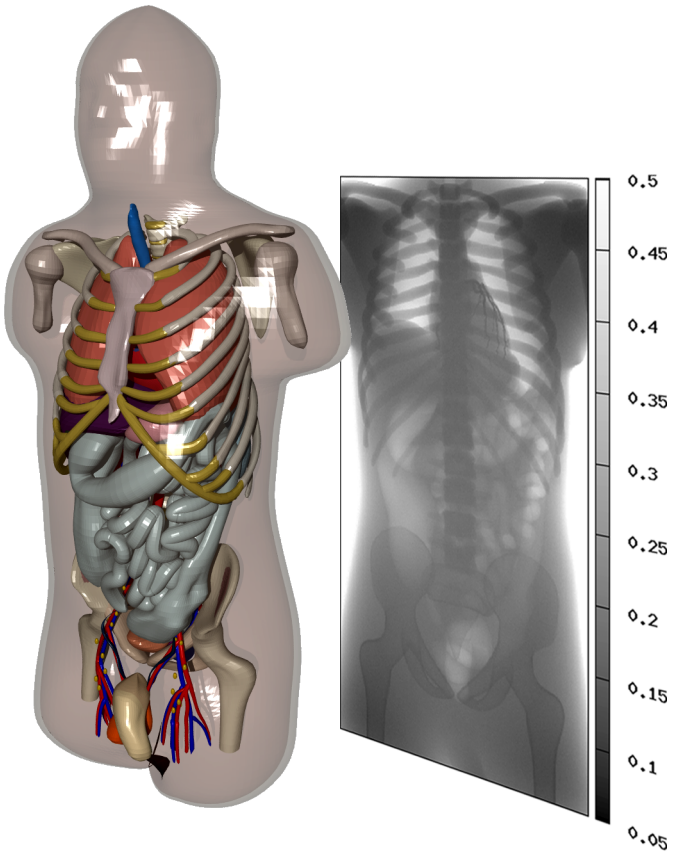


Development of advanced geometric models and acceleration techniques for Monte Carlo simulation in Medical Physics

Andreu Badal i Soler

DOCTORAL DISSERTATION
April 2008

Institut de Tècniques Energètiques
Universitat Politècnica de Catalunya

ACTA DE QUALIFICACIÓ DE LA TESI DOCTORAL

Reunit el tribunal integrat pels sota signants per jutjar la tesi doctoral:

Títol de la tesi: *Development of advanced geometric models and acceleration techniques for Monte Carlo simulation in Medical Physics*

Autor de la tesi: *Andreu Badal i Soler*

Acorda atorgar la qualificació de:

- ☐ No apte
- ☐ Aprovat
- ☐ Notable
- ☐ Excel·lent
- ☐ Excel·lent Cum Laude

Barcelona, …… de/d'…….. de .....….

El President El Secretari

....
(nom i cognoms) (nom i cognoms)

El vocal El vocal El vocal

....
(nom i cognoms) (nom i cognoms) (nom i cognoms)

A la meva família.

Acknowledgments

This thesis is the result of nearly four years of work at the *Institut de Tècniques Energètiques* and another year at the FDA's Center for Devices and Radiological Health. During these exciting years, I received the help and support from numerous people, without whom this dissertation would have never been completed.

First of all I would like to thank all of my family and friends, for their love and great encouragement. I also want to sincerely thank my supervisor, Josep Sempau, for guiding my research and always demanding the best from me. There is no doubt you and the other members of the PENELOPE group, especially José María Fernández Varea and Francesc Salvat, have teached me all I know about Monte Carlo simulation and scientific research—I am really indebted to you all.

I cannot forget the great help and inspiration from the friends I met in the United States. I want to thank especially my supervisors Iacovos Kyprianou and Aldo Badano, for their guidance and for introducing me to the field of medical imaging. I am proud of having had the opportunity to work with you in interesting medical problems, something I never expected while studing my Physics degree.

Finally, I gratefully acknowledge the financial support I have received from the Spanish *Fondo de Investigación Sanitaria* (project 03/0980), the *Ministerio de Educacion y Ciencia* (project FIS2006-07016) and the *Institut de Tècniques Energètiques*. My stay at the FDA was financed through an appointment at the Research Participation Program at the Center for Devices and Radiological Health administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U. S. Department of Energy and the Food and Drug Administration (FDA).

Contents

List of Figures

List of Tables

Part I

Introduction

1 Introduction

1.1 Overview and aim

Monte Carlo simulation of radiation transport is currently applied in a large variety of areas. This thesis is devoted to the development of advanced geometric models and acceleration techniques that facilitate the use of Monte Carlo simulation in medical physics applications involving detailed anatomical phantoms.

The geometric models implemented in most general-purpose codes impose limitations on the shape of the objects that can be defined. These models are not well suited to represent the free-form (i.e., arbitrary) shapes found in anatomic structures or complex medical devices. Therefore, some clinical applications that require the use of highly detailed phantoms can not be properly addressed. In addition, these simulations may require extremely long computations and this often makes the study of realistic settings unaffordable.

These problems can be tackled by having recourse to application-specific conceptual and geometric approximations. Nevertheless, these simplifications may compromise the accuracy of the simulations and reduce the applicability of the code. In this thesis these limitations were overcome by implementing advanced geometric models and computational tools that facilitate the description of complex objects typically encountered in medical applications. To this end, two new Monte Carlo codes, based on the PENE-

LOPE package (see section 2.3), have been developed. The first code, *penEasy*, implements a modular, general-purpose main program and provides various source models and tallies that can be readily used to simulate a wide spectrum of problems. Its associated geometry routines, *penVox*, extend the standard PENELOPE geometry, based on quadric surfaces, to allow the definition of voxelised phantoms. This kind of phantoms can be generated using the information provided by a computed tomography (CT) and, therefore, penVox is convenient for simulating problems that require the use of the anatomy of real patients (e.g., radiotherapy treatment planning). The second code, *penMesh*, utilises closed triangle meshes to define the boundary of each simulated object. This approach, which is frequently used in computer graphics and computer-aided design, makes it possible to represent arbitrary surfaces and it is suitable for simulations requiring a high anatomical detail (e.g., diagnostic imaging).

To mitigate the inconveniences associated to long execution times, the ray-tracing algorithms implemented in the new codes have been speeded up by using efficient computational techniques, such as an octree spatial data structure. Additionally, a set of software tools for the parallelisation of Monte Carlo simulations has been developed. These tools effectively reduce the simulation time by a factor that is roughly proportional to the number of processors available. It has been found that parallel computing is indispensable to perform complex simulations with detailed anatomical phantoms.

1.2 Thesis outline

This dissertation is organised in five parts. The first part provides an overview of the contents of the thesis and a brief introduction to the main topics to be addressed: medical applications of ionising radiation and computer simulation of radiation transport. This part also includes a description of PENELOPE, which is the basis of the codes developed in this work.

The second part contains the main body of the dissertation, that is, the description of the implemented geometric models and acceleration techniques. Chapter 3 describes penEasy and its voxelised geometry package, penVox, which allows the combination of quadric objects and voxels. Chapter 4 describes penMesh and its triangle mesh ray-tracing algorithm. Chapter 5 presents a study of the parallelisation of Monte Carlo simulations and introduces the script package clonEasy, its auxiliary code seedsMLCG, and a new version of PENELOPE's pseudo-random number generator (RANECU) with a much longer period.

The third part of the thesis provides several examples of the use of the previous tools in medical physics. These applications have been presented in scientific conferences or published in peer-reviewed journals during the course of the research reported in this thesis. Chapter 6 presents some simulations in the fields of radiotherapy and dosimetry that use objects described with quadric surfaces, voxels, or a combination of quadrics and voxels. Chapter 7 presents the study of two clinically-realistic diagnostic imaging cases that take advantage of a detailed anthropomorphic phantom defined with triangle meshes.

The fourth part of the dissertation summarises the main conclusions that arise from the investigations performed in this thesis and shows some future, and present, applications of the developed codes.

Finally, the appendices at the end of the dissertation, the fifth part, provide some auxiliary information. Appendix A contains a list of the publications and presentations at scientific conferences associated to this thesis. Appendix B presents an excerpt of the documentation of the two C++ classes implemented in penMesh.

2 Background

2.1 Medical applications of ionising radiation

Ionising radiations are routinely used in clinical facilities worldwide to diagnose and treat several diseases. It is well known that the exposure to radiations may be harmful, but their use is justified in those clinical situations where they can produce "more good than harm". This subjective value judgment has to take into account the available alternative treatments and social and economic factors (Lindell *et al.* 1998).

Medical applications of ionising radiation are divided in three main classes: diagnostic x-ray imaging, nuclear medicine and radiation therapy. Diagnostic x-ray imaging is the use of x-rays to visualise the patient internal anatomy for clinical purposes. An imaging modality of particular interest is computed tomography (CT), in which the patient body is reconstructed in 3D from a set of 2D projection images obtained at different angles around the patient. Nuclear medicine is a medical speciality characterised by the use of radioisotopes for treatment or imaging purposes. The most common nuclear medicine imaging techniques are single photon emission computed tomography (SPECT) and positron emission tomography (PET). These procedures use pharmaceuticals labelled with radionuclides to obtain 3D tomographic images that reflect biological processes that take place at the cellular level. Radiation therapy, finally, is the use of radia-

tion sources for treating diseases, mainly malignant tumours (cancers). There are two classes of radiation therapies: brachytherapy and teletherapy. The distinguishing characteristic of brachytherapy is that the sources are located close to, or inside, the treatment region. The radiation emitted by brachytherapy sources is typically composed of low energy photons or electrons (energy below a few hundred keV). Teletherapy, in turn, uses external radiation beams, such as high energy electron or photon beams produced by linear particle accelerators (energy between 6 an 20 MeV) or ^{60}Co beams (1.17 and 1.33 MeV photons).

The fast development of medical imaging technologies is significantly affecting the field of radiation oncology. The simultaneous acquisition of PET and CT scans, for instance, allows the obtention of images with functional information and high anatomical detail, which facilitates the detection, localisation and evaluation of some kinds of tumours. Another example is the advancement of intensity-modulated radiation therapy (IMRT). In IMRT, the planning tumour volume is irradiated from multiple directions using beams of varying intensity and shape, with the purpose of producing sharp dose gradients between the tumour and the healthy tissue. The use of accurate imaging modalities in the planning and delivery of IMRT treatments is of great importance to assure that the whole tumour is actually located inside the irradiated volume. An incorrect use of imaging in IMRT is potentially of greater importance than typical uncertainties produced in dose calibration (Li and Hendee 2007). For this reason, a significant research effort is currently devoted to the development of image-guided radiation therapy, a treatment modality that employs advanced imaging technologies to accurately define the treatment volume and reduce the delivery uncertainties (Xing *et al.* 2006).

2.2 Computer simulation of radiation transport

Computer simulations are a valuable tool to describe the transport of ionising radiation. In particular, they are useful to study a wide range of medical applications because they are inexpensive, safe and can provide some information that would be very difficult, or even impossible, to measure experimentally (e.g., perturbation factors in dosimetry, scatter-fractions, 3D fluence maps, etc.)

The propagation of radiation in matter is described by the Boltzmann transport equation (see, for example, the review by Zheng-Ming and Brahme 1993). This integro-differential equation can only be analytically solved in simple (e.g., semi-infinite) geometries. However, some numerical techniques can successfully reproduce the trans-

port process. There is general agreement that the Monte Carlo method (Kalos and Whitlock 1986) is the most accurate computational approach currently available. Actually, a detailed Monte Carlo simulation yields the exact solution to the transport equation for a given interaction model and within the statistical uncertainty inherent to the method. In contradistinction, other possible techniques, such as finite elements (Boman *et al.* 2005; Gifford *et al.* 2006) or semi-analytical models (Hogstrom *et al.* 1981; Ahnesjö 1989), provide only approximate solutions.

Some of the state-of-the-art general-purpose Monte Carlo codes that are currently used in medical physics are PENELOPE (described in section 2.3), EGS4 (Nelson *et al.* 1985), EGSnrc (Kawrakow and Rogers 2006), MCNP5 (Booth *et al.* 2003), MCNPX (Hendricks *et al.* 2007), FLUKA (Fasso *et al.* 2005) and Geant 4 (Agostinelli *et al.* 2003).

One of the reasons for the success of Monte Carlo algorithms lies in their conceptual simplicity and in the relative easiness with which they can be coded on a computer. These algorithms are based on the random sampling of a large number of independent particle tracks (or histories) and in the subsequent estimation of certain quantities of interest by averaging the contribution from each history. After a number N of histories has been completed, the expected value $\langle q \rangle$ of the quantity of interest q is estimated as

$$\overline{q} = \frac{1}{N} \sum_{i=1}^{N} q_i \,, \tag{2.1}$$

where q_i is the contribution from the i-th history. As a consequence of the stochastic nature of the algorithm, Monte Carlo results have an associated statistical uncertainty. For a sufficiently large number of histories, the central limit theorem applies and $\overline{q}$ follows a normal distribution with standard deviation σ. This implies that 95.4% of the estimated values will lie within a $\pm 2\sigma$ interval around $\langle q \rangle$. The variance, σ^2, is estimated by the expression

$$\sigma^2 \simeq \frac{1}{N} \left(\frac{\sum_{i=1}^{N} q_i^2}{N} - \overline{q}^2 \right) . \tag{2.2}$$

Therefore, σ is inversely proportional to the square root of the number of histories. This means, for example, that to reduce σ by a factor of two N has to be enlarged fourfold. For this reason Monte Carlo codes typically require the computation of a huge number of histories and, thus, long execution times. It is worth noting that, in addition to the statistical uncertainties, the accuracy of Monte Carlo results is also bound by the accuracy of the underlying cross sections (Andreo 1991).

In a detailed simulation, each particle track is described as a series of jumps between points in which the particle interacts with the medium. The distance s between two successive collisions is sampled according to

$$s = -\lambda \ln \xi \quad , \tag{2.3}$$

where ξ is a random number uniformly distributed between 0 and 1 (see section 5.1) and λ is the mean free path, which is inversely proportional to the total cross section. The kind of interaction, energy loss, and angular deflection are sampled using the probability distribution functions—the normalised differential cross sections—that characterise the different atomic interaction processes in the material. The trajectory between two consecutive interaction sites is assumed to be a straight line, an assumption that is valid, in the absence of external electromagnetic fields, for amorphous materials or when the particle wavelength is much smaller than the interatomic distances. The track ends when the particle's energy falls below a user-defined cutoff, called the absorption energy.

The detailed, i.e. collision-by-collision, simulation scheme described above is commonly used for photon transport due to their relatively large mean free path (14 cm for 1 MeV photons in water). For charged particles, however, λ is typically microscopic (0.3 μm for 1 MeV electrons in water) and the simulation of every individual collision may be unaffordable. This problem can be solved by having recourse to a condensed simulation scheme, that is, using a multiple scattering theory to describe the collective effect of the numerous interactions that take place in a macroscopic step (Berger 1963; Fernández-Varea *et al.* 1993; Kawrakow and Bielajew 1998; Salvat 1998). A shortcoming of this approach is that multiple scattering theories assume that particles move in an homogeneous medium and that the travelled distance is much larger than λ (i.e., there is actually multiple scattering). As a result, condensed simulation may produce artefacts in the vicinity of material interfaces and whenever it is applied to objects that are comparable in size to the mean free path.

Another limitation of condensed simulation is that it may not correctly account for the effect of "catastrophic" events, that is, individual interactions that significantly modify the particle energy or direction. This can be overcome by implementing a mixed simulation model in which condensed simulation is used to reproduce the contribution of interactions that change the particle energy or direction below certain user-defined thresholds (the so-called *soft* interactions) and detailed simulation is used for the remaining (*hard*) interactions (Berger 1963).

A convenient figure of merit to assess the performance of an Monte Carlo algorithm is given by the so-called simulation efficiency ϵ, defined as

$$\epsilon = \frac{1}{t\,\Delta^2}\,, \tag{2.4}$$

where t is the execution time and Δ is a measure of the uncertainty, for instance

$$\Delta = \frac{\sigma}{\overline{q}}\,. \tag{2.5}$$

An intrinsic efficiency can also be defined as

$$\epsilon_N = \frac{1}{N\,\Delta^2}\,. \tag{2.6}$$

Note that this quantity depends *only* on the algorithm, while

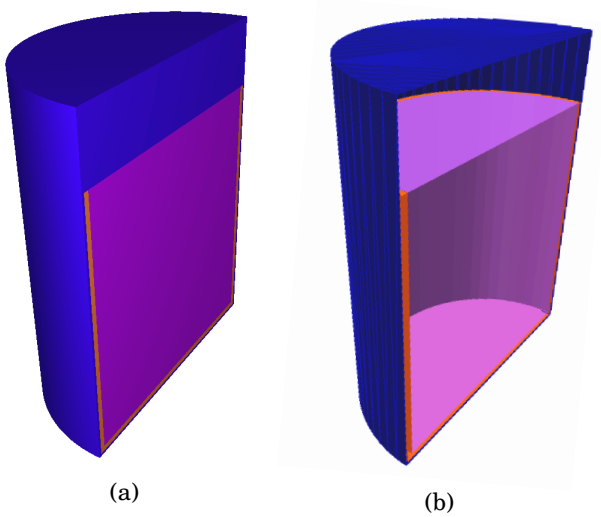
$$\epsilon = \epsilon_N \frac{N}{t}\,, \tag{2.7}$$

also depends on the simulation speed (N/t, histories per second), which varies for different computers and compilers.

Apart from the physical interactions with the medium, the simulation codes also have to handle the geometric aspect of particle transport. The algorithm that is used to transport particles across the simulated universe (the *geometry*) is called the *ray-tracer*. The ray-tracer determines the intersections between the particle path and the surfaces of the different objects that have been defined. The ray-tracer model determines how objects are defined and, thus, which kinds of shapes can be accurately modelled. It is also one of the most time-consuming parts of the code, employing more than 50% of the execution time in some applications.

The use of computer programs to create models of physical objects, commonly known as computer-aided design, is a well-established and rapidly evolving field due to its applications in industrial design and computer games, for instance. Two different approaches are generally used in computer-aided design to describe solid objects: constructive solid geometry and boundary representation. In constructive solid geometry, objects are created combining simple primitive objects with Boolean operations. This approach is employed by most Monte Carlo codes because it is robust and simple to implement. The primitives that are most commonly used are mathematical surfaces (e.g., quadrics) or simple objects (e.g., cubes, capped cylinders, etc.). A voxelised geometry is a particular case of constructive geometry in which the universe is described by the cells

(called volume elements or *voxels*) of a uniform 3D grid. In the boundary representation approach, solid objects are represented by its enclosing surface. The surface is typically represented using polygonal meshes or more sophisticated surfaces like Bezier patches or non-uniform rational B-splines (NURBS, Piegl and Tiller 1997). The boundary representation model is more flexible than the constructive solid geometry because it is not limited to combinations of primitive shapes and, therefore, it is able to represent any arbitrary object. An example of the use of the two geometry models, for the definition of a NaI detector, is shown in Fig. 2.1. Note that the first method defines volumes, while the second defines the surface that enclose the volumes. A review of advanced anthropomorphic phantoms for computer simulations can be found in Zaidi and Xu (2007) and in Xu *et al.* (2007).



(a) (b)

Figure 2.1: Definition of a NaI detector using two different geometry models: (a) constructive solid geometry model based on quadric surfaces; (b) boundary representation with triangles meshes. Only one half of the detector is represented to show the inner structure.

2.3 The Monte Carlo simulation package PENELOPE

PENELOPE (Salvat *et al.* 2006; Sempau *et al.* 1997; Baró *et al.* 1995), an acronym of PENetration and Energy LOss of Positrons and Electrons, is a general-purpose Monte Carlo simulation package that describes the transport of electrons, photons and positrons in any material and for the energy range from 50 eV to 1 GeV. PENELOPE, which is coded in FORTRAN 77, is free and open software. It is actively developed at the *Universitat de Barcelona* and distributed by the Nuclear Energy Agency (`http://www.nea.fr`).

PENELOPE uses detailed simulation for photon transport and a mixed scheme for electrons and positrons. Contrary to most other Monte Carlo codes, mixed simulation is consistently employed for all interaction mechanisms, that is, elastic, inelastic, and radiative collisions. The transport algorithm is controlled by a set of user-defined parameters: the cutoff energies for hard inelastic (`WCC`) and hard bremsstrahlung (`WCR`) events, the maximum allowed step length (`dsmax`), and `C1` and `C2`, which limit the average angular deflection and the maximum average energy loss between consecutive hard elastic events, respectively. If these parameters are all set to zero, PENELOPE effectively performs a detailed simulation of charged particle transport. An exception is made for bremsstrahlung events. Due to the fact that the differential cross section diverges for zero energy-losses, the minimum cuttoff value for radiative events is set to 10 eV. A strict collision-by-collision simulation can still be performed by entering a negative WCR value. In this case, the emission of bremsstrahlung photons with energy less than 10 eV is disregarded and all the remaining events are simulated in a detailed way.

PENELOPE has been extensively benchmarked with experiments and with other Monte Carlo codes in the past (Sempau *et al.* 2003; Ye *et al.* 2004; Cot *et al.* 2004; Vilches *et al.* 2007; Panettieri *et al.* 2007; Kryeziu *et al.* 2007; Fernández-Varea *et al.* 2007; Panettieri *et al.* 2007).

The PENELOPE package is the foundation of the computer codes introduced in chapters 3 and 4. It was employed in this thesis due to its wide dynamic energy range and outstanding physics models, especially for low energy radiation transport. These features allow, theoretically, the simulation of any electron or photon beam currently used in clinical facilities, from kilovoltage x-ray sources for diagnostic imaging to megavoltage linear accelerator beams for radiotherapy. Another advantage of PENELOPE compared to other general-purpose codes is the flexibility and simplicity of its simulation algorithm. In particular, the fact that the physics and geometric operations are clearly

separated was essential to implement new geometric models without compromising the accuracy of the physics part.

In order to use PENELOPE's subroutines in a specific application it is necessary to prepare a steering main program that defines the initial state of the particles (the radiation source), tallies the relevant quantities of interest, and reports the final results. To guide users in the development of their custom main program and to facilitate the simulation of common cases, some sample codes are provided with the standard PENELOPE distribution (for more details see Salvat *et al.* 2006, chapter 6):

- `penslab`: Main program to simulate the particle transport in an homogeneous and infinite material slab. A monoenergetic electron, photon or positron point source can be defined. The parameters tallied during the simulation include the angular and energy distribution of the emerging and backscattered particles, and the depth-distribution of deposited energy and charge.
- `pencyl`: Code to simulate in a multilayered cylindrical structure. The material, thicknesses and radii of the different cylindrical objects are defined in the input file. The code tallies the same parameters of `penslab`, as well as 2D dose distributions.
- `penmain`: Generic main program to simulate complex experiments. This general-purpose code includes a flexible source and several tallies. It uses the quadric geometry package PENGEOM, which is described below.

Each sample program provides a number of features that can be controlled by the user through an input file. Therefore, these codes can be readily used in many applications without any modification of the original source code. Equivalently, a compiled version of the code can be used in different studies without re-compiling. The input files describe the simulated universe, the radiation source, the quantities to be tallied, and some parameters of the Monte Carlo algorithm, such as the amount of particles that have to be simulated, the maximum execution time, the absorption energies, and the cutoff values for the mixed transport algorithm for electrons and positrons.

The options available in the sample codes do not cover all the possible applications of PENELOPE and advanced users may have to create a new main program or extend an existing one in order to get the maximum value from the simulations. It is also possible to use alternative steering programs that are not distributed with the standard distribution and that provide different features. An example of this is `penEasy`, a main program for PENELOPE that has been developed over the past five years at the *Institut*

de Tècniques Energètiques (*Universitat Politècnica de Catalunya*, Barcelona, Spain). PenEasy is a modular general-purpose main program that was created with the aim of simplifying the use of PENELOPE and, in particular, to extend its use in medical physics. This code is described in detail in chapter 3.1.

The standard geometry subroutines used in PENELOPE simulations are contained in the PENGEOM package. This package is distributed with PENELOPE and is used by the sample main program `penmain`. PENGEOM implements a constructive solid geometry model in which objects are defined by the volume enclosed between a set of quadric surfaces. The PENGEOM geometry is coded in a text file using an intuitive syntax described in the PENELOPE manual (Salvat *et al.* 2006, chapter 5). For each body included in the simulation universe the geometry file contains the following information: the equation of the quadric surfaces that limit the object, their side pointers (i.e., whether the object is in the positive or negative side of the quadric), and a list of other objects located inside it (if any). Quadrics can be defined either by giving its implicit equation, or by describing the geometrical transformations that have to be applied to one of the standard surfaces shown in Fig. 2.2, the so-called "reduced" quadrics. Figure 2.3 presents an example geometry file describing an object as the intersection of the volumes limited by a cone (defined in reduced form) and a pair of parallel planes (in implicit form).

The PENELOPE package provides two software tools, `gview2d` and `gview3d`, to visualise the quadric geometry in two and three dimensions, respectively. These programs employ the PENGEOM routines and, therefore, can be used to test that the objects have been correctly defined. Figure 2.4 presents two sample visualisations of the simple object defined in Fig. 2.3.

The quadric geometry model implemented in PENGEOM is simple and robust but it introduces a limitation on the kind of shapes that can be accurately represented. In particular, this model is not capable of representing free-form surfaces and, therefore, it can not accurately describe arbitrary anatomic structures. Idealised models of the human anatomy based on quadric surfaces have been developed in the past (Snyder *et al.* 1969). These models have been widely used in some fields, such as dosimetry and radiation protection, but they are not appropriate to simulate applications in which the anatomic detail is an essential factor, for example diagnostic imaging.

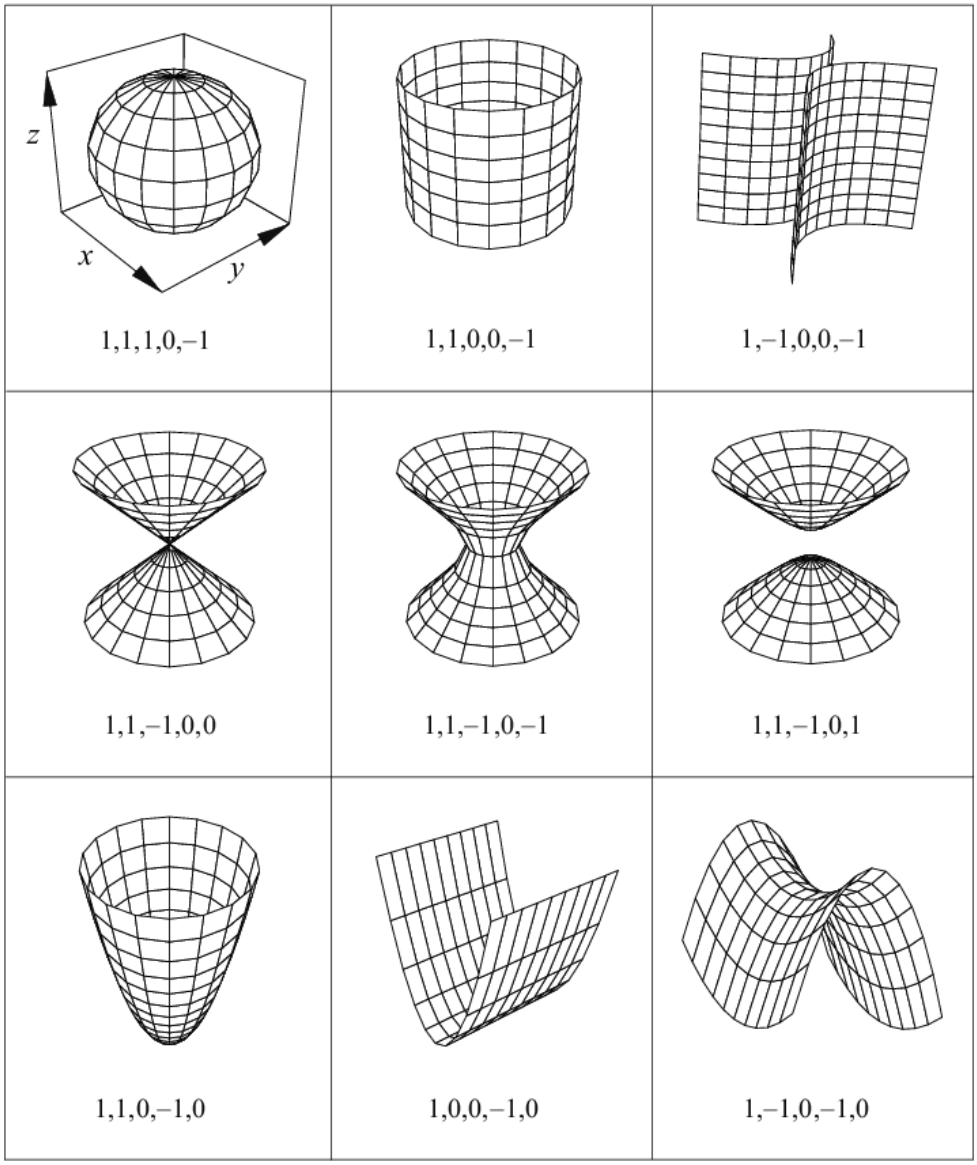
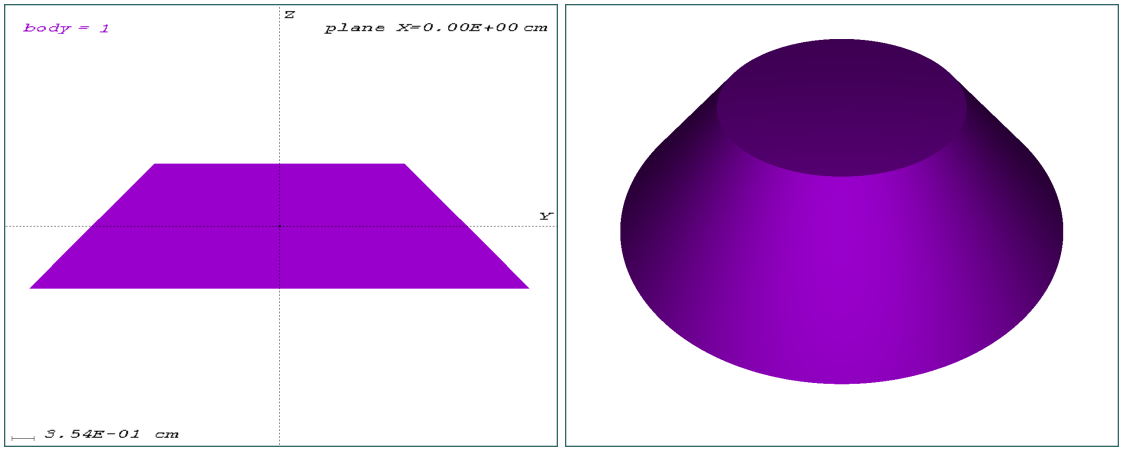


Figure 2.2: Graphical representation of non-planar quadric surfaces in reduced form (image taken from the PENELOPE manual; Salvat *et al.* 2006, page 174).

```
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   1)   Cone (in reduced form)
INDICES=( 1, 1,-1, 0, 0)
Z-SHIFT=(+3.000000000000000E+00,   0)              (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   2)   Pair of planes Z^2-1=0  (in implicit form)
INDICES=( 0, 0, 0, 0, 0)
    AZZ=(+1.000000000000000E+00,   0)              (DEFAULT=0.0)
     A0=(-1.000000000000000E+00,   0)              (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000000000
BODY    (   1)   Volume inside the pair of planes and the cone
MATERIAL(   1)
SURFACE (   1), SIDE POINTER=(-1)
SURFACE (   2), SIDE POINTER=(-1)
0000000000000000000000000000000000000000000000000000000000000000000000
END      0000000000000000000000000000000000000000000000000000000000000000
```

Figure 2.3: Sample PENGEOM quadric geometry file describing an object as the volume contained between a cone and a pair of parallel planes.



(a) (b)

Figure 2.4: Visualisation of a simple quadric object (a truncated cone) using the two viewers distributed with PENELOPE: gview2d (a), and gview3d (b).

Part II

New simulation tools and methods

3 Implementation of a voxelised geometry in PENELOPE

The simulation of radiation transport involves physical and geometrical operations. The first group includes the computation of distances between interactions and the sampling of particle states after those interactions. In the second group the basic tasks are the calculation of interface crossings and particle displacements. As it was explained in section 2.3, the general-purpose Monte Carlo simulation package PENELOPE handles the geometrical part with the help of PENGEOM, a subroutine package that defines homogeneous bodies as combinations of volumes limited by quadric surfaces. Although this geometric model is general enough for a large number of applications, it is not well suited in medical physics problems that require the simulation of realistic anatomical structures. In these cases the geometry is usually represented in terms of an uniform grid of parallelepipedic volume elements (voxels) with varying material composition and mass density. The main advantage of voxelised models is that the patient's anatomy can be obtained by segmenting computerised tomography (CT) scans.

