Nano-scale Monte Carlo simulations of proton transport

Johannes Tjelta

Thesis submitted for the degree of
Master in Medical physics
60 credits

Department of Physicss
Faculty of Mathematics and Natural sciences

UNIVERSITY OF OSLO

Spring 2021

Nano-scale Monte Carlo simulations of proton transport

Johannes Tjelta

Nano-scale Monte Carlo simulations of proton transport

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Acknowledgements

First of all, I would like to give my utmost gratitude to my supervisor Professor Eirik Malinen for his valuable and constructive feedback throughout this thesis. In addition, I would like to thank my second supervisor, Associate Professor Nina Frederike J Edin. Without them there would not have been a thesis. Furthermore, I would like to thank the BMF group for the daily academic spice.
Thank you to my older brothers for being supportive and setting me straight when I forget I'm the youngest of them. Thank you to all of my friends who listened to my whining these last months and listened to me blabber about medical physics for hours. Thank you to the PlayStation guys for fun times these two years. Finally, there are no words to articulate my profound gratitude towards my parents for the food, shelter and loving support throughout this thesis.

-Johannes Tjelta

Abstract

Introduction: By the year 2024, Norway will have two proton therapy centers although to this day not much is known about proton interaction with matter in the low energy domain. Therefore it is imperative to examine how low energy protons will deposit their energy throughout a cell culture on a nanometer scale.
Theory: Radiation and dose distribution are often calculated with Monte Carlo simulations. For larger objects such as reactors, a condensed history technique may be used. For smaller volumes such as a cell, analog Monte Carlo simulations may be employed but not with realistic dose levels.
In this thesis, the attempt is to bridge the gap between macroscopic and microscopic MC, simulating a cell culture with analog proton simulations from the Geant4-DNA toolkit implemented in a simulation program written in Python. The simulations where based on the experimental setup at the Oslo Cyclotron Laboratory for irradiating cells. In addition to irradiating a cell culture, irradiation of single cell nuclei where preformed with the same energy and dose as the irradiation of the cell culture. All event positions where stored and analyzed with several spatial autocorrelation algorithms.
Results: A dose distribution for clinical doses (1Gy-10Gy) for four different proton energies (1.2MeV, 1.5MeV, 1.8MeV and 8.7MeV) in cells and nuclei was obtained. The cell nucleus was seen to receive lower average dose than the whole cell, which was more pronounced at lower energies. In addition, the standard deviation in delivered dose per cell was seen to increase with decreasing proton energy. The same doses for the same energies were prescribed to the nuclear volume to investigate spatial clustering of events. A method called Moran's I did not provide meaningful results, but a dependency on energy derived from Geary's C algorithm was found. In addition, an Intra-track and Inter-track algorithm were derived to test spatial distribution between events and tracks, with no significant results.
Conclusion: The dose deficit between whole cells and nuclei might be a statistical phenomena not yet understood. In addition, some deviations seen indicate an underlying problem with the current simulations. Furthermore other algorithm needs to be used for spatial analytics to gain a meaningful result about linear energy transfer and how it correlates to cell death.

Contents

Chapter 1

Introduction

Wilhelm Conrad Röntgen is the man credited for the invention of the x-ray tube in 1895, producing the first x-ray picture. It took less than a month for this invention to be used for photographing a mummified cat, among other things. In addition, medical doctors began to irradiate superficial tumors such as skin carcinoma, although with much greater uncertainties than today's standards. Fast forward 100 years and we have radiation on demand for both therapeutic and diagnostic purposes.

While the x-ray tube is still in use, the technology has accelerated, literally, the development in particle accelerators. Now we have the ability to get high energy x-rays from linear accelerators and heavy charged particles accelerated to tremendous velocity with the cyclotron, both in use in the treatment of cancers. When treating cancers with radiation we want to limit the the dose to the healthy tissue while delivering a high dose to the tumor. For this purpose, the heavy charged particle are optimal. While a ionizing photons has a dose deposition throughout the patient, the heavy charged particle will deposit a large amount of its energy in a specific area, limiting the dose to healthy tissue. The virtually continuous large energy deposition by the charged particle is due to Coulomb force interactions. As the particle moves along it loses energy and thus the interactions will be greater the slower the particle moves. The small region in a medium where the particle loses the most energy is called the Bragg peak.

As most physicists are aware, quantum particles are simple but unforgivingly random. So random in fact that a special method called Monte Carlo simulations are used to simulate an many-particle outcomes.
A Monte Carlo simulations in radiation physics is a computational method where real-life scenarios are mimicked and the random nature of radiation physics is included. Monte Carlo is a method that has been used to simulate every random event one can think of, from determining the destructive power of thermonuclear weapons to predicting a stock value.

This study aims to use a Monte Carlo toolkit (Geant4-DNA) to preform nanodosimetric simulations on a cell culture with protons, mimicking ex-

perimental setup for irradiating cells at the Oslo Cyclotron Laboratory (OCL). Energies corresponding to protons in front, mid and distal end of a Bragg peak were considered. In addition, this study the aims to do spatial statistics for protons at different energies in cell nuclei to get a better grasp of how events are distributed.

Chapter 2

Theory

Sections 2.1, 2.2 and 2.5 are based on chapters 7,8 and 11 in Attix [18]
Section 2.3 is based on Microdosimetry and Its Applications [25]
Section 2.4 is based on chapter 8 in Fundamentals of Ionizing Radiation Dosimetry [36]
Section 2.6 is based on chapter 4 and 17 in The Cell [7] and Hall [19]
Section 2.7 is based on Hall [19] and [34]
Section 2.8.2 is based on [39] and [40]

2.1 Ionizing radiation

In its action on matter, ionizing radiation is highly efficient in transferring energy directly to the atom. Ionizing radiation is a broad term used for high-velocity charged particles, free neutrons, or electromagnetic radiation with ultra-high frequency. For it to be categorized as ionizing, it has to excite or ionize matter. Ionization is a transfer of energy from the ionizing radiation to an electron, liberating it from its atom or molecule. Excitation is the transfer of energy to an electron, moving it to a more energetic state but still bound to the atom. When the electron falls back to a lower energy state, it undergoes electron relaxation. An electromagnetic energy quanta is released from electron relaxation with the equivalent delta energy put into the excitation process.
When an electron is liberated from an atom, it is classified as directly ionizing radiation. Directly ionizing radiation is charged particles such as electrons and protons with enough kinetic energy to excite and ionize electrons, mainly through Coulomb interactions. Indirectly ionizing radiation is neutrons and highly energetic photons. These particles act as a liberator of charged particles, which is the main contributor to ionization clusters.
The energy needed to ionize matter, or ionization potential, is the amount of energy needed to ionize an electron from an atom or molecule. The energy needed to overcome the binding energy of the atom is in the range of 4eV-24eV, with the Nobel gasses requiring the most energy and the alkali metals requiring the least.

2.1.1 Cross section

The cross section is denoted σ and is measured in m^2 and is proportional to the interaction strength between the stationary and incoming particle. It is a measure of probability for an interaction to take place, in this context for an interaction between a ionizing particle and a atom.

Differential cross section

The differential cross section describes the likelihood of radiant energy scattered per solid angle. Classically, this can be explained with a ionizing particle scattering on a stationary particle.

