban's theses.

BUGS

Slow unless implemented in parallel!

AUTHORS

David Murray and Peter Hoban, University of Waikato, Hamilton, N.Z.

Appendix I
PC Hardware Configuration and Switch Settings

Transtech TMB08 Motherboard

Jumper Settings

Jumper	*Setting*
I (1)	Link in (module 0 reset = UP)
O (2)	Link out (module N reset from IBM)
R (3)	Link out (ROM not selected)
1 (4)	Link in (link adaptors disabled)
2 (5)	Link in (link adaptors disabled)
3 (6)	Link in (link adaptors disabled)
S (7)	Link out (20 MHz links)

Link Patch

PL0 to M0L0 (pipehead to Link 0)

Socket Adaptor

UP connected to subsystem control on Videoputer
Link 8 (patch link 0) connected to Link 1 of Videoputer

Microway Videoputer

Switches

Switch	*Setting*
1	1 (20 MHz processor speed)
2	1 (20 MHz processor speed)
3	1 (20 MHz processor speed)
4	0 (20 MBits/sec link speed)
5	0 (20 MBits/sec link speed)
6	0 (20 MBits/sec link speed)
7	0 (20 MBits/sec link adaptor speed)
8	1 (root processor mode)
9	0 (default hex 150 I/O base address)
10	0 (video memory in high address space)

Link Socket Connections

Subsystem Control connected to UP on TMB08
Link 1 connected to Link 8 (patch link 0) of TMB08

Strap Fields

Jumper	*Setting*
J1	Middle (4 wait states)
J2	Link socket
J3	Left (normal palette)
J4	Left (active low H Sync)
J5	Left (active low V Sync)
J6	Enable

Appendix J

Configuration Script and Example Output File

Configuration Script

```
# s_config
# script to generate 3L configuration file for linear array
# David Murray, 1991
# Usage: s_config proc_no > s_trans_calc.cfg
# (where proc_no is the number of transputers in the array)
#
echo '\! Configuration file for '$1' processors'
echo '\!'
echo 'Processor Host'
@ i = 0
while ($i < $1)
  echo "Processor Proc$i"
  @ i = $i + 1
end
echo '\!'
echo 'Wire ? Host[0] Proc0[0]'
@ i = 0
while ($i < ( ${1} - 1 ))
  @ j = $i + 1
  echo "Wire ? Proc${i}[2] Proc${j}[1]"
  @ i = $i + 1
end
echo '\!'
echo 'Task DevDriver Ins=1 Outs=1'
@ i = 0
while ($i < $1)
  echo "Task Filter${i} File="'"s_filter.b4" Ins=3 Outs=3 Data=20k Urgent'
  echo "Task Worker${i} File="'"s_worker.b4" Ins=1 Outs=1 Stack=2k Heap=?' \
       'Opt=Stack Opt=Code'
  @ i = $i + 1
end
echo '\!'
echo 'Connect ? DevDriver[0] Filter0[0]'
echo 'Connect ? Filter0[0] DevDriver[0]'
@ i = 0
while ($i < $1)
  echo "Connect ? Filter${i}[1] Worker${i}[0]"
  echo "Connect ? Worker${i}[0] Filter${i}[1]"
  if ($i > 0) then
    @ j = $i - 1
    echo "Connect ? Filter${i}[0] Filter${j}[2]"
    echo "Connect ? Filter${j}[2] Filter${i}[0]"
  endif
  @ i = $i + 1
end
@ i = $1 - 1
echo "Bind Input Filter${i}[2] Value=0"
echo "Bind Output Filter${i}[2] Value=0"
echo '\!'
echo 'Place DevDriver Host'
@ i = 0
while ($i < $1)
  echo "Place Filter${i} Proc${i}"
  echo "Place Worker${i} Proc${i}"
  @ i = $i + 1
end
# end of s_config
```