In this chapter, two new Monte Carlo tools are introduced. In the following section we present penEasy, a versatile main program for PENELOPE. In section 3.2, we describe a set of geometry subroutines integrated in penEasy, named penVox, which allow the

simulation of objects formed by combinations of quadric surfaces and voxels, or by voxels alone. Two simulations with simple geometries that can be represented with either voxels or quadrics are presented, with the purpose of validating that the new geometry routines do not affect the results. Some details on the approximations underlying the transport of radiation in voxelised geometries are addressed in section 3.2.1.

The penEasy system has already been successfully used in a wide range of applications. In the field of medical physics it has been applied, for instance, to obtain the response of MOSFET dosimeters (see section 6.1), to characterise ionisation chambers (Kryeziu *et al.* 2007; Panettieri *et al.* 2008), to compute stopping power ratios (Fernández-Varea *et al.* 2007), and to simulate radiotherapy treatments (Panettieri *et al.* 2007). It has also been the starting point for the development of two new Monte Carlo codes for medical imaging applications, namely, MANTIS (Badano and Sempau 2006), which offers the possibility of transporting optical photons in order to study indirect digital detectors (see, e.g., Badano *et al.* 2006); and penMesh—described in chapter 4—which uses a geometric model based on triangle meshes that has been employed to generate realistic x-ray images of a detailed human phantom (Badal *et al.* 2007; Badal *et al.* 2008). Some example applications of penEasy, with and without penVox, are presented in more detail in chapter 6, including a benchmark with other general-purpose Monte Carlo codes and with experimental data (see section 6.3).

3.1 PenEasy

PenEasy is a modular, general-purpose main program for PENELOPE that includes various source models and tallies. The rationale is to provide a tool suitable for a wide spectrum of problems so that users do not need to develop a specific code for each new application, a process that is usually error-prone and time consuming. For those cases where these models are insufficient and some additional coding or adaptation needs to be done, its modular structure is designed to reduce the programming effort to a minimum.

PenEasy is free and open software, and can be readily downloaded from the website `http://www.upc.es/inte/downloads/penEasy.htm`. The source code is mostly written in FORTRAN 77, although it has recourse to some extensions included in Fortran 95. A version of the `main()` function in C++ is also available from the same website mentioned above. An auxiliary header file handles the different calling conventions used by C++ and Fortran within the GNU compiler collection (`http://gcc.gnu.org`),

allowing the use of PENELOPE's subroutines and common blocks from C++ source code.

The skeleton of the main program is displayed in Fig. 3.1. This structure follows, with slight modifications, the flow diagram included in the PENELOPE manual (Salvat *et al.* 2006, page 214). Uppercase names denote routines from the PENELOPE (and PENGEOM) kernel, whereas lowercase subroutines are provided with the penEasy package. The code is structured in an initialisation phase, followed by three nested loops and the reporting of results. Each cycle of the loop named "history" performs the simulation of one primary particle and all its descendants, that is, of an electromagnetic shower. Each cycle of "particle" simulates a single photon, electron or positron. Each cycle of "interact" reproduces a single interaction, or the crossing of an interface if the distance to the intersection is shorter than the distance to the next interaction. Thus the algorithm essentially consists of repeating the sequence of calls to subroutines `JUMP`, `STEP` and `KNOCK`: `JUMP` computes the distance to the next interaction event, returned through the variable `ds`; `STEP` determines if an interface is crossed before completing the step `ds` and displaces the particle; and `KNOCK` simulates the effect of the interaction and returns the energy lost by the particle (variable `de`). If an interface is crossed (`ncross` not zero) the trajectory is truncated at the boundary, no interaction takes place and a new "interact" cycle begins. The simulation of a particle ends when it leaves the material system (`mat` is zero), or when its kinetic energy (`e`) falls below some user-defined absorption energy (`eabs`), which may depend on the material and particle type.

The generation of initial particle states is performed by the subroutine `source`, which defines the position, direction, energy, statistical weight and particle type. A call to `tally` (not shown in Fig. 3.1 for brevity) is performed every time there is a change in the state of the particle to allow the scoring of the quantities of interest. In fact, the solely task of `tally` (or `source`) is to call in succession all the individual tally (source) routines, which are described in section 3.1.2 (3.1.1). Tallies or sources that are inactive return the control to the calling procedure without further actions. Detailed documentation for each source and tally is included in the distribution package.

The communication between the main program and the PENELOPE kernel—in order to, for example, allow a source to set the initial state—is done via a common block that holds the dynamical state of the particle being simulated at any time. Some tallies also take advantage of this method.

The simulation can be stopped by any of the following events: (i) a predetermined number of histories has been completed; (ii) the allotted (either real or CPU) time has been exhausted; (iii) a set of predetermined statistical uncertainties (one for each tally) has

```
call init; n=0                        ! Initialisation
history: do                           ! One history
  n = n+1                             ! Update history counter
  call CLEANS                         ! Clear stack
  call source                         ! Create primary particle
  particle: do                        ! One particle
    call SECPAR(left)                 ! Retrieve particle from stack
    if (left.eq.0) exit particle      ! Stack was empty
    call START                        ! Reset transport mechanics
    interact: do                      ! One interaction/intersection
      call JUMP(dsmax(mat),ds)        ! Distance to interaction
      call STEP(ds,dsef,ncross)       ! Geometry check
      if (ncross.ne.0) then           ! Interface crossed
        if (mat.eq.0) exit interact   ! Particle is gone
        call START                    ! Reset transport mechanics
        cycle interact                ! Start over
      endif
      call KNOCK(de,icol)             ! Simulate an interaction
      if (e.lt.eabs(kpar,mat)) exit interact   ! Absorbed
    enddo interact
  enddo particle
  if (endsim(n)) exit history         ! Check end-of-simulation
enddo history
call report
```

Figure 3.1: Basic structure of the Fortran version of the main program of penEasy. Calls to tallying and other auxiliary subroutines have been removed for clarity. See text for further details.

been reached; or (iv) the user sends a "stop" command through an external file that is read at regular time intervals. After completion, a report is written for each tally. These reports can also be obtained at regular intervals during the execution of the program, to visualise its progress.

In order to reduce the computation time required for some ill-conditioned problems, a variance reduction technique known as interaction forcing can be applied. Our scheme is based on the approach adopted in the variance reduction subroutines provided with PENELOPE, which create "virtual" interactions along a particle trajectory with an effective (reduced) mean free path. To keep the results unbiased, the appropriate statistical weight is assigned to any secondary particle or energy loss involved in these events. Weight windows can be defined to control the range of values of the statistical weight for which this method is to be applied.

Another useful characteristic is the possibility of initialising PENELOPE's pseudo-random number generator with integer numbers (the "seeds") read from an external file. This feature can be used in conjunction with the package of Linux scripts clonEasy

(see chapter 5) to parallelise the execution in a straightforward way, without altering the source code. The seeds required for the parallel execution can be obtained with the code seedsMLCG, described in section 5.2.2.

3.1.1 Source models

PenEasy includes two configurable source models. Their configuration parameters are set by the user through a global input file. Only one source can be active in any given simulation.

The source called BIGS (an acronym for Box Isotropic Gauss Spectrum) allows the definition of volumetric sources limited by quadric surfaces. The type of particle can be either a photon, electron or positron with an arbitrary energy spectrum of emission or, alternatively, with the spectrum defined by means of a Gaussian function. The angular distribution has a constant probability per unit solid angle and can be limited by a cone with arbitrary orientation in space. This includes, as limiting cases, point sources, 4π isotropic sources and pencil beams. A point source can also be configured to "illuminate" a distant field of arbitrary (quadric) shape, which can be useful, e.g., to define an electron beam from an accelerator head that produces a square field on the patient surface.

The source called PSF (Phase-Space File) reads the initial state of the particles to simulate from an external file, hence the name. Usually, this file is created by penEasy in a previous run by means of the PSF tally (see below), for example to reproduce the radiation field produced by a linac before entering the patient's body. The PSF contains information to identify all particles, either primary or secondary, belonging to the same history (i.e., descendants of the same primary particle). This feature plays an important role in the computation of the statistical uncertainty (Sempau *et al.* 2001).

The particles read from the PSF can be rotated, for instance to adapt it to the gantry angle of a clinical linear accelerator, and also translated. Particle splitting, a variance reduction technique intended to reduce the statistical uncertainty by creating K identical copies, with statistical weight $1/K$, of each initial particle is also implemented. This method—also called particle recycling by some authors, see e.g. Walters *et al.* (2002)—should not be confused with restarting the PSF, which implies that the PSF itself is recycled as a whole, that is, it is reread from the beginning after the last particle has been used up. The second method should be avoided because it does not account for correlations between between identical copies of the particles, thus producing an incorrect estimation of the statistical uncertainty.

3.1.2 Tallies

PenEasy includes subroutines to score the most common quantities of interest. The tallies available in the current version are listed below.

- Spatial (3D) distributions of absorbed dose in homogeneous regions. These can be scored in parallelepipedic bins, cylindrical shells or spherical shells. In the case of parallelepipeds, the distribution can be integrated along the x, y and/or z directions to obtain 2D or 1D results. The dose is estimated using the collision estimator, i.e., scoring the energy deposited on the spot in each interaction.

- When voxelised geometries are used, the absorbed dose distribution can also be scored. To speed up the tallying process, the scoring and voxels grids are the same.

- Fluence, differential in energy, in the detection material. This tally is based on the relation, pointed out by Chilton (1978), between the track length of particles visiting a certain volume and the average fluence in it. The total energy deposited in the detector is also reported. The latter is estimated by combining track lengths of charged particles, restricted stopping powers and "residual" deposition events—such as the so-called track-ends (Nahum 1978).

- Pulse height spectra, that is, the distribution of energy deposition events (per completed history) in a certain detection material. The total energy deposited in the detector is reported as well.

- Energy spectra and total number of particles of each type (photon, electron or positron) entering a certain material.

- Particle trajectories. This tally allows the representation of the particle tracks in a 3D graph. To this end, particle coordinates after each interaction are written to an external file.

- Phase-space file. Generates a file with the phase space (type of particle, energy, position, direction, etc.) of all particles that reach the detection material. The PSF is stored on disk in ASCII format (that is, plain text).

Note that tallying the dose distribution in voxels is similar to using the parallelepipedic bins. However, since penVox allows voxels and quadrics to be combined in the same simulation (see section 3.2), some voxels may be partly overlapped by a quadric body. Hence, the mass of each voxel needs to be determined by integrating the mass density

over its volume. This process is performed for voxels during the initialisation stage, but not for parallelepipedic bins. For the latter, the mass density is simply taken from the center of the bin. As a result, dose distributions produced by the tallies in the first item of the list may be inaccurate inside the voxelised region or, in general, for non-homogeneous bins.

All the reported quantities are normalised per unit simulated history. A set of gnuplot[1] script files is provided to automatically create plots of the data generated by each tally. It is also important to remark that all the tally routines are called using a common interface, which facilitates the development of new ones by end users.

3.2 PenVox

Realistic models of the human body can be obtained using anatomic data obtained from a CT scan. The CT data can be processed to estimate the chemical composition and mass density of each voxel (Schneider *et al.* 2000; Reynaert *et al.* 2007), information that is required by most radiation transport simulation systems. Other imaging modalities—most notably magnetic resonance imaging (MRI)—can also be used to create detailed anatomic phantoms.

In this section a new tool, named penVox, that allows the use of voxels in a PENELOPE simulation is introduced. PenVox is fully integrated in penEasy, and its operation is controlled through sections of the same global input file used for the latter. Other authors have developed subroutine packages for PENELOPE that are capable of handling voxelised geometries—see for example Moskvin and Papiez (2005), Taranenko and Zankl (2005) and García *et al.* (2007)—but penVox is, to the best of our knowledge, the only one that allows the superposition of quadric objects and voxelised regions and that works in conjunction with a general-purpose main program. Other Monte Carlo codes have been adapted to voxelised geometries in the past, e.g., Geant 4, FLUKA, or DOSXYZnrc (Walters *et al.* 2007) for EGSnrc.

The quadric geometry, consisting of combinations of bodies delimited by quadric (that is, second order) surfaces—planes, spheres, cones, cylinders, etc—is handled by invoking subroutines from PENELOPE's standard geometry package PENGEOM. The tracking across voxels, in turn, is performed using an algorithm inspired on the DPM code (Sempau *et al.* 2000). To produce the superposition of both elements users must provide

[1] Gnuplot is an open source, freely available plotting program that can be downloaded from http://www.gnuplot.info.

two geometry input files. The first one is a standard PENGEOM file. For details on its format and on the associated visualisation tools the reader may consult the manual (see also Fig. 2.3). The other file, which we shall call the voxels file, defines the number of voxels to consider, their dimensions and a list of material and mass densities, one pair for each voxel. The voxels boundary box is assumed to be located in the first octant ($\{x, y, z\} > 0$), with its vertex at the origin of coordinates. An example of voxels file is given in Fig. 3.2.

```
HEADER section:
  <Comment line, use it to describe the simulation>
Nx,Ny,Nz (No. of voxels along each Cartesian axis):
  128    128    128
dx,dy,dz (Voxel dimensions in each direction, in cm):
  0.3    0.3    0.3
Material# : Mass_density_(g/cm^3):
1 1.
1 1.
2 2.699
... (etc)
```

Figure 3.2: First lines of a voxels file for penVox, showing the header section and the beginning of the two-column list with the material number and density for every voxel.

In order to embed the voxelised region into the quadric geometry users must tag one of the quadric bodies as transparent. This will cause penVox to treat it as if it were a "hole" in the quadric geometry, thus being able to "see" the voxels underneath. An example of a non-trivial geometry defined in this way is represented in Fig. 3.3. For cases that do not require the explicit definition of quadric objects, the preparation of a PENGEOM file may be omitted. A suitable parallelepipedic bounding box enclosing all voxels is then automatically defined (using quadrics) by penVox without the users' intervention.

During the initialisation of the penVox system voxels are classified in three groups: (i) voxels that are fully visible; (ii) those that are partly overlapped by a quadric body; and (iii) those that are invisible, that is, that are completely covered by quadrics and, therefore, effectively nonexistent from the viewpoint of the simulation. A particle that is moving in a partly overlapped voxel can cross a boundary and enter a quadric body in a

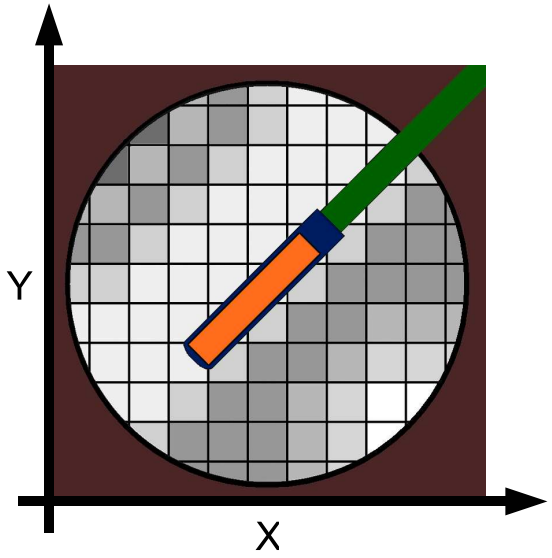


Figure 3.3: A notional example of a combination of voxels (a portion of a fictitious CT scan) and quadrics (a brachytherapy source). Notice that the voxels region is also bound with a quadric surface.

single step and, hence, every step it takes requires that both geometries be interrogated to find the closest intersection.

The mass in each voxel (either covered, partly overlapped or fully visible) is also determined during the initialisation by integrating the density using the ray tracing capabilities of PENGEOM. The average absorbed dose in each voxel is computed as the ratio between the deposited energy and its mass. Note that the dose map is computed for all voxels, including those that are totally covered by quadric bodies.

Voxels files can be obtained by processing CT scans; they can also be created by converting a quadric geometry into a grid of volume elements with the help of genVox, a stand-alone program included in the penVox package. GenVox is intended to produce voxelised geometries that describe simple objects such as homogeneous or multilayer phantoms. This functionality was used to simulate dose distributions for several layered phantoms irradiated with photon and electron beams of various energies with both PENGEOM (using the quadric version of the geometry) and penVox (using the voxelised version). In all the studied cases the differences were negligible, i.e., compatible with the statistical uncertainties obtained, thus validating the new tracking algorithm.

Two example benchmarks performed with PENGEOM and penVox are presented. The source used in these studies was a 1.5×1.5 cm^2, 20 MeV electron beam. Figure 3.4

shows the comparison for an inhomogeneous geometry composed of an aluminum cylinder (1 cm in diameter and 1 cm in length) in water (Bielajew 1993). The reported dose profile was tallied 1 mm downstream the cylinder, in a line of voxels perpendicular to the beam central axis. In the penVox simulation the water was described with voxels and the cylinder with quadrics. The second example, shown in Fig. 3.5, was proposed in the ICCR 2000 workshop by Rogers and Mohan (2000). The geometry consisted in layers of water, aluminum, lung, and water. The central axis depth-dose and the relative difference with respect to penEasy's maximum dose are represented.

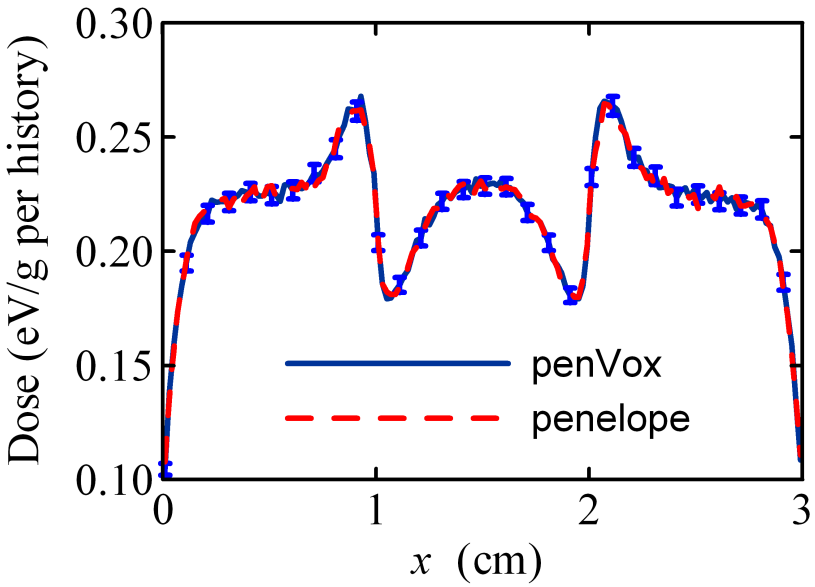


Figure 3.4: Comparison of PENGEOM and penVox for the simulation of an aluminum cylinder in water. See text for details.

3.2.1 The density effect

PENELOPE requires the definition of a specific material for each different density, even when the chemical composition is the same. This poses a problem because, in general, a CT scan contains millions of voxels with varying mass densities. Since the initialisation time and the memory allocated increase linearly with the number of materials, the definition of a large number of them is impractical.

A simple method that overcomes this difficulty is based on the fact that the cross section

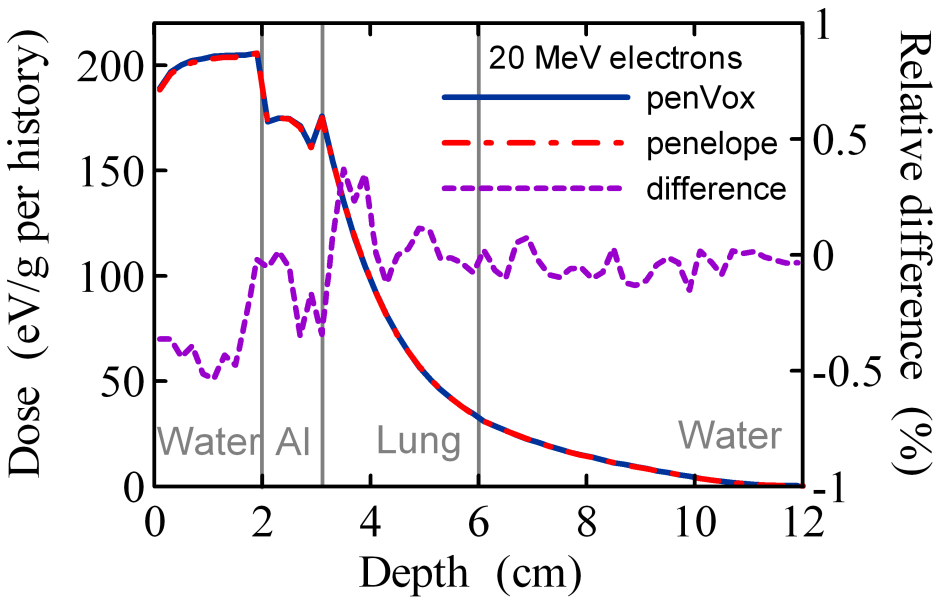


Figure 3.5: Central-axis depth dose calculated with PENGEOM and penVox for a layered geometry. The difference, relative to penEasy's maximum dose, is presented in the scale on the right. The statistical uncertainty in the simulations was below 1% at 2σ.

models for the relevant interactions are, with the exception of inelastic collisions of charged particles, independent of the local mass density ρ of the medium. The distance s between two consecutive interactions is a random variable that follows an exponential distribution. In a medium with total cross section σ, s can be sampled according to the expression

$$s = -\lambda \ln \xi \tag{3.1}$$

where ξ is a random number uniformly distributed between 0 and 1 (see section 5.1 for a description of random number generators) and

$$\lambda = \frac{A}{N_{\mathrm{A}} \sigma \rho} \tag{3.2}$$

is the mean free path. The symbols N_{A} and A represent Avogadro's number and the molecular mass in atomic units, respectively. Eqs. (3.1) and (3.2) show that, everything else being equal, s in inversely proportional to the density and, hence, spatial density variations can be readily taken into account by scaling the distance to travel as ρ_0/ρ, with ρ_0 the material nominal density.

Note that, for charged particles, the mass collision stopping power S_{c}/ρ can be written

in terms of $\sigma_{\mathrm{c}}(W)$, the inelastic cross section differential in the energy loss W, as

$$\frac{S_{\mathrm{c}}}{\rho} = \frac{N_{\mathrm{A}}}{A} \int \mathrm{d}W \; W\sigma_{\mathrm{c}}(W). \tag{3.3}$$

The polarisation of the medium caused by the incoming charged projectile produces a partial screening of its electric field and, as a result, a decrease of the inelastic cross section, a phenomenon known as the *density effect* (Sternheimer 1952; Fano 1963). The density effect involves the introduction of a term in the differential cross section that does depend on the mass density, a dependency that, by virtue of (3.3), propagates to the collision stopping power.

Fortunately for our purposes, the variation of S_{c}/ρ with ρ is relatively small even for fairly large density changes. This is illustrated in Fig. 3.6 for electrons in water, where it can be seen that S_{c}/ρ deviates by less than 1% with respect to its nominal value when the density changes by as much as 20%. The conclusion is that, for most practical purposes, the transport can be carried out assuming that the cross sections for the material with nominal density apply. This is in fact the approach taken in most Monte Carlo codes that use a voxelised geometry (Jiang and Paganetti 2004; Reynaert *et al.* 2007).

It is important to point out that, in order to encapsulate the algorithm that moves particles inside voxels and to keep the penEasy main program simple, the scaling of flight distances with $1/\rho$ is performed inside the tracking routine of penVox. A limitation of this operating procedure is that the fluence tally cannot be employed inside the voxel region because the main program is oblivious to the actual path lengths travelled in it.

The relatively little importance of the density effect correction on the absorbed dose can be observed in the depth-dose profiles shown in Fig. 3.7, for a 20 MeV electron beam. This figure compares depth-dose profiles obtained with PENGEOM and penVox for water with densities 0.8 g/cm^3 and 1.2 g/cm^3. For the PENGEOM profiles, fictitious materials defined with the PENELOPE database for water with varying density values were employed (i.e., the density effect correction was calculated for the actual material density). For the penVox case, the phantom was made of water with its nominal density (i.e., with the density effect correction for 1.0 g/cm^3 water) and the particle tracks were re-scaled according to the voxel density. In this situation the depth-dose profile, in units of MeV cm^2/g, does not vary for different voxel densities (the profile corresponding to voxels with density 0.8 g/cm^3 is represented). It can be observed that the density effect just slightly modifies the dose deposition, even for this extreme case in which the material is 20% heavier (or lighter) than the nominal water.

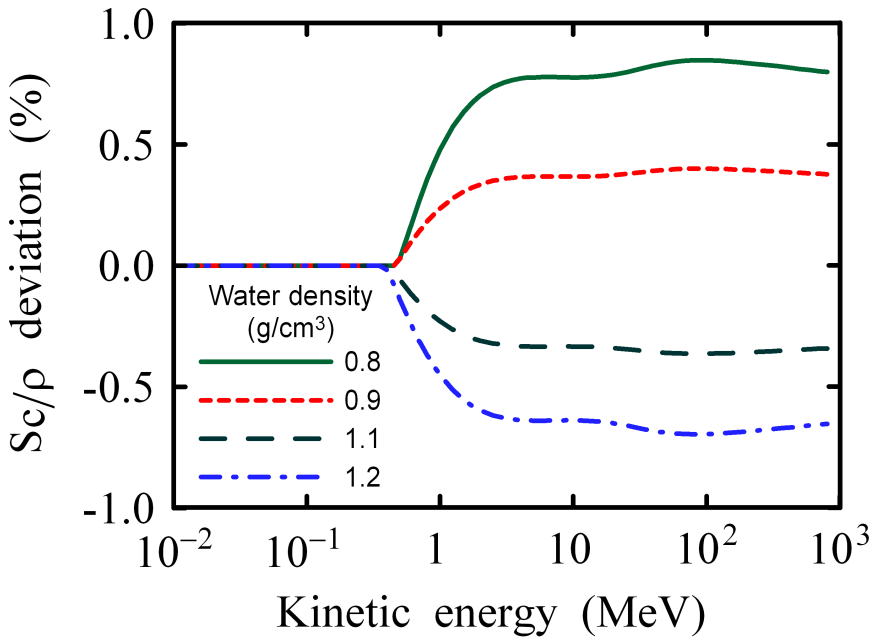


Figure 3.6: Variation, in percentage, of the electron collision mass stopping power (S_c/ρ) in water of various densities, with respect to its nominal value (i.e., for ρ=1). Stopping powers were obtained with the PENELOPE model and assuming a constant (independent of ρ) mean ionisation energy.

The remaining question of how variations of the density effect could be taken into account in a simple manner is still of some interest. A possible, albeit only partially correct, scheme is outlined below. Note that the method that is to be described has not been implemented in penVox due to its limited practical impact.

The density effect correction δ_F on the stopping power for electrons with velocity β (in units of the speed of light) and for a material with atomic number Z can be computed as (Fano 1963)

$$\delta_F = \frac{1}{Z}\int_0^\infty \mathrm{d}W \, \frac{\mathrm{d}f(W)}{\mathrm{d}W} \ln\left(\frac{W^2+L^2}{W^2}\right) - L^2\frac{(1-\beta^2)}{\Omega^2}. \tag{3.4}$$

In this expression $\mathrm{d}f(W)/\mathrm{d}W$ is the so-called optical oscillator strength (OOS), which characterises the response of the medium in distant (i.e. with low momentum transfer) inelastic collisions. The OOS, which depends only on the structure of the atomic target and on the energy W lost by the projectile, is closely related to the photoelectric cross section for photons of energy W. The symbol Ω represents the plasma energy of an equivalent free-electron gas and, for a given material, Ω^2 is proportional to the mass

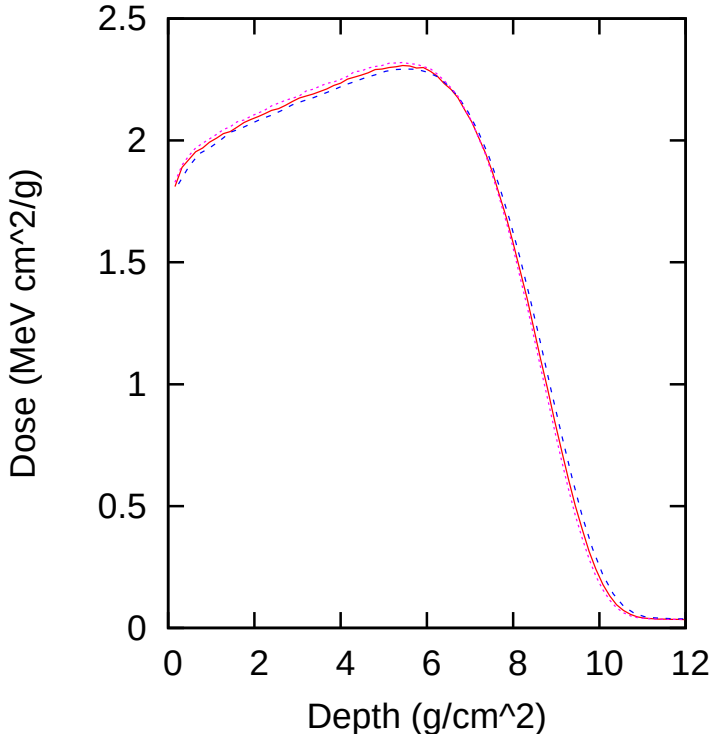


Figure 3.7: Depth-dose profiles calculated with PENGEOM for water with densities 0.8 g/cm^3 (dots) and 1.2 g/cm^3 (dashes). The solid line corresponds to the profile calculated with penVox using the nominal water cross sections (i.e., including the density effect correction for 1.0 g/cm^3 water).

density ρ. Finally, L is a function of β^2 given by the largest root of the equation

$$\frac{1}{Z}\int_0^\infty \mathrm{d}W\,\frac{\mathrm{d}f(W)}{\mathrm{d}W}\frac{L^2}{W^2+L^2} - L^2\frac{(1-\beta^2)}{\Omega^2} = 0. \tag{3.5}$$

Notice that, formally, β (or, equivalently, the kinetic energy of the projectile) and ρ appear in Eqs. (3.4) and (3.5) *only* through the combination $(1-\beta^2)/\Omega^2$. Since Ω^2 is proportional to ρ, the density effect correction that corresponds to a certain combination of β and ρ can be computed using the nominal density ρ_0 of the material in question and a velocity β' such that

$$\frac{1-\beta^2}{\rho} = \frac{1-\beta'^2}{\rho_0}. \tag{3.6}$$

PENELOPE tabulates the density correction given by Eq. (3.4) (with a convenient, simple OOS model) for ρ_0 and a grid of kinetic energies T of the electron or positron. Thus, in terms of T, the computation of δ_F for a voxel with density ρ can be done using ρ_0

instead and an equivalent energy T' given by

$$T' = \frac{1}{\sqrt{1-\beta'^2}} - 1 = \sqrt{\frac{\rho}{\rho_0}}\,(T+1) - 1 \tag{3.7}$$

where use has been made of Eq. (3.6). In this relation the kinetic energy is expressed in units of the electron rest energy (approximately 511 keV) and the relativistic relation between T and β has been used.

Strictly speaking, this method of accounting for density changes from voxel to voxel is not exact. It neglects the fact that the OOS itself also depends, to a certain extent, on ρ. Let us consider, for the sake of concreteness, the PENELOPE model for the OOS of an insulator, which is formed by a superposition of resonances of the form

$$\frac{\mathrm{d}f(W)}{\mathrm{d}W} = \sum_k Z_k\,\delta(W - W_k), \tag{3.8}$$

where the symbol δ represents Dirac's distribution, Z_k is the number of electrons occupying the k-th shell and the W_k's are the resonance energies. Following Sternheimer (1952), the resonance energy of each bound-shell oscillator is expressed with a semiempirical formula that involves the ionisation energy of that shell, the mass density and a free parameter; this parameter (and hence the resonances) is determined by requiring that the equation

$$Z \ln I = \int_0^\infty \mathrm{d}W\ \frac{\mathrm{d}f(W)}{\mathrm{d}W} \ln W, \tag{3.9}$$

is satisfied. The mean ionisation energy I, which plays a key role in the familiar Bethe stopping power formula, can be inferred from experimental measurements. In order to be rigorously correct, therefore, different values of I should be employed for different densities so that the new W_k's could be re-computed.

At this point it is important to bear in mind that variations in the mass density of the materials that are likely to appear in a CT scan are ultimately artefacts caused by the presence of inhomogeneities inside voxels or by inaccuracies in the CT acquisition and reconstruction processes. Therefore, it can be argued that the adaptation of the inelastic cross section to the local voxel density may not only be a costly procedure in terms of CPU time, but it can also worsen the agreement with experiments because density variations are not factual.

Another, less fundamental, disadvantage of the recipe given by Eq. (3.7) is that it requires changing the inner workings of the physics routines of PENELOPE, a course of action that would be against the design principles of penEasy.

4 Implementation of a triangle mesh geometry in PENELOPE

As discussed in previous chapters, the geometric model is a key feature of a Monte Carlo code, because it limits the type of applications that can be handled in practice. Most general-purpose codes employ geometries based on the combination of simple primitive objects, such as quadric surfaces. These primitives are convenient to simulate simple or idealised objects, but they are not well suited to model objects that have free-form (i.e., arbitrary) shapes, such as many organic structures. The use of a voxelised geometry (e.g., penVox, described in section 3.2) facilitates the development of anatomical phantoms from segmented CT scans. However, a shortcoming of voxelised phantoms is that they have a limited resolution, which is determined by the voxel size. Furthermore, they are not flexible, that is, objects included inside the phantom can not be easily translated or deformed (although a software tool for the manual reassignment of voxels was developed by Becker *et al.* 2007).