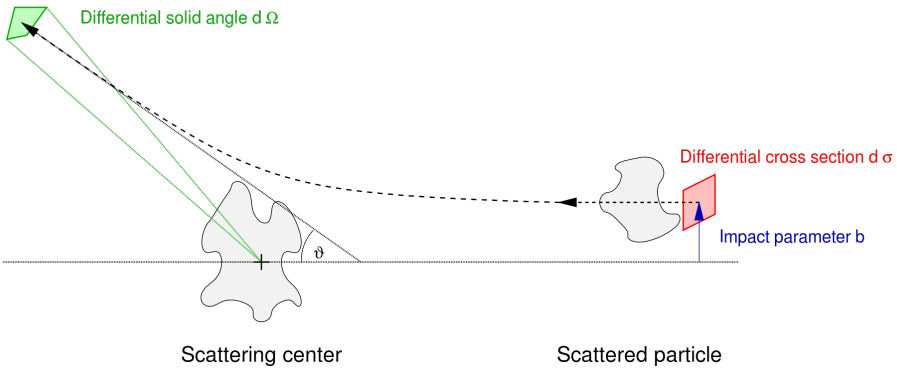


Figure 2.1: Illustration of the scattering and the differential cross section. [38]

In figure 2.1 the impact parameter b is the offset for the incoming particle parallel to the scattering center. When the particle moves along towards the scattering center, a force will act on it from the scattering center. The traveling particle will be scattered at an angle θ. In a classical sense, this is not a problem, but generally, when working on the atomic scale, the parameters b and scattering center can not be controlled (the process is stochastic). Solving this requires several scattering events and measuring the beam/particle's scatter angle.
The differential cross section can be deduced to be a plane perpendicular to the incident particle; $d\sigma = bd\phi d\theta$ as shown in figure 2.1. When a incoming particle scatters of the scattering center, it will have a probability to scatter in a solid angle. The solid angle is described as $\Omega = \mathrm{A}/r^2$ where A is the area of the sphere and r is the distance to the scatter center. The infinitesimal amount of a solid angle is $d\Omega = d\mathrm{A}/r^2 = sin(\theta)d\theta d\phi$. If there is cylindrical symmetry the differential angle $d\phi$ integrated over all angles becomes 2π so $d\sigma = 2\pi bd\theta$ and $d\Omega = 2\pi sin\theta d\theta$. The differential of these two are known as the differential scattering cross section.

$$\frac{d\sigma}{d\Omega} \tag{2.1}$$

The total cross section (σ) may be recovered by integrating over the total solid angle for the differential scattering cross section. This can be seen in

equation 2.2:

$$\sigma = \oint_{4\pi} \frac{d\sigma}{d\Omega} d\Omega \tag{2.2}$$

Not only is the differential cross section useful to predict the scattering angle but the solid angle may be replaced with energy or momentum transfer to provide the likelihood for a given energy transfer.

2.1.2 Photon interaction

As mentioned, highly energetic photons are indirectly ionizing radiation. Categories of photons with enough energy to ionize are extreme ultraviolet, X-rays and gamma rays. An ionizing photon will only have a couple of interactions in a medium. There are two reasons for this, the interaction cross section is relatively small, and the energy transfer cross section is large. Generally, a photon will rarely interact, but if it is interacting, it will release a relatively large amount of energy. The three most important interactions are the Compton effect, the photoelectric effect and pair production.

Compton effect

The Compton effect is a collision between an electron and a photon. The result of this collision is a photon with less energy and a liberated electron with an amount of kinetic energy equal to the energy the photon lost. First, the kinematics of the Compton effect. Kinematics is the mathematical construct for calculating energies and angles for both particles participating in the interaction. In kinematics, the electron is approximated to be stationary.

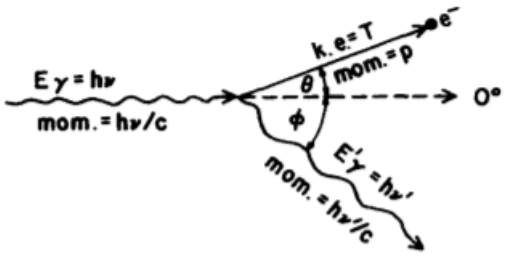


Figure 2.2: A simple sketch portraying the kinematics of the Compton effect. The wave is the photon in the interaction, the straight line is the scattered electron. ϕ and θ are the scattering angles for photon and electron respectively.[18]

Kinematics, the mathematical construct of a singular high energy photon and a free electron. The incoming photon has an energy $E_\gamma = h\nu$ with the electron at rest. The photon will collide with the free-electron giving the electron some kinetic energy T and an angle θ. The photon going out of the interaction will have reduced energy $h\nu' = h\nu - T$.
Conservation of momentum along the photon's original path (0^o) can be expressed:

$$h\nu = h\nu' cos(\theta) + pccos(\phi)$$

p is the momentum of the electron after the interaction.
The Compton effect's probability for interaction for a given energy and atomic number Z are given by the differential cross section. From the differential cross section it is possible to calculate deflection angle probability for the photon and electron. The differential cross section might also be modified with respect to energy. This results in equation 2.3:

$$\frac{d\sigma}{d(h\nu')} = \frac{\sigma}{d\Omega}\frac{d\Omega}{d(h\nu')} \tag{2.3}$$

This will indicate what energy transfer is expected in the interactions.
The cross section for Compton scattering depends on the photon energy and the effective atomic number. A rule of thumb is that the cross section decreases with increasing effective atomic number and increases with photon energy until pair production becomes prominent.

Photoelectric effect

The photoelectric effect is one of Einstein's many discoveries which got him a Nobel prize in physics. In his paper, he described photons as small quanta of energy crashing with electrons and liberating them from its atoms. In this process, the photon is absorbed, and almost all its energy is manifested in the electron's kinetic energy. In the photoelectric effect, one must consider the electron's binding energy when calculating the electron's kinetic energy. There is also a small contribution to the atom recoiling to conserve momentum along the incoming photon path ($\theta = 0^o$), as shown in figure 2.3. Effectively, the atom will not contribute significantly to further interactions. One could compare this interaction to throwing a marble at a bowling ball, where the bowling ball is the atom.

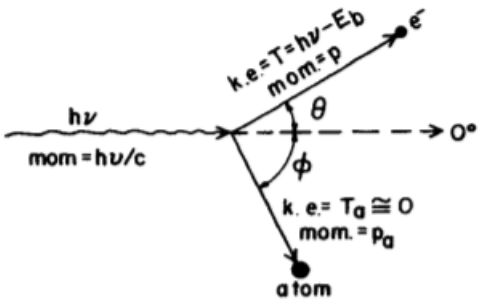


Figure 2.3: As in figure 2.2 a simple sketch to show the kinematics for the photoelectric effect. The photon are the wave, the upper straight arrow is the electron and the lower is the momentum transfer to the atom.[18]

The kinematics for photoelectric shown in figure 2.3, an incoming photon with energy $h\nu$ and an outgoing electron with the kinetic energy $T = h\nu - E_b$. E_b is the binding energy for the electron. The reason the binding energy is included in the photoelectric effect and not Compton scattering is that the photoelectric effect's energies are lower. Therefore, the binding energy comprises a non-negligible part of the equation. Compared to the

Compton effect, it can be seen in figure 2.3 that the photoelectric effect does not leave a residual photon.
The photoelectric effect is most likely to happen when the photon involved in the interaction has relatively low energy. It is more likely to happen when the material is composed of a high atomic number.

$$\tau \propto \frac{Z^m}{h\nu^n} \tag{2.4}$$

$m \in (4,5)$ and $n \in (1,3)$. τ is the denotation for the cross section for the photoelectric effect. The likelihood for this interaction is exponentially larger for a higher atomic number. An example of this might be bone, having a high concentration of calcium and soft tissue, mostly composed of water. This is exactly the prinsipal behind an X-ray image, where the X-rays are more likely to be absorbed in the bone and pass trough soft tissue.