Example Output File

```
 1 ! Configuration file for 4 processors
 2 !
 3 Processor Host
 4 Processor Proc0
 5 Processor Proc1
 6 Processor Proc2
 7 Processor Proc3
 8 !
 9 Wire ? Host[0] Proc0[0]
10 Wire ? Proc0[2] Proc1[1]
11 Wire ? Proc1[2] Proc2[1]
12 Wire ? Proc2[2] Proc3[1]
13 !
14 Task DevDriver Ins=1 Outs=1
15 Task Filter0 File="s_filter.b4" Ins=3 Outs=3 Data=20k Urgent
16 Task Worker0 File="s_worker.b4" Ins=1 Outs=1 Stack=2k Heap=? Opt=Stack Opt=Code
17 Task Filter1 File="s_filter.b4" Ins=3 Outs=3 Data=20k Urgent
18 Task Worker1 File="s_worker.b4" Ins=1 Outs=1 Stack=2k Heap=? Opt=Stack Opt=Code
19 Task Filter2 File="s_filter.b4" Ins=3 Outs=3 Data=20k Urgent
20 Task Worker2 File="s_worker.b4" Ins=1 Outs=1 Stack=2k Heap=? Opt=Stack Opt=Code
21 Task Filter3 File="s_filter.b4" Ins=3 Outs=3 Data=20k Urgent
22 Task Worker3 File="s_worker.b4" Ins=1 Outs=1 Stack=2k Heap=? Opt=Stack Opt=Code
23 !
24 Connect ? DevDriver[0] Filter0[0]
25 Connect ? Filter0[0] DevDriver[0]
26 Connect ? Filter0[1] Worker0[0]
27 Connect ? Worker0[0] Filter0[1]
28 Connect ? Filter1[1] Worker1[0]
29 Connect ? Worker1[0] Filter1[1]
30 Connect ? Filter1[0] Filter0[2]
31 Connect ? Filter0[2] Filter1[0]
32 Connect ? Filter2[1] Worker2[0]
33 Connect ? Worker2[0] Filter2[1]
34 Connect ? Filter2[0] Filter1[2]
35 Connect ? Filter1[2] Filter2[0]
36 Connect ? Filter3[1] Worker3[0]
37 Connect ? Worker3[0] Filter3[1]
38 Connect ? Filter3[0] Filter2[2]
39 Connect ? Filter2[2] Filter3[0]
40 Bind Input Filter3[2] Value=0
41 Bind Output Filter3[2] Value=0
42 !
43 Place DevDriver Host
44 Place Filter0 Proc0
45 Place Worker0 Proc0
46 Place Filter1 Proc1
47 Place Worker1 Proc1
48 Place Filter2 Proc2
49 Place Worker2 Proc2
50 Place Filter3 Proc3
51 Place Worker3 Proc3
```

Bibliography

3L, *Parallel C User Guide.* 3L Limited, Livingston, Scotland (1988).

3L, *Parallel FORTRAN User Guide.* 3L Limited, Livingston, Scotland (1988).

Agarwal R.C. and Cooley J.W., New algorithms for digital convolution. *Proceedings of the 1977 International Conference on Acoustics, Speech and Signal Processing (Hartford), p360* (1977).

Ahnesjö A., Collapsed cone convolution of radiant energy for photon dose calculation in heterogeneous media. *Med. Phys.* (1989), **16** 577.

Ahnesjö A., Andreo P. and Brahme A., Calculation and application of point spread functions for treatment planning with high energy photon beams. *Acta Oncologica* (1987), **26** 49.

Andreo P., Stopping-power ratios for dosimetry. In *Monte Carlo Transport of Electrons and Photons, p485*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Andrews G.R. and Schneider F.B., Concepts and notations for concurrent programming. *Comput. Surv.* (1983), **15 (1)** 3.

Athas W.C. and Seitz C.L., Multicomputers: Message-passing concurrent computers. *Comput.* (1988), **21 (8)** 9.

AT&T, *Unix System V Programmer's Reference Manual.* Prentice-Hall, Englewood Cliffs, New Jersey (1987).

Babb R.G., Storc L. and Hiromoto R., Developing a parallel Monte Carlo transport algorithm using large-grain data flow. *Parallel Comput.* (1988), **7** 187.

Batho H.F., Lung corrections in cobalt 60 beam therapy. *J. Can. Assoc. Radiol.* (1964), **15** 79.

Battista J., Field C., Santon L. and Barnett R., Radiotherapy planning on a VAX-11/780 computer. In *Proceedings of the Eighth International Conference on the Use of Computers in Radiation Therapy,* July 9-12, 1984.