Another problem of current general-purpose codes is that the implemented geometry models are usually incompatible. This means that the geometry files can not be used in

different codes and have to be created and visualised using tools specifically developed for each code.

To overcome some of these limitations we have developed penMesh, a general-purpose Monte Carlo code based on the PENELOPE subroutine package that employs a standard computer-aided design (CAD) geometry model: triangle meshes. triangle meshes can approximate any arbitrary surface and, therefore, can represent the boundary of virtually any object. Furthermore, they can be easily generated, manipulated, stored and displayed using the powerful and well-established tools from the CAD and computer games communities. Other Monte Carlo codes have also been adapted to CAD geometries, such as GEANT (Sulkimo and Vuoskoski 1996) or EGS4 (Tabary and Glière 2000). A triangle mesh geometry package for PENELOPE had been previously developed but the code had an extremely low simulation efficiency and is not publicly available (Borglund *et al.* 2004).

In this chapter we present a detailed description of the penMesh code and its triangle mesh ray-tracer algorithm. A benchmark with experimental measurements and an analysis of the code performance are also provided in sections 4.3 and 4.4, respectively. Example applications of this code in diagnostic imaging are provided in chapter 7.

4.1 PenMesh

The simulation code penMesh (Badal *et al.* 2007; Kyprianou *et al.* 2007; Badal *et al.* 2008) implements a general-purpose Monte Carlo algorithm that simulates the transport of electrons, positrons and photons in a geometry composed of homogeneous bodies limited by triangle meshes. The interaction between matter and radiation is simulated by the state-of-the-art physics from the PENELOPE 2006 package (see section 2.3). The particle tracking is handled by a new set of subroutines implementing the ray-tracing algorithm described in section 4.2.2.

PenMesh uses a hybrid geometry model that combines the simplicity of objects defined by quadric surfaces and the geometric detail of objects represented by triangle meshes. The standard PENELOPE quadric geometry package (PENGEOM) is used to define a bounding box that encloses the triangles and to track the particles outside this box. The tracking inside is handled by the novel triangle mesh ray-tracer package.

The penMesh code is based on the penEasy package, the modular general-purpose main program for PENELOPE described in section 3.1. Most of the features of penEasy are made available to penMesh, such as its flexible source models and tallies for the most

common quantities of interest (e.g., 3D dose distributions, energy fluence spectra, energy deposition pulse height spectra, etc.). Taking advantage of the main program modular structure, a new tally for the creation of radiographic images has been developed and implemented in penMesh, as explained in section 7.2.

PenMesh and the new geometry package are coded in C++. However, they make use of the original penEasy and PENELOPE subroutines and common blocks in Fortran. Therefore, the code takes advantages of the advanced features of C++ (object-oriented programming, dynamic memory, recursive functions, pointers, etc.) and the computational efficiency and simplicity of Fortran. The multi-language compilation is facilitated by a header file that harmonises the naming conventions for the C++ and Fortran languages.[1] The documentation for the new source code has been prepared using the automatic documentation generator system `doxygen` (`http://www.doxygen.org`). Appendix B provides a summary of the documentation for the two C++ classes employed by penMesh.

4.2 Triangle mesh geometry

PenMesh uses a boundary representation geometry model in which the shape of each object is described by a triangle mesh. The implemented algorithm is not limited to any specific application and imposes only two restrictions on the meshes: being watertight closed (a particle may not enter or exit the mesh without crossing a triangle) and not self-intersecting manifolds (triangles from the same mesh must not intersect). The latter requirement is imposed so that a particle may not enter the triangle mesh from within itself. The implemented algorithm can handle intersecting meshes from different objects, meshes contained inside other meshes, and overlapping coplanar triangles.

Some advantages of triangle meshes compared to other mathematical surfaces—such as quadrics, non-uniform rational B-splines, or Bezier patches—are that triangles can be fitted to any arbitrary surface, can be exactly ray-traced (costly for cubic or higher-order parametric surfaces), and can be easily generated, manipulated, stored and displayed using existing CAD software. An obvious drawback of the triangle meshes is that curved surfaces can only be approximated up to the smallest triangle size used. Nevertheless, the size of the triangles can be adapted to the required level of detail and

[1] PenMesh has been designed to be compiled in the Linux operating system and with the GNU compiler collection (GCC; `http://gcc.gnu.org`) or compatible compilers (such as Intel®Fortran). To use Fortran code from C++ functions, the Fortran subroutines are simply declared as `extern "C" void` functions and with an underscore attached at the end of the original name in lowercase. Similarly, common block variables are accessed as `extern "C"` structures.

no artifacts should appear in the simulation due to tessellation inaccuracies.

In penMesh, the mesh corresponding to each object used in the simulation is stored in an external file in standard ASCII `ply` format (Stanford 3D Scanning Repository). The `ply` file must contain three different sections: a header giving the number of triangles and vertices in the mesh (other properties, like normals or color, may be included but are not used); a list of the coordinates of each vertex; and, finally, a list of the vertices associated to each triangle (adjacent triangles may share two vertices in a closed mesh). Even though only a few features of the `ply` format are employed by penMesh, the objects can still be created and handled by most of the available CAD software. There are also numerous programs that can export/import these `ply` files to/from other popular CAD formats.

4.2.1 Spatial data structures: the octree

Every time a particle moves across the triangle mesh geometry it is necessary to check if the particle's trajectory intersects any triangle. A typical penMesh geometry contains millions of triangles and, therefore, checking the intersection with every triangle at each step would result in an extremely slow computation (Borglund *et al.* 2004). This can be avoided by using a spatial data structure to group adjacent triangles inside bounding boxes. Using this technique only those triangles located inside the boxes crossed by the ray will be checked for intersection, therefore the total number of ray–triangle intersection tests will be significantly reduced at the expense of adding ray–bounding box intersection tests. Fortunately, computing the distance from a point to the wall of an axis-aligned box is very fast and does not introduce a significant overhead in the simulation (see Eq. 4.6 in section 4.2.3). The data structures most commonly used for ray-tracing are the uniform grid, the octree, and the k-D tree (Chang 2001).

The *uniform grid* divides the space in axis-aligned equal-size boxes (voxels). The grid ray-tracing is straightforward and can be accelerated by accounting for the periodicity of the intersection between the ray and the voxel walls (Amanatides and Woo 1987). The drawback of uniform grids is that they are inefficient in heterogeneous geometries, i.e., when there are regions with high density of triangles and regions with low density or without triangles. In this situation, using small voxels significantly increases the required computer memory while forcing the program to calculate many unnecessary intersections with empty voxels in the low density region. On the other hand, if large voxels are employed the code has to calculate many ray–triangle intersections inside the voxels located on the high density region and the code execution is slow.

The *octree* (Meagher 1982; Glassner 1984; Chang 2001) is a hierarchical tree structure that subdivides the space in different size boxes (nodes). The first node (root) is an axis-aligned parallelepiped enclosing all the triangles. This node is divided in eight octants of the same size, which are recursively subdivided in the same way. This subdivision is applied until all the space is partitioned according to a predetermined termination condition, on which we will elaborate later. The index of subdivision is called the octree level and the final nodes are called leaves. Constructing and ray-tracing an octree is fast and simple because the size and position of each node depends only on its level and on the shape of the root node (Revelles *et al.* 2000).

The *k-D tree* (Chang 2001) is a binary space-partitioning hierarchical structure where the space is recursively divided in two parts using axis-aligned planes which alternate the dividing axis in the k dimensions (e.g., alternating planes along the x and y axis in 2D). The plane position is calculated according to a predetermined splitting condition (e.g., the resulting volumes may contain exactly half of the triangles in the original volume). The uniform grid and the octree are particular cases of a k-D tree with simple splitting conditions. Therefore a k-D tree is theoretically the most efficient spatial data structure when the splitting condition is correctly optimised for the given application and geometry. Nevertheless, the creation and ray-tracing of a general k-D tree is not as simple as for the other structures and the resulting computer code may require a longer execution time.

Based on the expected efficiency and low overhead, an octree structure was implemented in penMesh. The octree is generated on execution time using an heuristic termination condition that subdivides a node whenever the amount of triangles it contains is larger than or equal to the value of the node level. This restrictive condition favours the generation of higher level nodes tightly fit on the object surface and it has exhibited optimum computing performance in our benchmark tests. The maximum octree level can be chosen by the user depending on the complexity of the geometry (which depends on the total number of triangles and their structure in space) and the available RAM memory in the computer where the simulation will be executed. As a rule of thumb, it is good practice to allow as many subdivisions as possible since ray-tracing a node is much faster than checking a triangle intersection (an analysis of the octree performance is given in section 4.4).

Figure 4.1 shows a sample level 9 octree sorting the triangles from the anthropomorphic phantom described in section 7.1. The represented plane contains triangles corresponding to the skin, lungs, heart, liver, ribs, sternum and a vertebra meshes. A 2D view of the octree structure, in EPS (encapsulated postscript) image format, is automat-

ically generated by penMesh to allow the user to visually validate that the triangles are correctly sorted and completely enclosed inside the defined bounding box. The octree plane to be drawn, the maximum octree level, the bounding box size and location, and the name of the meshes that are included in the simulation are specified in an external input file.

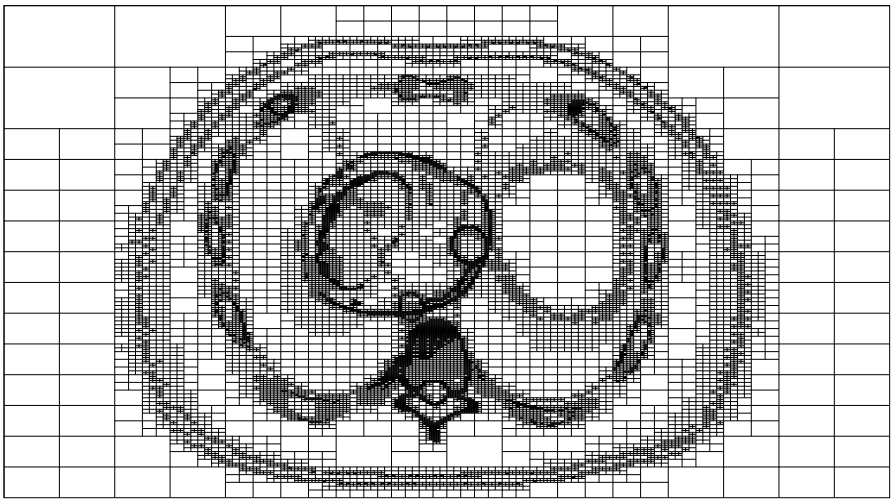


Figure 4.1: Node structure of a level 9 octree sorting the triangles from a realistic anthropomorphic phantom.

The EPS image is created taking advantage of the vector graphic capability of the postscript language. This means that the walls of the octree nodes are not represented by pixels, which would limit the image resolution and require a lot of computer space for high level octrees, but defined as line segments by using the postscript commands `moveto` and `lineto`. Figure 4.2 shows the header and first lines of a sample EPS image created by penMesh.

4.2.2 Ray-tracing algorithm

The new geometric routines implement a robust and exact algorithm to track the movement of particles across triangles contained inside octree leaves.

Ray-tracing inside the triangle region begins by locating the particle position in the octree structure. This is done searching recursively from the root node to the subnodes

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 596 344
%%Title: octree_structure.eps
%%Creator: penMesh
%%Description: Drawing of the octree plane Z = 35.4725 cm
%%             Octree definition constant: MAX_LEVEL = 11
%%             Maximum octree subdivision level is 8
%%Page: 1 1

/cm {28.3465 mul} def   % Convert cm->inches->PS points (1/72 in)
0.5 cm 0.5 cm translate % Translate the page origin
0.37037 0.37037 scale   % Scale figure coordinates to fit page
28 cm 22.5 cm translate % Translate figure origin to page origin

% --Aliased used to reduce the file size:
/N {newpath} bind def
/M {moveto} bind def
/L {lineto} bind def
/LS {lineto stroke} bind def

% --Octree bounding box:
newpath -28 cm -22.5 cm moveto 26 cm -22.5 cm lineto 26 cm 7.5 cm
         lineto -28 cm 7.5 cm lineto closepath stroke

% --Octree leaves (using alias):
N -28 cm -22.5 cm M -21.25 cm -22.5 cm L -21.25 cm -18.75 cm LS
N -21.25 cm -22.5 cm M -14.5 cm -22.5 cm L -14.5 cm -18.75 cm LS
N -28 cm -18.75 cm M -21.25 cm -18.75 cm L -21.25 cm -15 cm LS
N -21.25 cm -18.75 cm M -14.5 cm -18.75 cm L -14.5 cm -15 cm LS
N -14.5 cm -22.5 cm M -7.75 cm -22.5 cm L -7.75 cm -18.75 cm LS
  (...)
```

Figure 4.2: Header and first lines from a sample EPS image created by penMesh.

until the octree leaf that contains the particle position is found. Finding the subnode that contains the particle is trivial because the splitting planes are located exactly at the middle of the node (in each dimension). Then, PENELOPE's physics routines sample the distance to the next interaction using a pseudo-random number generator and taking into account the medium interaction cross sections (see Eq. 2.3). The ray-tracer takes the sampled distance and searches for possible interface crossings in the rectilinear path. If the interaction point is found in the current node and no triangle is intersected, the interaction is processed. If the trajectory crosses a triangle, the particle is stopped on its surface and the medium material is updated. This triangle is not checked for intersection in the following leap to avoid getting trapped on the triangle surface due to computer rounding errors. Finally, if the particle reaches the leaf wall it is stopped and the neighbour leaf is loaded. The leap continues in the new leaf without re-sampling the interaction distance, because octree walls are virtual boundaries that merely sort the triangles and do not represent real material interfaces.

A variety of techniques can be used to find the neighbouring nodes and transport the particles from one leaf to another (Chang 2001). A top-down approach consists of finding the next leaf through a search from the root node. A bottom-up approach moves up to the leaf parent and grand-parent nodes and then goes down looking for the neighbouring node. Finally, the fastest approach—which we have implemented—takes advantage of pointers to the six leaf neighbour nodes (with the same or lower level) precalculated in the initialisation stage using one of the previous methods. This approach requires more computer memory (six extra pointer variables per leaf), but it simplifies the transport algorithm considerably.

A general and robust ray-tracing algorithm has to address two complicated situations: the intersection of triangle meshes from different objects and the presence of overlapping coplanar triangles. These situations may represent the real geometry or may have been artificially produced during the phantom generation or surface tessellation. Obviously, the triangle mesh geometry must be thoroughly inspected to assure that unrealistic object overlapping does not affect the simulation outcome.

PenMesh deals with triangle mesh intersections by storing the surfaces that are crossed during the particle track in a virtual particle *flight log*. Every time a particle crosses a surface the log is searched for the index of the corresponding object. If the surface was not previously crossed, the particle enters a new object and its index is recorded in the log. If the index is found in the log, the particle is exiting the object and its index is deleted from the log. In the regions where multiple objects overlap the flight log contains more than one entry of object indices. In such cases, the appropriate material at the current position is determined by using a lookup table that establishes an object priority hierarchy (see Segars 2001, Appendix A, and Kyprianou *et al.* 2007).

The flight log is necessary to determine the material found at any point inside the geometry and the material that will be found when the particle exits the current object. For this reason the generated secondary particles have to inherit a copy of the flight log at the point they are created. It is possible to define a radiation source inside the triangle mesh region but in such a case the source routine must load the appropriate flight log for each particle. The log can be automatically calculated by transporting a virtual ray from any point outside the triangle bounding box back to the source location.

A particle originating outside the octree region that enters and crosses the whole octree will end up having only one entry in the flight log. This index corresponds to the quadric body that defines the octree bounding box. Having more than one index in the log when leaving the octree signals an inconsistency in the geometry. Object indices may remain on the flight log if their triangle meshes are not correctly closed or they extend outside

the defined bounding box.

The second complicated situation mentioned above is the handling of overlapping coplanar triangles. During the ray-tracing process, particles are stopped on the surface of the intersected triangles. In case the distance to the next triangle is zero—as for coplanar triangles—a particle can get trapped, i.e., continuously jump between the coplanar triangles without changing its actual location. In a real computer the situation may be even worse because the intersection distance is calculated with a finite precision and a null distance may be represented by a value close, but not equal, to zero. In this situation the particle could continue the track but the flight log would be corrupted and the particle would move in an incorrect material. To avoid this problem the function that looks for triangle intersections in penMesh has been adapted to detect coplanar triangles, defined as those for which the intersection distance is smaller than 10^{-9} cm. All coplanar triangles are crossed at the same time and the corresponding object indices are recorded in the flight log. The active material is determined as usual.

4.2.3 Calculating ray–triangle, ray–box and triangle–box intersections

The most time-consuming part of the triangle mesh ray-tracing is the calculation of ray–triangle intersections (e.g., it accounts for 30% of the total simulation time in the case analysed in section 4.4). PenMesh calculates these intersections using the efficient algorithm developed by Möller and Trumbore (1997). This algorithm calculates the intersection point in terms of the so-called barycentric coordinates (u, v). In this coordinate system, a point with position vector $\mathbf{P}$, located on the plane of the triangle with vertices given by the position vectors $\mathbf{V}_0$, $\mathbf{V}_1$, $\mathbf{V}_2$, is expressed as

$$\mathbf{P}(u, v) = (1 - u - v)\mathbf{V}_0 + u\mathbf{V}_1 + v\mathbf{V}_2 \, . \tag{4.1}$$

The point $\mathbf{P}$ is found inside the triangle if, and only if, $u \geq 0$, $v \geq 0$ and $u + v \leq 1$. Given a ray $\mathbf{R}$ in parametric form

$$\mathbf{R}(t) = \mathbf{O} + t\,\hat{\mathbf{d}} \, , \tag{4.2}$$

where $\mathbf{O}$ is the position vector of the ray origin, $\hat{\mathbf{d}}$ is a unitary direction vector and t is a free parameter, it is possible to find the point of intersection between the ray and the triangle, (u, v), and the distance to the intersection, t, by solving the equality $\mathbf{R}(t) = \mathbf{P}(u, v)$, that is,

$$\mathbf{O} + t\,\hat{\mathbf{d}} = (1 - u - v)\mathbf{V}_0 + u\mathbf{V}_1 + v\mathbf{V}_2 \, . \tag{4.3}$$

As pointed out by Möller and Trumbore, the previous equation can be rearranged as

$$-t\hat{\mathbf{d}} + (\mathbf{V}_1 - \mathbf{V}_0)u + (\mathbf{V}_2 - \mathbf{V}_0)v = \mathbf{O} - \mathbf{V}_0 \ , \tag{4.4}$$

and this system of equations can be solved by applying Cramer's rule, obtaining

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\mathbf{F} \cdot \mathbf{E}_1} \begin{bmatrix} \mathbf{Q} \cdot \mathbf{E}_2 \\ \mathbf{F} \cdot \mathbf{T} \\ \mathbf{Q} \cdot \mathbf{D} \end{bmatrix} \ , \tag{4.5}$$

with $\mathbf{E}_1 = \mathbf{V}_1 - \mathbf{V}_0$, $\mathbf{E}_2 = \mathbf{V}_2 - \mathbf{V}_0$, $\mathbf{T} = \mathbf{p} - \mathbf{V}_0$, $\mathbf{F} = \mathbf{D} \times \mathbf{E}_2$, and $\mathbf{Q} = \mathbf{T} \times \mathbf{E}_1$. Note that the triangle plane equation does not have to be calculated in this algorithm.

The intersections with the walls of the octree nodes also take a significant amount of execution time (about 15%). Since the nodes are aligned with the axis the ray–box intersection can be found with a simple and fast computation. For a particle located at (x_p, y_p, z_p) inside a box and moving with a direction vector (u_p, v_p, w_p) with $u_p > 0$, the distance d_x to the nearest box wall perpendicular to the x axis is

$$d_x = \frac{x_{max} - x_p}{u_p} \ , \tag{4.6}$$

where x_{max} is the maximum value of x inside the box (the minimum x would be used in the case $u_p < 0$). Equivalent equations can be used to find the intersection with the y and z walls.

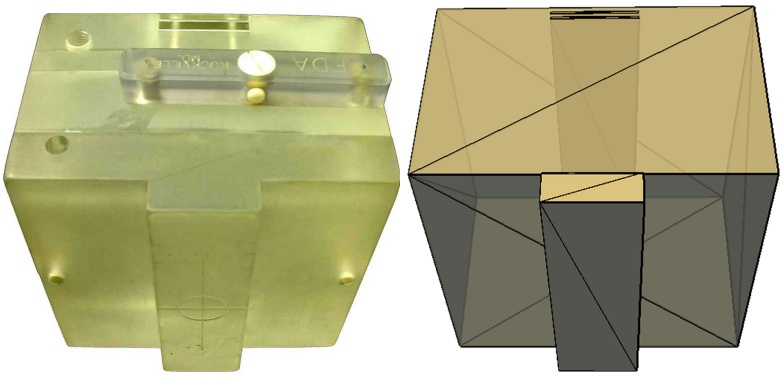
The triangle–box intersections that have to be calculated to distribute the triangles into the leaves during the octree generation are computed using the algorithm developed by Akenine-Möller (2001). This algorithm takes advantage of the separating axis theorem, which states that two convex polyhedra, A and B, are disjoint if they can be separated along either an axis parallel to a normal of a face of the polyhedra, or along an axis formed from the cross product of an edge from A with an edge from B. For the particular case of a triangle and an axis aligned box, the implemented algorithm requires checking 13 conditions.

4.3 Code validation

The performance of a new Monte Carlo code is typically evaluated by simulating a simple case that can be reproduced with previously well-established codes as well as with laboratory experiments. In spite of the fact that penMesh uses the original physics

routines from PENELOPE and the Monte Carlo algorithm from penEasy, which have been extensively used and validated with experimental data in the past (see chapter 3), it is still necessary to check that the new geometric models have been correctly implemented.

Scatter fraction measurements provide a simple way to quantitatively compare experimental and simulated data. Furthermore, scatter is one of the most important factors of image quality degradation and, therefore, the accurate simulation of scattered radiation is essential for the assessment of imaging systems. In order to analyse the accuracy of penMesh, scatter fractions were measured, and simulated, in two regions located 10 cm behind the CDRH[2] phantom, an idealised model of the human abdomen and lumbar spine described in the AAPM report 31 (1990). Figure 4.3 shows a picture of the phantom used in the lab and the triangle mesh version used by penMesh, containing 4 independent objects and 48 triangles in total. The triangulated phantom was designed with ParaView[3]. Scatter fraction measurements were also used by Mathy *et al.* (2003) as a preliminary validation of a CAD geometry package for EGS4.



(a) (b)

Figure 4.3: CDRH abdomen and lumbar spine phantom (AAPM report 1990) used in the scatter fraction measurements: (a) picture of the real phantom used in the lab; (b) triangle mesh version containing four independent objects, with 48 triangles in total.

[2]Center for Devices and Radiological Health (CDRH), U. S. Food and Drug Administration (FDA).

[3]ParaView is an open-source, multi-platform application designed to visualise and manipulate digital data. It provides a user-friendly graphical interface to the VTK library (http://www.vtk.org). This program is freely available at the website http://www.paraview.org.

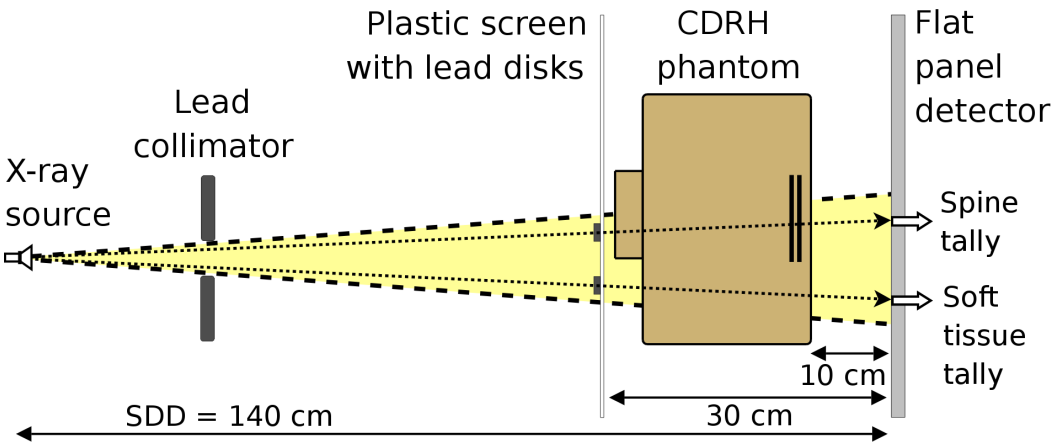


Figure 4.4: Experimental setting for the scatter fraction measurement (not to scale). The arrows mark the places where the fractions were tallied. The simulation geometry did not include the lead discs.

The scatter fraction (SF) of a given image acquisition is defined as

$$\mathrm{SF} = \frac{S_s}{S_s + S_p} \,, \tag{4.7}$$

where S_s is the signal produced by the scattered x rays and S_p is the signal produced by the primary (non-scattered) x rays (Chan and Doi 1983), as recorded in a standard energy integrating x-ray detector. The scatter fraction can be experimentally measured by blocking the primary beam with various size lead discs placed between the source and the imaged object; this process is known as the beam-stop method (Brezovich and Barnes 1977). A lead disc blocks the incident beam and projects a dark disc in the image. Any signal detected at the center of the projected disc is generated only by the radiation scattered within the object. The scatter fraction can be measured by dividing the signal detected at the center of the projected disc (only scatter) and the signal detected at the same location by removing the disc from the field (scatter plus unscattered radiation). A diagram of the experimental setting used to measure the scatter fraction is given in Fig. 4.4. Naturally, the diameter of the beam blocker affects the measured scatter. The value corresponding to a disc with zero diameter (i.e., the true scatter fraction) can be estimated by taking multiple measurements for discs with decreasing diameters and extrapolating with a linear fit (Brezovich and Barnes 1977; Chan and Doi 1983).

Measuring the scatter fraction in a Monte Carlo simulation is straightforward because every x-ray interaction is simulated individually and the scattered and primary par-

ticles can be differentiated without using beam blocks. In penMesh the particles are labelled according to the interactions that they have suffered in the track. There are four different kinds of particles: non-scattered (primaries), single elastic scattered, single inelastic scattered, and multiple scattered. Using this information, and assuming that the signal detected in the x-ray detector is proportional to the total energy of the particles entering the scintillator (Chan and Doi 1983), the scatter fraction can be directly measured with Eq. 4.7.

A 90 kVp x-ray source, with 10×10 and 20×20 cm^2 fields and a source-detector distance (SDD) of 140 cm, was used. The scatter fraction was tallied 10 cm downstream the phantom, in a point located behind the soft tissue region (containing 16.91 cm of lucite) and a point behind the spine region (composed of 0.46 cm of aluminum and 18.95 cm of lucite). The detector used in the experimental setup was a digital flat panel, model Varian 4030CB (Varian Corp. Salt Lake City, UT, USA), with 2048×1596 195×195 μm^2 pixels and a 600-μm-thick columnar CsI(Tl) scintillator. The x-ray tube was a Varian B180 (Varian Corp., Salt Lake City, UT, USA) with 0.3 mm nominal focal spot and a tube filtration of 1 mm aluminum.

The simulation used a point source emitting a realistic 90 kVp energy spectrum computed with the software provided by Cranley *et al.* (1997). The square fields were conformed using a completely absorbing lead collimator defined with quadric surfaces. The scatter fraction was estimated scoring the number of primary and scattered particles that entered a 1.0 cm^2 sensitive area centered at the point corresponding to the center of the projected discs in the experimental setting. The lead discs were not included in the simulation geometry because the scatter fraction was directly tallied counting the particles that arrived at the sensitive area. The number of initial x rays for the 10×10 and 20×20 cm^2 fields were $3{\cdot}10^9$ and $9{\cdot}10^9$, respectively. The simulation speeds, in an Intel®Core™ 2 processor at 2.4 GHz, were 37127 x-rays/second for the first field and 32974 x-rays/second for the second.

Table 4.1 summarises the scatter fractions experimentally measured and simulated with penMesh. Figure 4.5 displays the experimental measurements for different disc diameters and the linear fits that produced the reported fractions. The agreement between the measured and simulated scatter fractions was quite good, taking into account the large experimental (8%) and simulation (3–5%) uncertainties. However, differences of up to 6.5% for some of the measurements were observed. These differences were expected because the techniques used for the measurements were different: beam blockers were used in the lab, while the simulation calculated the scatter fraction directly labelling the particles that had been scattered. In addition, it was assumed that there

was no transmission of radiation through the lead discs and no scatter coming from the collimator. Furthermore, the scatter fraction was experimentally tallied in the 9 pixels closer to the center of the projected discs (sensitive area of $3.6{\cdot}10^{-3}$ cm^2) while, for simulation efficiency reasons, the sensitive area in the simulation was much bigger (1 cm^2). Other sources of discrepancy were the idealised models used to simulate the radiation source (point focal spot, nominal energy spectrum, etc.) and the detector (signal proportional to the deposited energy, 100% detection efficiency, no scattering in the scintillator, no optical photon transport or energy dependent optical photon generation, etc.). All these factors affect the experimental scatter fraction measurement using the beam stop method.

Table 4.1: Experimental and simulated scatter fraction behind the soft tissue and spine regions of the CDRH torso phantom, for a 90 kVp x-ray source. The statistical uncertainty in the simulation is given as 2σ; the uncertainty corresponding to the experiment was estimated as 8%.

Field	Region	Experiment	Simulation	Difference
10×10	soft	0.276	0.258 ($\pm$ 3%)	-6.5%
	spine	0.405	0.400 ($\pm$ 4%)	-1.2%
20×20	soft	0.491	0.498 ($\pm$ 4%)	1.4%
	spine	0.626	0.664 ($\pm$ 5%)	6.1%

An interesting feature of Monte Carlo simulations is that they can give more information about the scatter than what can be actually measured experimentally. A simulation not only provides the scatter fraction but can also determine the fraction of particles scattered by each interaction process, or which part of the geometry produced the scattered particles (Zaidi 1999). As an example, Table 4.2 shows the simulated results from Table 4.1 separated into the fractions corresponding to the x-rays that suffered a single elastic interaction, an inelastic interaction, or multiple interactions in the phantom.

4.4 Efficiency analysis

In order to assess the efficiency of penMesh and the octree structure in a typical case, the performance of the coronary angiography simulation presented in section 7.3

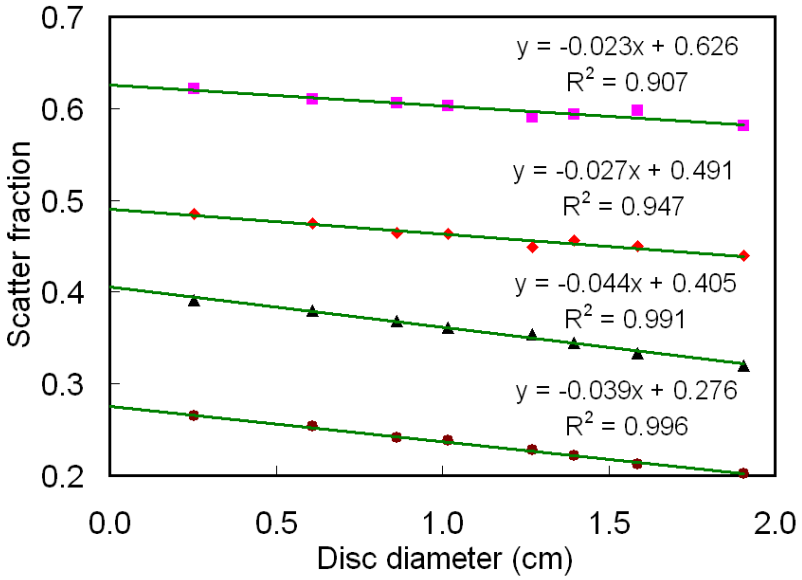


Figure 4.5: Scatter fractions experimentally measured for discs with different diameters. The presented data correspond to: 20×20 field, spine region (squares); 20×20 field, soft tissue region (diamonds); 10×10 field, spine region (triangles); and 10×10 field, soft tissue region (circles). The equations of the linear fits are provided above the corresponding data sets.

(page 97) was analysed. A profiling[4] of the penMesh source code for this particular application is presented in Fig. 4.6. The profiling displays the percentage of CPU time used by the PENELOPE routines, the quadric geometry, the octree traversal, and the ray–triangle intersection tests. It can be observed that the triangle mesh geometry consumes 46% of the simulation time. This is of the same order of magnitude as the time expended by the standard quadric geometry in a typical PENELOPE simulation, although in our case the objects have much more detail. It is worth noting that the quadric geometry (i.e., PENGEOM) expends a significant amount of time (9.5%), even though it was only used to track the particles in the air volume surrounding the octree region.

The performance of the octree structure was also analysed. Figure 4.7 shows an axial slice, at the center of the phantom heart, of an octree with levels 3, 5, 7, and 9 (the phantom is described in section 7.1). The simulation performance, in terms of required computer memory and simulation speed[5] as a function of the octree level, is presented

[4]The profiling was performed using the free software GNU `gprof`, which can be obtained at the website http://www.gnu.org/software/binutils/manual/gprof-2.9.1/gprof.html.

[5]In this application penMesh was compiled using the Intel®Fortran and C++ compilers for Linux

Table 4.2: Simulated scatter fractions for x rays scattered by different interaction mechanisms (statistical uncertainty given as 2σ). The relative importance of the different interactions with respect to the total fractions given in Table 4.1 is also shown.

Field	Region	Scattering	Simulated fraction	% Total
10×10	soft	elastic	0.073 (± 3%)	28.3
		inelastic	0.048 (± 2%)	18.6
		multiple	0.136 (± 2%)	52.7
	spine	elastic	0.086 (± 3%)	21.5
		inelastic	0.083 (± 4%)	20.8
		multiple	0.230 (± 4%)	57.5
20×20	soft	elastic	0.077 (± 4%)	15.5
		inelastic	0.118 (± 4%)	23.7
		multiple	0.303 (± 4%)	60.8
	spine	elastic	0.076 (± 5%)	11.4
		inelastic	0.158 (± 5%)	23.8
		multiple	0.430 (± 5%)	64.8

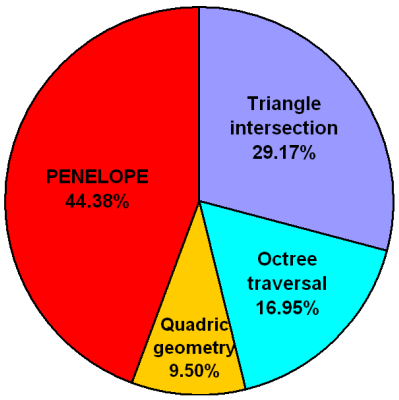
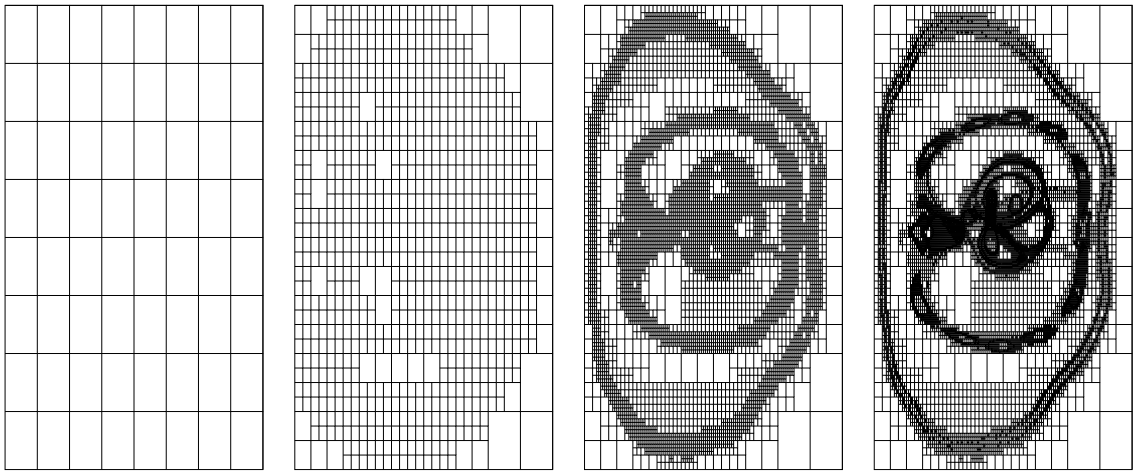


Figure 4.6: Profiling of the penMesh code for the 90 kVp angiography simulation described in section 7.3. The percentage of CPU used by the PENELOPE physics routines, the quadric geometry (PENGEOM), the octree traversal, and the ray–triangle intersection tests, are displayed.

in Fig. 4.8. Only the memory allocated for the octree is shown; the simulation requires an additional 346 MB of memory for storing the PENELOPE variables and the raw triangles.

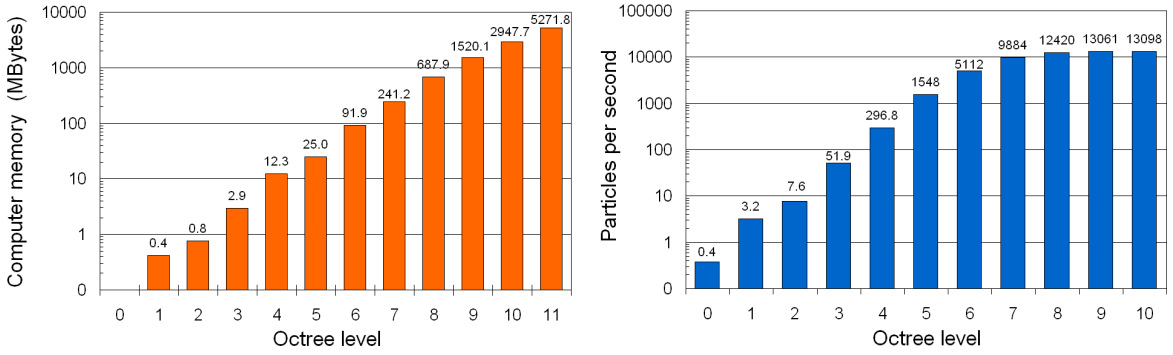
The presented results show that the octree spatial data structure plays a key role to

(option `-O3 -ipo`). For the timing tests, the simulation was run in a single CPU from a Dual-Core AMD®Opteron™2 GHz processor.



(a) (b) (c) (d)

Figure 4.7: Axial slice of the octree data structure sorting the triangles from the tessellated NCAT phantom, at the height of the heart (see Fig. 7.1). Four different octree levels are shown: (a) level 3, (b) level 5, (c) level 7, and (d) level 9.



(a) (b)

Figure 4.8: Performance of the penMesh code, for the 90 kVp angiography simulation, using different levels of the octree structure: (a) computer memory required for storing the octree structure (346 MB are allocated for the rest of the code); (b) simulation speed (primary x rays per second in a single CPU).

increase the code efficiency. The octree accelerates the simulation five orders of magnitude: from less than one particle per second, to more than 10^4 particles per second. For octree levels higher than seven the acceleration saturates and the simulation speed is almost constant. Indeed, the simulation is expected to run slower above a certain octree level due to the extra time expended in the octree traversal. The number of

octree nodes increases exponentially with the octree level, because the implemented termination condition is very restrictive and many nodes are subdivided up to the maximum level. The computer memory used to allocate the octree structure also increases exponentially, although only above seven octree divisions the memory expended in the octree is larger than the memory used to store the raw triangles. Analysing the timing and memory usage results we found that a level eight octree gives an optimum performance in our application, with memory requirements available on standard workstations (about 1 GB RAM).

5 Parallelisation of Monte Carlo simulations

The statistical variance of Monte Carlo calculations is, when everything else remains the same, inversely proportional to the simulation time (see Eq. 2.2). For this reason, and in spite of the fact that fast computers are available nowadays at low cost, there are many situations where obtaining a reasonably low statistical uncertainty involves a prohibitively large amount of computing time.

A simple way to overcome the previous limitation is by having recourse to parallel computing, that is, by executing the simulation in multiple CPUs at the same time. Various strategies and tools have been developed to facilitate the implementation of this solution, e.g., MPI, openMP, PVM and openMOSIX. All these have been successfully used with Monte Carlo codes, but they have the drawback of requiring the modification of the sequential code and the installation of additional software, which may be inconvenient for some users.

In general, the parallelisation of an algorithm is not a straightforward task and it may even be unfeasible. Fortunately, it can be achieved relatively easily in the case of a

Monte Carlo radiation transport simulation code because each random history is sampled independently from the others. The particles can be simply grouped into several batches and executed in parallel on different computers or in different computing kernels of multiprocessor computers. The results yielded by the independent runs of the same code can be combined *a posteriori* to obtain a single final result with a reduced statistical uncertainty. Some authors call this approach process replication (or domain replication) to distinguish it from a possible alternative, domain decomposition, which involves segmenting the space of possible states of the system, e.g., by splitting the geometry or the energy range into pieces and assigning each piece to a different CPU. The "natural" simplicity of the process replication scheme has led to the denomination "embarrassingly parallel" used by some authors to refer to Monte Carlo algorithms (see, e.g., Srinivasan *et al.* 2003).

In order to sample the probability distributions that characterise the system to be simulated it is necessary to have a source of randomness, typically a string of numbers uniformly distributed between 0 and 1. True random numbers can be generated using physical devices such as radiation detectors or semiconductors, but they are too slow compared with computer speeds and their unpredictable nature makes it difficult to debug the programs or check the results. For these reasons most Monte Carlo codes use pseudo-random number generators, which provide a cyclic sequence of numbers produced by deterministic algorithms (L'Ecuyer 1990; Knuth 1998). These algorithms must be carefully chosen so that the correlations between the generated numbers are not apparent. Thus, they are required to pass several statistical tests intended to check the uniformity and independence of the produced sequence (Coddington 1994; Srinivasan *et al.* 2003).

In this chapter, a brief description of the basic properties of pseudo-random number generators used in Monte Carlo codes, and a discussion of their use in parallel calculations, are provided. Section 5.2.1 introduces a set of Linux scripts and FORTRAN programs, named clonEasy, that implement the process replication approach on a set of "clone" CPUs governed by a "master" computer. This software package readily permits parallel computations to be carried out without requiring additional software or significant modifications of the original, sequential, code. Finally, a related software tool named seedsMLCG, which provides the information necessary to initialise disjoint sequences of any pseudo-random number generator based on a multiplicative linear congruential algorithm, is presented in section 5.2.2. This auxiliary tool can be used, for example, to obtain disjoint sequences of RANECU, the generator used in PENELOPE.

5.1 Pseudo-random number generators

The best pseudo-random number generators are those that perform as well as the true random number generators in the randomness tests and have a solid mathematical basis that justifies their essential properties and cycle length. A theoretical proof of their quality is an important point since biased results due to subtle correlations have been reported in the past (see e.g., Ferrenberg and Landau 1992).

Among the most well studied and extensively used algorithms for pseudo-random number generators there are those based on recursions with modular arithmetic. These algorithms take a few input integer numbers, called *seeds*, and produce a cyclic sequence of integers which can be subsequently transformed into real values uniformly distributed between 0 and 1. A family of algorithms that belong to this class is the so-called multiplicative linear congruential generators (MLCGs). Although more recent generators are known to perform better in some aspects and have therefore superseded MLCGs for most applications, it can be argued that congruential generators are conceptually simpler and their weaknesses well understood. In fact, generators based on combinations of MLCGs (such as RANECU, which is described in section 5.1.1) have been extensively used in the past and are still in use by numerous computer codes. The Monte Carlo code PENELOPE, for instance, has been amply benchmarked against experimental results and other codes since its first release in 1996, and no bias that can be attributed to its pseudo-random number generator has been detected to date.

An MLCG is initialised with a single seed, an integer value S_0. It produces each term of the sequence (S_i, $i = 0, 1, \ldots$) by multiplying the previous value by an integer a and calculating the modulo m, i.e., computing the remainder of the integer division by m. The possible remainders are all the integers from 0 to $m-1$. A null remainder, however, should be avoided to prevent the generator from collapsing into a sequence of zeroes. A real value u_i in the interval [0,1) can be obtained by dividing S_i by m. The resulting sequence can thus be expressed by

$$S_{i+1} = (aS_i) \,\mathrm{MOD}\, m \;, \quad u_i = \frac{S_i}{m} \;. \tag{5.1}$$

The largest possible period of an MLCG is $m-1$. In case m is restricted to be representable in a digital computer by a 32-bit-long signed integer, the maximum attainable period is $2^{31}-1 \sim 2{\cdot}10^9$. This period is clearly insufficient for present-day computers since a single CPU working at a clock speed of 3 GHz can perform nearly $3{\cdot}10^9$ operations per second and, therefore, it could generate the whole sequence in a few seconds.

This limitation can be overcome by combining two or more MLCGs, as it is explained below.

A known weakness common to all MLCGs is, as pointed out by Marsaglia (1968), that if groups of n successive random values are used as the Cartesian coordinates of points in an n-dimensional space, they do not uniformly fill up the volume. Instead, they lie on a relatively small number of parallel hyperplanes producing a lattice structure. The maximal distance between adjacent hyperplanes is a convenient measure of the quality of the generator and its determination is the goal of the so-called spectral test (Knuth 1998). When the distance between hyperplanes is small the illusion that points are uniformly distributed in the hypercube is reinforced. This criterion is thus frequently employed to find the best multiplier and modulus for an MLCG.

Another widely used algorithm for pseudo-random number generation, based on a recursion with modular arithmetic, is the lagged Fibonacci. The generator RCARRY implements an extension of this algorithm called subtract-and-borrow (Marsaglia *et al.* 1990). Its initialisation requires 24 seeds (S_i, $i = 0, \ldots, 23$) and a carry bit c_{23} (equal to either 0 or 1), and it generates the elements of the sequence as

$$S_i = (S_{i-10} - S_{i-24} - c_{i-1}) \text{ MOD } 2^{24} \,, \quad i > 23 \,, \tag{5.2}$$

where c_{i-1} ($i > 24$) is 0 if $(S_{i-11} - S_{i-25} - c_{i-2}) \geq 0$ and it is 1 otherwise. This generator has a very long period ($\sim 5 \cdot 10^{171}$) but it fails some statistical tests (Vattulainen *et al.* 1995) and, therefore, its use may compromise the reliability of the simulation results. By studying RCARRY from the viewpoint of the dynamics of chaotic systems Luescher (1994) showed that the detected correlations are short-ranged and that they can be eliminated by simply discarding $(p - 24)$ elements of every p consecutive elements of the sequence, where p is a fixed user-defined parameter that determines the quality (i.e., randomness) of the resulting generator. It can be argued that the corresponding computer code, named RANLUX (James 1994), has therefore been proven to produce random sequences of the highest quality, but the price to be paid is a low efficiency, that is, the quantity of pseudo-random numbers produced per unit time is relatively small compared with other algorithms.

Another heavily used generator is the Mersenne Twister[1], which is based on the state-of-the-art algorithm proposed by Matsumoto and Nishimura (1998). This algorithm implements a version of the linear feedback shift register (Knuth 1998), which involves

[1]More information on the Mersenne Twister generator, and sample implementations in different computer languages, can be obtained at the website:
http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html.

binary operations (i.e., modulo 2 operations) with the bits of the initial 624 seeds. The Mersenne Twister is becoming increasingly popular due to the fact that it is considerably faster than RANLUX and RANECU and that it has passed the most relevant randomness tests available. However, although it has been demonstrated that the generated values are equidistributed in 623 dimensions and that the cycle period is of the order of 10^{6001}, the theoretical basis for its random properties is not as established as it is for RANLUX and its possible weaknesses are not well understood.

The string of random numbers employed by each processor participating in a parallel Monte Carlo simulation should be uncorrelated with those used by the other processors in order to guarantee the statistical independence of the different partial results. The ability to produce these sequences is thus an important feature of a pseudo-random number generator. While some algorithms can generate long independent strings in a simple way (e.g., RANLUX produces one of these sequences for each integer value given to its initialisation routine, see James 1994), more elaborated techniques are required in other cases (for the Mersenne Twister they can be produced using a slightly modified version of the generator in each node, as proposed by Matsumoto and Nishimura 2000). MLCGs can not intrinsically produce independent sequences, but these can be obtained using a simple procedure, as explained in section 5.1.2. The software library "Scalable Parallel Random Number Generators Library" (Mascagni 2000) can be used to produce uncorrelated sequences for a number of popular generators.

5.1.1 RANECU generator

The pseudo-random number generator RANECU was developed by L'Ecuyer (1988). It combines the sequences $S_i^{(1)}$ and $S_i^{(2)}$ ($i = 0, 1, \ldots$) from a pair of MLCGs with moduli $m^{(1)}$ and $m^{(2)}$ and multipliers $a^{(1)}$ and $a^{(2)}$, respectively, to produce a new sequence S_i defined by

$$S_i \;=\; (S_i^{(1)} - S_i^{(2)}) \,\mathrm{MOD}\, (m^{(1)} - 1) \;\; . \tag{5.3}$$

The parameters of the two MLCGs used in RANECU, chosen so as to yield optimal results in the spectral test, are shown in Table 5.1. The period of the combined generator is the least common multiple of the periods of $S_i^{(1)}$ and $S_i^{(2)}$, and its lattice structure is considerably better than that of its individual components, as explained below.

RANECU was coded in FORTRAN 77 by James (1990), for computers using registers with a minimum of 32 bits. The FORTRAN code included in PENELOPE, which differs from the one proposed by James mainly in that it returns a single value at each call instead of an array of values, is displayed in Fig. 5.1. The reliability of RANECU stems

Table 5.1: Parameters of the two multiplicative linear congruential generators used by RANECU.

Generator	Modulus (m)	Multiplier (a)
$S_i^{(1)}$	2147483563	40014
$S_i^{(2)}$	2147483399	40692