Pair production

Pair production is a process where the photon is totally absorbed in its interaction with the atom's nuclear field, resulting in one positron (anti-particle of an electron) and one electron with kinetic energy T+ and T- respectively.
Due to the nature of converting energy to mass and the fact that energy cannot be created nor destroyed, the threshold for this effect is 1.022MeV. As in the photoelectric effect and the Compton effect, the nucleus will have a recoil effect to conserve momentum, though this is negligible.

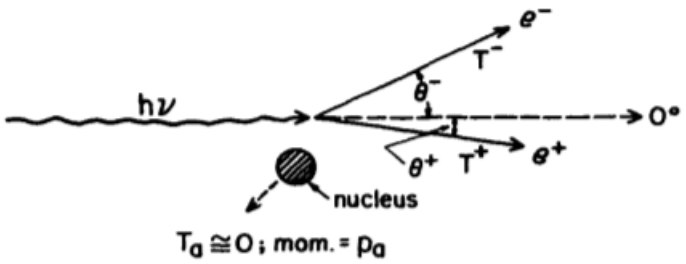


Figure 2.4: A simple sketch of the kinematics behind pair production. The incoming photon is the wave and the positron/electron are the straight lines.[18]

An estimation of the cross section of pair production can be expressed in the form of equation 2.5:

$$\kappa \propto Z^2 P(h\nu) \tag{2.5}$$

Where κ is the standard notation for the pair production cross section.

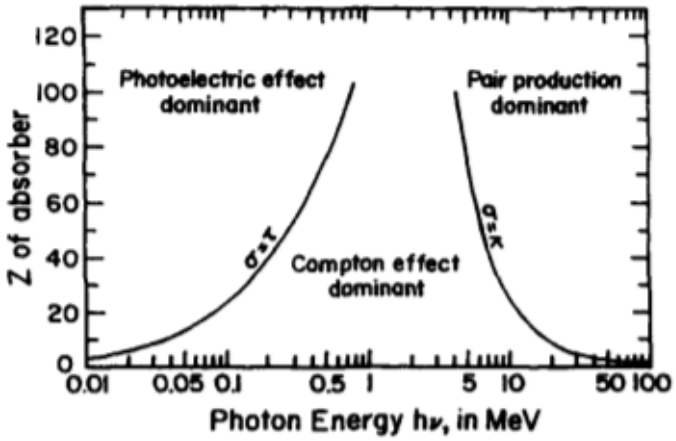


Figure 2.5: A simple plot showing where the dominance of the different interactions take place. Effective atomic number on the y axis and photon energy at the x axis.[18]

Figure 2.5 provides an overview of when the 3 different interactions discussed are most dominant. The photoelectric effect will be most dominant for materials with high effective atomic number (Z_{eff}) and low energies. The pair production will be more dominant for high Z_{eff} and high energies. The Compton effect is a place in the middle of these two, for low Z_{eff} and mid-range energies.

2.1.3 Fluence

A sphere is placed in a radiation field. The sphere is in this field from time t_0 to t. The fluence is expressed as the number of particles dN per area da, and the differential quotient between these two.

$$\Phi = \frac{dN}{da} \tag{2.6}$$

So in its simplest form, the fluence is the number of particles (dN) traversing through a plane (da) at a point and is an expectation value, and is expressed in the unit m^{-2} and independent of the beam's energy and rate. The rate of fluence from a time $t = t_0$ to $t = t_{max}$ can define the fluence rate.

$$\phi = \frac{d\Phi}{dt} \tag{2.7}$$

It has the unit of $m^{-2}s{-}2$, and covers the amount of particles per unit time. Solving the energy dependency is not that simple. Energy fluence considers the particles traversing the surface by integrating over all the beam's energy levels.

$$\Psi = \int_0^{T_{max}} T\Phi_T dT$$

Where Φ_T is the differential fluence of particles per area and energy interval. The power of the energy fluence is the fact the equation considers the number of particles and their energy.

2.1.4 Attenuation

Attenuation is defined as the loss of intensity of a beam, normally of photons, as it traverses through a medium. Loss of intensity in the beam is due to scattering and absorption. The beam will have a loss of dN when traversing through a plane with dx thickenss. The number dN is dependent on the original number of particles N and a constant μ, to be discussed in section 2.1.5.

$$dN = N\mu dx$$

By integrating, we get the formula for the number of primary photons at a given depth.

$$N = N_0 e^{-\mu x} \tag{2.8}$$

From equation 2.8 it is possible to calculate when the primary beam is reduced to half the original intensity $N = 1/2N_0$ where N_0 is the original number of particles, and then solve for x. The thickness of material needed is called the half-value layer, for equation 2.8 this is $x = -ln(1/2)/\mu$.
Furthermore, if the primary beam is reduced to $\approx$ 37% it can be derived from equation 2.8 that $1/\mu = x_{37}$, and is known as the mean free path. The mean free path is the average distance traveled for a given particle in a given medium before interacting. For example with $\mu = 0.1183 cm^2 g^{-1}$ in solid ^{214}Pb with γ-ray with a energy of 241.98keV (found in [2]) the mean free path is 8.45cm. The mean free path will increase with increasing energy. μ is referred to as the linear attenuation coefficient or just attenuation coefficient.

2.1.5 Attenuation coefficients

Each interaction discussed previously for photons has its own cross section for interaction. By summarizing these cross sections divided by the density of the material (ρ), the total mass attenuation coefficient is attainable in equation 2.9:

$$\frac{\mu}{\rho} = \frac{\tau}{\rho} + \frac{\sigma}{\rho} + \frac{\kappa}{\rho} + \frac{\sigma_R}{\rho} \tag{2.9}$$

μ/ρ gives an attenuation estimation for a beam of photons. The σ_R component is from an interaction called Rayleigh scattering. This is a coherent scattering process where the photon does not lose energy and will not be further discussed.

The mass attenuation coefficient summarizes the total cross section for the photon interactions discussed previously. The mass attenuation coefficients are an indicator of primary photons still left in a beam.

Suppose it is of interest to find the amount of energy transferred to a medium. μ_{tr} is the mass energy-transfer coefficient. It is found much in the same manner as 2.3 only with regards to the mean energy of the electrons ($\bar{T}$) for each individual interaction. This is done for all of the cross sections

for the different interactions and summed in the same manner as in equation 2.9.

$$\frac{\mu_{tr}}{\rho} = \frac{\mu}{\rho}\frac{\bar{T}}{h\nu}$$

The μ_{tr}/ρ is the mass-energy transfer coefficient and considers the energy transferred to the medium as this coefficient only gives an estimate of the loss of energy from a primary beam, transferred to charged particles as kinetic energy.

Not all of the energy transferred to the electrons is deposited locally. This needs correcting and brings us to the phenomena known as bremsstrahlung, which is the loss of kinetic energy from an electron near an atom as it redirect its trajectory, this will result in a residual photon. This process happens with a fraction (g) for electrons with a certain kinetic energy, and are more likely to happen with increasing electron kinetic energy. Since it only happens to a fraction of the energy imparted, and this fraction is constant for a given situation, it can be implemented in the mass-energy transfer coefficient. The contribution of bremsstrahlung is g, so (1-g) is the total fraction deposited locally since photons created in the bremsstrahlung effect will likely leave the local volume.

$$\frac{\mu_{en}}{\rho} = \frac{\mu_{tr}}{\rho}(1-g)$$

$\mu_{en}/$ is the mass energy-absorption coefficient. This contribution is a bit tricky. The reason for this is the likelihood for a photon to interact locally is slight. Therefore it is factored out of the energy deposited locally.