Bell G., The future of high performance computers in science and engineering. *Commun. ACM* (1989), **32 (9)** 1091.

Berger M.J., Monte Carlo calculations of the penetration and diffusion of fast charged particles. In *Methods in Computational Physics, Vol. 1, p135*, edited by Alder B., Fernbach S., and Rotenberg M., Academic Press, New York (1963).

Berger M.J., ETRAN — experimental benchmarks. In *Monte Carlo Transport of Electrons and Photons, p183,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Berger M.J. and Seltzer S.M., *Stopping Powers and Ranges of Electrons and Positrons.* NBSIR 82–2550, National Bureau of Standards, Washington D.C. (1982).

Berger M.J. and Wang R., Multiple-scattering angular deflections and energy-loss straggling. In *Monte Carlo Transport of Electrons and Photons, p21*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Bielajew A.F. and Rogers D.W.O., PRESTA: The parameter reduced electron-step transport algorithm for electron Monte Carlo transport. *Nucl. Instr. and Meth.* (1987), **B18** 165.

Bielajew A.F. and Rogers D.W.O., Electron step-size artefacts and PRESTA. In *Monte Carlo Transport of Electrons and Photons, p115*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Bielajew A.F. and Rogers D.W.O., Variance-reduction techniques. In *Monte Carlo Transport of Electrons and Photons, p407*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Bielajew A.F., Rogers D.W.O., Cygler J. and Battista J.J., A comparison of electron pencil beam and Monte-Carlo calculational methods. In *The Use of Computers in Radiation Therapy, p65*, edited by Bruinvis I.A.D. *et al.*, Elseviere Science Publishers, Holland (1987).

Bielajew A.F., Rogers D.W.O. and Nahum A.E., The Monte Carlo simulation of ion chamber response to ^{60}Co — resolution of anomalies associated with interfaces. *Phys. Med. Biol.* (1985), **30** 419.

Boyer A.L. and Mok E.C., A photon dose distribution model employing convolution calculations. *Med. Phys.* (1985), **12** 169.

Boyer A.L. and Mok E.C., Calculation of photon dose distributions in an inhomogeneous medium using convolutions. *Med. Phys.* (1986), **13** 503.

Boyer A.L., Wackwitz R. and Mok E., A comparison of the speeds of three convolution algorithms. *Med. Phys.* (1988), **15** 224.

Boyer A.L., Zhu Y., Wang L. and Francois P., Fast Fourier transform convolution calculations of x-ray isodose distributions in homogeneous media. *Med. Phys.* (1989), **16** 248.

Brigham E.O, *The Fast Fourier Transform.* Prentice-Hall, Englewood Cliffs, N.J. (1974).

British Institute of Radiology Supplement 17,*Central Axis Depth Dose Data for Use in Radiotherapy.* British Institute of Radiology, London (1983).

Brown C.M., *Human-Computer Interface Design Guidelines.* Ablex Publishing Corporation, Norwood, New Jersey (1988).

Brown F.B. and Martin W.R., Monte Carlo methods for radiation transport analysis on vector computers. *Prog. in Nucl. Ener.* (1984), **14** 269.

Chan H.-P. and Doi K., Monte Carlo simulation in diagnostic radiology. In *Monte Carlo Simulation in the Radiological Sciences, p103*, edited by Morin R.L., CRC Press (1988).

Chui C. and Mohan R., Off-center ratios for three-dimensional dose calculations. *Med. Phys.* (1986), **13** 409.

Chui C. and Mohan R., Extraction of pencil beam kernels by the deconvolution method. *Med. Phys.* (1988), **12** 138.

Cook A.J., *Mortran3 Users Guide.* SLAC Computation Research Group Technical Memorandum CGTM–209 (1983).

Cowan R. and Nelson W.R., *Producing EGS4 Shower Displays with Unified Graphics.* Stanford Linear Accelerator Center Report TN–87–3 (1987).

Cunningham J.R., Tissue inhomogeneity corrections in photon-beam treatment planning. In *Progress in Medical Radiation Physics, Volume 1, p103,* edited by Orton C.G. Plenum Press, New York (1982).