```
      FUNCTION RAND(DUMMY)
C  This is an adapted version of subroutine RANECU written by F. James
C  (Comput. Phys. Commun. 60 (1990) 329-344), which has been modified to
C  give a single random number at each call.
C
C  The 'seeds' ISEED1 and ISEED2 must be initialized in the main program
C  and transferred through the named common block /RSEED/.
C
C  Some compilers incorporate an intrinsic random number generator with
C  the same name (but with different argument lists). To avoid conflict,
C  it is advisable to declare RAND as an external function in all sub-
C  programs that call it.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER*4 (I-N)
      PARAMETER (USCALE=1.0D0/2.147483563D9)
      COMMON/RSEED/ISEED1,ISEED2
C
      I1=ISEED1/53668
      ISEED1=40014*(ISEED1-I1*53668)-I1*12211
      IF(ISEED1.LT.0) ISEED1=ISEED1+2147483563
C
      I2=ISEED2/52774
      ISEED2=40692*(ISEED2-I2*52774)-I2*3791
      IF(ISEED2.LT.0) ISEED2=ISEED2+2147483399
C
      IZ=ISEED1-ISEED2
      IF(IZ.LT.1) IZ=IZ+2147483562
      RAND=IZ*USCALE
C
      RETURN
      END
```

Figure 5.1: FORTRAN code of the pseudo-random number generator RANECU, which is included in the PENELOPE package.

from the well known mathematical basis of congruential generators and the fact that it has successfully passed a number of statistical tests (L'Ecuyer 1988; Knuth 1998; Coddington 1994; Gammel 1998). Moreover, it has been satisfactorily used for many years in a number of Monte Carlo codes without showing any apparent artefact. A version of RANECU (called RAN2) that incorporates an additional shuffling algorithm is supplied with the book Numerical Recipes (Press *et al.* 1997).

RANECU's two MLCGs fulfil the following conditions:

1. m is a large prime number and a is a primitive root modulo m.
2. $a^2 < m$.
3. $(m^{(1)} - 1)/2$ and $(m^{(2)} - 1)/2$ are relatively prime.

Condition 1 is equivalent to requiring that the MLCG attains its maximal period $m-1$, i.e., that all the integer values between (and including) 1 and $m-1$ are produced once before repeating the initial seed. The second condition permits an MLCG to be coded in an efficient and portable way with integers of bit length b such that $m < 2^{b-1}$ by having recourse to the so-called *approximate factoring* method (L'Ecuyer and Côté 1991). This method is used to compute the product of the two integers S_i and a modulo m without overflow hazard (i.e., without exceeding the length of a signed long integer) by taking advantage of the identity

$$(aS_i) \,\mathrm{MOD}\, m = [a\ (S_i \,\mathrm{MOD}\, q) - \lfloor S_i/q \rfloor\ r] \,\mathrm{MOD}\, m\ , \tag{5.4}$$

where $q = \lfloor m/a \rfloor$ and $r = m \,\mathrm{MOD}\, a$ are the quotient and remainder, respectively, of the integer division of m by a. This equation is directly implemented in the RANECU source code shown in Fig. 5.1: for the first MLCG, $q = 53668$ and $r = 12211$; for the second, $q = 52774$ and $r = 3791$; variables `I1` and `I2` contain the quotients $\lfloor S_i/q \rfloor$; and `IZ` is the combined seed, which is converted into a double-precision real value between 0 and 1 multiplying by `USCALE`. Since RANECU uses moduli slightly smaller than 2^{31}, its MLCGs can be coded using integer arithmetic in 32 or more bits. Finally, the third condition ensures that the combination of the two MLCGs produces a generator which also attains its maximal possible period, $(m^{(1)}-1)(m^{(2)}-1)/2$, which is the least common multiple of the individual periods $m^{(1)} - 1$ and $m^{(2)} - 1$. For the moduli of Table 5.1 this yields a total period of $2305842648436451838 \simeq 2.3 \cdot 10^{18}$.

It is a well-known fact that some generators have long-range correlations and, hence, it is not advisable to use a large fraction of the sequence in a single simulation. This, compounded with the increasing computing power available per monetary unit and the widespread use of computer clusters, is bound to render RANECU obsolete in the long run. Presently, it would take of the order of 10^4 years to cycle RANECU on a single-processor computer working at a clock speed of 3 GHz (assuming that a call to the RANECU function takes about 40 clock cycles to complete).

Using the general formula for the combination of MLCGs described by l'Ecuyer (L'Ecuyer 1988), of which Eq. 5.3 is a particular case, it is possible to extend RANECU

with additional MLCGs, provided that all of them meet the three conditions presented above (the third condition must then be applied to any pair of m values). Particularly, the MLCG with multiplier $a^{(3)} = 45742$ and modulus $m^{(3)} = 2147482739$, also studied by l'Ecuyer (L'Ecuyer 1988), can be used to produce an extension of RANECU with a sequence defined by

$$S_i \;=\; (S_i^{(1)} - S_i^{(2)} + S_i^{(3)}) \,\mathrm{MOD}\, (m^{(1)} - 1) \;\; . \tag{5.5}$$

This "extended" RANECU uses three initial seeds and has a period of $\sim 5 \cdot 10^{27}$.

The structural properties of combined MLCGs are much better than those of single MLCGs. To study the lattice structure of the generators, a plot of points with Cartesian coordinates consisting of two (or three) consecutive random values in the interval (0,1) can be employed. It can be calculated that for the two MLCGs in Table 5.1 and the third one proposed above as an extension the number of parallel hyperplanes (straight lines in the 2D case) that contain all these points is 65535 at most (Marsaglia 1968). In contradistinction, RANECU requires of the order of 10^9 lines. As shown in Figs. 5.2 and 5.3, the lattice structure of the MLCGs is apparent in the represented x interval, whereas the points from the extended RANECU (and similarly for the original version) seem to be truly randomly distributed due to the much larger number of hyperplanes present.

5.1.2 Parallel execution with multiplicative congruential generators

Statistically independent sequences of pseudo-random numbers for parallel executions can be obtained from an MLCG (L'Ecuyer and Côté 1991; Mendes and Pereira 2003) by applying the methodology described by L'Ecuyer (1988). As he states, an important property of MLCGs is that any term in the cycle can be obtained without calculating the intermediate values. Indeed, given an initial seed S_0, the term S_i can be found directly using (cf. Eq. 5.1)

$$S_i = (a^i S_0) \,\mathrm{MOD}\, m = [(a^i \,\mathrm{MOD}\, m)\, S_0] \,\mathrm{MOD}\, m \;\; . \tag{5.6}$$

In order to compute $a^i \,\mathrm{MOD}\, \mathrm{m}$, for large values of i, we have adapted to modular arithmetic the right-to-left binary method for exponentiation described by Knuth (1998, page 462). The basic idea behind this method is that a high power of a can be obtained by successively squaring a. For instance, since 6 is written as 110 in binary, a^6 can be expressed (by reading the binary from right to left) as $(a^2)^2 \cdot a^2$, which involves evaluating

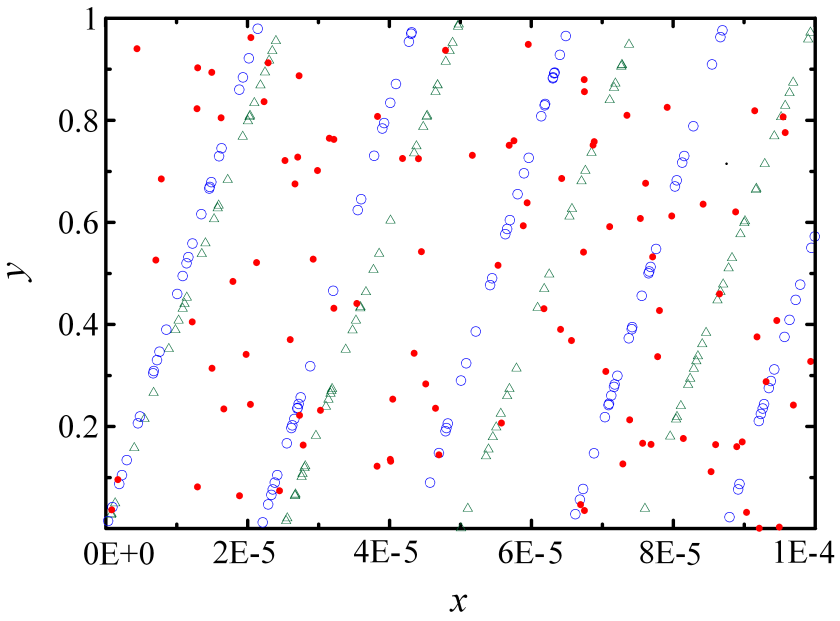


Figure 5.2: Graphical representation of the lattice structure of the extended RANECU generator (dots) and of its MLCG components. Triangles correspond to the first MLCG employed in RANECU (the second MLCG, not shown, produces a similar pattern) and open circles to the third MLCG included in the extended generator (see text).

only three products. Notice that the approximate factoring method described above (see Eq. 5.4) may not be applicable here to compute a^6 (or any other power) because some intermediate factors, a^2 and $(a^2)^2$ in our example, may not fulfil the condition that their squares are less than m, as required by condition 2 in the previous section. To overcome this difficulty and to calculate the product without overflow, we have recourse to the algorithm proposed by L'Ecuyer and Côté (1991). It consists of using the so-called Russian peasant multiplication scheme (basically, halving one factor while doubling the other) until the halved factor complies with the aforesaid condition 2, at which point the approximate factoring method is applied.

For backward jumps (that is, $i < 0$ in Eq. 5.6) we define

$$a^i \,\mathrm{MOD}\, m \equiv \tilde{a}^{(-i)} \,\mathrm{MOD}\, m \tag{5.7}$$

where $\tilde{a}$ is the multiplicative inverse of a modulo m, that is,

$$a\,\tilde{a} \,\mathrm{MOD}\, m = 1\,. \tag{5.8}$$

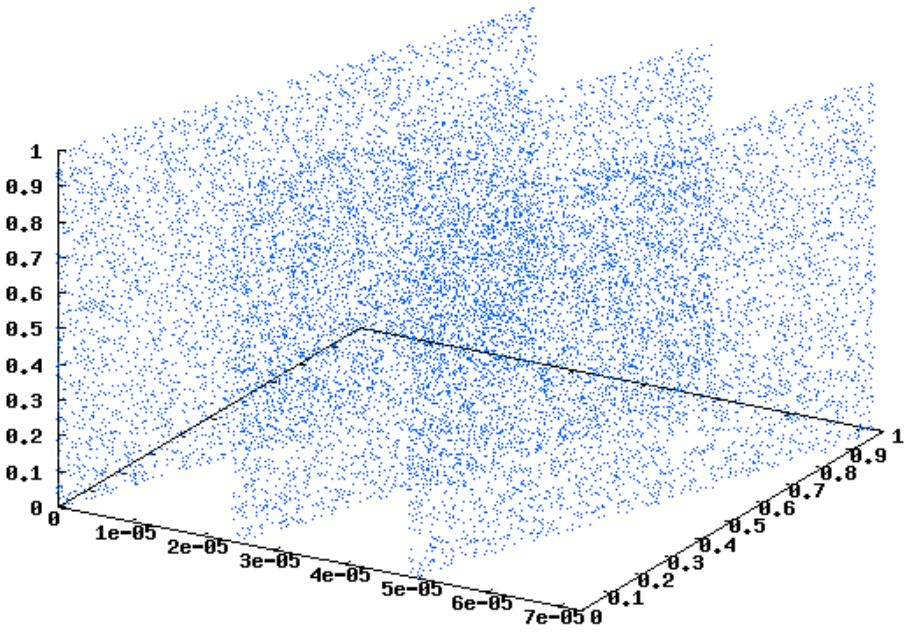


Figure 5.3: Three-dimensional lattice structure of the second MLCG used by RANECU.

The value of $\tilde{a}$ can be evaluated as $\tilde{a} = a^{(m-2)}$ since, when a and m are coprime,

$$a^{(m-1)} \,\mathrm{MOD}\, m = 1 \;, \tag{5.9}$$

an identity known as Fermat's little theorem (see e.g., Knuth 1973). Alternatively, the inverse can be obtained using the so-called extended Euclid's algorithm (Knuth 1973; Knuth 1998). The latter is an adaptation of the algorithm conceived by Euclid circa 300 B.C. to find the greatest common divisor (GCD) of two integer numbers m and a by taking advantage of the fact that, if r is the remainder of the integer division of m by a, then

$$\mathrm{GCD}(m, a) \;=\; \mathrm{GCD}(a, r) \;. \tag{5.10}$$

By iterating the former expression, a division with null remainder is eventually found, yielding the GCD as the last non-null remainder. The extended algorithm uses the quotients and remainders of the intermediate iterations to find integers $\tilde{m}$ and $\tilde{a}$ such that

$$a\,\tilde{a} + m\,\tilde{m} \;=\; \mathrm{GCD}(m, a) \tag{5.11}$$

by recursively expressing each residue as a linear combination of the original a and m values. When m and a are coprime, as is the case with the MLCGs considered here, we

find

$$a\,\tilde{a} + m\,\tilde{m} \;=\; 1\;, \tag{5.12}$$

known as Bézout's identity—despite some authors attribute it to Bachet de Méziriac. Now, since $m \,\mathrm{MOD}\, m = 0$, it follows that $\tilde{a} \,\mathrm{MOD}\, m$ is the sought inverse. We have found that the extended Euclid's algorithm takes less computer time than evaluating $a^{(m-2)}$ and, hence, the former is the method of choice in our routines.

With all these ingredients, each CPU in a parallel calculation can be fed with a seed that initiates a string of pseudo-random numbers that does not overlap other CPU sequences. Indeed, if J is an integer larger than the number of calls to the pseudo-random number generator performed by any of the CPUs during the simulation, then the k-th CPU ($k = 0, 1, \ldots, K-1$) is provided with the sequence $\{S_{Jk+j} \,|\, j = 0, 1, \ldots, J-1\}$. This method is known as *sequence splitting* and has the advantage of using the original generator without alterations. A drawback of this approach are the long-range correlations of the MLCG, which may become particularly relevant if J is related to m, for instance if both are powers of 2.

A different procedure to obtain disjoint sequences is the *jumping* or *leapfrog* technique. This method consists of jumping a fixed distance K along the generator cycle before the next seed is obtained. The resulting subsequence for the k-th CPU is thus $\{S_{Kj+k} \,|\, j = 0, 1, \ldots, J-1\}$. Since, from Eq. 5.6, $S_{(Kj)} = (a^K)^j \, S_0 \,\mathrm{MOD}\, m$, an MLCG can be modified to leapfrog the sequence by merely replacing the multiplier a by a^K or, equivalently, by $a^K \,\mathrm{MOD}\, m$. Nonetheless, the generator with the new multiplier may be cumbersome to code in a portable way because it may not comply with condition 2 mentioned above. Furthermore, it is not clear whether the resulting sequence of pseudo-random numbers is as good as the original since all the tests (e.g., the spectral test) have been performed using the latter. In other words, the correlations between numbers K positions apart in the original sequence are, generally speaking, less well known. Taking these considerations into account, we have opted for the sequence splitting method.

5.2 New parallelisation software

5.2.1 clonEasy

The clonEasy package is a collection of Linux scripts and auxiliary FORTRAN programs that implement Secure Shell-based communication[2] between a "master" computer and a set of "clones". They perform simple operations, such as upload and download files or compile and execute programs, with the aim of running a code that performs a Monte Carlo simulation on all the clones at the same time. The scripts that are provided with clonEasy are briefly explained below:

`clon-setup`	: sets up the environment for clonEasy to work.
`clon-upload`	: uploads files from the master to the clones.
`clon-make`	: executes a compilation script on each clone.
`clon-run`	: uploads the pseudo-random number generator seeds and executes a program on all clones.
`clon-download`	: downloads files from the clones to the master.
`clon-remove`	: removes files and directories from all clones.

Most of these scripts take arguments; when not enough of them are provided, a message is issued to the screen to inform the user. The scripts are written using the syntax of the bash shell and, consequently, they may run under various Unix flavours, although we have only tested them on Linux distributions. The README file distributed with the package gives a detailed description of their use.

Secure Shell is used to execute commands on a remote machine and Secure Copy is used to move files across the net. Thus, any computer on the Internet with a Secure Shell server installed can be used as a node of a virtual computer cluster for parallel calculations. It is not necessary to install any additional software on the clones or the master and the sequential Monte Carlo source code does not need to be modified, except for the fact that the initial random seeds for the pseudo-random number generator are required to be read from an external file. A FORTRAN program included in clonEasy and invoked by the scripts mentioned above prepares the communication with all the clones that are listed in an external ASCII file, named `clon.tab` hereafter. In this file each clone is identified by a name or IP address and it is given a unique nickname that will be used as a prefix for the files downloaded to the master. Additionally, `clon.tab`

[2]For a description of the features and use of an open distribution of Secure Shell, the reader may consult, for instance, the website http://www.openssh.com.

also specifies the compilation command and the seeds for the generator that are to be used for each clone. By default clonEasy expects to find two integer seeds per clone, which may be for example the initialisation values of RANECU, or the "luxury level" and the seed that are needed to initialise RANLUX. If more than two seeds are needed the Monte Carlo code can be adapted so as to use the first introduced number as an index in a list of predefined set of seeds or, alternatively, the clonEasy source code could be adapted conveniently. As it was explained before, since the same code is executed on all the nodes with the same initial conditions, the user should carefully choose these seeds to produce independent sequences in each processor. As an example, the sample file `clon.sample.tab` distributed with the package is displayed in Fig. 5.4.