2.1.6 Charged particles

Charged particles will interact with matter in the form of Coulomb-force interactions. The way charged particles interact can be described by an impact parameter b (from section 2.1.1), and the atomic radius a. This is shown in figure 2.6.

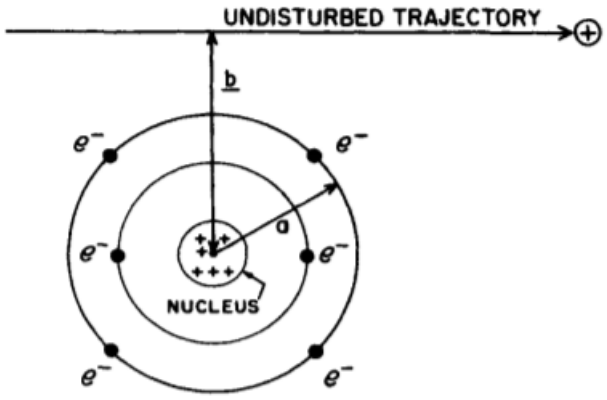


Figure 2.6: The parameters b and a is distance from the atom nuclei and the charged particle and distance from the classical atomic radius. Here shown by the Bohr atom model.[18]

A charged particle's likelihood to interact with matter is much larger than photons due to the Coulomb force interaction. This force will be imparted on almost every atom along the particle's trajectory. Furthermore, when inspecting the total cross section for an charged particle ($10^{18} - 10^{-17} cm^{-2}$) it is much larger than the photon cross section ($10^{-24} - 10^{-22} cm^{-2}$). It is possible to calculate that an electron will have one interaction per $10^{-6} - 10^{-5}$cm. This can be compared to the γ-ray example (section 2.1.4) which have a mean free path of $\approx 8cm$.

Stopping power

We can describe the expectation value for loss of energy for a charged particle given a distance x, the kinetic energy T, the medium's atomic number Z and the charge of the particle z. This is called stopping power and is the quotient of differential energy over differential distance, dE/dx. The loss of energy will result in events (ionizations, excitations, etc.) and is a measure of ionization density. The unit of stopping power is often given in MeV/cm or $keV/\mu m$.
The local depositions for a charged particle is refereed to as restricted linear energy transfer (LET_Δ), it is measured in the same units as stopping power (MeV/cm or $keV/\mu m$). The LET_Δ is the energy transfer for a minimum energy E_{min} to a max Δ, and can be calculated by integrating the differential energy transfer cross section from E_{min} to Δ:

$$LET_\Delta = \int_{E_{min}}^{\Delta} E \frac{d\sigma}{dE} dE \tag{2.10}$$

if the $\Delta = E_{max} = \infty$ the LET_Δ becomes unrestricted linear energy transfer (LET) and is equal to the stopping power.
If the density of the material in question is a gaseous substance like air, the traversing particle will lose less energy per unit length. If the substance is a solid, the particle will lose more energy per unit length. The atoms in the solid substance are packed tighter than in the gas and the traversing particle will have more interactions. To avoid the problem, stopping power is divided by the density, ρ, to give the mass stopping power [$MeV/cm^2 g$].

The stopping power term is by convention divided into two categories, the soft collision term and the hard collision term, as shown by equation 2.11:

$$\left(\frac{dT}{\rho dx}\right)_c = \left(\frac{dT_s}{\rho dx}\right)_c + \left(\frac{dT_h}{\rho dx}\right)_c \tag{2.11}$$

The soft collision term

When the charged particle passes through a medium, the bulk of interactions between the charged particle and atoms will be soft collisions. In a soft collision, the particle pass at a considerable distance from the atom, i.e., $b >> a$ from figure 2.6.
This interaction will result in the ionization of electrons with low binding

energy, such as the valence bond electrons and excitations. The energy transfer of the interaction is often in the low end of the spectrum, transferring a few eV in each interaction. The first part of equation describes the soft collision term 2.11.

$$\left(\frac{dT_h}{\rho dx}\right)_c = \frac{2Cm_0c^2z^2}{\beta^2}\left[ln\left(\frac{2m_0c^2\beta^2H}{I^2(1-\beta^2)}\right) - \beta^2\right] \tag{2.12}$$

In equation 2.12 β is the relativistic speed of the particle, and if the speed increases the stopping power decreases. The $C = \pi(N_AZ/A)r_0^2$ where the N_AZ/A is the number of electrons per gram in the medium and $r_0^2 = e^2/m_c^2 = 2.818 \times 10^{-13}cm$ m_0c^2 is the rest-mass of the electron ($= 511keV$) The $2Cm_0c^2z^2/\beta^2$ can be simplified to $k = 0.1535Zz^2/\mathrm{A}\beta^2$. The stopping power is also dependent on the traversing medium's electron density (Z/A) linearly and the charge (z^2) quadratically. The I in equation 2.12 is the mean ionization potential or often referred to as the mean excitation potential. It is a mean value of all of the excitation and ionization potentials of an atom in a given medium. It was first thought to be dependent on the particle's velocity, but, in the end, through experiments, it was shown only to be dependent on the medium.

$$\bar{I} \propto kZ \tag{2.13}$$

Where k is an constant, $k \sim 10eV$. Equation 2.13 was Bloch's first approximation. Later, I/Z versus Z has been shown to not be constant for different Z.

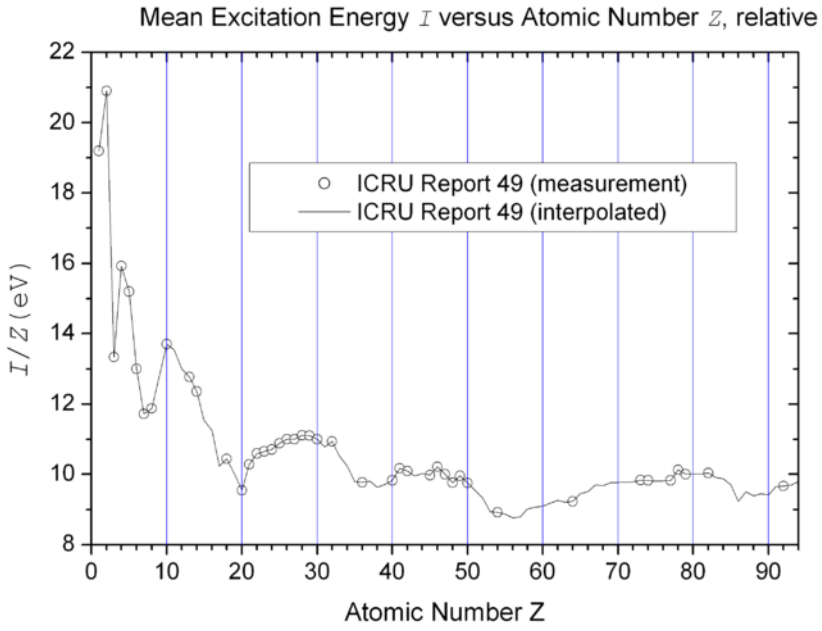


Figure 2.7: ICRU reported numbers for I for different atomic numbers. [1]

In figure 2.7 it can be seen that the ionization potential is a non-analytical quantity and needs to be empirical data. It is often measured with a thin film of material and heavy nuclei to interact. The heavy nuclei need to have sufficient energy to surpass the contribution of elastic collisions ($T_{prot} > 20keV$ and $T\alpha > 150keV$), and not too high energy, so the contribution of polarization is negligible. (elastic collisions will be discussed in **Elastic collisions** and polarization in **Corrections to the Bethe-Bloch**)