Cvetanovic Z., The effect of problem partitioning, allocation, and granularity on the performance of multiple-processor systems. *IEEE Trans. Computers* (1987), **36** 421.

Dean R.D., A scattering kernel for use in true three-dimensional dose calculations. *Med. Phys.* (1980), **7** 429.

Del Guerra A. and Nelson W.R., Positron emission tomography applications of EGS. In *Monte Carlo Transport of Electrons and Photons, p469,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Desrochers G.R., *Principles of Parallel and Multi-processing.* McGraw-Hill, New York (1987).

Dudgeon D.E. and Mersereau R.M., *Multidimensional Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, N.J. (1984).

Electronics, Supercomputers: The Proliferation Begins (Special Issue). *Electronics* (1988), **61 (5)** (March 3), 51.

Elliott D.F. and Rao K.R., *Fast Transforms: Algorithms, Analyses, Applications.* Academic Press, Orlando, Florida (1982).

Field G.C. and Battista J.J., Photon dose calculations using convolution in real and Fourier space: Assumptions and time estimates. In *The Use of Computers in Radiation Therapy, p103,* edited by Bruinvis I.A.D. *et al.*, Elsevier Science Publishers (North-Holland) (1987).

Flynn M.J., Very high speed computing systems. *Proc. IEEE* (1966), **54** 1901.

Flynn M.J., Some computer organizations and their effectiveness. *IEEE Trans. Computers* (1972), **21** 948.

Ford R.L. and Nelson W.R., *The EGS Code System: Computer Programs for the Monte Carlo Simulation of Electromagnetic Cascade Showers (Version 3).* Stanford Linear Accelerator Center Report SLAC–210 (1978).

Goitein M., Limitations of two-dimensional treatment planning programs. *Med. Phys..* (1982), **9** 580.

Goitein M., Calculation of the uncertainty in the dose delivered during radiation therapy. *Med. Phys.* (1985), **12** 608.

Gonzalez R.C. and Wintz P.A., *Digital Image Processing (2nd edition).* Addison-Wesley, Reading, Massachusetts (1987).

Greening J.R., *Fundamentals of Radiation Dosimetry (2nd edition).* Medical Physics Handbook Number 15, Adam Hilger Ltd, Bristol (1985).

Halbleib J., Structure and operation of the ITS Code System. In *Monte Carlo Transport of Electrons and Photons, p249,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Han K., Ballon C., Chui C. and Mohan R., Monte Carlo simulation of a cobalt-60 beam. *Med. Phys.* (1987), **14** 414.

Harbison S.P. and Steele G.L., *A C Reference Manual.* Prentice-Hall, Englewood Cliffs, New Jersey (1984).

Harper N.R., *The Design, Construction, and Use of a Thin Window Ionization Chamber.* University of Waikato M.Sc. Thesis (1990).

Haviland K. and Salama B., *Unix System Programming.* Addison-Wesley, Wokingham, England (1987).

Hayward E. and Hubbell J., The albedo of various materials for 1–MeV photons. *Phys. Rev.* (1954), **93 (5)** 955.

Heitler W., *The Quantum Theory of Radiation.* Oxford University Press, New York (1954).

Hendee W.R., *Radiation Therapy Physics.* Year Book Medical Publishers, Chicago (1981).

Hoare C.A.R., *Communicating Sequential Processes.* Prentice-Hall, Englewood-Cliffs, New Jersey (1985).

Hoban P.W., *Lateral Electron Disequilibrium in Radiotherapy Treatment Planning.* University of Waikato D.Phil. Thesis (1991).

Hoban P.W., Murray D.C., Metcalfe P.E. and Round W.H., Superposition dose calculation in lung for 10MV photons. *Australas. Phys. Eng. Sci. Med.* (1990), **13 (2)** 81.

Hockney R.W. and Jesshope C.R., *Parallel Computers: Architecture, Programming and Algorithms.* Adam Hilger Ltd, Bristol (1981).

Hogstrom K.R., Mills M.D. and Almond P.R., Electron beam dose calculations. *Phys. Med. Biol.* (1981), **26 (3)** 445.