```
# >>>> CLONES TABLE >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
#   Sample clone table for clonEasy.
#   Copy and modify to fit your configuration.
#   Lines starting with '#' are skipped.

# Clone ID       Nickname        Random seeds       Make file  Comments
# ------------------------------------------------------------------------
joe@myserver.es  fastMachine   918882992  858672133  make_ifc  None
192.168.1.101    myPC_1       2069007070 1309916099  make_g95  Dual CPU
192.168.1.102    myPC_2        944675654 1438406465  make_g95  Dual CPU
# neverWorks.es  devil         149156960  257442270  make_g77  Inactive

[EndOfTable]
```

Figure 5.4: Sample list of clones (`clon.sample.tab`) distributed with clonEasy.

The various processes that run in parallel do not need to communicate with each other. When all of them have finished, their outputs are downloaded to the master where they can be combined to obtain the global result with a reduced statistical uncertainty. The FORTRAN program COMBINE, supplied with clonEasy, can be used to perform this task. Its input file can be adapted to read a wide variety of output file formats. COMBINE looks for some user specified keywords inside the output files and reads the value of the quantity of interest q_k ($k = 0, 1, \ldots, K-1$) produced by each clone. It can also process output files containing a data set, e.g., an energy spectrum. After N_k histories have been simulated by the k-th clone producing the value q_k with variance $\sigma^2(q_k)$, the global average value $\overline{q}$ and its standard deviation $\sigma(\overline{q})$ are obtained using the relations

$$\overline{q} \;=\; \frac{1}{N} \sum_{k=1}^{K-1} N_k \, q_k \tag{5.13}$$

and

$$\sigma(\overline{q}) \;=\; \frac{1}{N}\,\sqrt{\sum_{k=1}^{K-1} N_k^2\;\sigma^2(q_k)}\;, \tag{5.14}$$

where

$$N \;=\; \sum_{k=1}^{K-1} N_k \tag{5.15}$$

is the total number of histories simulated. Notice that Eq. 5.14 is derived from 5.13 by taking advantage of the independence of the various q_k's. The overall relative uncertainty Δ is obtained as

$$\Delta \;=\; 100\;\frac{\sigma(\overline{q})}{\overline{q}}\;. \tag{5.16}$$

The intrinsic and absolute simulation efficiencies are defined by

$$\varepsilon_N \;=\; \frac{1}{N\;\Delta^2} \tag{5.17}$$

and

$$\varepsilon \;=\; \frac{N}{t}\;\varepsilon_N\;, \tag{5.18}$$

respectively, where N/t stands for the total simulation speed, in histories per unit clock time (i.e., real time), which would be achieved if all the clones were running in single-process mode. We have

$$\frac{N}{t} = \sum_{k=1}^{K-1} \frac{N_k}{t_k}\;, \tag{5.19}$$

where t_k is the CPU time (that is, user time) employed by the k-th clone to simulate N_k histories. Note that ε_N depends only on the performance of the simulation algorithm per unit simulated history, regardless of any timing considerations—hence the term "intrinsic".

The absolute simulation efficiency increases approximately linearly with total CPU power, or what is equivalent in the case that all CPUs are identical, with the number of processors. This assertion is quite obvious from the considerations made above, since no time is spent intercommunicating clones or sending information between these and the master, except when required by the user. In consequence, N/t is proportional to the computer power available and, since ε_N is independent of this quantity, Eq. 5.18 reflects the claimed linearity.

5.2.2 seedsMLCG

The seedsMLCG code uses Eq. 5.6 to calculate elements of the sequence of an MLCG that are located a certain distance ahead, or behind, the input seed. This code is intended to provide the set of initial seeds required to implement the sequence splitting technique and, thus, allow the parallelisation of the simulations.

On execution, the user is prompted to introduce the modulus and multiplier of an MLCG and the separation (that is, the number J of intermediate random values) to be jumped between consecutive seeds. Since integer numbers with sign are restricted to 32 bits in most computers, the input separation is limited to the value $2^{31} - 1$. For this reason, the program also allows the user to input the separation as the exponent of a power of 10, in order to efficiently split the sequence of generators with very long periods. Negative distances can be entered to jump the sequence backwards, with the aid of Eq. 5.7. This capability may be useful for debugging purposes (for instance, to reproduce a certain past history). Figure 5.5 shows the output from a seedsMLCG test run.

The FORTRAN 77 source code of seedsMLCG is accompanied by three sample input files containing the parameters that define the two MLCGs employed by RANECU and the MLCG used in the extended version proposed in previous sections. These files are read by the program by redirecting the keyboard input with the '$<$' sign from the system command line. In Table 5.2 a set of 30 seeds calculated with the three sample input files is presented. Each seed starts a disjoint subsequence with 10^{15} pseudo-random numbers. The generation of each seed takes a less than one millisecond on a modern computer.