The hard collision term

When the particle traverse's through the shell of the atom, i.e., $b \approx a$, the particle has a chance of having a hard collision. The hard collision is an interaction between electromagnetic fields of two particles and can be defined by equation 2.14:

$$\frac{dT_h}{\rho dx} = k\left[ln\left(\frac{T_{max}}{H}\right) - \beta^2\right] \tag{2.14}$$

It can be favorable to distinguish between the soft term and the hard term. When $b \approx a$, the energy transfer can be considerable. A liberated electron will have enough energy to ionize matter by itself.
Whenever an electron is ejected via a hard collisions, it has considerable kinetic energy. This electron type is often referred to as an δ-electron or a knock-on electron. If this is the case, the electron can be considered "free".
The hard collision term will account for about 50% av the lost energy. Equation 2.12 and 2.14 can be combined to obtain the mass collision stopping power:

$$\left(\frac{dT}{\rho dx}\right)_c = k\left[ln\left(\frac{2m_0c^2\beta^2T_{max}}{I^2(1-\beta^2)}\right) - 2\beta^2\right] \tag{2.15}$$

Corrections to the Bethe-Bloch

Two scenarios needs to be considered when discussing a charged particle traversing through a medium. One part is the relativistic nature of particles with sufficient energy. The other is the implementation of Born approximation. Born approximation assumes the incident particle has a velocity (v) greater than the maximum Bohr-orbit velocity (u), $v = \beta c > u$.

First, the relativistic properties of a particle. From Einstein's theory of relativity, it is known as an object gets closer to the speed of light (c), the energy needed to accelerate the object further increases exponentially. The same is true for charged particles. This is implemented in β in the Bethe formula.

$$T = M_0c^2\left[\frac{1}{\sqrt{1-\beta^2}-1}\right] \quad \text{and} \quad \beta = \left[1-\left(\frac{1}{(T/M_0c^2)+1}\right)^2\right]^{1/2} \tag{2.16}$$

Where $\beta \in [0, 1\rangle$

Secondly, Bethe's formula for the low energy case. The assumption for Bethe's formula is $(Z_{eff}/137\beta)^2 \ll 1$, where Z_{eff} is the effective atomic number.

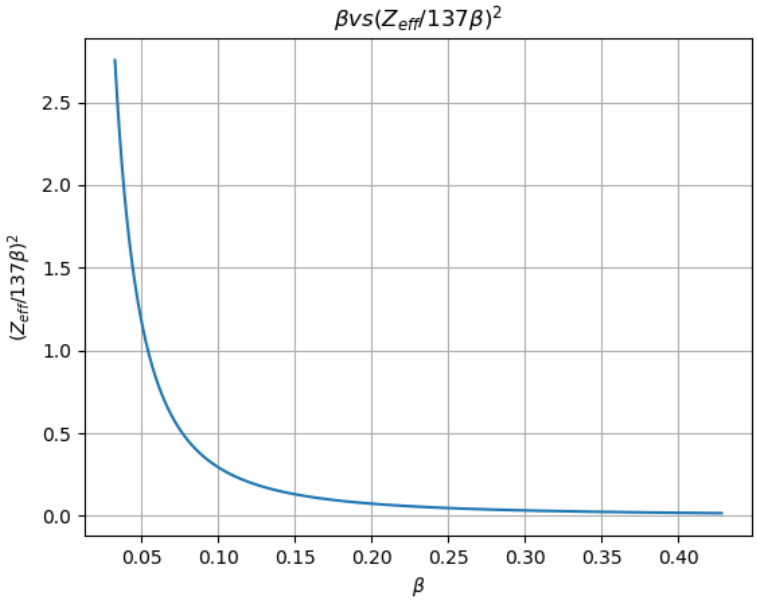


Figure 2.8: β vs $(Z_{eff}/137\beta)^2$. In this specific program, the parameters were water as the irradiated substance and protons as the source. (Own program)

By plotting the assumption versus charged particle velocity as shown in figure 2.8 it is possible to find where the assumption fails. Figure 2.8 indicates that this happens around $\beta < 0.1$. The particle velocity is less than the orbiting electrons in this energy range. This is a cascading effect, first happening to the K-shell electrons since they are the fastest. Implementing a correction term (C/Z) in the Bethe-Block equation's brackets makes it possible to correct this. C/Z is constant for all charged particles with equal velocity.

$$\left(\frac{dT}{\rho dx}\right)_c = k\left[ln\left(\frac{2m_0c^2\beta^2T_{max}}{I^2(1-\beta^2)}\right) - 2\beta^2 + \frac{C}{Z}\right] \tag{2.17}$$

One effect which should be mentioned is the polarization or density effect. This effect is not significant for gasses but relevant in condensed media.
The density effect impacts the soft collision term of the Bethe-block equation. In essence, the effect tackles the impact of how dipole distortion in a condensed medium by atoms near the track decreases the Coulomb force felt by more distant atoms. The effect will decrease the stopping power, and hence the particle will travel further and have lower LET. Correction for this effect is denoted δ and included in the equation's brackets 2.17.

Elastic collisions

When the impact parameter is much smaller than the atomic radius, i.e., a»b, the particle may be deflected. This deflection is due to the Coulomb interaction between the incident particle and the atomic nucleus. Even though this is an elastic collision, the incident particle will lose some of its kinetic energy due to the conservation of momentum along the particle's original path. This energy is imparted onto the atom. But for heavier nuclei such as protons, alpha, and so on, this effect will be negligible at higher energies. It will account for about 1% of total energy loss down to 20keV and 150keV for protons and alpha, respectively. Bellow this kinetic energy threshold, the elastic collisions will increase exponentially. When considering particles below these energies, the elastic collisions need to be accounted for.

CSDA and projected range

Continuously Slowing Down Approximation is a good approximation to calculate the expected range because charged particles lose energy in a quasi-continuous manner. CSDA for our purpose is the same as range ($\Re$). The range is defined as the expected value of path-length (p) a charged particle in a given medium. CSDA can be expressed by equation 2.18

$$\Re_{CSDA} = \int_0^{T_0} \left(\frac{dT}{\rho dx} \right)^{-1} dT \tag{2.18}$$

T_0 is the starting energy of the particle. The path of a given charged particle is not necessarily straight, and $\Re_{CSDA}$ will give an estimation for the total length traversed by a particle. For heavy charged particles, $\Re_{CSDA}$ is a close approximation of penetration depth but not for light charged particles, for example electrons.
Projected range $\langle t \rangle$ is an estimate of penetration depth for the initial particle with a given energy in a given medium. Of course some particles will go further and some will not reach this depth.

2.2 Dosimetry

A dose is defined by the energy deposited in a volume in the form of ionizations and excitations divided by the volume's mass. Dose is thus often measured as J/kg and is named gray (Gy).