Huang T.S. (editor), *Topics in Applied Physics Volume 6 — Picture Processing and Digital Filtering (2nd edition).* Springer-Verlag (1979).

Ibbett R.N. and Topham N.P., *Architecture of High Performance Computers (Volume I): Uniprocessors and Vector Processors.* MacMillan, London (1989).

Ibbett R.N. and Topham N.P., *Architecture of High Performance Computers (Volume II): Array Processors and Multiprocessor Systems.* MacMillan, London (1989).

International Commission on Radiation Units and Measurements, *Determination of Absorbed Dose in a Patient Irradiated by Beams of X or Gamma Rays in Radiotherapy Procedures.* ICRU Report 24, Washington D.C. (1976).

International Commission on Radiation Units and Measurements, *Radiation Quantities and Units.* ICRU Report 33, Washington D.C. (1980).

Inmos, *IMS–B012 User Guide and Reference Manual.* Inmos, Bristol, U.K. (1988).

Inmos, *The Transputer Reference Manual.* Prentice-Hall, U.K. (1988).

Inmos, *The Transputer Applications Notebook.* Inmos, Bristol, U.K. (1989).

Inmos, *Transputer Technical Notes.* Prentice-Hall, U.K. (1989).

Inmos, *The T9000 Transputer Products Overview Manual.* Inmos, Bristol, U.K. (1991).

Ito A., Electron track simulation for microdosimetry. In *Monte Carlo Transport of Electrons and Photons, p361*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Jamieson L.H., Mueller P.T. and Siegel H.J., FFT algorithms for SIMD parallel processing systems. *J. Parallel Dist. Comput.* (1986), **3** 48.

Jenkins T.M., Nelson W.R. and Rindi A. (editors), *Monte Carlo Transport of Electrons and Photons.* Plenum Press, New York (1988).

Johns H., Bates L., and Watson T., 1000 Curie cobalt units for radiation therapy: I. The Saskatchewan cobalt-60 unit. *Br. J. Radiol.* (1952), **25** 296.

Johns H.E. and Cunningham J.R., *The Physics of Radiology (4th edition).* Charles C. Thomas, Springfield, Illinois (1983).

Kahn H., *Applications of Monte Carlo.* USAEC Report AECU-3259 (1954), 62.

Kalet I.J. and Jacky J.P., Radiation therapy treatment planning using concurrent programming. *Comput. Programs Biomed.* (1988), **26** 115.

Kernighan B.W. and Ritchie D.M., *The C Programming Language.* Prentice-Hall, Englewood Cliffs, New Jersey (1978).

Klevenhagen S.C., *Physics of Electron Beam Therapy.* Medical Physics Handbook Number 13, Adam Hilger Ltd, Bristol (1985).

Knuth D.E., *The Art of Computer Programming, Volume II.* Addison Wesley, Reading, Massachusetts (1981).

Krishnamurthy E.V., *Parallel Processing: Principles and Practice.* Addison-Wesley, New York (1989).

Kruatrachue B. and Lewis T., Grain size determination for parallel processing. *IEEE Softw.* (1988), **5 (1)** 23.

Lawrence H.P., A public domain PC based treatment planning system. *Phys. Med. Biol.* (1990), **35 (6)** 787.

Levitan S.P., Evaluation criteria for communication structures in parallel architectures. In *Proceedings of the International Conference on Parallel Processing, 1985.* IEEE, New York (1985).

McCracken D.D., The Monte Carlo method. *Sci. Am.* (1955), **192** 90.

McCullough E.C. and Holmes T.W., Acceptance testing computerized radiation therapy treatment planning systems: Direct utilization of CT scan data. *Med. Phys.* (1985), **12** 237.

McGrath E.J. and Crawford D.F., *Techniques for Efficient Monte Carlo Simulation, Vols I,II and III.* Radiation Shielding Information Center, Oak Ridge National Laboratory Report ORNL-RSIC-38 (1975).

Mackie T.R., *The study of megavoltage x-ray beams.* University of Alberta Ph.D. Thesis (1985).

Mackie T.R., Bielajew A.F., Rogers D.W.O. and Battista J.J., Generation of photon energy deposition kernels using the EGS Monte Carlo Code. *Phys. Med. Biol.* (1988), **33** 1.