```
     ************************************************
     **                                            **
     **  CALCULATING FUTURE OR PAST ELEMENTS OF THE  **
     **     SEQUENCE OF A MULTIPLICATIVE LINEAR      **
     **        CONGRUENTIAL GENERATOR (MLCG)         **
     **                                            **
     **  [P. L Ecuyer,  Commun. ACM 31 (1988) p.742] **
     **                                            **
     ************************************************
- Initial integer seed:  S(i) =
      1
- Number of new seeds to be calculated =
     10
- Is the interval to the next seed a power of 10? [0=no/1=yes]
    Yes
- Distance between seeds (negative value for a backward jump):
  j=10**k, k =
     15
- Multiplier of the MLCG: a =
     40014
- Modulus of the MLCG:    m =
     2147483563
- The input generator is the first MLCG from RANECU.
- RESULTS:  11 seeds, separated 10** 15 units:
                1
        918882992
       2069007070
        944675654
        149156960
        360537627
       1446789139
        888673974
           258943
       1434784182
        698429770
```

Figure 5.5: Test run output for seedsMLCG.

Table 5.2: Seeds that start 30 disjoint subsequences separated by 10^{15} numbers, for the two MLCGs in RANECU and for the third MLCG proposed as an extension in Eq. 5.5.

RANECU seed 1	RANECU seed 2	Extension seed
1	1	1
918882992	858672133	35977198
2069007070	1309916099	62205517
944675654	1438406465	392697167
149156960	257442270	820143318
360537627	133123709	609065445
1446789139	1248992867	917376822
888673974	2014364429	382392929
258943	664687714	1007129025
1434784182	1598489021	804921119
698429770	1978724894	1737229562
1590179518	797341276	755191382
658812725	1829379549	1732001184
1141813962	1863091192	672741026
970004038	827424326	1888778569
156172654	2028805919	1036204498
479486796	238097920	979439700
1926622811	1383430112	1507290784
1024191502	938910203	145727789
914109165	803254235	275064188
996688518	2093478795	206594468
1343291889	1203423504	1065781201
1783116228	282077422	1592352438
552569869	1861318300	1246750375
839738220	1673889598	1459353822
1436544680	196014388	1665060735
1590006138	1754830438	1909717237
88748046	1574175136	170682952
940315134	1848214072	1826881820
1676463528	1603536943	322770548

5.3 Computer cluster Argos

The clonEasy package and the seedsMLCG code were employed in most of the studies presented in this thesis. In order to perform the simulations shown in sections 6.1, 6.2 and 7.4, the clonEasy input file was adapted to the Linux-based heterogeneous cluster Argos (argos.upc.es), located at the Institute of Energy Technologies (Universitat Politècnica de Catalunya; Avinguda Diagonal 647, 08028 Barcelona, Spain).

At the time this thesis was concluded the Argos cluster was composed of 35 computers, with a total of 74 processors[3], interconnected with a 1 Gbps Ethernet. Each computer was accessed through the Secure Shell protocol using the computer IP address inside the private local area network. A short description of the computers found in the cluster is provided in Table 5.3.

The status of the Argos cluster and the utilisation of its resources are controlled by the distributed monitoring system Ganglia (`http://ganglia.sourceforge.net`) and the command line utility Jmon (`http://www.hlrn.de/doc/jmon`). The information collected by the Ganglia server is available in real-time at the public website `http://argos.upc.es/ganglia`. This web interface also provides comprehensive statistics of the cluster usage during the last year.

Table 5.3: Description of the computers in the Argos cluster. The data provided for each computer are: number of CPUs, processor brand and model, clock speed and available RAM memory.

Description	Processors	Speed	RAM memory
Main server[3] (argos.upc.es)	4 × Intel Xeon™ 5130	2.0 GHz	4.0 GBytes
Legacy web server[3] (sparc)	4 × TI UltraSparc™ II	0.25 GHz	0.25 GBytes
Nodes 1 to 10	1 × AMD Athlon™ XP	1.5 GHz	0.5 GBytes
Nodes 11 to 16	1 × Intel Pentium™ 4	3.0 GHz	0.5 GBytes
Nodes 17 to 22	2 × Intel Pentium™ 4	3.4 GHz	0.5 GBytes
Nodes 23 to 33	4 × Intel Xeon™	3.0 GHz	2.0 GBytes

[3]The main server and the web server were not used in the calculations.

Part III

Applications

6 Radiotherapy and dosimetry

In this chapter the simulation tools developed in chapters 3, penEasy, and 5, clonEasy, are applied to selected problems of radiotherapy and dosimetry. In section 6.1, the simulation of the response of a metal-oxide-semiconductor field-effect transistor (MOSFET) detector employed in entrance *in vivo* dosimetry during radiotherapy treatments is presented. This work was presented at the *XV Congreso Nacional de Física Médica* (June 28, 2005; Pamplona, Spain), the biannual conference of the Spanish Medical Physics Society. An extended version of the presented work was published by Panettieri *et al.* (2007). Section 6.2 presents the simulations of a teletherapy and a brachytherapy treatments. These simulations were intended to test penVox, the voxelised geometry package employed by penEasy. Some preliminary results of these two studies were presented at the *XIV Congreso Nacional de Física Médica* (June 17, 2003; Vigo, Spain), and at the *IV Jornades de Recerca en Enginyeria Biomèdica* (June 9, 2004; Barcelona,

Spain), organised by the CREBEC (Catalan Biomedical Engineering Research Center). Finally, section 6.3 presents the simulation of an internal dosimetry procedure performed with a voxelised knee phantom and two semiconductor detectors. A benchmark of the penEasy data with experimental measurements and with the Monte Carlo codes EGS 4 and MCNPX 2.5 is provided. This simulation was performed in the framework of an international comparison on Monte Carlo modelling proposed by the European Union Coordinated Network for Radiation Dosimetry (Gómez-Ros *et al.* 2007). A detailed description of this study and the results of the intercomparison, including our simulation, has recently been published by Gómez-Ros *et al.* (2008).

6.1 MOSFET detectors for entrance dosimetry

An important application of Monte Carlo simulation is the study of radiation detectors and dosimeters. The simulation of ionisation chambers, for instance, has played an important role because an accurate characterisation of these devices is essential to assess the reliability of dosimetric measurements (Andreo 1991; Nahum 1996).

The simulation of clinical dosimeters is far from trivial due to two reasons. First, in most applications the radiation field is much larger than the sensitive region and, therefore, only a small fraction of the simulated histories contributes to the deposited energy. This gives rise to very inefficient simulations (i.e., long computing times), a problem that can be tackled by using variance reduction techniques and parallelising the execution. Second, dosimeters may have an optically thin sensitive volume, which implies that the number of interactions per passing particle is also small. In this situation the so-called interface artifacts become prominent, thus challenging the condensed history algorithm employed in virtually all general-purpose electron Monte Carlo codes (see section 2.2). A careful selection of the transport parameters inside and in the close vicinity of the sensitive volume is required to limit the influence of these effects on the results. For instance, the configuration of EGSnrc and PENELOPE for the simulation of ionisation chambers has been discussed in detail by Kawrakow (2000a) and by Sempau and Andreo (2006), respectively.

In this section the simulation of radiation detectors based on the dual bias dual MOSFET technology is presented. A detailed description of these devices was provided by Soubra *et al.* (1994). MOSFETs can be used for entrance *in vivo* dosimetry in radiotherapy treatments. The small size, isotropic response and low dependence on temperature changes allow the placement of these detectors on the patient skin and in-

side the treatment field without introducing a significant perturbation. Furthermore, the immediate read-out capability can provide a real-time validation of the delivered treatment, which is interesting for quality assurance purposes.

In order to measure entrance dose the dosimeter has to be covered with a build-up cap to guarantee that the sensitive region is under electronic equilibrium conditions. For high-energy photon beams, the response under different build-up caps was experimentally studied by Jornet *et al.* (2004) at the *Hospital de la Santa Creu i Sant Pau* (Barcelona, Spain). An intriguing result of their work was that the response of the MOSFET detector depended on the cap composition and, in particular, that the response under a metallic cap was significantly higher than under water-equivalent caps. The purpose of the simulations presented in this section was to reproduce the experimental results and explain the observed overresponse under the metallic cap with the sensor exposed to a 18 MV photon beam.

The preliminary work presented here was extended and published by Panettieri *et al.* (2007). The main improvements introduced in the mentioned paper are the use of: (i) more aggressive variance reduction techniques; (ii) a more realistic description of the geometry that was provided by the manufacturer; (iii) several photon sources, namely, monoenergetic beams from 2 to 10 MeV, a ^{60}Co machine beam, and 6 and 18 MV beams generated by a linear accelerator; and (iv) experimental measurements obtained with thermoluminescent detectors.

6.1.1 MOSFET geometry

A high-sensitivity MOSFET model TN-1002RD manufactured by Thomson and Nielsen Electronics, Ltd. (Ottawa, Canada) was modelled with the PENELOPE quadric geometry package PENGEOM (see section 2.3). The geometry was based on the simplified description given by Wang *et al.* (2004), which ignores the dual transistor internal structure and the electric contacts. The MOSFET sensitive region was defined as a 1-μm-thick SiO_2 box with an area of 0.2×0.2 mm^2, centered on the top of a $1.0\times1.0\times0.525$ mm^3 silicon substrate. The silicon dice was covered with a semi-ellipsoid epoxy bulb (1.0 mm thick, 2.0 mm wide) and attached to a kapton cable (0.25 mm thick, 2.0 mm wide), as shown in Fig. 6.1.

Three different build-up caps were defined. The simplest cap was a $3\times3\times3$ cm^3 cube made of water-equivalent bolus material (density 1.02 g/cm^3). The second cap consisted of a 2 cm radius hemisphere made of polystyrene (density 1.03 g/cm^3). Finally, the third case reproduced a metallic cap from a P30 diode detector. This cap was made of a

tungsten shell (external diameter 10.0 mm and thickness 1.6 mm, density 17.0 g/cm^3) covered by a thin plastic layer (thickness 0.5 mm, density 1.03 g/cm^3). The empty space between the tungsten shell and the detector—a MOSFET detector is smaller than a diode—was filled with bolus. A cross section of the detector inside the metallic cap is shown in Fig. 6.1. A $10\times10\times10$ cm^3 water phantom was defined downstream the MOSFET.

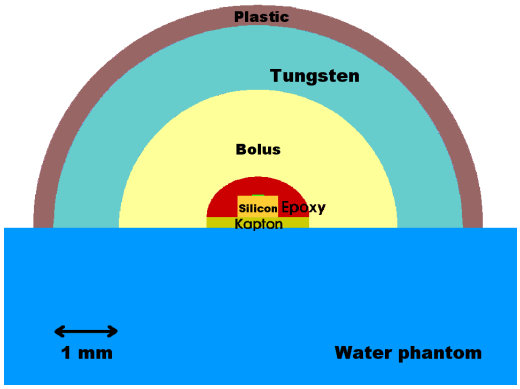


Figure 6.1: MOSFET detector under a P30 metallic build-up cap. See text for details.

6.1.2 PENELOPE configuration

The difficulty of the MOSFET simulation stems from the small size of its sensitive region: just $4{\cdot}10^{-5}$ mm^3. PENELOPE can accurately reproduce the response of these devices thanks to its ability to perform detailed, that is, collision-by-collision, transport of charged particles, as explained in section 2.3. The detailed simulation completely eliminates the aforementioned interface artefacts, although its application in the whole geometry would be extremely time consuming and, thus, impractical. Instead, it was only used inside the sensitive SiO_2 layer. Mixed simulation with very conservative transport parameters was employed in two small volumes of epoxy and silicon (the *skin*) surrounding the sensitive region. Larger step lengths were allowed in regions located far from the detector. The transport parameters for the P30 case are given in Table 6.1.

To speed up the simulation, two variance reduction techniques were implemented, namely, interaction forcing and range rejection. Interaction forcing artificially in-

Table 6.1: PENELOPE transport parameters used in the simulation of a MOSFET detector under a P30 cap. The electron, photon and positron absorption energies (E_{abs}) are given in eV; the maximum step length (`dsmax`) in cm. Note that detailed simulation was used in the SiO_2 layer.

$E_{abs}(e^-)$	E_{abs}(ph)	$E_{abs}(e^+)$	C1	C2	WCC	WCR	dsmax	Description
1.0e2	1.0e2	1.0e2	0.00	0.00	0.0e0	0.0e0	—	SiO_2
1.0e2	1.0e2	1.0e2	0.02	0.02	1.0e2	1.0e2	3.5e-3	Silicon skin
1.0e4	1.0e3	1.0e4	0.03	0.03	1.0e4	1.0e3	1.0e-3	Silicon cube
1.0e2	1.0e2	1.0e2	0.02	0.02	1.0e2	1.0e2	3.5e-3	Epoxy skin
1.0e4	1.0e4	1.0e4	0.03	0.03	1.0e5	1.0e3	5.0e-2	Bolus
1.0e4	1.0e4	1.0e4	0.03	0.03	1.0e5	1.0e3	5.0e-3	Epoxy
1.0e4	1.0e4	1.0e4	0.05	0.05	1.0e5	1.0e3	3.0e-3	Kapton
1.0e4	1.0e4	1.0e4	0.03	0.03	1.0e5	1.0e3	2.0e-2	Tungsten
1.0e6	1.0e5	1.0e6	0.03	0.03	1.0e6	1.0e3	5.0e-3	Plastic
5.0e4	1.0e4	5.0e4	0.05	0.05	5.0e4	5.0e3	1.0e30	Water

creased the photon interaction probability inside the skin to produce, on average, one virtual interaction per crossing photon. To keep the simulation unbiased, the appropriate statistical weight was assigned to the generated secondary particles. Range rejection was applied to electrons and positrons that had a very low probability of reaching the sensitive region. At each electron (positron) step, the range (within the continuous slowing down approximation) was read from a table kept in memory by PENELOPE. The particle was discarded whenever the distance to the silicon oxide was more than 1.33 times its range, which was estimated in the MOSFET lowest density material (epoxy) and using a tabulated energy above the actual value to avoid time consuming interpolations. As a result the particle range was overestimated, which reduced the chance of removing particles that could contribute to the measurement. Range rejection produced a significant acceleration of the simulation because most secondary electrons and positrons were created in the cap or the water phantom, quite far from the MOSFET sensitive region.

The evaluation of MOSFET responses was the first application that made extensive use of the clonEasy parallelisation package described in chapter 5. The simulations were executed in the Argos cluster (see section 5.3).

6.1.3 Response using different build-up caps

This section presents the preliminary results of the MOSFET simulations. The dosimeters were irradiated with a simplified model of an 18 MV photon beam produced by a linear accelerator. The model consisted in a cone beam, with 1° aperture, located 1 m above the detector. The energy of the emitted photons was sampled from a realistic energy spectrum previously simulated by Duch (1999). This source model produced a circular photon field with a diameter of 3.5 cm at the phantom plane. We opted not to use a phase-space file due to the huge amount of particles that were required in the simulation.

The response of the detector was estimated by scoring the absorbed dose inside the sensitive volume. This approach assumed that the response was proportional to the deposited energy, which is a good approximation for dual bias dual MOSFET dosimeters (Soubra *et al.* 1994). Table 6.2 shows the simulated responses for different build-up caps.

Table 6.2: Simulated response of the MOSFET detector under different build-up caps. The statistical uncertainty is given at 2σ level.

Build-up cap	Response [eV/hist]	Uncertainty	Histories	CPU time [h]
P30	$9.88 \cdot 10^{-4}$	3.4%	$1.43 \cdot 10^{10}$	214
Bolus	$6.31 \cdot 10^{-4}$	4.6%	$1.12 \cdot 10^{10}$	184
Polystyrene	$5.52 \cdot 10^{-4}$	5.1%	$1.07 \cdot 10^{10}$	67

These results can be compared with the experimental data obtained by Jornet *et al.* (2004). The responses of the P30 and the polystyrene caps normalised to the bolus case are given in Table 6.3. Taking into account the statistical uncertainty and the numerous simplifications assumed in the simulation (particularly in the MOSFET geometry and the source model), the experimental and simulated Polystyrene/Bolus ratios show a reasonable agreement. For the metallic (P30) cap, both the simulation and the experiment show an increase in the response with respect to the bolus, although the agreement in this case is not so good. A more accurate geometry description, a realistic radiation source, and the simulation of more particle tracks allowed us to improve the agreement considerably, well within the associated uncertainty (see Panettieri *et al.* 2007 for details).

Table 6.3: Experimental and simulated MOSFET responses under the P30 and the polystyrene build-up caps normalised to the response under the 3 cm bolus. The statistical uncertainty is indicated in parenthesis at 2σ. The differences, relative to the experimental values, are also provided in percent.

Response ratio	Experimental	Simulated	Rel. diff.
P30/Bolus	1.395 ($\pm$ 3.8 %)	1.566 ($\pm$ 5.8 %)	-12%
Polystyrene/Bolus	0.948 ($\pm$ 3.4 %)	0.874 ($\pm$ 6.9 %)	7.8%

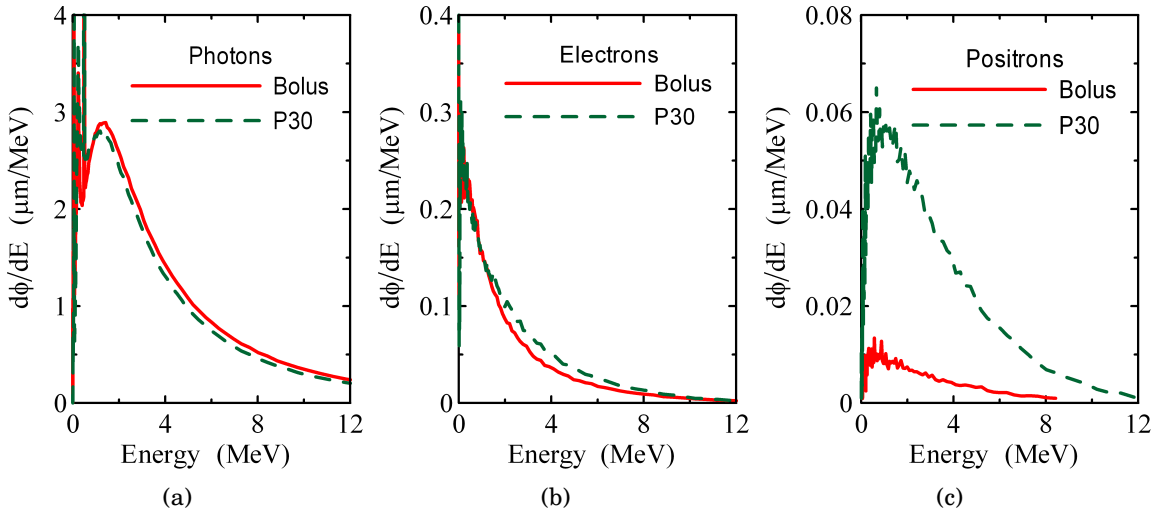


Figure 6.2: Photon (a), electron (b) and positron (c) average fluence spectra inside the MOSFET sensitive region under a bolus cap (solid line) and a metallic P30 cap (dashed line).

In order to study the observed increase in energy deposition under the metallic cap, the particle fluence spectra, differential in energy, was tallied (see chapter 3 for details on the penEasy fluence tally). The photon, electron and positron spectra for the bolus and the P30 cases are plotted in Fig. 6.2.

These spectra suggested that positrons could play a significant role in the observed overresponse. To analyse the contribution of positron transport, the P30 and the bolus simulations were repeated with effectively infinite absorption energy for these particles. Electrons created in pair production events and annihilation photons were followed as usual. The resulting responses are given in Table 6.4.

The data in Tables 6.2 and 6.4 revealed that the ratio P30/bolus decreases from 1.57 to 1.19 when positron transport was disconnected. These results confirm that positrons

Table 6.4: Simulated response of the MOSFET detector, under the P30 and bolus caps, disregarding positron transport.

Build-up cap	Response [eV/hist]	Uncertainty	Histories	CPU time [h]
P30	$6.86 \cdot 10^{-4}$	10%	$2.14 \cdot 10^{9}$	17.4
Bolus	$5.77 \cdot 10^{-4}$	9.7%	$2.14 \cdot 10^{9}$	16.2

have a substantial contribution to the observed effect. More positrons are created in the tungsten layer due to the fact that its pair production cross section is larger than that of elements with lower atomic number, such as those found in typical water equivalent materials (e.g., bolus).

6.2 Dose distributions in radiotherapy treatments

Computer simulations are routinely used to plan and optimise radiotherapy treatments. The purpose of a treatment planning system is to assist the radiation physicist to ensure that the defined treatment volume receives the prescribed dose while the dose to the surrounding healthy tissue is minimised.

The codes that are most commonly used today in the clinic for treatment planning are based on pencil beam and convolution/superposition algorithms (Hogstrom *et al.* 1981; Ahnesjö 1989; Sterpin *et al.* 2007).

Several studies have shown that, in certain situations, these semi-analytic approximations may introduce significant errors in the calculation of dose distributions, especially in inhomogeneous regions (Knöös *et al.* 1995; Krieger and Sauer 2005) or when there is a lack of electronic equilibrium (Martens *et al.* 2002; Carrasco *et al.* 2004; Knöös *et al.* 2006). However, these algorithms are still used because they can compute the treatment dose distribution in the time frame required in the clinic, i.e., a few minutes.

Recently, a new generation of simulation codes based on accelerated Monte Carlo algorithms have been developed and it is expected that they will substitute the semi-analytic codes in the near future (Reynaert *et al.* 2007; Chetty *et al.* 2007). Some of the Monte Carlo codes that are capable of computing the dose distribution produced by a real radiotherapy treatment in a few minutes are Macro Monte Carlo (MMC, Neuenschwander and Born 1992), Super Monte Carlo (SMC, Keall and Hoban 1996), PEREGRINE (Hartmann-Siantar *et al.* 1995), Voxel Monte Carlo (VMC++, Kawrakow 2000b), Dose Planning Method (DPM, Sempau *et al.* 2000), Monte Carlo Dose Engine

(MCDE, Reynaert *et al.* 2004), and ORANGE (van der Zee *et al.* 2005).

General-purpose Monte Carlo codes—such as EGS, FLUKA, Geant, MCNP, or PENELOPE—are, in principle, more accurate than the semi-analytic or accelerated codes, but they are too slow to be routinely used in the clinic. In spite of this fact, these Monte Carlo codes are still useful to validate the accuracy of the commercial planning systems and as a source of reference data. In exceptional situations they can also be employed to improve the planning of treatments in situations where the other codes are not reliable (e.g., near metallic implants).

In the following subsections we present the simulation of two radiotherapy treatments with penVox, the voxelised geometry package for PENELOPE described in chapter 3. The two studied cases are: a teletherapy treatment using an electron beam from a linear particle accelerator; and a brachytherapy treatment with a ^{192}Ir high dose-rate (HDR) source. The voxelised anthropomorphic phantom used in these simulations is described below. The aim of these simulations was to demonstrate that the presented geometric subroutines can successfully handle a voxelised phantom, thus making feasible the use of PENELOPE to simulate realistic radiotherapy treatments.

6.2.1 Voxelised anthropomorphic phantom

A voxelised phantom of the human head and neck region was created segmenting a CT scan of an anthropomorphic Alderson-Rando male phantom[1] (Alderson Research Laboratories Inc., Long Island City, New York, USA) described in detail by Alderson *et al.* (1962) and in ICRU Report 48 (1992). The CT voxel size was $0.58\times0.58\times3.0$ mm^3 and the original number of voxels in the x, y and z directions was $512\times512\times100$, respectively. In order to optimise the use of computer memory the voxel matrix was reduced to $300\times425\times70$ removing unnecessary air layers surrounding the head.

The CT segmentation, that is, the mapping from CT Hounsfield units (HU) to media composition, was performed using the generic conversion table provided by the TMS-Helax planning system (Helax AB, Uppsala, Sweden). CT units are defined as

$$\mathrm{HU} = 1000\left(\frac{\mu}{\mu_{\mathrm{water}}} - 1\right) , \tag{6.1}$$

where μ is the average attenuation coefficient of the voxel. This conversion table, reproduced in Table 6.5, is based on the relations between HU and electron density given by Knöös *et al.* (1985). The table defines an interval of HU values and a nominal den-

[1]The CT scan was obtained by Dr. Alberto Sánchez-Reyes at the *Hospital Clínic de Barcelona* (Spain).

sity for each body tissue. The conversion from HU to material density was performed defining linear functions between the nominal densities of consecutive materials.

Table 6.5: Generic mapping from CT Hounsfield units to media composition provided by the TMS-Helax treatment planning system.

Hounsfield units	Density (g/cm^3)	Material
$[-1000, -481]$	0.00121	Air
$[-480, -97]$	0.500	Lung
$[-96, 47]$	0.950	Adipose
$[48, 127]$	1.050	Muscle
$[128, 527]$	1.100	Cartilage
$[528, 975]$	1.334	Cartilage 67%, Bone 33%
$[976, 1487]$	1.603	Cartilage 33%, Bone 67%
$[1488, 1823]$	1.850	Bone
$[1824, 2223]$	2.100	Bone 2
$[2224, 2639]$	2.400	Bone 50%, Aluminum 50%
$[2640, 2831]$	2.700	Aluminum
$[2832, 3000]$	2.830	Aluminum 2

The simple approach for CT segmentation that was used to create the phantom, equivalent to a thresholding segmentation, is not accurate and may introduce significant artifacts in the voxel composition (Cozzi *et al.* 1998; Reynaert *et al.* 2007; Chetty *et al.* 2007). One of the limitations of this segmentation method is that it does not use any *a priori* information of the tissue distribution and, therefore, the voxel composition may be mis-assigned due to partial volume effects (e.g., a voxel containing muscular tissue and bone may be erroneously mapped as cartilage). Apart from this, the fact that the conversion table was not adapted to the particular scanner used to obtain the initial phantom CT may produce a systematic bias in the mapping from HU to material composition. In spite of these shortcomings, the simple mapping was used to facilitate the comparison of the Monte Carlo and the TMS-Helax simulations. This method is also used in other general-purpose codes that can handle voxelised geometries.

It has been shown that mis-assignment of media composition can produce significant errors in the dose distributions calculated by treatment planning system (Verhaegen and Devic 2005). Therefore, a more sophisticated segmentation method should be used in case the simulations are intended for clinical use or to perform quantitative comparisons with experimental measurements (Schneider *et al.* 2000; Reynaert *et al.* 2007).

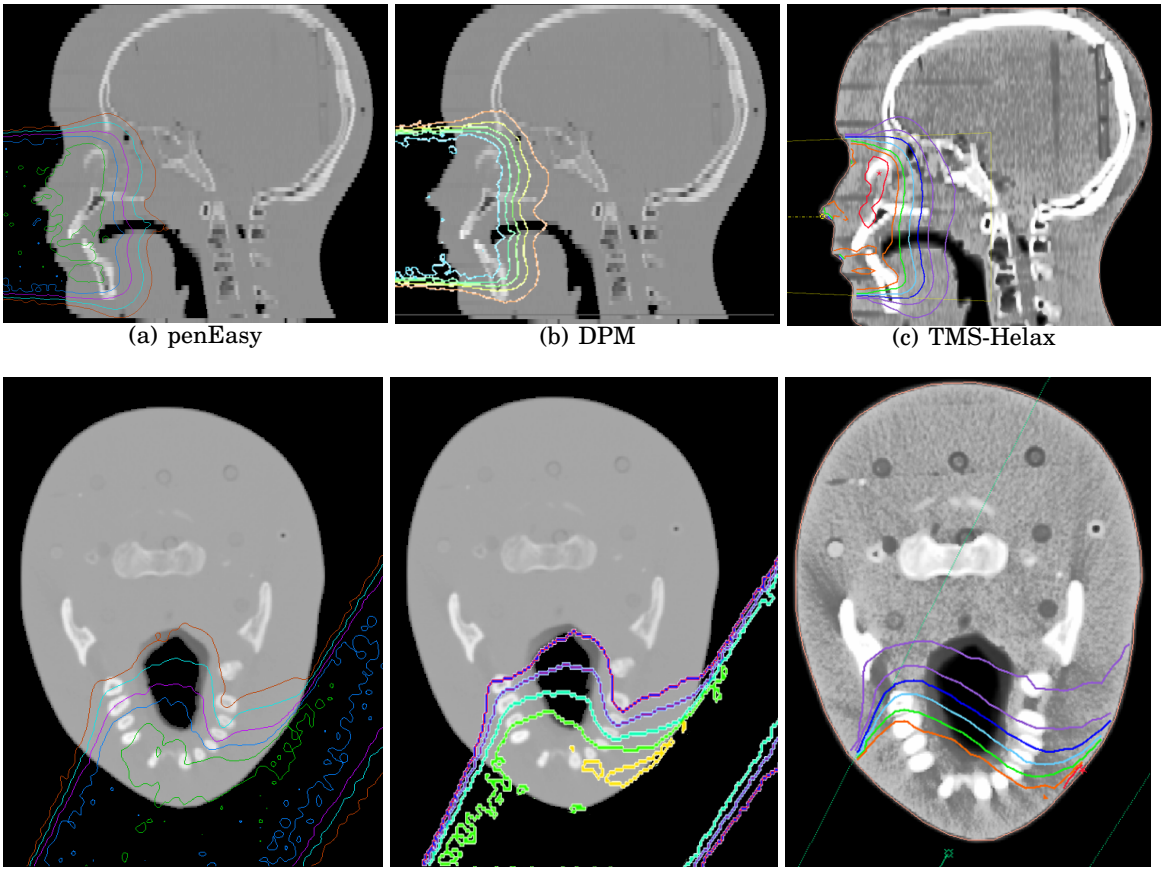
In the presented studies, the accuracy of the segmentation process was not relevant because the purpose of the simulations was just to test the new geometric subroutines and not to validate the phantom anatomy.

6.2.2 Teletherapy treatment

A teletherapy, or external beam radiotherapy, treatment was simulated with penVox and the phantom described in the previous section. The radiation source employed in this case was a 10×10 cm^2 electron beam with a nominal energy of 12 MeV, impinging on the patient head with an inclination of 30 degrees, at the level of the nose. The beam was created simulating a linear accelerator Mevatron KDS (Siemens AG, Munich, Germany) with PENELOPE, as explained by Sempau *et al.* (2001). The beam was stored in a phase-space file (PSF) containing the position, movement direction and energy of $2.5{\cdot}10^6$ particles. The PSF was read by penEasy's PSF source routine (see section 3.1.1), with a splitting factor of 100 (i.e., each particle was cloned 100 times with a statistical weight of 1/100).

The isodose curves simulated with the combination of penEasy and penVox are graphically represented in Fig. 6.3 (a) and (d). This radiotherapy treatment had been previously simulated by Sempau *et al.* (2003) with the accelerated Monte Carlo code DPM and the commercial planning system TMS from Helax (Helax AB, Uppsala, Sweden). The isodoses calculated with DPM and TMS-Helax are also shown in Fig. 6.3. Due to the reduced number of particles in the PSF, and the small voxel size, the dose distribution maps calculated by penEasy and DPM had significant uncertainties. In spite of the fact that the particles were splitted 100 times, the PSF latent variance still limited the accuracy of the simulation (see Sempau *et al.* 2001, Appendix B). For this reason, the presented isodose curves were obtained combining 5×5 adjacent voxels on the xy planes.

This simulation demonstrated that penEasy is capable of simulating a radiotherapy treatment with an anthropomorphic phantom created from a CT scan. This feature can be used to compare PENELOPE with the accelerated codes used in the clinic. In the presented case, we found that the isodoses calculated by DPM were in better agreement with PENELOPE than those calculated by the semi-analytical code TMS.



(a) penEasy (b) DPM (c) TMS-Helax

(d) penEasy (e) DPM (f) TMS-Helax

Figure 6.3: Radiotherapy treatment simulated with penEasy, TMS-Helax and DPM. Figures (a), (b) and (c) provide the isodoses obtained for the sagittal plane X=118; (d), (e) and (f) for the coronal plane corresponding to Z=48. Arbitrary isodoses are represented.

6.2.3 Brachytherapy treatment

PenVox and the voxelised phantom described in section 6.2.1 were also used to simulate a fictitious brachytherapy treatment. A high dose-rate ^{192}Ir source—a simplified version of the microSelectron HDR source (Nucletron B.V., Veenendaal, The Netherlands)—was modeled with quadric surfaces. As shown in Fig. 6.4, the source was composed of an iridium cylinder (diameter 0.65 mm and length 3.60 mm) enclosed inside a stainless steel encapsulation (diameter 0.90 mm and length 4.50 mm); a piece of the steel cable that is used to deploy the source was also simulated (diameter 0.70 mm).

To reproduce a brachytherapy treatment, the source was vertically inserted inside the throat of the voxelised phantom. The source was located inside an air region, in contact

with the esophagus wall and close to the spine. This location was chosen to study the effects of material heterogeneities (air, muscle and bone) on the isodose curves. The relevance of this study comes from the fact that most commercial brachytherapy treatment planning systems are based on Monte Carlo simulations in homogeneous water phantoms and disregard the presence of inhomogeneities (Nath *et al.* 1995; Anagnostopoulos *et al.* 2004). For the sake of simplicity, the ^{192}Ir emission espectrum was reduced to a single 340 keV gamma.

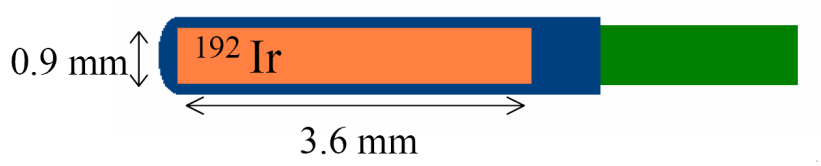
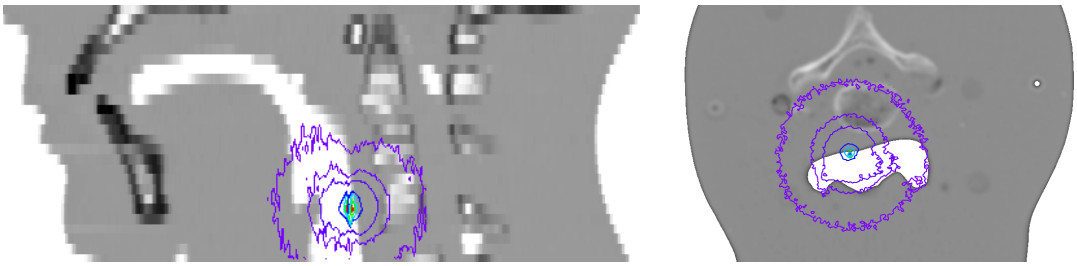


Figure 6.4: Quadric geometry of a high dose-rate ^{192}Ir brachytherapy source (simplified version of the source model microSelectron HDR from Nucletron).

The presented study provides an example of the convenience of combining quadric surfaces and voxels. Since the brachytherapy source is smaller than a voxel, it would have been impossible to represent it using the phantom voxel size. Reducing the voxels to a size capable of representing the internal source structure would require an unaffordable amount of computer memory. By using quadrics surfaces to describe the source in detail and voxels to model patient anatomy, we benefit from the best features of each geometric model.

The simulated dose distributions are shown in Fig. 6.5. An anisotropy in the dose distribution can be clearly seen in the sagittal view (Fig. 6.5 (a)). The anisotropy is caused by the cylindrical shape of the radioactive source and by the presence of the steel cable. The 3% isodose (with respect to the maximum voxel dose) is located approximately at 1.5 cm from the source. Therefore, most of the dose is delivered within a short distance, which is a desirable feature in brachytherapy treatments. As expected, the isodoses are affected by the presence of different surrounding materials. However, the perturbation is small and may not have clinical relevance in the simulated treatment, as suggested by Anagnostopoulos *et al.* (2004). This simulation was executed in parallel in 4 computers from the Argos cluster (AMD Athlon™ XP 1800+ processors at 1533 MHz and 1 GByte RAM; see section 5.3 for more details) using the clonEasy package (see chapter 5). The total number of primary particles was $4{\cdot}10^7$ and the overall CPU time was

approximately 55 hours (∼200 histories/second per computer).



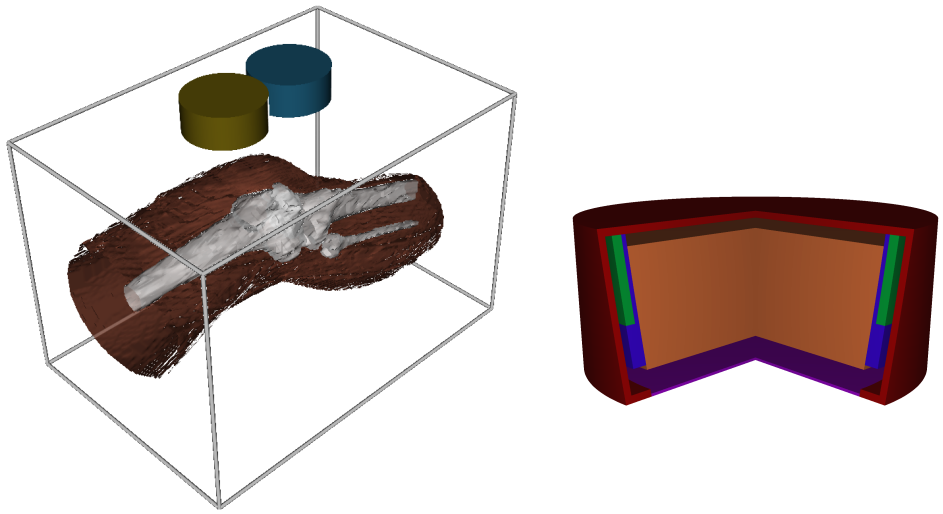
(a) (b)

Figure 6.5: Simulation of a brachytherapy treatment using a ^{192}Ir brachytherapy source: (a) sagittal view of the isodose curves and the corresponding CT slice (X=118, the voxel material is shown); (b) axial view of the isodoses and the CT slice (Z=64, the voxel density is shown). In both cases, the four most external isodoses correspond to a 0.5%, 1.5%, 3% and 15% of the voxel dose maximum, respectively. The axis show the voxel number.

The work presented in this section shows that penEasy and penVox can be useful to assess brachytherapy treatments. Ongoing work in collaboration with the *Hospital Universitario Puerta del Mar* (Cádiz, Spain) aims at applying these tools in a realistic setup.

6.3 Internal dosimetry with Germanium detectors

The penEasy system with quadric geometries and voxels, i.e., using penVox, has been benchmarked with other Monte Carlo codes and with experimental measurements in an international comparison on Monte Carlo modelling organised by the European Union Coordinated Network for Radiation Dosimetry (CONRAD, Gómez-Ros *et al.* 2007). One of the cases proposed in the intercomparison was a measurement of the gamma radiation field produced by an homogeneous distribution of ^{241}Am in the osseous tissue of a knee. This problem required the superposition of quadric objects (two germanium detectors) and voxels (a segmented CT of a knee phantom). Figure 6.6 shows a 3D rendering of the simulation setting and the detector inner structure. The detectors were located inside the voxels bounding box, thus masking or partly overlapping some air voxels.



(a) (b)

Figure 6.6: Volume rendering of the full simulation geometry for the CONRAD intercomparison experiment (a), and inner structure of the Ge detectors described with quadric surfaces (b). The detectors are inside the voxels bounding box, which is shown as a parallelepiped in (a). See text for details.

In this study the BIGS source from penEasy (see section 3.1.1) was configured to emit, from all voxels containing bone, photons with the energy spectrum of americium and with a uniform emission probability per unit volume. The pulse height spectrum tally (see section 3.1.2) served to obtain the energy distribution of pulses in the sensitive region of both Ge detectors. This spectrum was subsequently convolved with a Gaussian distribution to account for the full width at half maximum (FWHM) experimentally obtained for this detection system.

The doses absorbed in the voxels were also scored with the corresponding penEasy tally. A sample 2D slice of the dose distribution map is displayed in Fig. 6.7. It can be observed that the energy deposited in the bone voxels, which contain the radioactive material, is approximately four times higher than that in the muscle voxels.