2.2.1 KERMA

Kerma is an acronym for kinetic energy released per mass. We can derive kerma by summarizing the individual parts of energy transferred to a volume V.

$$\epsilon_{tr} = R_{in} - R_{out} - \Sigma Q \tag{2.19}$$

ϵ_{tr} is the total energy transferred from photons to electrons in the volume and is a non-stochastic quantity. The equation relates to the radiative energy going into a volume (R_{in}) minus the energy out (R_{out}) equals the energy departed in the volume.
The ΣQ is a sum of all the energy converted to mass and vice versa. An example of this can be annihilation, by positron emission from a pair production. This process will result in two photons with energy hν=511keV. These photons will likely leave the volume V due to the photons' relatively small cross section. If a dose calculation is done, the energy-mass conversion needs to be accounted for.
We can define the kerma at a point P in volume V with the infinitesimal mass dm.

$$K = \frac{d\epsilon_{tr}}{dm}$$

The term kerma consists of two parts, collision kerma (K_c) and radiative kerma (K_r).

$$K = K_c + K_r$$

The collision kerma relates to the energy transferred to charged particles that dissipate their energy as ionizations and excitations due to interactions with the atomic electrons. Radiative kerma is the production of radiative photons due to interaction with atomic electrons; bremsstrahlung.
The radiation, both charged and uncharged going in and out of a volume is defined in equation 2.20:

$$\bar{\epsilon} = R_{in_u} - R_{out_u} + R_{in_c} - R_{out_c} + \Sigma Q \tag{2.20}$$

In equation 2.20, $\bar{\epsilon}$ is the energy imparted as defined by ICRU [42]. This considers the full picture of energy released in a volume. $\bar{\epsilon}$ in 2.20 is a stochastic quantity and can fluctuate. In an infinitesimal finite volume V with an infinitesimal energy ϵ imparted, the dose can be derived as.

$$D = \frac{d\bar{\epsilon}}{dm} \tag{2.21}$$

The volume contains the infinitesimal mass dm. In the case of equation 2.21 the dose is an expected dose.
For charged particles the dose can be estimated with the fluence of charged particles and it's unrestricted mass collision stopping power as long as the depth of the medium are greater than the range of the secondary electrons.

$$D = \Phi \left(\frac{S_c}{\rho} \right) \tag{2.22}$$

This is a more practical approach to the dose calculation as the only parameter needed is the fluence (Φ), knowledge of the material and $\bar{T}$ of the particles. The S_c/ρ is the mass-collision stopping power.

Charged particle equilibrium

Charged particle equilibrium or CPE is a term often designated to a large volume (V) containing a small volume (v) where V is inside a photon-field. CPE is the phenomenon where the amount of charged particles going into the volume v is the same as goes out of the volume. The particles entering and exiting the volume v must be of the same type and same energy.
There are some conditions to have CPE; V must be homogeneous and photon attenuation must be negligible. If a non-homogeneous material is irradiated, the photons will dispatch their energy with different intensities throughout the material, resulting in a non-equilibrium in ionizations. Photon attenuation must be negligible because a less intense beam will create a different amount of ionizations and thus secondary electrons.

Fano's Theorem

Fano's theorem states that if an infinite medium with the same atomic structure is exposed to a homogeneous radiation field, charged particle equilibrium will prevail throughout the medium for both primary and secondary particles.

$$S(E,\mathbf{u}) - N(E,\mathbf{u}) \int_o^E dE' \int_{4\pi} d\mathbf{u}' k(E,E',\mathbf{u}\cdot\mathbf{u}') + \int_E^\infty dE' \int_{4\pi} d\mathbf{u}' k(E,E',\mathbf{u}\cdot\mathbf{u}') N(E',\mathbf{u}') = 0 \tag{2.23}$$

Equation 2.23 is Fano's proof for an infinite medium for gamma rays. Where $S(E,\mathbf{u})$ is the number of electrons liberated from the primary source contributed from the primary source with variable energy E and direction $\mathbf{u}$ generated per unit mass, where evidently the $S(E,\mathbf{u})$ should be uniform through the medium. $N(E,\mathbf{u})$ is the uniform fluence for electrons for all energies and direction. The $d\mathbf{u}'k(E,E',\mathbf{u}\cdot\mathbf{u}')$ term is the probability for electrons to scatter inelastic and thus have the energy E' and direction $\mathbf{u}'$. The first term of equation 2.23 is the source contribution as mentioned. The second term relates to depletion of energy due to such processes as scattering and absorption, this happens to electrons up to a threshold energy E and electrons can scatter in all solid angles, it will subtract from the fluence. The third term relates processes of higher energies and/or direction will contribute to the fluence.

When inspecting the prof, it becomes apparent that the equation is not dependant on the fluence of the γ. The only thing it depends on is the uniform fluence of electrons that the gamma produces. Fano's theorem does not take the mean ionization potential into account as well as the polarization effect so it is not applicable for materials of different atomic number or density.

2.3 Microdosimetry

As discussed, ionizing radiation is uniquely efficient because it transfers the energy directly to the atom, with subsequent liberation of electrons. Absorbed dose is an expectation value for energy absorbed in a volume, but radiation itself is stochastic. When the volume decreases the stochastic nature of radiation becomes more prominent and the field of microdosimetry is needed.
Microdosimetry was developed to quantify the minute spatial distribution of ionizing radiation through a system of concepts. Microdosimetry can provide a window into the early stages of radiation-induced processes that determine the ultimate outcome.
As in most physics branches, microdosimetry has a theoretical and an experimental part, the experimental, due to the capabilities of detectors are in the range of $2\mu m - 30\mu m$. The theoretical part attempts to quantify the spatial and temporal aspects of radiation in concepts such as lineal energy, the site concept, etc.

2.3.1 Specific energy

In dosimetry, the dose is defined and measured as J/kg. When interested in a dose at a point (which is an expectation value) in an object and is defined by equation 2.21. $\bar{\epsilon}$ in dosimetry are an average while in microdosimetry the ϵ is a energy released per event, dose at a point can be described as the differential equation 2.24:

$$D = \frac{d\epsilon}{dm} \tag{2.24}$$

Equation 2.24 and 2.21 are for all intents and purposes the same. When in a volume V surrounding the site of energy deposition, the specific energy (z) equal to dose can be defined as:

$$z = \frac{\epsilon}{m} \tag{2.25}$$

Where m is the mass that is affected by the energy deposition ϵ. It should be mentioned the quantity z is a stochastic quantity.

2.3.2 Lineal energy

Lineal energy is defined as:

$$y = \frac{\epsilon}{\bar{l}} \tag{2.26}$$

This entity's physical dimension is J/m, but by convention, it is often expressed as keV/μm. $\bar{l}$ is the mean cord length of a volume. If this volume is, for example, a sphere, the mean cord length would be $\bar{l} = 2/3d$ where d is the diameter. ϵ is the energy imparted in said volume.
As previously mentioned, LET is the energy deposition for a given charged particle along its path in a medium, similarly to lineal energy. We assume an almost deterministic view for dose deposition on a macroscopic scale.

However, it should be mentioned that dose is an expectation value, in the microscopic domain it is the event probability within a volume. Therefore, lineal energy is a stochastic quantity and is derived through statistical analysis.
A note should be made regarding ϵ in the specific energy and lineal energy. When discussing dose in dosimetry, $\bar{\epsilon}$ is defined as the energy imparted on a predefined volume or mass. ϵ in the microdosimetric domain is defined as energy imparted in a singular event.
The lineal energy has a probability density function f(y). f(y) is mainly considered the probability for a particle to have a specific lineal energy.

2.3.3 The site concept

Depending on a radiation field's LET, the effects and event density can differ greatly, indicating an impact of the local concentration of absorbed energy. The site concept quantifies the local concentration of events restrained to a volume. This can be illustrated by a 2D grid looked at from a beam's eye view for different LET fields in figure 2.9.