Mackie T.R., El-Khatib E., Battista J.J., Scrimger J., Van Dyk J., and Cunningham J.R., Lung dose corrections for 6 and 10 MV x-rays. *Med. Phys.* (1985), **12** 327.

Mackie T.R., Kubsad S.S., Rogers D.W.O. and Bielajew A.F., The OMEGA Project: Elec-

tron dose planning using Monte Carlo simulation. Paper BB1 of the 32nd annual meeting of the AAPM, 1990, listed in the Addendum to Scientific Paper Abstracts and Sessions, *Med. Phys.* (1990), **17** 510. See also abstracts F20 and P25 of the 33rd annual meeting of the AAPM, *Med. Phys.* (1991), **18 (3)**.

Mackie T.R., Scrimger J.W., and Battista J.J., A convolution method of calculating dose for 15-MV x rays. *Med. Phys.* (1985), **12** 188.

Magnenat-Thalmann N. and Thalmann D., *Image Synthesis.* Springer-Verlag, Tokyo (1987).

Metcalfe P.E., *X-ray Beam Modelling in Radiotherapy: The Effect of Lung Inhomogeneities.* University of Waikato D.Phil. Thesis (1990).

Metcalfe P.E. and Beckham W.A., Radiotherapy planning accuracy in terms of C.T. numbers and inhomogeneity correction techniques. *Australas. Radiol.* (1988), **32** 371.

Metcalfe P.E, Beckham W.A., Long B.H. and Battista J.J, The effect of patient density variation on radiotherapy dose calculations. *Australas. Phys. Eng. Sci. Med.* (1988), **11 (3)** 107.

Metcalfe P.E., Hoban P.W., Murray D.C. and Round W.H., Modelling polychromatic high energy photon beams by superposition. *Austral. Phys. Eng. Sci. Med.* (1989), **12 (3)** 138.

Metcalfe P.E., Hoban P.W., Murray D.C. and Round W.H., Beam hardening of 10MV x-rays: Analysis using a convolution/superposition method. *Phys. Med. Biol.* (1990), **35 (11)** 1533.

Meyer H.A. (editor), *Symposium on Monte Carlo Methods.* John Wiley and Sons, New York (1954).

Microway, *Videoputer User Manual.* Microway Limited, Surrey, U.K. (1988).

Miura K., EGS4-V: Vectorization of the Monte Carlo cascade shower simulation code EGS4. *Comput. Phys. Comm.* (1987), **45** 127.

Miura K. and Babb R.G., Tradeoffs in granularity and parallelization for a Monte Carlo shower simulation code. *Parallel Comput.* (1988), **8** 91.

Mohan R., Dose calculations for three-dimensional radiation treatment planning. *Austral. Phys. Eng. Sci. Med.* (1989), **12 (4)** 241.

Mohan R., Barest R., Brewster L.J., Chui C.S., Kutcher G.J., Laughlin J.S. and Fuks Z., A comprehensive three-dimensional radiation treatment planning system. *Int. J. Rad. Onc. Biol. Phys.* (1988) **15 (2)** 481.

Mohan R. and Chui C., Validity of the concept for separating primary and scatter dose. *Med. Phys.* (1985), **12** 726.

Mohan R., Chui C. and Lidofsky L., Energy and angular distributions of photons from medical linear accelerators. *Med. Phys.* (1985), **12** 592.

Mohan R., Chui C. and Lidofsky L., Differential pencil beam dose computation model for photons. *Med. Phys.* (1986), **13** 64.

Morin R.L., Raeside D.E., Goin J.E., and Widman J.C., Random number generation and testing. In *Monte Carlo Simulation in the Radiological Sciences, p37*, edited by Morin R.L., CRC Press (1988).

Mould R.F., *Radiotherapy Treatment Planning (2nd edition)*. Medical Physics Handbook Number 14, Adam Hilger Ltd, Bristol (1985).

Murray D.C., Hoban P.W., Metcalfe P.E. and Round W.H., 3-D superposition for radiotherapy treatment planning using fast Fourier transforms. *Australas. Phys. Eng. Sci. Med.* (1989), **12 (3)** 128.