The experimental and the penEasy pulse height spectra, in units of counts per second per kilobecquerel (cps/kBq), are plotted in Fig. 6.8 (a). Figure 6.8 (b) presents a comparison of our result with equivalent simulations performed with the Monte Carlo codes EGS 4 and MCNPX 2.5 (data from the CONRAD report, Gómez-Ros *et al.* 2008). The results show that penEasy, and the other codes, can correctly reproduce the shape of the main peaks of the pulse height spectrum. The differences observed in the low energy

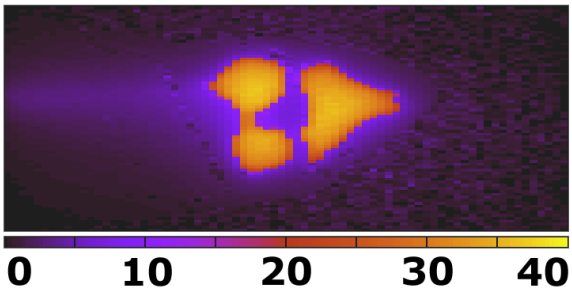


Figure 6.7: Dose distribution map tallied by penEasy inside the voxelised knee phantom, which is composed of bone, muscle and air voxels. The scale is given in eV/g per primary history.

region could be caused by scattering radiation coming from parts of the experimental setting that were not included in the simulation (such as the cryostat located above the detectors and the phantom support table) or from an inaccurate description of the phantom materials. The shape of the peaks in this region is also related to the electron transport models considered by the different codes and, according to Ménard (2004), to the modelling of the Doppler broadening effect for inelastic photon interactions.

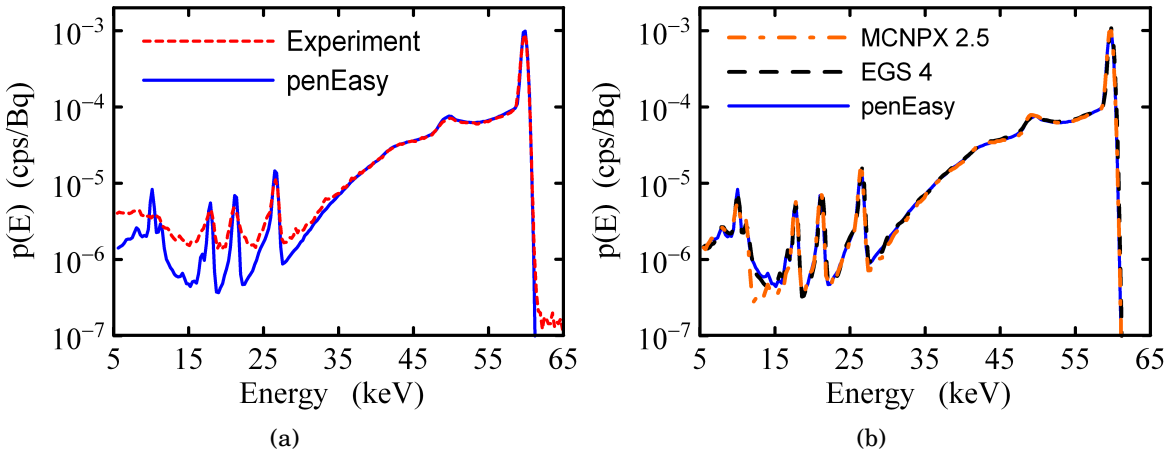


Figure 6.8: Energy deposition pulse height spectra measured inside the sensitive regions of the two Ge detectors for the ^{241}Am source in a knee phantom: (a) penEasy and experimental data; (b) comparison of penEasy, EGS 4 and MCNPX 2.5. The average statistical uncertainty was less than 1% for penEasy.

The measured, and simulated, peak efficiency at 59.54 keV, defined as the area under

the previous spectrum in the interval from 57.7 to 64.4 keV, is provided in Table 6.6. The three Monte Carlo codes estimate a consistent value of the peak efficiency. Simulated results are higher than the measurement, but fall within the experimental uncertainty ($\pm$8%). The latter was estimated taking into account the inaccuracies in the position and orientation of the detectors around the reference measurement position. In particular, it was found that a $\pm$1 cm variation in the distance between the source (knee bone) and the Ge detectors produced a 7% modification of the peak efficiency (see Gómez-Ros *et al.* 2008, Table 4). Therefore, the systematic discrepancies could be explained by geometric differences between the simulated and the experimental settings.

Table 6.6: Peak efficiencies (in cps/kBq) simulated and experimentally measured at 59.54 keV. Uncertainties given as 2σ when available.

Method	Peak efficiency
Experimental	3.56 ± 0.28
penEasy	3.69 ± 0.01
MCNPX 2.5	3.70
EGS 4	3.66

7 Medical imaging

In this chapter we present two practical applications of the penMesh code—described in chapter 4—in the field of medical imaging. The clinically-realistic anthropomorphic phantom used in these simulations is described in the following section. Section 7.2 introduces the simple detector and source models that have been implemented in penMesh for the simulation of radiographic imaging systems. Section 7.3 presents the simulation of coronary angiography images. This work was presented at the *SPIE Medical Imaging 2007 symposium, Physics of Medical Imaging conference* (San Diego, February 19, 2007) and published in the conference proceedings (Badal *et al.* 2007). Finally, section 7.4 shows the simulation of an x-ray imaging procedure performed during a prostate brachytherapy treatment. This simulation was presented at the *9th Biennial ESTRO Meeting on Physics and Radiation Technology for Clinical Radiotherapy* (Barcelona, September 10, 2007). A paper describing this work (Badal *et al.* 2008) was published in the journal of the European Society for Therapeutic Radiology and Oncology (ESTRO).

7.1 Triangle mesh-based anthropomorphic phantom

The triangle mesh geometry model implemented in penMesh can represent any three-dimensional shape. This feature is particularly convenient for the description of organic structures. A comprehensive review of computational anthropomorphic models was provided by Zaidi and Xu (2007). Phantoms described by standard boundary representation models can be tessellated and exported to triangle meshes using existing computer-aided design software. One of the most accurate phantoms of this kind is the NURBS-based Cardiac-Torso (NCAT) phantom (Segars 2001). The NCAT is a highly detailed male anatomical model created by fitting non-uniform rational B-splines (NURBS) to the high-resolution CT scans from the Visible Human Project (National Library of Medicine). An interesting feature of the NCAT is that it is a 4D phantom, that is, it can be deformed to reproduce the cardiac and the respiratory motions (Segars *et al.* 2003). This phantom is extensively used to simulate medical imaging applications, especially in nuclear medicine.

In order to use penMesh in medical applications, we prepared a triangle mesh-based version of the NCAT phantom (Badal *et al.* 2007; Kyprianou *et al.* 2007). This tesselated phantom, shown in Fig. 7.1, is composed of 330 closed triangle meshes and comprises, in total, 5 million triangles. A lower resolution version, with 1.5 million triangles, was also prepared. The ray-tracing of the low resolution phantom is significantly faster than the high resolution one because the ray–triangle intersection tests spend a significant fraction of the execution time, as shown in section 4.4.

A detailed triangle mesh-based heart phantom was also developed by Banh *et al.* (2007) and registered into the phantom described above. The new heart model, shown in Fig. 7.2, was created segmenting a high-resolution CT angiography scan, fitting smooth surfaces on the organ boundaries, and tessellating the resulting surfaces into triangle meshes using the VTK[1] and ITK[2] software libraries, as described by Kyprianou *et al.* (2007).

The material composition of the organs included in the phantoms was approximated by the average stoichiometric compositions measured by Woodard and White (1986).

[1]The *Visualization Toolkit* (VTK) is an open-source software system for 3D computer graphics, image processing, and visualisation. It is freely available at http://www.vtk.org.

[2]The *Insight Segmentation and Registration Toolkit* (ITK) is an open-source software toolkit for performing registration and segmentation of digital images. It is freely distributed by the U. S. National Library of Medicine at http://www.itk.org.

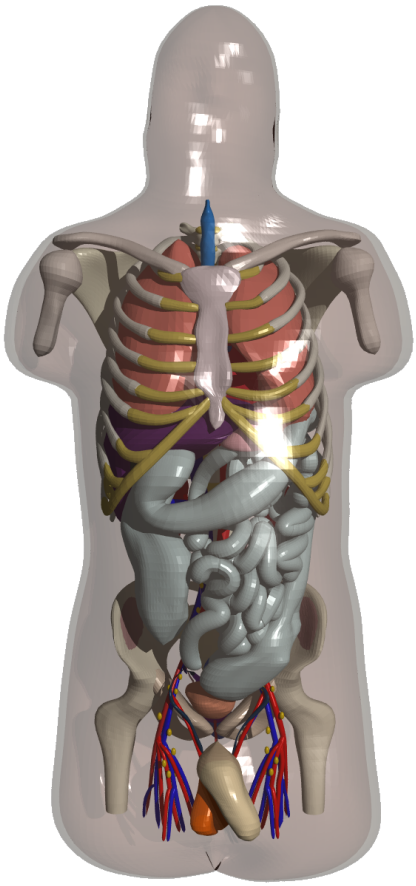


Figure 7.1: High-resolution tessellated version of the NCAT phantom adapted to penMesh. In total, 330 closed triangle meshes, with over 5 million triangles, are used to define the shape of the organs.

7.2 X-ray detector and source models

In order to simulate a medical imaging system it is necessary to create a model of the image formation process. With this purpose, a tally routine to create radiographic images has been developed according to penEasy's specifications and included in penMesh. Due to the wide energy range in which PENELOPE can be applied, the new tally can cover from low-energy x-ray imaging (e.g., mammographic systems) to radiotherapy portal imaging. This tally can also be used with penEasy, but it is not included in the standard distribution for consistency with its general-purpose approach.

The new tally employs a simple model in which an image is formed by scoring the energy deposited inside the detecting object (typically a thin layer of CsI or some other

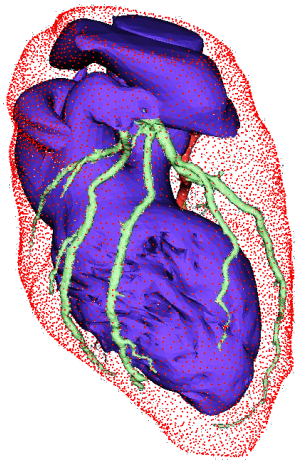


Figure 7.2: Triangle-mesh based heart model segmented from a high-resolution CT angiography scan.

scintillating material). The detector geometry can be defined with triangle meshes, or with quadric surfaces in case the detector is located outside the octree bounding box. The detector location, size, orientation, and the number of pixels in which it is divided, are introduced through the simulation input file. The image tally reports a text, or binary, file containing the energy deposited in each pixel, in units of eV/cm^2 per simulated particle. A gnuplot script provided with penMesh can be used to convert the reported file into an image in which the grey level of each pixel is directly proportional to the deposited energy. The use of an absolute scale allows for a straightforward comparison of images obtained using different geometries or simulation parameters.

For the sake of simplicity, and to maximise the simulation efficiency, the material of the detector layer was defined to absorb all the incoming radiation in the simulations presented in the current chapter. This was done defining an absorption energy larger than the maximum source energy. In this situation, the detection efficiency is 100% and the pixel values correspond to the energy fluence on the detector surface. A drawback of this method is that x-ray scattering inside the detector is neglected.

The presented image tally includes an alternative method to generate images, which consists in a pure ray-tracing of the simulation geometry. In this deterministic method, i.e., not based on Monte Carlo techniques, a virtual ray is cast from the focal spot to the centre of each pixel and the probability that an x ray does not interact in the full path is scored as the pixel value. The interaction probability is calculated using the cross-sections from the PENELOPE database for the mean energy of the input x-ray spectrum (multiple rays with different energies could be simulated for each pixel, but

this would naturally increase the simulation time). The resulting image corresponds to the image that would be generated in an ideal detector disregarding beam hardening effects and any radiation scattered inside the phantom (or, equivalently, using an ideal anti-scatter grid). This tally is extremely fast (requires of the order of 1 minute of computing time for a full body image) and produces high contrast, noise-free images, because it does not include the effect of scattered radiation or the statistical uncertainty associated with an Monte Carlo simulation.

The x-ray sources used in the presented simulations were modelled taking advantage of the standard penEasy source BIGS (Box Isotropic Gauss Spectrum), which was described in section 3.1.1. The x rays were emitted from a point focal spot, forming a cone beam. A version of the BIGS source in which the focal spot is a surface defined by a combination of three gaussian functions was prepared, but it was not used in the studies reported in this chapter.

The source cone beam was collimated to a square field using a completely radiopaque collimator defined with quadric surfaces between the source and the octree region. The emitted energy spectrum was precalculated using the data from the IPEM report 78 (Cranley *et al.* 1997), with 0.5 keV energy bins, a ripple-free high voltage source, a tungsten anode with 10° angle, and 3 mm aluminum filtration.

7.3 Simulation of coronary angiography

Coronary angiography is the radiographic visualisation of the coronary vessels after injection of radiopaque contrast media (Scanlon *et al.* 1999). The ability to simulate clinically-realistic angiography images in the computer may be valuable for the optimisation of angiographic systems, as well as to study in detail the physical processes involved in the formation of the images (Kyprianou *et al.* 2007).

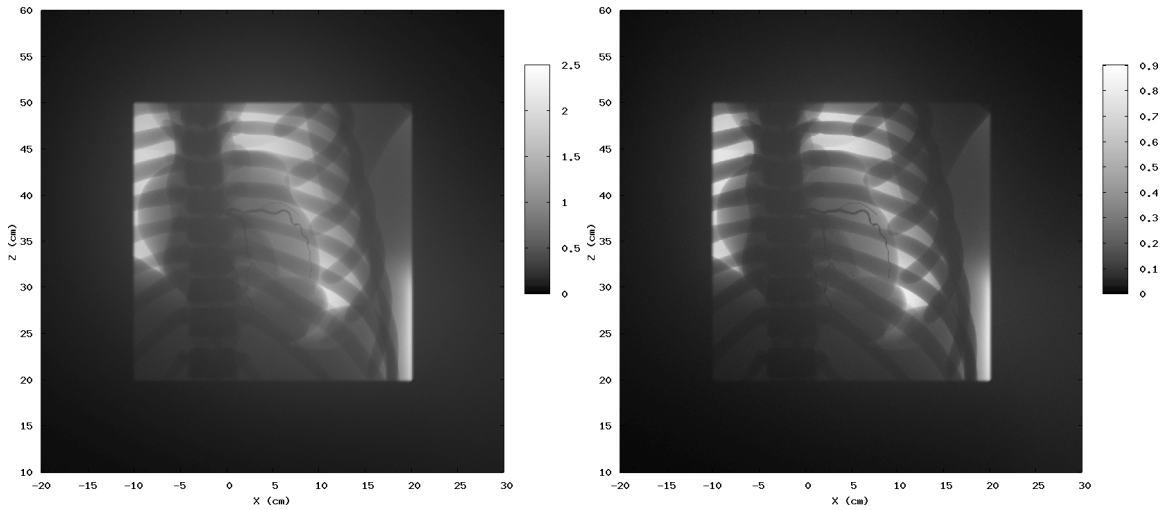
PenMesh and the anthropomorphic phantom described above were used to simulate coronary angiograms. The right and left coronary arteries from the NCAT heart were filled with contrast media composed of 10% per volume of iodine and 90% of blood. The phantom triangles were sorted with a level 8 octree spatial data structure. The triangle meshes corresponding to the intestines were not included in the simulation geometry. The source-to-detector distance was 100 cm, and the air gap between the phantom and the detector was 15 cm. In this configuration the coronary arteries were approximately located at 75 cm from the source, giving a geometrical magnification of 1.33 in the image plane. The PENELOPE absorption energies were set to 3 keV for

photons and 20 keV for electrons. At these energies, the photon mean free path and electron CSDA (continuous slowing down approximation) range in air were 5 cm and 0.8 cm, respectively.

Simulated coronary angiography images for a 90 kVp and a 60 kVp x-ray energy spectra and a 50×50 cm^2 detector with 1000×1000 pixels (pixel size of 500 μm) are presented in Fig. 7.3. Each simulation was performed in parallel using the script package clonEasy (see chapter 5) in 100 CPUs of the computer cluster at the Center for Devices and Radiological Health (U. S. Food and Drug Administration). In total, $9.5{\cdot}10^{10}$ primary x rays were simulated for each case. The 90 kVp simulation required 24 hours of parallel computing, while the 60 kVp required 20 hours. The image grey scale corresponds to the energy deposited in the detector plane per unit area and per primary particle (eV/cm^2 per x ray). This makes it possible to quantitatively compare images obtained using different simulation parameters (e.g., different energy spectra). Taking advantage of the capabilities of the Monte Carlo simulated data, Fig. 7.4 presents a decomposition of the 90 kVp angiography image based on the interactions that each individual x ray underwent while traversing the triangle mesh phantom.

The simulated angiograms are equivalent to the ones obtained in the clinic, within the limitations of our anatomical phantom and detector model. More clinically-realistic images could be obtained including the inner structure of the lungs and differentiating trabecular and cortical bone in the phantom. The iodine-filled coronary arteries can be clearly observed. As expected, the contrast in the 60 kVp image is better than in the 90 kVp (the vessels are darker than the background). Studying the contribution of scatter to the generated images, it can be seen that Compton scattering (which is dominant in the multiple scattering image) is the main scattering process, while the contribution of Rayleigh interactions is much smaller. Nevertheless, Compton scattering produces large angular deflections and does not preserve anatomical information, while Rayleigh scattering gives small angular deflections and produces a blurred image with reduced contrast.

Apart from the image tally, penEasy's pulse height energy spectra and energy deposition tallies were also used in the simulations. The spectra tallied at the detector during the generation of the previous scatter images, and the initial spectrum without phantom, are presented in Fig. 7.5. The integral of the pulse height spectrum is proportional to the number of particles per history that arrive at the detector. The spectra shows that most of the primary particles are absorbed in the body. Indeed, only 1.5% of the primary x rays arrive at the detector. Additionally, the majority of the detected particles were scattered in the body. Despite this, the full image—including scatter—still shows

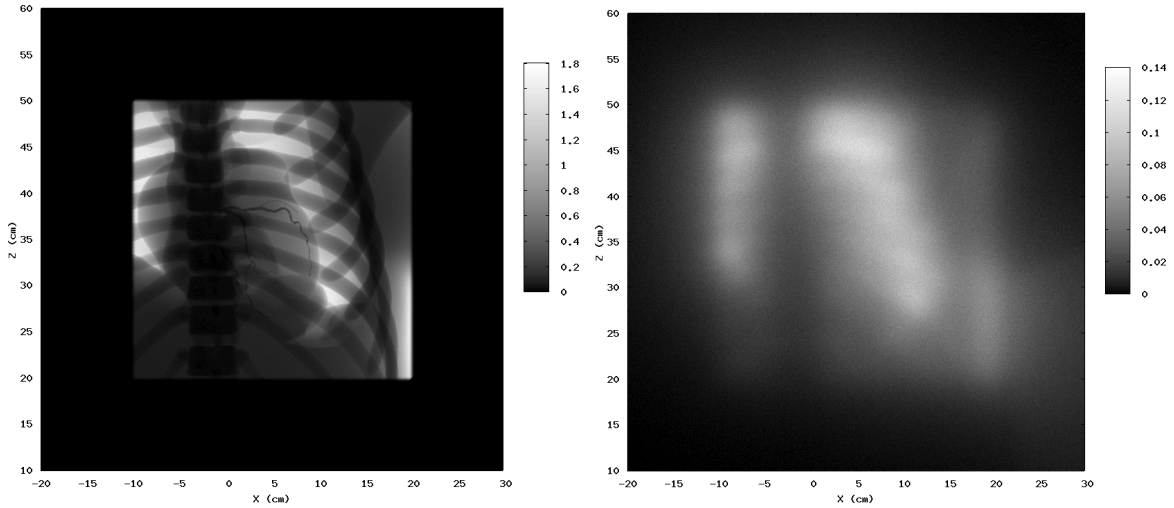


(a) (b)

Figure 7.3: Simulated coronary angiography images for a 90 kVp (a) and 60 kVp (b) x ray source. The detector pixel size was 500×500 μm. For each simulation, nearly 10^{11} primary x rays were simulated in one day of parallel computing in 100 CPUs. The image gray scale has units of eV/cm^2, and its value corresponds to the energy deposited in the flat panel detector per unit area and per primary particle.

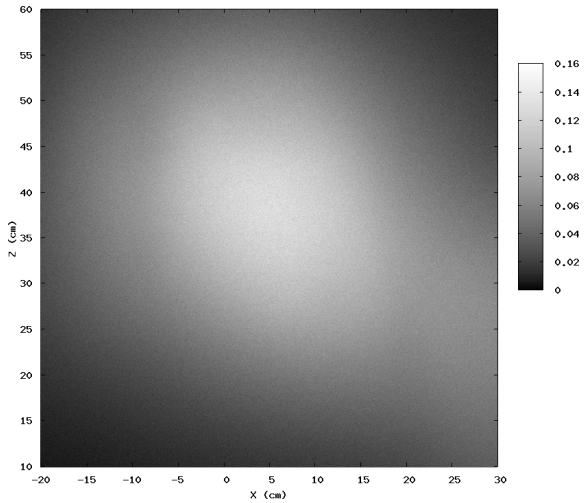
a clear projection of the anatomy, because the scattered radiation is fairly uniformly distributed, while non-scattered particles preserve the anatomic information. Another interesting feature that can be observed in the spectra is that inelastic scattered photons have, as expected, less average energy than the non-scattered. In particular, most of the non-scattered radiation has an energy above 55 keV, while most of the multiply scattered radiation is below this energy. This suggests that a detector optimised for detecting particles with energy higher than 55 keV could produce an image with less scatter and, hence, with better quality.

The energy deposited per primary particle in the volume of some organs, for the 90 kVp and 60 kVp simulations, is given in Table 7.1. The statistical uncertainty in the energy deposition was below 1% for all organs. The radiosensitive tissues that receive more radiation dose are the bones (including the ribs and the spine) and the skin. For all the organs except the skin the energy deposited per primary particle increases between 20% and 30% for the 90 kVp spectrum with respect to the 60 kVp. Note that the relative difference in the deposited energy for the two spectra equals the relative difference in the average absorbed dose in the organ volume. However, this calculation does not take into account the fact that, in a real situation, the 60 kVp image will require a larger number of primary x-rays. The energy deposited in the skin volume is considerable,

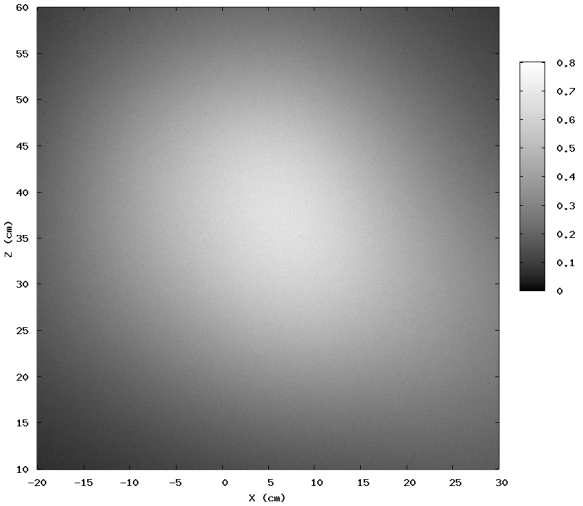


(a) Non scattered primary particles.

(b) Single Rayleigh (elastic) interaction.



(c) Single Compton (inelastic) interaction.



(d) Particles scattered more than once.

Figure 7.4: Decomposition of the 90 kVp angiography image depending on the interactions that each individual x-ray underwent while traversing the triangle mesh phantom (in eV/cm^2 per primary particle).

and the resulting absorbed dose per primary particle is 12.5% higher for the 60 kVp case. The reason for this increase is that the body attenuates low energy radiation more efficiently. Indeed, as it can be seen from the pulse height spectra, most of the radiation below 30 keV is completely absorbed in the body.

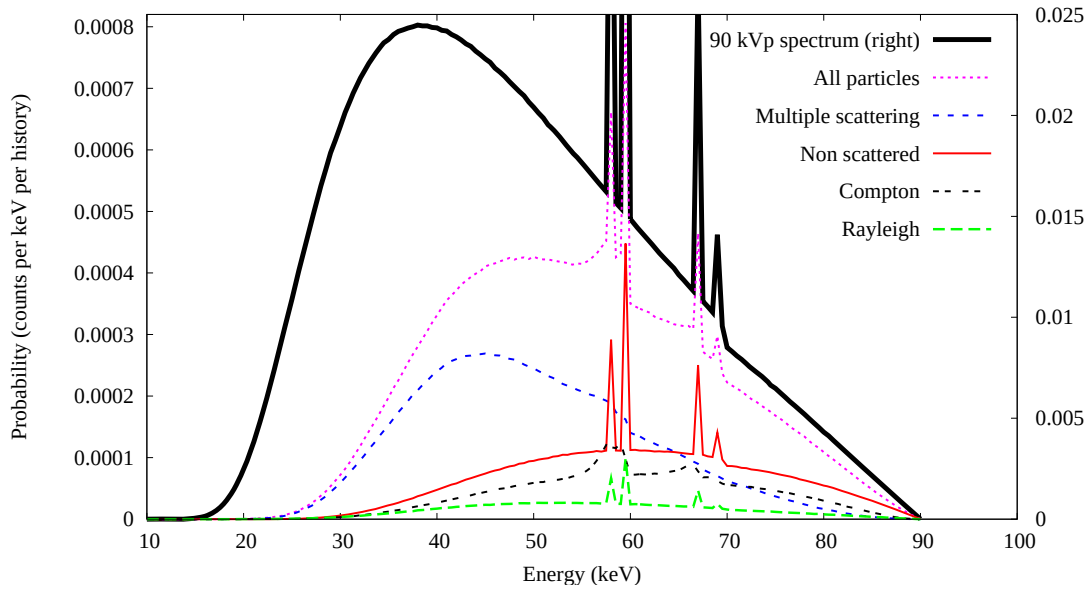


Figure 7.5: Pulse height spectra tallied at the detector plane for the 90 kVp image. The spectra corresponding to all the particles and to the different kinds of scattering are provided. The initial normalised source spectrum is also presented, with the scale shown at the right of the plot.

Table 7.1: Energy deposited in the phantom organs during angiographic imaging with 60 kVp and 90 kVp x-rays (in keV per primary particle). The increase in the deposited energy for the 90 kVp with respect to the 60 kVp is also provided (in %). The statistical uncertainty was below 1% for all organs.

Organ	60 kVp image	90 kVp image	% difference
Bones	4.21	5.62	+33.5
Skin	3.11	2.72	-12.5
Liver	0.77	1.06	+37.7
Heart muscle	0.56	0.76	+35.7
Stomach	0.32	0.40	+25.0
Left lung	0.57	0.69	+21.1
Right lung	0.23	0.29	+26.1
Rest of the body	7.96	9.10	+14.3
TOTAL	17.73	20.64	+16.4

7.3.1 New heart model

The previous angiographies were obtained using the heart included in the original NCAT phantom (2001 version). With the aim of increasing the realism of the simulated images, the study was repeated using the new heart model described in section 7.1. Figure 7.6 shows the obtained results, for the 90 kVp x-ray source and the two image formation models described in section 7.2. In this case a large detector, with an area of 50×100 cm^2 and 500×1000 pixels, was defined. Due to the relatively large pixel size (1 mm^2), the low resolution version of the NCAT phantom, shown in Fig. 7.6 (a), was employed (see section 7.1 for details). The full Monte Carlo simulation, Fig. 7.6 (b), used 10^{11} x rays and was executed in parallel for approximately 4 days in 30 computers from the Argos cluster (described in section 5.3) using the clonEasy scripts (see chapter 5). A level 8 octree was used in the simulation. For each execution, 386 MByte of memory were required and the simulation speed was 14420 x-rays/second per CPU.[3] Comparing this performance with that of the previous case, we see that the use of the low resolution phantom (i.e., less triangles) significantly reduces the required memory and increases the speed. The average statistical uncertainty in the reported pixel values, for pixels with values above half the image maximum value, was below 1%. Finally, the ray-tracing simulation, Fig. 7.6 (c), required only 3 minutes of execution in a single CPU. This time includes about 2 minutes of initialisation (the time spent reading the triangles, creating the octree structure, reading the material database, etc.).

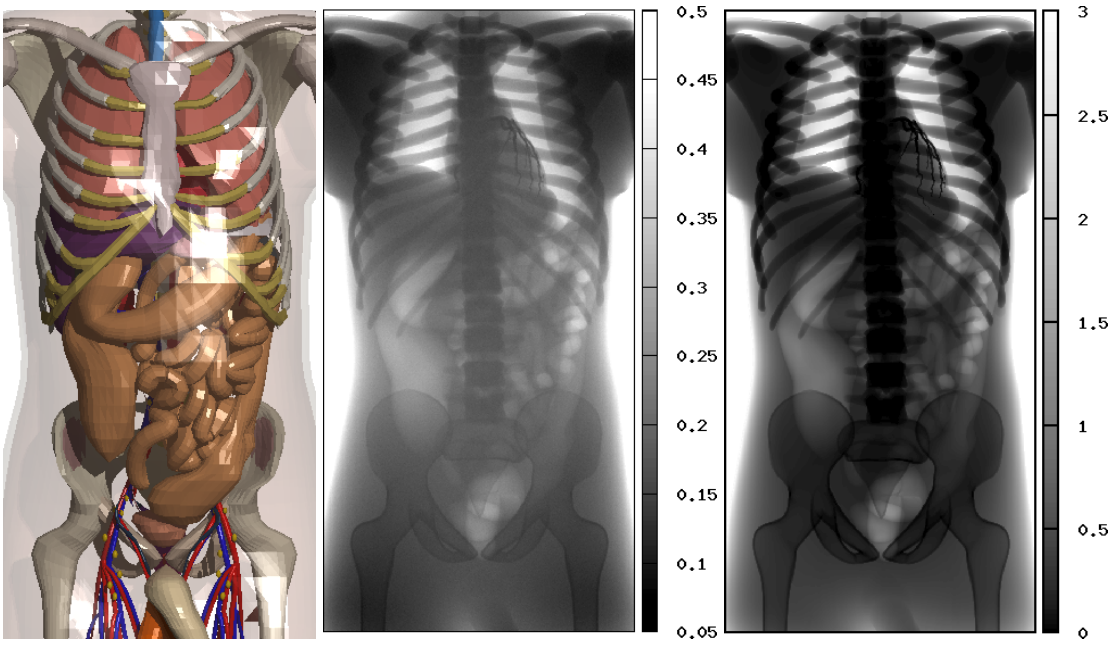
In summary, we have shown that penMesh can accurately simulate coronary angiography using an advanced triangle mesh geometry. The simulation provided relevant information that can be employed in the optimisation of imaging systems.

7.4 Simulation of prostate brachytherapy imaging

As a second example of the capabilities of penMesh, we simulated projection images of the human pelvis region. Several radioactive seeds were included inside the phantom's prostate to simulate the assessment of the seed positioning during a clinical prostate brachytherapy treatment. A detailed description of this imaging procedure can be found in the ESTRO recommendations on prostate brachytherapy (Ash *et al.* 2000; Salembier *et al.* 2007).

A simplified model of a brachytherapy seed, as described in the ICRU Report 72 (2004),

[3] In this case the reported timing results correspond to an Intel® Xeon™ 5130 CPU at 2.0 GHz and with 4 GByte of RAM. The code was compiled with GCC (option `-O3`).



(a) (b) (c)

Figure 7.6: Whole body angiography with the low resolution version of the NCAT phantom and the new heart model: (a) 3D rendering of the phantom, containing 1.5 million triangles; (b) Monte Carlo simulated image (grey scale: eV/cm^2 per initial x ray); (c) fast ray-tracing projection (grey scale: detection probability in %).

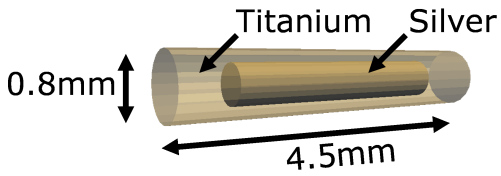


Figure 7.7: Simple model of a ^{125}I brachytherapy seed, composed of two cylinders tessellated into 80 triangles.

was created with the open-source program ParaView (see http://www.paraview.org). The seed, shown in Fig. 7.7, was modelled using two concentric cylinders. The internal cylinder (3 mm in length and 0.5 mm in diameter) represented a silver marker that allows the seeds to be readily seen in radiographic images, and the external cylinder (4.5 mm in length and 0.8 mm in diameter) represented a titanium encapsulation. Radioactive ^{125}I is absorbed on the surface of the silver rod in a real seed but it was not included in our simple model.

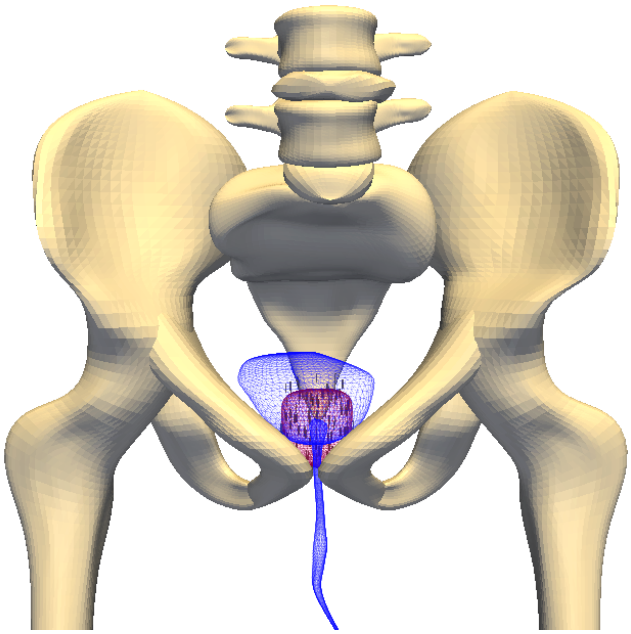


Figure 7.8: Closeup of the simulated treatment. The 112 ^{125}I brachytherapy seeds can be seen inside the phantom's prostate.

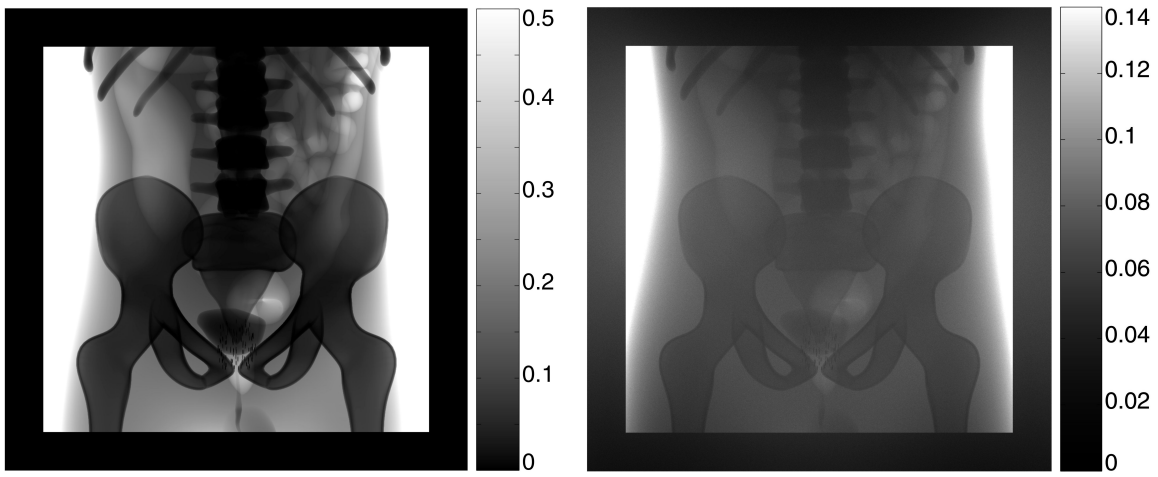
The seed cylinders were tessellated into 80 triangles and stored as triangle meshes in ASCII `ply` format (Stanford 3D Scanning Repository), which is readable by penMesh (see section 4.2). The meshes were cloned to produce 112 identical seeds and distributed forming a cylindrical pattern around the prostate to reproduce a brachytherapy treatment. A small random shift was applied to each seed to account for a small drift inside the prostate and implantation inaccuracies. A closeup of the treatment setting is provided in Fig. 7.8. The bladder, prostate and urethra are represented as a wire frame (i.e., only the triangle edges are shown) to allow the visualisation of the seeds.

In the simulations presented in this section, the source-to-detector distance was 100 cm. The detector was located 20 cm behind the patient, producing a 1.3 geometrical magnification of the prostate region. A completely radiopaque collimator was used to define 50×50 cm^2 and 10×10 cm^2 fields on the detector plane. A dilute barium sulphate contrast (3% weight/volume concentration) was injected inside the bladder and the urethra to facilitate the visualisation of these organs in the projection images, as recommended by (Ash *et al.* 2000). PenMesh was compiled using the GCC compilers and executed in parallel, using clonEasy, in 30 nodes of the computer cluster Argos (see section 5.3). The computer used to report the simulation speed had an Intel®Xeon™3.0 GHz CPU and 2 GByte of RAM memory. An octree with 8 levels of subdivision was used. Each

execution required 260 MBytes of computer memory.

Figure 7.9 shows the simulated radiographic images for a 70 kVp x-ray source with a 50×50 cm^2 field, and a 60×60 cm^2 detector with 500×500 μm^2 pixels. Figure 7.9 (a) was created with the ray-tracing model described in section 7.2. The grey scale corresponds to the probability, in %, that an x ray will arrive at the pixel without interacting. According to the obtained pixel values, more than 99.5% of the x rays that cross the phantom are absorbed or scattered before reaching the detector surface. This simulation required just 3 minutes of computing time in a single CPU.

Figure 7.9 (b) displays the full Monte Carlo simulation. The image was created scoring the energy of the particles arriving on the detector plane and, therefore, the grey scale has units of eV/cm^2 per emitted x ray. The simulation speed was 17450 x rays per second. In total 10^{11} rays were simulated; this would have required more than 66 days of computation in a single CPU (but only 2.2 days in the computer cluster). The speed of this simulation using an octree with level 10, 4 and 0 (i.e., no octree) was 17725, 3105 and 4 x rays per second respectively. Therefore the octree accelerated the simulation more than 4300 times and it is definitely an essential component of the algorithm.



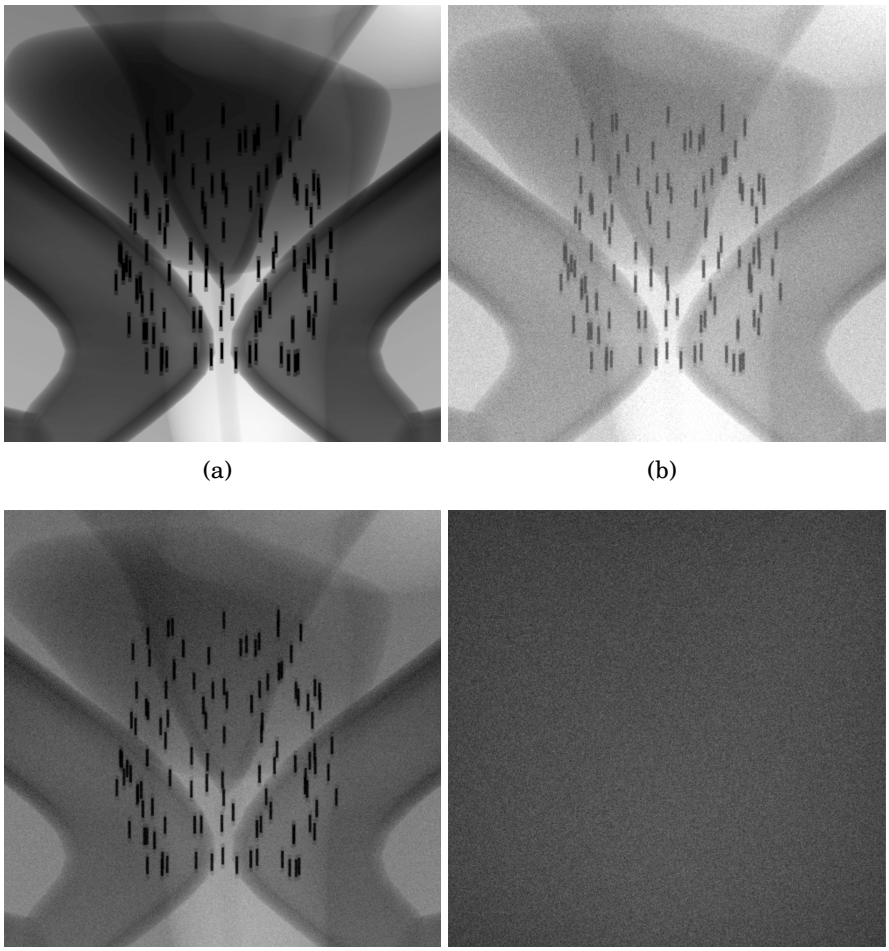
(a) (b)

Figure 7.9: Simulated anteroposterior radiographic images produced by a 70 kVp x-ray source collimated to a 50×50 cm^2 field, on a 60×60 cm^2 detector with 500 μm pixels: (a) ray-traced image (grey scale: detection probability in %), (b) full penMesh simulation (grey scale: eV/cm^2 per initial x ray). Pixels with values higher than the scale maximum are displayed in white.

The presented large-field images show the high level of detail of the phantom. The bladder and the urethra, which contain a radiopaque contrast, can be clearly observed, as well as the intestines, which do not attenuate as much radiation as the surrounding tissue. The radioactive seeds can also be seen but not clearly because the image pixels are almost as big as the seeds.

In order to visualise the inner structure of the seeds a simulation with a higher resolution detector was performed. Figure 7.10 shows the simulation of a 10×10 cm^2 field on a 10×10 cm^2 detector with 200×200 μm^2 pixels. Four images are displayed, corresponding to: (a) the ray-tracing model, (b) the full simulation, (c) the image formed only by x rays that have not interacted in the phantom (this is also the image that would be obtained with an ideal anti-scatter grid), and (d) the image created by scattered x rays and secondary radiation (including electrons). All these images have a linear grey scale with black at the value 0, and white at 0.4% for the ray-traced image (a) and at 1.5 eV/cm^2 per history for the rest. In this case 10^{10} x rays were simulated and the speed in the reference CPU was 12175 rays per second (about 9.5 days in a single CPU).

By analysing the data from the previous simulation it was found that only 0.24% of the emitted x rays arrived at the detector, and 0.077% of the rays were detected after being scattered in the phantom. This gives an estimated scatter fraction of 0.32 at the detector surface. Note that this value is similar to those provided in Table 4.1 for the same irradiation field and x ray source and a simple lucite torso phantom.



(a) (b)

(c) (d)

Figure 7.10: Simulated radiographies of the prostate region for a 70 kVp source with 10×10 cm^2 field, on a 10×10 cm^2 detector with 200 μm pixels: (a) ray-traced image, (b) full penMesh simulation, (c) non-scattered x rays, (d) scattered x rays and secondary radiation.

Part IV

Conclusions

8 Conclusions

This thesis has been devoted to the design and implementation of new Monte Carlo tools that facilitate the accurate simulation of medical physics applications involving complex geometries. The main conclusions that arise from the presented work are:

1. penEasy, the general-purpose Monte Carlo package for PENELOPE presented in section 3.1, can accurately simulate clinical applications of ionising radiation. The modular structure of its main program and the comprehensive set of tallies and source models that it includes allow the simulation of a wide variety of cases and facilitate the implementation of new tallies, sources, variance reduction techniques and geometric models.

2. penVox, the geometry package for penEasy presented in section 3.2, extends the applicability of PENELOPE to voxelised objects. This code can be used to compute, for example, dose distributions produced by radiotherapy treatments in anatomic phantoms created from CT scans.

3. penMesh, the simulation code presented in section 4.1, combines the features of penEasy with the flexibility of a geometric model based on triangle meshes. Its new ray–tracing algorithm can efficiently handle highly realistic phantoms. This code has been used to simulate diagnostic imaging applications in which the geometric detail was an essential factor.

4. clonEasy and its auxiliary program seedsMLCG, described in section 5.2, provide a simple way to run Monte Carlo simulations in parallel. These software tools produce an effective reduction of the execution time that is roughly proportional to the number of processors available. The use of parallel computing is indispensable to perform complex simulations with detailed anatomical phantoms.

During the course of the research reported in this thesis the developed codes have been tested in realistic medical physics applications, and also compared with other Monte Carlo codes and experimental data. As a result, the new codes, which are free and open software, are ready to be publicly distributed.

8.1 Future work

Some researchers from various institutions are already using the presented software tools. For example, penEasy and penVox are currently used to simulate stereotactic body radiotherapy treatments at the Karolinska Institute (Stockholm, Sweden; see Panettieri 2008) and to obtain mammographic images at the Center for Devices and Radiological Health (CDRH, U. S. Food and Drug Administration; see Park 2008). PenMesh is also used at the CDRH for the simulation of coronary angiographies with heart phantoms containing realistic pathology models Kyprianou *et al.* 2007.

It is expected that the development of the codes will continue in the future. A feature that will soon be implemented in penMesh is a more realistic x-ray detector model. The new image formation model, which has already been described by Kyprianou *et al.* (2008), will take advantage of depth- and energy-dependent distribution functions calculated by the MANTIS code (Badano and Sempau 2006) for columnar CsI detectors. Ongoing work also aims at implementing more advanced variance reduction techniques in penEasy (e.g. azimuthal splitting, see Bush *et al.* 2007), developing an interface to read/write phase-space files in IAEA format (Capote *et al.* 2006) and preparing a new source model that will automatically generate the cascade of beta and gamma (or x) rays from radioactive elements. Another useful tool under development is a converter that will take quadric objects defined with the PENGEOM model and tessellate them into triangle meshes suited for penMesh. This tool will facilitate the visualisation of PENGEOM geometries and it will also allow the use of some complex geometries already defined with quadrics, such as particle accelerators and multileaf collimators, in penMesh simulations.

Part V

Appendices

A List of publications and presentations

This appendix lists the most relevant publications and presentations at scientific conferences that were performed during the course of this thesis (in chronological order).

Papers published in (or submitted to) peer-reviewed journals:

- **A. Badal** and J. Sempau, A package of Linux scripts for the parallelization of Monte Carlo simulations, Comput. Phys. Commun. 175, pp. 440–450, 2006.
- **A. Badal**, I. Kyprianou, A. Badano, and J. Sempau. Monte Carlo simulation of a realistic anatomical phantom described by triangle meshes: application to prostate brachytherapy imaging. *Radiother. Oncol.*, 86, pp. 99–103, 2008.
- J. Sempau and **A. Badal**. PenEasy, a modular main program and voxelised geometry package for PENELOPE, *submitted*.
- **A. Badal**, I. Kyprianou, D. P. Banh, A. Badano, and J. Sempau. penMesh—Monte Carlo radiation transport simulation in a triangle mesh geometry, *submitted*.

Papers published in scientific conferences proceedings:

- **A. Badal**, I. Kyprianou, A. Badano, J. Sempau and K. J. Myers. Monte Carlo package for simulating radiographic images of realistic anthropomorphic phantoms described by triangle meshes. In J. Hsieh and M. J. Flynn, editors, *Medical Imaging 2007: Physics of Medical Imaging*, volume 6510 of *Proc. SPIE*, 2007.
- I. S. Kyprianou, **A. Badal**, A. Badano, D. P. Banh, M. Freed, K. J. Myers, and L. Thompson. Monte Carlo simulated coronary angiograms of realistic anatomy and pathology models. In K. R. Cleary and M. I. Miga, editors, *Medical Imaging 2007: Visualization and Image-Guided Procedures*, volume 6509 of *Proc. SPIE*, 2007.

Conferences and workshops where I have shown my work as an oral presentation or poster:

- **A. Badal** and J. Sempau. Noves eines per a la planificació radioterapèutica amb un codi de simulació Monte Carlo. *IV Jornades de Recerca en Enginyeria Biomèdica*, Barcelona (Spain), June 9, 2004.
- **A. Badal** and J. Sempau. Nuevas herramientas para paralelizar una simulación con el programa Monte Carlo PENELOPE. *XV Congreso Nacional de Física Médica*, SEFM, Pamplona (Spain), June 28–July 1, 2005.
- **A. Badal**, I. Kyprianou, J. Sempau, A. Badano and K. J. Myers. A novel Monte Carlo Package for simulating x-ray images. *USMS Workshop: Imaging as a Biomarker*, NIST, Gaithersburg (USA), September 14–15, 2006.