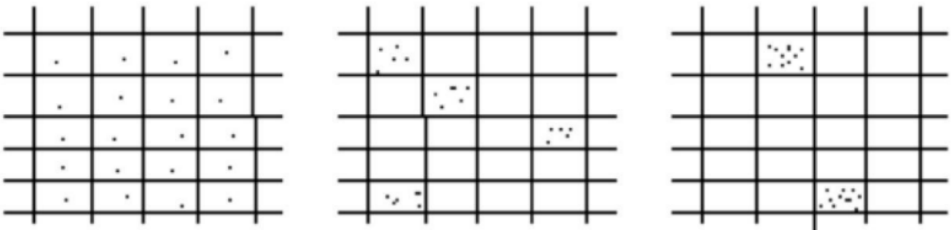


Figure 2.9: The squares in this figures are sites and the black dots are events. Most left figure displays a beam of low LET, where events will be distributed uniformly. Middle figure are mid-range LET. Right is equivalent to a high LET. Each of the figures contain 20 events.

Figure 2.9 is an illustration of how LET will affect the distribution of events within a volume, the volume is divided into small squares (sites) as illustrated. In accordance with equation 2.22, with increasing LET (S/ρ), the fluence of the beam will decrease for an equal dose, increasing the local concentrations in a site. The sites seen in figure 2.9 are squares but for example the radial event distribution for a proton will be more like a cylinder around the proton with decreasing probability further away from the proton.
The probability for events from particle path after interaction often takes the form of a sphere or cylinder, depending on the particle's properties and kinetic energy. For a sphere and a charged particle with infinite range (R), the mean events ($\bar{\epsilon}_m$) within the sphere will be:

$$\bar{\epsilon}_m = \bar{l}L \tag{2.27}$$

L is the LET and is assumed to be constant, but a charged particle does not have a infinite range. Keller had this problem in 1980 and derived the mean track length $\bar{s}$ for a given particle and energy. He derived the mean track

length for a particle with a given LET within and a volume with a mean cord length $\bar{l}$:

$$\bar{s} = \left(\frac{1}{\bar{l}} + \frac{1}{\bar{R}} \right)^{-1} \tag{2.28}$$

R is the expected range of a particle derived from Cauchy's theorem. Equation 2.28 expresses the mean track length and in some sense the ionization density for a charged particle within a site.

2.3.4 The interface effect

Energy departed in a medium depends on the properties of the beam and the properties of the medium. The interface effect explains what happens when a charged particle traverses from a medium such as for example glass to another medium such as atmospheric gas: the mean excitation potential, density and effective atomic number change. Even though the charged particle interaction changes, liberated electrons in one medium will travel over to the other medium. In addition to changes in interaction, the charged particles will have new differential cross sections. This will change deflection angle, CSDA, LET and change the particle's behavior. More often than not, such interactions are not calculated by hand and almost always calculated using Monte Carlo.

2.4 Monte Carlo simulations

As previously mentioned, radiation in its nature is random; while calculating fluence, HVL, etc will give a good average estimate of dose propagation, a more precise method is needed to understand how radiation will behave for different scenarios. The method derived by John Von Neumann and Stanley Ulam called Monte Carlo simulations was developed and is today the "gold standard" in radiation transport calculations. Though this method was first used to calculate the chances of winning a game of solitaire, the aftermath of this discovery has been plenty, from medical treatment to thermonuclear weapons. Monte Carlo is a numerical method to simulate the behavior of random mathematical or physical systems.

The basic premise of all Monte Carlo methods is to find an outcome based on a probability distribution function (PDF). For a dice with 6 sides, the PDF would be $Prob_{dice} = 1/6$ for every outcome. If the probability distribution function is continuous in the interval a to b, a method called inverse transform sampling can be applied. The inverse transform sampling generates random values, distributed according to any PDF by using the inverse cumulative distribution function ($F^{-1}(x)$). Getting $F^{-1}(x)$ is done through two steps (as seen in equation 2.29, f(x) as the PDF), first deriving the cumulative distribution function (CDF). Deriving the CDF is straightforward by integrating the PDF from a-x (equation 2.29b). It can be preferable, not necessary, to normalize the PDF to unity since this is compliant with most

random algorithms which are optemized to draw random numbers from 0 to 1. When the CDF is acquired the inverse can be achieved through inverting the CDF (equation 2.29c).

$$f(x) = \alpha^{-\alpha x} \tag{2.29a}$$

$$\int_a^x f(x) = F(x) = 1 - e^{-\alpha x} \tag{2.29b}$$

$$F^{-1}(x) = x = -\frac{1}{\alpha} log(1 - \xi) \tag{2.29c}$$

Generating random values distributed according to the PDF is done by producing a random number $\xi \in [0,1]$, using $F^{-1}(x)$ to find an x value and determine the outcome. In figure 2.10 it is shown how this works in the case of equation 2.29a.

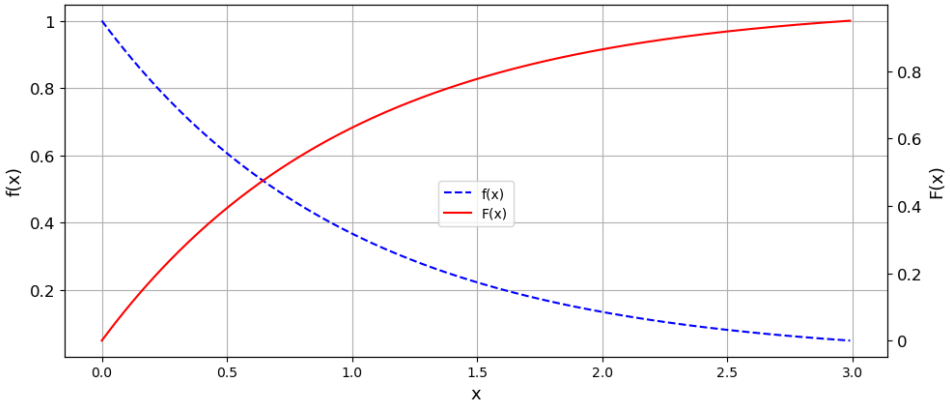


Figure 2.10: A continuous function (f(x)) as a probability function and the integrated function (F(x)) as the cumulative probability function. (A.17)

In figure 2.10 f(x) is the probability distribution, F(x) is the cumulative distribution function, following the recipe given by equation 2.29, $F^{-1}(x)$ is not shown in the figure.
One might run into the situation where the probability distribution is discrete in the interval a-b. Then a technique named the rejection technique can be applied. If we have a PDF in one dimension ($f(x)$), the technique can be applied by randomly sampling 2 points within a set range, one x and one y, rejecting those who do not fall in the function's boundaries. This technique can also be used for higher dimension PDF's. A traditional example of a non-continuous function is the circumference of a circle calculation and the estimation of π. Here, the numbers exceeding $\sqrt{x^2 + y^2} > 1$ is rejected, by counting the total of non rejected numbers and divide by the total of numbers, we get a rough estimation of pi. In figure 2.11 the estimate was 3.1415 ± 0.001 for a simulation run for 10^8 points.