Murray D.C., Hoban P.W., Round W.H., Graham I.D, and De Vel O.Y., Radiotherapy treatment planning using transputers. *N.Z. Journal of Computing* (1989), **1 (2)** 30.

Murray D.C., Hoban P.W., Round W.H., Graham I.D. and Metcalfe P.E., Superposition on a multicomputer system. *Med. Phys.* (1991), **18** 468.

Nahum A.E., Overview of photon and electron Monte Carlo. In *Monte Carlo Transport of Electrons and Photons, p3*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Nahum A.E., Simulation of dosimeter response and interface effects. In *Monte Carlo Transport of Electrons and Photons, p523*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Nath R., Monte Carlo simulations in radiation therapy. In *Monte Carlo Simulation in the Radiological Sciences, p209*, edited by Morin R.L., CRC Press (1988).

Neblett D.L. and Hogan S.E., Local area networks as a multiprocessor treatment planning system. In *The Use of Computers in Radiation Therapy, p567*, edited by Bruinvis I.A.D. *et al.*, Elsevier Science Publishers (North Holland) (1987).

Nelson W.R., Hirayama H. and Rogers D.W.O., *The EGS4 Code System*. Stanford Linear Accelerator Center Report SLAC-265 (1985).

Nelson W.R. and Jenkins T.M., *Writing Subroutine HOWFAR for EGS4*. Stanford Linear Accelerator Center Report TN–87–4 (1988).

Nelson W.R. and Rogers D.W.O., Structure and operation of the EGS4 Code System. In *Monte Carlo Transport of Electrons and Photons, p287*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Newman W.M. and Sproull R.F., *Principles of Interactive Computer Graphics (2nd edition)*. McGraw-Hill (1979).

Niemierko A. and Goitein M., The influence of the size of the grid used for dose calculation on the accuracy of dose estimation. *Med. Phys.* (1989), **16** 239.

Niemierko A. and Goitein M., The use of variable grid spacing to accelerate dose calculations. *Med. Phys.* (1989), **16** 357.

Nussbaumer H.J., *Fast Fourier Transform and Convolution Algorithms* (2nd Edition). Springer-Verlag, N.Y. (1982).

Nussbaumer H.J. and Quandalle P., Fast computation of discrete Fourier transforms using polynomial transforms. *IEEE Trans. Acoust., Speech, Signal Processing* (1979), **ASSP-27** 169.

O'Connor J.E., The variation of scattered x-rays with density in an irradiated body. *Phys. Med. Biol.* (1957), **1** 352.

Orton C.G. Mondalek P.M. Spicka J.T. Herron D.S. and Andres L.I., Lung corrections in

photon beam treatment planning: Are we ready? *Int. J. Rad. Onc. Biol. Phys.* (1984), **10** 2191.

Press W.H., Flannery B.P., Teukolsky S.A. and Vetterling W.T., *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, U.K. (1986).

Quinn M.J., *Designing Efficient Algorithms for Parallel Computers.* McGraw-Hill, New York (1987).

Raeside D.E., Monte Carlo principles and applications. *Phys. Med. Biol.* (1976), **21** 181.

Reed D.A. and Fujimoto R.M., *Multicomputer Networks: Message-Based Parallel Processing.* MIT Press, Cambridge, Massachusetts (1987).

Rice R.K., Mijnheer B.J. and Chin L.M., Benchmark measurements for lung dose corrections for x-ray beams. *Int. J. Rad. Onc. Biol. Phys.* (1988), **15** 399.

Rogers D.W.O, More realistic Monte Carlo calculations of photon detector response functions. *Nucl. Inst. and Meth.* (1982), **199** 531.

Rogers D.W.O., Low energy electron transport with EGS. *Nucl. Instr. and Meth.* (1984), **A227** 535

Rogers D.W.O. and Bielajew A.F., Experimental benchmarks of EGS. In *Monte Carlo Transport of Electrons and Photons, p~~249~~ 367,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Rogers D.W.O. and Bielajew A.F., A comparison of EGS and ETRAN. In *Monte Carlo Transport of Electrons and Photons, p323*, edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Rogers D.W.O., Bielajew A.F. and Nahum A.E., Ion chamber response and A_{wall} correction factors in a ^{60}Co beam by Monte Carlo simulation. *Phys. Med. Biol.* (1985), **30** 429.