- **A. Badal**, I. Kyprianou, A. Badano, J. Sempau and K. J. Myers. Monte Carlo package for simulating radiographic images of realistic anthropomorphic phantoms described by triangle meshes. *SPIE Medical Imaging 2007: Physics of Medical Imaging*, February 19, 2007.
- **A. Badal**, J. Sempau, I. Kyprianou and A. Badano. Nuevas herramientas para la descripción de geometrías complejas con PENELOPE: aplicación a la modelización de maniquís antropomórficos. *XVI Congreso Nacional de Física Médica*, SEFM, Granada (Spain), May 23–25, 2007.
- **A. Badal**, I. Kyprianou, A. Badano and J. Sempau. Monte Carlo simulation with PENELOPE and a realistic anthropomorphic phantom described by triangle meshes. *9th Biennial ESTRO Meeting on Physics and Radiation Technology For Clinical Radiotherapy*, Barcelona (Spain), September 10, 2007.

B penMesh documentation

A comprehensive manual describing penMesh's source code has been prepared by means of the automatic system `doxygen` (http://www.doxygen.org). This free and open-source documentation generator parses the C++ code and extracts information on the code structure and also descriptions of the defined functions and variables that were written as comments during the code development.

An excerpt of the documentation generated for the two C++ classes used by penMesh is presented in the following sections.

B.1 octree_node Class Reference

This class describes a nodes of the octree structure. Each node in the octree is an instance of this class, from the root node, which has level 0 and encloses the whole structure, to the leaves, which are the ending subnodes and contain may contain triangles inside (file `<octree_definition.h>`).

Public Member Functions

- **octree_node** ()

  Default class constructor.

- **octree_node** (double x0, double y0, double z0, double x1, double y1, double z1, char level0, **octree_node** *father0)

  Explicit class constructor.

- ∼**octree_node** ()

  Class destructor.

- void **set_node_parameters** (double x0, double y0, double z0, double x1, double y1, double z1, char level0, **octree_node** ∗father0)

  Sets the values of the node attributes (variable).

- **octree_node** ∗ **get_node** (double &px, double &py, double &pz, char level_limit=char(-1))

  Searches from the input node (the octree root usually) for the subnode that contains the input point and that has the maximum level specified (this is needed in function 'set_neighbors' because the stored neighbors can not have a higher level, i.e., smaller size).

- **octree_node** ∗ **get_node_fast** (double &px, double &py, double &pz)

  Searches from the input node for the subnode that contains the input point. This function assumes that the point is already located inside the node and the execution is faster than for the previous function.

- void **create_subnodes_with_triangles** (int &input_max_level)

  Generates the octree structure and distribute the triangles (to be called recursively).

- void **clean_octree** ()

  Free the dynamically allocated memory that is not required after the octree creation.

- double **get_wall_distance** (double &**x**, double &**y**, double &**z**, double &invers_vx, double &invers_vy, double &invers_vz, int &num_neighbor)

  Calculates the distance and neighbor number of the nearest node, for the input position and inverse value of the direction cosines.

- void **set_neighbors** (**octree_node** ∗in_node)

  Set the 'neighbor' class variable: pointers to the 6 neighbor nodes of each node.

- double **get_triangle_intersection** (double &**x**, double &**y**, double &**z**, double &**u**, double &**v**, double &**w**, **triangle_octree** ∗∗triangle_found_array)

Calculates the intersection with internal triangles and returns the distance and a pointer to the intersected triangle.

- **octree_node** ∗ **step_octree** (double &ds, double &dsef, int &ncross)

  Moves the current particle (in common /track/) across the octree and the triangular mesh. Equivalent to PENGEOM's STEP subroutine.

- void **store_secondary_flight_log** (int &old_nsec)

  Check if there are new secondary particles and store the corresponding flight_log.

- void **get_flight_log** (int &track_label)

  Loads the flight_log saved when the secondary particle was generated inside the octree, and set the static class variable 'last_triangles' to NULL.

- void **init_flight_log** (int &current_organ, int &current_material, int &track_label)

  Clears the class static variables 'flight_log' and 'last_triangles', and store the input organ and material in the log.

Public Attributes

- double **x_min, y_min, z_min**

  Lower corner of the octree node.

- double **x_max, y_max, z_max**

  Upper corner of the octree node.

- char **level**

  Recursion level of the current node in the octree hierarchy (0 for the root node).

- **octree_node** ∗ **son** [8]

  Pointers to the 8 descendants of the node (NULL for leaves).

- **octree_node** ∗ **father**

  Pointer to the father of this node (NULL for root).

- **octree_node** ∗ **neighbor** [6]

  Array with pointers to the 6 neighbor nodes (NULL for root).

- **triangle_octree** ∗∗ **triangle_list**

  List of pointers that point to the triangles that are found inside this sub-node.

- int **num_triangles**

  Number of triangles that are found inside—or intersect—this node.

Static Public Attributes

- static **triangle_octree** ∗ **triangle_mesh**

  Static variable that stores the collection of triangles that define the simulation geometry.

- static int **triangle_mesh_size**

  Size of the triangle mesh (i.e., number of triangles).

- static int **amount_nodes** [MAX_LEVEL+1]

  Amount of nodes for each octree level.

- static int **amount_leaves** [MAX_LEVEL+1]

  Amount of leaves in each level of the octree.

Private Member Functions

- bool **termination_condition** (int &max_level)

  Checks the condition that decides whether the node is a leaf (final node) or it has to be further sub-divided.

- void **sort_triangles** ()

  Distributes the triangles in the octree structure, that is, set the 'triangle_list' class member variable of each leaf (final node) assigning pointers to the triangles inside the node.

- void **update_flight_log** (int found_organ, int found_material, int &new_organ, int &new_material)

Updates the data in the flight_log using the information of the new organ found.

Static Private Attributes

- static **triangle_octree** ∗ **last_triangles** [MAX_OVERLAP]

  Array of pointers to the triangles intersected in the last step (or NULL).

- static int **flight_log** [2 ∗MAX_OVERLAP]

  Vector with a list of all the organs that the current particle has entered but not exited yet (the organ and material numbers are stored in consecutive positions in the array).

- static int **secondary_flight_log** [MAX_SECONDARIES][2 ∗MAX_OVERLAP]

  Array of flight logs to store the flight_log at the moment a secondary particle was generated.

- static int **old_warnings** [MAX_WARNING]

  Array with the ID of the overlapping warnings that have been displayed.

Friends

- void **draw_octree_ps** (**octree_node** ∗octree, int &octree_max_level, double &z_-plane, const char ∗file_ps, bool first_call=true)

  Draws the octree structure at the input z plane in postscript format (one square for each leaf in the plane).

- void **triangles_per_leaf** (**octree_node** ∗octree, int &octree_max_level, double &mean_tri, double &sigma_tri, int &leaves_full, int &leaves_empty, bool first_-call=true)

  Calculates the mean quantity, and standard deviation, of triangles inside the leaves with maximum level, and the amount of empty leaves.

B.2 triangle_octree Class Reference

This class describes each triangle that is part of the simulation geometry (file `<triangle_octree.h>`).

Public Member Functions

- **triangle_octree** ()

  Implicit constructor for the triangle objects.

- **triangle_octree** (double new_vert0_X, double new_vert0_Y, double new_vert0_Z, double new_vert1_X, double new_vert1_Y, double new_vert1_Z, double new_vert2_X, double new_vert2_Y, double new_vert2_Z, short int new_organ, short int new_material)

  Explicit class constructor.

- void **set_triangle_members** (double new_vert0_X, double new_vert0_Y, double new_vert0_Z, double new_vert1_X, double new_vert1_Y, double new_vert1_Z, double new_vert2_X, double new_vert2_Y, double new_vert2_Z, short int new_organ, short int new_material)

  Sets the value of all the class members.

- bool **intersect** (double &orig_X, double &orig_Y, double &orig_Z, double &dir_X, double &dir_Y, double &dir_Z, double &distance)

  Calculates whether the input ray intersects this triangle or not using the Moller-Trumbore algorithm.

- bool **inside_box** (double x_min, double y_min, double z_min, double x_max, double y_max, double z_max)

  Calculates whether the present triangle is inside or overlaps the input box (octree node).

Public Attributes

- double **vert0_X, vert0_Y, vert0_Z**

  Coordinates of the first vertex of the triangle.

- double **vert1_X, vert1_Y, vert1_Z**

  Coordinates of the second vertex of the triangle.

- double **vert2_X, vert2_Y, vert2_Z**

  Coordinates of the third vertex of the triangle.

- short int **organ**

  Identification number of the organ which owns this triangle.

- short int **material**

  Identification number of the organ material.

Epilogue

Peregrino

¿Volver? Vuelva el que tenga,
Tras largos años, tras un largo viaje,
Cansancio del camino y la codicia
De su tierra, su casa, sus amigos.
Del amor que al regreso fiel le espere.
Mas ¿tú? ¿volver? Regresar no piensas,
Sino seguir siempre adelante,
Disponible por siempre, mozo o viejo,
Sin hijo que te busque, como a Ulises,
Sin Ítaca que aguarde y sin Penélope.
Sigue, sigue adelante y no regreses,
Fiel hasta el fin del camino y tu vida,
No eches de menos un destino más fácil,
Tus pies sobre la tierra antes no hollada,
Tus ojos frente a lo antes nunca visto.

Luis Cernuda (1961)

References

AAPM report 1990 Standardized Methods For Measuring Diagnostic X-Ray Exposures Technical Report 31, American Institute of Physics

Agostinelli S, Allison J and Amako *et al.* 2003 Geant4—a simulation toolkit *Nucl. Instrum. Meth. Phys. Res. A* **506** 250–303

Ahnesjö A 1989 Collapsed cone convolution of radiant energy for photon dose calculation in heterogeneous media *Med. Phys.* **16** 577–592

Akenine-Möller T 2001 Fast 3D Triangle-Box Overlap Testing *Jour. Graphics Tools 6*(1) 29–33

Alderson S W, Lanzl L H, Rollins M and Spira J 1962 An instrumented phantom system for analog computation of treatment plans *Am. J. Roentgenol. Radium. Ther. Nucl. Med.* **87** 185–195

Amanatides J and Woo A 1987 A fast voxel traversal algorithm for ray tracing *Eurographics* **87** 3–10

Anagnostopoulos G, Baltas D, Pantelis E, Papagiannis P and Sakelliou L 2004 The effect of patient inhomogeneities in oesophageal ^{192}Ir HDR brachytherapy: a Monte Carlo and analytical dosimetry study *Phys. Med. Biol. 49*(12) 2675–2685

Andreo P 1991 Monte carlo techniques in medical radiation physics *Phys. Med. Biol. 36*(7) 861–920

Ash D, Flynn A, Battermann J, Reijke T, Lavagnini P and Blank L 2000 ESTRO–EAU–EORTC recommendations on permanent seed implantation for localized prostate cancer *Radiother. Oncol.* **57** 315–321

Badal A, Kyprianou I, Badano A and Sempau J 2008 Monte Carlo simulation of a realistic anatomical phantom described by triangle meshes: application to prostate brachytherapy imaging *Radiother. Oncol.* **86** 99–103

Badal A, Kyprianou I, Badano A, Sempau J and Myers K J 2007 Monte Carlo package for simulating radiographic images of realistic anthropomorphic phantoms

described by triangle meshes In J. Hsieh and M. J. Flynn (Eds.), *Medical Imaging 2007: Physics of Medical Imaging*, Volume 6510 of *Proc. SPIE*

Badano A, Kyprianou I S and Sempau J 2006 Anisotropic imaging performance in indirect x-ray imaging detectors *Med. Phys.* **33** 2698–2713

Badano A and Sempau J 2006 MANTIS: combined x-ray, electron and optical Monte Carlo simulations of indirect radiation imaging systems *Phys. Med. Biol. 51*(6) 1545–1561

Banh D P, Kyprianou I S, Paquerault S and Myers K J 2007 Morphology-based three-dimensional segmentation of coronary artery tree from CTA scans In J. P. W. Pluim and J. M. Reinhardt (Eds.), *Medical Imaging 2007: Image Processing*, Volume 6512 of *Proc. SPIE*

Baró J, Sempau J, Fernández-Varea J M and Salvat F 1995 PENELOPE: An algorithm for Monte Carlo simulation of the penetration and energy loss of electrons and positrons in matter *Nucl. Instrum. Meth. Phys. Res. B* **100** 31–46

Becker J, Zankl M and Petoussi-Henss N 2007 A software tool for modification of human voxel models used for application in radiation protection *Phys. Med. Biol. 52*(9) N195–N205

Berger M J 1963 Monte Carlo calculation of the penetration and diffusion of fast charged particles In B. Alder, S. Fernbach, and M. Rotenberg (Eds.), *Methods in Computational Physics (vol. 1)*, pp. 135–215 Academic Press (New York)

Bielajew A F 1993 The effect of strong longitudinal magnetic fields on dose deposition from electron and photon beams *Med. Phys. 20*(4) 1171–1179

Boman E, Tervo J and Vauhkonen M 2005 Modelling the transport of ionizing radiation using the finite element method *Phys. Med. Biol. 50*(2) 265–280

Booth T E, Brown F B, Bull J S, Forster R A, Goorley J T, Hughes H G, Mosteller R D, Prael R E, Sood A, Sweezy J E, Zukaitis A, Boggs M and Martz R 2003 MCNP–A General Monte Carlo N-Particle Transport Code, Version 5 Technical Report LA-UR-03-1987, Los Alamos National Laboratory

Borglund N, Eriksson J, Fumero E, Kjäll P, Mårtensson L, Orsholm C and Sikora T 2004 Geometry package for Monte Carlo simulations on CAD files *Nucl. Instrum. Meth. Phys. Res. A* **525** 417–420

Brezovich I and Barnes G 1977 A new type of grid *Med. Phys.* **4** 451–453

Bush K, Zavgorodni S F and Beckham W A 2007 Azimuthal particle redistribution for the reduction of latent phase-space variance in Monte Carlo simulations *Phys.*

Med. Biol. **52** 4345–4360

Capote R, Jeraj R, Ma C M, Rogers D W O, Sánchez-Doblado F, Sempau J, Seuntjens J and Siebers J V 2006 Phase-Space Database for External Beam Radiotherapy Technical Report INDC(NDS)-0484, International Atomic Energy Agency, Nuclear Data Section

Carrasco P, Jornet N, Duch M A, Weber L, Ginjaume M, Eudaldo T, Jurado D, Ruiz A and Ribas M 2004 Comparison of dose calculation algorithms in phantoms with lung equivalent heterogeneities under conditions of lateral electronic disequilibrium *Med. Phys.* **31** 2899–2911

Chan H P and Doi K 1983 The validity of Monte Carlo simulation in studies of scattered radiation in diagnostic radiology *Phys. Med. Biol. 28*(2) 109–129

Chang A 2001 A Survey of Geometric Data Structures for Ray Tracing Technical Report TR-CIS-2001-06, Polytechnic University (New York)

Chetty I J, Curran B, Cygler J E, DeMarco J J, Ezzell G, Faddegon B A, Kawrakow I, Keall P J, Liu H, Ma C M C, Rogers D W O, Seuntjens J, Sheikh-Bagheri D and Siebers J V 2007 Report of the AAPM Task Group No. 105: Issues associated with clinical implementation of Monte Carlo-based photon and electron external beam treatment planning *Med. Phys. 34*(12) 4818–4853

Chilton A B 1978 A note on the fluence concept *Health Phys.* **34** 715–716

Coddington P D 1994 Analysis of random number generators using Monte Carlo simulation *Int. Jour. Mod. Phys. C* **5** 547–560

Cot A, Sempau J, Pareto D, Bullich S, Pavía J, Calvino F and Ros D 2004 Study of the point spread function (PSF) for ^{123}I SPECT imaging using Monte Carlo simulation *Phys. Med. Biol.* **49** 3125–3136

Cozzi L, Fogliata A, Buffa F and Bieri S 1998 Dosimetric impact of computed tomography calibration on a commercial treatment planning system for external radiation therapy *Radiother. Oncol.* **48** 335–338

Cranley K, Gilmore B J, Fogarty G W A and Desponds L 1997 Catalogue of Diagnostic X-ray Spectra and Other Data Technical Report 78, Institute of Physics and Engineering in Medicine

Duch M A 1999 *Desarrollo de técnicas dosimétricas para su aplicación en dosimetría in vivo en terapia de alta energía* Ph. D. thesis, INTE, Universitat Politècnica de Catalunya

Fano U 1963 Penetration of protons, alpha particles and mesons *Ann. Rev. Nucl. Sci.* **13** 1–66

Fasso A, Ferrari A, Ranft J and Sala P R 2005 FLUKA: a multi-particle transport code Technical Report INFN/TC_05/11, SLAC-R-773, CERN-2005-10

Fernández-Varea J M, Carrasco P, Panettieri V and Brualla L 2007 Monte Carlo based water/medium stopping-power ratios for various ICRP and ICRU tissues *Phys. Med. Biol.* **52** 6475–6483

Fernández-Varea J M, Mayol R, Baró J and Salvat F 1993 On the theory and simulation of multiple elastic scattering of electrons *Nucl. Instrum. Meth. B* **73** 447–473

Ferrenberg A M and Landau D P 1992 Monte Carlo simulations: Hidden errors from "good" random number generators *Phys. Rev. Lett.* **69** 3382–3384

Gammel B M 1998 Hurst's rescaled range statistical analysis for pseudorandom number generators used in physical simulations *Phys. Rev. E* **58** 2586–2597

García E, Jiménez J and Puimedón J 2007 Dose calculation in patients with PENELOPE/PENGEOM *J. Phys.: Conf. Series* **74** 012006

Gifford K A, Jr J L H, Wareing T A, Failla G and Mourtada F 2006 Comparison of a finite-element multigroup discrete-ordinates code with Monte Carlo for radiotherapy calculations *Phys. Med. Biol. 51*(9) 2253–2265

Glassner A 1984 Space Subdivision for Fast Ray Tracing *IEEE Computer Graphics and Applications* **4** 15–22

Gómez-Ros J M, de Carlan L, Franck D, Gualdrini G, Lis M, López M A, Moraleda M and Zankl M 2007 Monte Carlo modelling for in vivo measurements of Americium in a knee voxel phantom: general criteria for an international comparison *Radiat. Prot. Dosim.* **(in press)**

Gómez-Ros J M, de Carlan L, Franck D, Gualdrini G, Lis M, López M A, Moraleda M, Zankl M, Badal A, Capello K, Cowan P, Ferrari P, Heide B, Henniger J, Hooley V, Hunt J, Kinase S, Kramer G H, Löhnert D, Lucas S, Nuttens V, Packer L W, Reichelt U, Vrba T, Sempau J and Zhang B 2008 Monte Carlo modelling of Germanium detectors for the measurement of low energy photons in internal dosimetry: results of an international comparison *Radiation Measurements* **(accepted)**

Hartmann-Siantar C L, Chandler W P, Rathkopf J A, Svatos M M and White R M 1995 PEREGRINE: An all-particle Monte Carlo code for radiation therapy In *Proc. Int. Conf. on Mathematics and Computations, Reactor Physics, and Environmental Analyses*, La Grange Park, Illinois, pp. 857–865 American Nuclear Society Press

Hendricks J S, McKinney G W, Fensin M L, James M R, Johns R C, Durkee J W, Finch J P, Pelowitz D B, Waters L S and Gallmeier F X 2007 MCNPX, Version 26E Technical Report LA-UR-07-6632, Los Alamos National Laboratory

Hogstrom K R, Mills M D and Almond P R 1981 Electron beam dose calculations *Phys. Med. Biol. 26*(3) 445–459

ICRU Report 48 1992 *Phantoms and Computational Models in Therapy, Diagnosis and Protection* International Commission on Radiation Units and Measurements

ICRU Report 72 2004 Dosimetry of Beta Rays and Low-Energy Photons for Brachytherapy with Sealed Sources *Journal of the ICRU* **4** 21–28

James F 1990 A review of pseudorandom number generators *Comput. Phys. Commun.* **60** 329–344

James F 1994 RANLUX: A Fortran implementation of the high-quality pseudorandom number generator of Luscher *Comput. Phys. Commun.* **79** 111–114

Jiang H and Paganetti H 2004 Adaptation of GEANT4 to Monte Carlo dose calculations based on CT data *Med. Phys.* **31** 2811–2818

Jornet N, Carrasco P, Jurado D, Ruiz A, Eudaldo T and Ribas M 2004 Comparison study of MOSFET detectors and diodes for entrance in vivo dosimetry in 18 MV x-ray beams *Med. Phys. 31*(9) 2534–2542

Kalos M H and Whitlock P A 1986 *Monte Carlo Methods (Volume 1)* New York: John Wiley and Sons

Kawrakow I 2000a Accurate condensed history Monte Carlo simulation of electron transport. II. Application to ion chamber response simulations *Med. Phys. 27*(3) 499–513

Kawrakow I 2000b VMC++, electron and photon Monte Carlo calculations optimized for Radiation Treatment Planning In A. Kling, F. Barao, M. Nakagawa, L. Távora, and P. Vaz (Eds.), *Proc. Monte Carlo 2000 Meeting*, pp. 229–238

Kawrakow I and Bielajew A F 1998 On the condensed history technique for electron transport *Nucl. Instrum. Meth. Phys. Res. B* **142** 253–280

Kawrakow I and Rogers D W O 2006 The EGSnrc Code System: Monte Carlo Simulation of Electron and Photon Transport Technical Report PIRS-701, National Research Council of Canada

Keall P J and Hoban P W 1996 Super-Monte Carlo: A 3-D electron beam dose calculation algorithm *Med. Phys. 23*(12) 2023–2034

Knöös T, Ahnesjö A, Nilsson P and Weber L 1995 Limitations of a pencil beam approach to photon dose calculations in lung tissue *Phys. Med. Biol. 40*(9) 1411–1420

Knöös T, Nilsson M and Ahlgren L 1985 A method for conversion of hounsfield number to electron density and prediction of macroscopic pair production cross-sections *Radiother. Oncol.* **5** 337–345

Knöös T, Wieslander E, Cozzi L, Brink C, Fogliata A, Albers D, Nyström H and Lassen S 2006 Comparison of dose calculation algorithms for treatment planning in external photon beam therapy for clinical situations *Phys. Med. Biol. 51*(22) 5785–5807

Knuth D E 1973 *The Art of Computer Programming (Volume 1)* (Second ed.). Addison-Wesley

Knuth D E 1998 *The Art of Computer Programming (Volume 2)* (Third ed.). Addison-Wesley

Krieger T and Sauer O A 2005 Monte Carlo versus pencil-beam/collapsed-cone dose calculation in a heterogeneous multi-layer phantom *Phys. Med. Biol. 50*(5) 859–868

Kryeziu D, Tschurlovits M, Kreuziger M and Maringer F J 2007 Calculation of calibration figures and the volume correction factors for ^{90}Y, ^{125}I, ^{131}I and ^{177}Lu radionuclides based on Monte-Carlo ionization chamber simulation method *Nucl. Instrum. Meth. A* **580** 250–253

Kyprianou I S, Badal A, Badano A, Banh D P, Freed M, Myers K J and Thompson L 2007 Monte Carlo simulated coronary angiograms of realistic anatomy and pathology models In K. R. Cleary and M. I. Miga (Eds.), *Medical Imaging 2007: Visualization and Image-Guided Procedures*, Volume 6509 of *Proc. SPIE*

Kyprianou I S, Brackman G, Badal A, Myers K J and Badano A 2008 An efficient depth- and energy-dependent Monte Carlo model for columnar CsI detector In *Medical Imaging 2008: Physics of Medical Imaging*, to be published in Proc. SPIE

L'Ecuyer P 1988 Efficient and Portable Combined Random Number Generators *Commun. ACM* **31** 742–749

L'Ecuyer P 1990 Random numbers for simulation *Commun. ACM* **33** 85–97

L'Ecuyer P and Côté S 1991 Implementing a random number package with splitting facilities *ACM Trans. Math. Soft.* **17** 98–111

Li X A and Hendee W R 2007 Radiation Oncology Physicists Will Need to Better Understand Medical Imaging *J. Am. Coll. Radiol.* **4** 40–44

Lindell B, Dunster H J and Valentin J 1998 *International Commission on Radiological Protection: History, Policies, Procedures* ICRP booklet

Luescher M 1994 A Portable High-Quality Random Number Generator for Lattice Field Theory Simulations *Comput. Phys. Commun.* **79** 100–110

Marsaglia G 1968 Random numbers fall mainly in the planes *Proc. Nat. Acad. Sci.* **61** 25–28

Marsaglia G, Narasimhan B and Zaman A 1990 A random number generator for PC's *Comput. Phys. Commun.* **60** 345–349

Martens C, Reynaert N, Wagter C D, Nilsson P, Coghe M, Palmans H, Thierens H and Neve W D 2002 Underdosage of the upper-airway mucosa for small fields as used in intensity-modulated radiation therapy: A comparison between radiochromic film measurements, Monte Carlo simulations, and collapsed cone convolution calculations *Med. Phys.* **29** 1528–1535

Mascagni M 2000 SPRNG: A scalable library for pseudorandom number generation *ACM Trans. Math. Soft.* **26** 436–461

Mathy F, Guillemaud R and Picone M 2003 Experimental validation of a coupled photon Monte Carlo and CAD software *Nucl. Instrum. Meth. Phys. Res. A* **504** 317–320

Matsumoto M and Nishimura T 1998 Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator *ACM Trans. Modeling Comput. Simul.* **8** 3–30

Matsumoto M and Nishimura T 2000 Dynamic Creation of Pseudorandom Number Generators In *Monte Carlo and Quasi-Monte Carlo Methods*, pp. 56–69 Springer

Meagher D 1982 Geometric modeling using octree encoding *Computer Graphics and Image Processing* **19** 129–147

Ménard S 2004 Summary of the P7 problem: peak efficiencies and pulse height distribution of a photon Ge spectrometer in the energy range below 1 MeV In G. Gualdrini and P. Ferrari (Eds.), *Intercomparison on the usage of computational codes in radiation dosimetry*, pp. 289–303 ENEA (Italy)

Mendes B and Pereira A 2003 Parallel Monte Carlo Driver (PMCD)—a software package for Monte Carlo simulations in parallel *Comput. Phys. Comm.* **151** 89–95

Möller T and Trumbore B 1997 Fast, Minimum Storage Ray-Triangle Intersection *Jour. Graphics Tools 2*(1) 21–28

Moskvin V and Papiez L 2005 Voxel penelope for radiation therapy applications in proceedings of MC2005, American Nuclear Society Topical Meeting in Monte Carlo, Chattanooga, TN, USA, April 17-21 Available online from http://rsicc.ornl.gov/rsiccnew/MC2005_Presentations/MoskvinPresentation.pps

Nahum A E 1978 Water/air stopping power ratios for megavoltage photon and electron beams *Phys. Med. Biol.* **23** 24–38

Nahum A E 1996 Perturbation effects in dosimetry: Part I. Kilovoltage x-rays and electrons *Phys. Med. Biol. 41*(9) 1531–1580

Nath R, Anderson L L, Luxton G, Weaver K A, Williamson J F and Meigooni A S 1995 Dosimetry of interstitial brachytherapy sources: Recommendations of the AAPM Radiation Therapy Committee Task Group No. 43 *Med. Phys. 22*(2) 209–234

National Library of Medicine The Visible Human Project http://www.nlm.nih.gov/research/visible

Nelson W R, Hirayama H and Rogers D W O 1985 The EGS4 code system Technical Report SLAC-265, Stanford Linear Accelerator Center

Neuenschwander H and Born E J 1992 A macro monte carlo method for electron beam dose calculations *Phys. Med. Biol. 37*(1) 107–125

Panettieri V 2008 personal communication

Panettieri V, Duch M A, Jornet N, Ginjaume M, Carrasco P, Badal A, Ortega X and Ribas M 2007 Monte Carlo simulation of MOSFET detectors for high-energy photon beams using the PENELOPE code *Phys. Med. Biol. 52*(1) 303–316

Panettieri V, Sempau J and Andreo P 2008 Monte Carlo simulation of three ionisation chambers in a ^{60}Co beam *(in preparation)*

Panettieri V, Wennberg B, Gagliardi G, Duch M A, Ginjaume M and Lax I 2007 SBRT of lung tumours: Monte Carlo simulation with PENELOPE of dose distributions including respiratory motion and comparison with different treatment planning systems *Phys. Med. Biol. 52*(14) 4265 – 4281

Park S 2008 personal communication

Piegl L A and Tiller W 1997 *The Nurbs Book* Springer

Press W H, Teukolsky S A, Vetterling W T and Flannery B P 1997 *Numerical Recipes in Fortran 77: The Art of Scientific Computing. (2nd edition)* Cambridge University Press

Revelles J, Ureña C and Lastra M 2000 An Efficient Parametric Algorithm For Octree Traversal In *Proc. Winter School On Computer Graphics*

Reynaert N, Smedt B D, Coghe M, Paelinck L, Duyse B V, Gersem W D, Wagter C D, Neve W D and Thierens H 2004 MCDE: a new Monte Carlo dose engine for IMRT *Phys. Med. Biol. 49*(14) N235–N241

Reynaert N, van der Marck S C, Schaart D R, der Zee W V, Vliet-Vroegindeweij C V, Tomsej M, Jansen J, Heijmen B, Coghe M and Wagter C D 2007 Monte Carlo treatment planning for photon and electron beams *Radiat. Phys. Chem.* **76** 643–686

Rogers D W O and Mohan R 2000 Questions for comparison of clinical Monte Carlo codes In W. Schlegel and T. Bortfeld (Eds.), *The Use of Computers in Radiation Therapy* Springer

Salembier C, Lavagnini P, Nickers P, Mangili P, Rijnders A, Polo A, Venselaar J and Hoskin P 2007 Tumour and target volumes in permanent prostate brachytherapy: A supplement to the ESTRO–EA–EORTC recommendations on prostate brachytherapy *Radiother. Oncol.* **83** 3–10

Salvat F 1998 Simulation of electron multiple elastic scattering *Radiat. Phys. Chem.* **53** 247–256

Salvat F, Fernández-Varea J M and Sempau J 2006 *PENELOPE–2006: A code system for Monte Carlo simulation of electron and photon transport* Nuclear Energy Agency (OECD) Issy-les-Moulineaux. Available in PDF at http://www.nea.fr

Scanlon P J, Faxon D P, Audet A M, Carabello B, Dehmer G J, Eagle K A, Legako R D, Leon D F, Murray J A, Nissen S E, Pepine C J and Watson R M 1999 ACC/AHA guidelines for coronary angiography: executive summary and recommendations: a report of the American College of Cardiology/American Heart Association Task Force on Practice Guidelines (Committee on Coronary Angiography) *Circulation* **99** 2345–2357

Schneider W, Bortfeld T and Schlegel W 2000 Correlation between CT numbers and tissue parameters needed for Monte Carlo simulations of clinical dose distributions *Phys. Med. Biol. 45*(2) 459–478

Segars W P 2001 *Development of a new dynamic NURBS-based cardiac-torso (NCAT) phantom* Ph. D. thesis, University of North Carolina

Segars W P, Tsui B M W, Frey E C and Fishman E K 2003 Extension of the 4D NCAT phantom to dynamic X-ray CT simulation *Nucl. Sci. Symp. Conf. IEEE* **5** 3195–3199

Sempau J, Acosta E, Baró J, Fernández-Varea J M and Salvat F 1997 An algorithm for Monte Carlo simulation of coupled electron-photon transport *Nucl. Instrum.*

Meth. Phys. Res. B **132** 377–390

Sempau J and Andreo P 2006 Configuration of the electron transport algorithm of PENELOPE to simulate ion chambers *Phys. Med. Biol. 51*(14) 3533–3548

Sempau J, Asenjo J, Sánchez-Reyes A and Badal A 2003 DPM: cálculo de la dosis en radioterapia externa mediante simulación Monte Carlo Acelerada. Comparación con el planificador TMS de Helax In SEFM (Ed.), *Proc. XIV Congreso Nacional de Física Médica, Vigo (Spain)*

Sempau J, Fernández-Varea J M, Acosta E and Salvat F 2003 Experimental benchmarks of the Monte Carlo code PENELOPE *Nucl. Instrum. Meth. Phys. Res. B* **207** 107–123

Sempau J, Sánchez-Reyes A, Salvat F, Oulad ben Tahar H, Jiang S B and Fernández-Varea J M 2001 Monte Carlo simulation of electron beams from an accelerator head using PENELOPE *Phys. Med. Biol. 46*(4) 1163–1186

Sempau J, Wilderman S J and Bielajew A F 2000 DPM, a fast, accurate Monte Carlo code optimized for photon and electron radiotherapy treatment planning dose calculations *Phys. Med. Biol. 45*(8) 2263–2291

Snyder W S, Ford M R, Warner G G and Jr H L F 1969 Estimates of absorbed fractions for monoenergetic photon sources uniformly distributed in various organs of a heterogeneous phantom Technical Report 5, MIRD pamphlet

Soubra M, Cygler J and Mackay G 1994 Evaluation of a dual bias dual metal oxide-silicon semiconductor field effect transistor detector as radiation dosimeter *Med. Phys. 21*(4) 567–572

Srinivasan A, Mascagni M and Ceperley D 2003 Testing parallel random number generators *Parallel Computing* **29** 69–94

Stanford 3D Scanning Repository PLY format http://www-graphics.stanford.edu/data/3Dscanrep/

Sternheimer R M 1952 The Density Effect for the Ionization Loss in Various Materials *Phys. Rev.* **88** 851–859

Sterpin E, Tomsej M, Smedt B D, Reynaert N and Vynckier S 2007 Monte Carlo evaluation of the AAA treatment planning algorithm in a heterogeneous multilayer phantom and IMRT clinical treatments for an Elekta SL25 linear accelerator *Med. Phys.* **34** 1665–1677

Sulkimo J and Vuoskoski J 1996 Particle tracking in sophisticated CAD models for simulation purposes *Nucl. Instrum. Meth. Phys. Res. A* **371** 434–438

Tabary J and Glière A 2000 Coupling Photon Monte Carlo Simulation and CAD Software. Application to X-ray Nondestructive Evaluation In A. Kling, F. Barao, M. Nakagawa, L. Távora, and P. Vaz (Eds.), *Proc. Monte Carlo 2000 Meeting*

Taranenko V and Zankl M 2005 Photon and electron transport simulation in voxel geometry with PENELOPE *Biomed. Tech.* **50 suppl. 1** 271–272

van der Zee W, Hogenbirk A and van der Marck S C 2005 ORANGE: a Monte Carlo dose engine for radiotherapy *Phys. Med. Biol. 50*(4) 625–641

Vattulainen I, Kankaala K, Saarinen J and Ala-Nissila T 1995 A Comparative Study of Some Pseudorandom Number Generators *Comput. Phys. Commun.* **86** 209–226

Verhaegen F and Devic S 2005 Sensitivity study for CT image use in Monte Carlo treatment planning *Phys. Med. Biol. 50*(5) 937–946

Vilches M, Garcia-Pareja S, Guerrero R, Anguiano M and Lallena A M 2007 Monte Carlo simulation of the electron transport through thin slabs: A comparative study of PENELOPE, GEANT3, GEANT4, EGSnrc and MCNPX *Nucl. Instrum. Meth. Phys. Res. B* **254** 219–230

Walters B, Kawrakow I and Rogers D W O 2007 DOSXYZnrc Users Manual Technical Report PIRS-794, National Research Council of Canada

Walters B R B, Kawrakow I and Rogers D W O 2002 History by history statistical estimators in the BEAM code system *Med. Phys.* **29** 2745–2752

Wang B, Kim C and Xu X G 2004 Monte Carlo modeling of a High-Sensitivity MOSFET dosimeter for low- and medium-energy photon sources *Med. Phys. 31*(5) 1003–1008

Woodard H Q and White D R 1986 The composition of body tissues *Br. J. Radiol.* **59** 1209–1218

Xing L, Thorndyke B, Schreibmann E, Yang Y, Li T F, Kim G Y, Luxton G and Koong A 2006 Overview of image-guided radiation therapy *Med. Dosim. 31*(2) 91–112

Xu X G, Taranenko V, Zhang J and Shi C 2007 A boundary-representation method for designing whole-body radiation dosimetry models: pregnant females at the ends of three gestational periods—RPI-P3, -P6 and -P9 *Phys. Med. Biol. 52*(23) 7023–7044

Ye S J, Brezovich I A, Pareek P and Naqvi S A 2004 Benchmark of PENELOPE code for low-energy photon transport: dose comparisons with MCNP4 and EGS4 *Phys. Med. Biol. 49*(3) 387–397

Zaidi H 1999 Relevance of accurate Monte Carlo modeling in nuclear medical imaging *Med. Phys.* **26** 574–608

Zaidi H and Xu X G 2007 Computational Anthropomorphic Models of the Human Anatomy: The Path to Realistic Monte Carlo Modeling in Radiological Sciences *Annu. Rev. Biomed. Eng.* **9** 1.1–1.30

Zheng-Ming L and Brahme A 1993 An overview of the transport theory of charged particles *Radiat. Phys. Chem.* **41** 673–703