Scheifler R.W., Gettys J. and Newman R., *X Window System C Library and Protocol Reference.* Digital Press, Massachusetts (1988).

Seltzer S.M., An overview of ETRAN Monte Carlo methods. In *Monte Carlo Transport of Electrons and Photons, p153,* edited by Jenkins T.M., Nelson W.R. and Rindi A., Plenum Publishing Corporation (1988).

Seltzer S.M., Hubbell J.H., and Berger M.J., Some theoretical aspects of electron and photon dosimetry. In *National and International Standardisation of Radiation Dosimetry, Vol. II, p3*, IAEA publication STI/PUB/471 (1978).

Sherouse G.W., *GRATIS Programmer's Manual.* University of North Carolina, Chapel Hill (1990).

Shore J.E., Second thoughts on parallel processing. *Comput. Elect. Eng.* (1973), **1** 95.

Siddon R.L., Solution to treatment planning problems using coordinate transformations. *Med. Phys.* (1981), **8** 766.

Siddon R.L., Fast calculation of the exact radiological path for a three-dimensional CT array. *Med. Phys.* (1985), **12** 252.

Silverman H.F., An introduction to programming the Winograd Fourier transform algorithm (WFTA). *IEEE Trans. Acoust., Speech, Signal Processing* (1977), **ASSP-25** 152.

Sontag M.R. and Cunningham J.R., Corrections to absorbed dose calculations for tissue inhomogeneities. *Med. Phys.* (1977), **4** 431.

Sontag M.R. and Cunningham J.R., The equivalent tissue-air ratio method for making absorbed dose calculations in a heterogenous medium. *Radiology* (1978), **129 (3)** 787.

Sternheimer R.M. and Peierls R.F., General expression for the density effect for the ionization loss of charged particles. *Phys. Rev. B* (1971), **3** 3681.

Straker E.A., Stevens P.N., Irving D.C., and Cain V.R., *MORSE-CG, General Purpose Monte-Carlo Multigroup Neutron and Gamma-ray Transport Code with Combinatorial Geometry*. Radiation Shielding Information Center, Oak Ridge National Laboratory Report Number CCC–203 (1976).

Sun Microsystems, *Network Programming Guide*. Sun Microsystems, Mountain View, California (1990).

Transtech, *TMB–08 Installation and User Guide*. Transtech Devices Limited, Buckinghamshire, U.K. (1988).

Turner J.E., Wright H.A., and Hamm R.N., Review article: A Monte Carlo primer for health physicists. *Health Phys.* (1985), **48** 717.

Ullman J.D., *Computational Aspects of VLSI*. Computer Science Press, Rockville, Maryland (1984).

Werner T, Herrman G., Schlegel W. and Lorenz W.J., Parallel processing in radiotherapy treatment planning. In *The Use of Computers in Radiation Therapy, p21*, edited by Bruinvis I.A.D. *et al.*, Elsevier Science Publishers (North Holland) (1987).

Widman J.C., Monte Carlo simulation in nuclear medicine. In *Monte Carlo Simulation in the Radiological Sciences, p193*, edited by Morin R.L., CRC Press (1988).

Williamson J.F., Monte Carlo evaluation of kerma at a point for photon transport problems. *Med. Phys.* (1987), **14** 567.

Williamson J.F., Monte Carlo simulation of photon transport phenomena: Sampling techniques. In *Monte Carlo Simulation in the Radiological Sciences, p53*, edited by Morin R.L., CRC Press (1988).

Wilson R.R., Monte Carlo study of shower production. *Phys. Rev.* (1952), **86** 261.

Winograd S., A new principle for fast Fourier transformation. *IEEE Trans. Acoust., Speech, Signal Process.* (1976), **ASSP–24** 264.

Wong J.W. and Henkelman R.M., A new approach to CT pixel-based photon dose calculations in heterogeneous media. *Med. Phys.* (1981), **10** 199.

Zhu Y. and Boyer A., X-ray dose computations in heterogeneous media using 3-dimensional FFT convolution. *Phys. Med. Biol.* (1990), **35** 351.