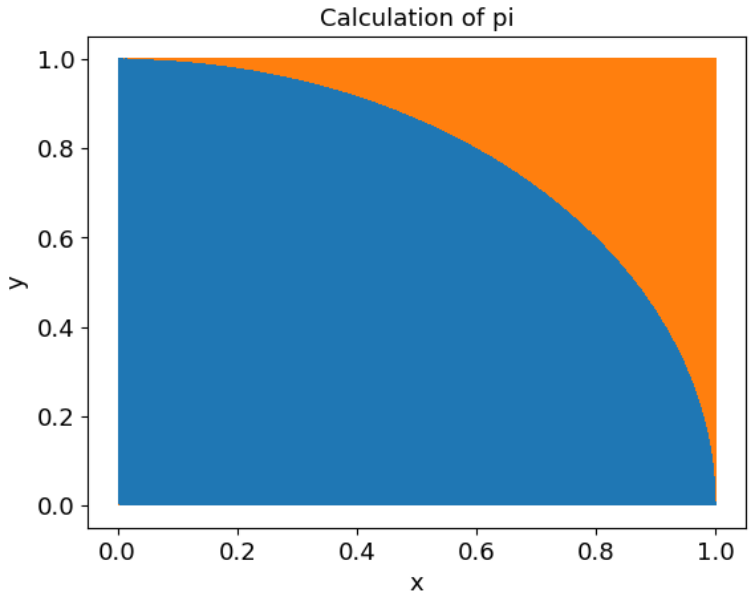


Figure 2.11: The classical example for calculating pi by rejection technique. The colored area are sampled points and are either rejected (orange) or accepted (blue). (A.18)

The PDF used in the rejection technique Monte Carlo does not need to be discrete. Though this method is robust for discrete PDF's, the algorithm needs to draw several random numbers; hence the time needed to calculate is longer than the inverse transform sampling.

2.4.1 Particle tracks

When simulating a particle track, a few approximations are needed to make the code work efficiently. The particle track is segmented into small steps (Δs). The delta will vary in size depending on the purpose of the simulation. For example, the simulation of a nuclear reactor and its contribution to the surrounding area does not need the same step size and intricate calculations as a simulation of protons on a cellular level.
The particles in question are considered to have well-defined trajectories with scattering centers distributed throughout the medium. The reason not to simulate every cross section for each atom in the medium is fully due to computer hardware limitations and the gain from simulating every cross section is not significant.
When a particle traverses a medium, it will have a mean free path. The mean free path is a sum of steps it takes through the medium. The probability for interaction in a segment of the path Δs is assumed to be $\Sigma_j \Delta s$. Here, Σ_j is the probability for the interaction j for a number of scattering sections N, so $\Sigma_j = N\sigma_j$. σ_j is the microscopic interaction cross section for the j'th process (knock-on collisions, soft collisions, etc.)
Then the probability (p(s)ds) for the mean free path to be interrupted for a given trajectory is:

$$p(s)ds = \Sigma e^{-\Sigma s} ds \tag{2.30}$$

A phase space coordinate describes each particle (E, r, Ω). Here, the particle's energy is E. r(x,y,z) is the position and Ω is the direction.
The particle starts with the phase space (E_0, r_0, Ω_0). When it has traveled a distance Δs it will interact and give off an amount of energy ΔE. The new phase space will be (E_1, r_1, Ω_1). This will continue until cut parameters are met. These can be set by the user or by the model's validity in use.

Monte Carlo methods

The MC simulations purpose is to mimic a experimental setup no matter the physical size. In addition to size of an experimental setup, the question is what information gain is of interest, perhaps the information of interest is to calculate shielding material needed from a source of radiation (ex, linac and cyclotron) or the physical properties of a proton beam on the nm scale.

There are two main methods called analog Monte Carlo simulations and condensed history Monte Carlo simulations. When simulating on a small scale, it can be preferable to do the analog Monte Carlo method. This is a method in which raw cross section data is used to simulate each interaction along a singular particle's track, and is most often used for small size problems to investigate the nature of radiation. This is still done in the manner of the probability of interaction along the mean free path, simulating each interaction. Though in this simulation method, the limitation is only the limits of cross section data. There is one drawback with this method, simulating every interaction, will require tremendous computational power. Computational power is the ability for a computational framework to execute a set amount of instructions in a given time. When the size of the experimental setup increases to such a degree that analog MC is not feasible due to computational constraint (the framework not able to compute instruction is a time frame given), the condensed history Monte Carlo simulations is often used. A method grouping individual particle collisions together using multiple scatter theory. The drawback of condensed history compared to analog MC is the accuracy of the simulations.

Variance Reduction Techniques

Variance Reduction Techniques (VRT) are methods to accelerate code based on simple approximations for the track simulations.
The first of the VRT's is reciprocity. If a detector and absorber almost have the same properties (radiation-wise), they can be interchanged due to the interface effect is negligible. So we might exchange one for the other.
Another method of shortening the run time of the code is to see if the projected range (CSDA) is shorter than the target's distance. If CSDA is shorter, the program will terminate the particle.
In addition, use of polar coordinates when cylinder symmetry applies, omit processes that do not impact the outcome of the simulation, and sampling of individual tracks will improve the run time.
Of course, there is a plethora of VRT's not discussed here. The main aim

for this section is acknowledging the existence of these techniques.

2.5 Non-radioactive radiation sources

The fact is, we have the opportunity through scientific discoveries to get radiation on demand. The most common method is through accelerators. Accelerating ions and electrons to tremendous velocity, using them as the ionizing radiation or creating X-rays via the bremsstrahlung effect.

There are several types of accelerators applied for different purposes. The Rontgen tube is mostly used for diagnostic purposes due to the nature of how relatively low energy photons (keV range) interact with low and high Z materials (see section 2.1), for example, soft tissue and bone, respectively.

2.5.1 Linear accelerator

The linear accelerator (linac) produces X-rays in the range of $0.5MeV - 15MeV$. Using oscillating electromagnetic waves, electrons become accelerated through a linear tube (hence the name linac) and hits a target of high Z_{eff}, resulting in forwarding scattered X-rays.

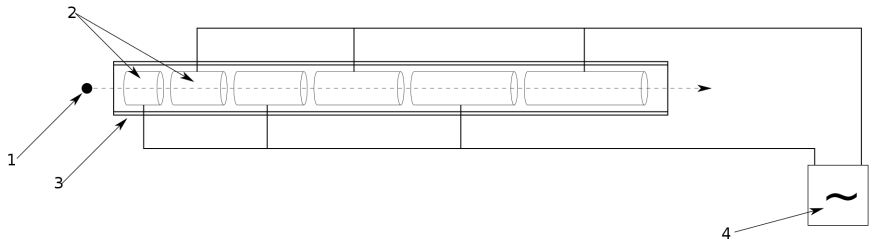


Figure 2.12: Principle of the linear accelerator. The electrons will be ejected by the electron gun (1) and accelerated trough the wave guide (2) on electric oscillating fields provided by a magnetron (4).[10]

The electron gun will send out electrons in a pulse. How electrons accelerates through a linear tube can vary with the manufacturer. The electrons will feel a force $\bar{F} = -q\bar{E}$ from an oscillating electromagnetic field. The charge of the electron is negative, and will thus accelerate the electron in the positive direction, towards the end of the linac. In the relatively short length of the tube, the electrons will accelerate to near light speed. This results in a pulse of high-velocity electrons.
Depending on the company manufacturing the linac, the electrons will be guided to a target at the end of the tube, often at $\approx 112^o$ to the wave-guide. A bending magnet steers the electrons to a tungsten target, resulting in forwarding scattered X-rays. The bending magnet has two purposes: making the system more compact and making the beam narrower. The

magnet system will vary with the manufacturer. If a pulsed electron beam are needed instead of X-rays, one would only remove the tungsten filter.

2.5.2 cyclotron

Although particle accelerators' diversity is large, the main workhorse for medical heavy charged particle therapy is the cyclotron. By accelerating charged particles through an oscillating electric field, the particle will build up kinetic energy until the particle exits the cyclotron.
The beam that exits can be modified using magnets to steer the particle beam into a target. The main function of a cyclotron is to accelerate heavy charged nuclei such as protons, alpha particles (atom core of a helium atom), and carbon ions. The reason not to use electrons is mainly that the geometry of the cyclotron needs to be different for such light particles. Therefore the linac is easier to use for electrons.

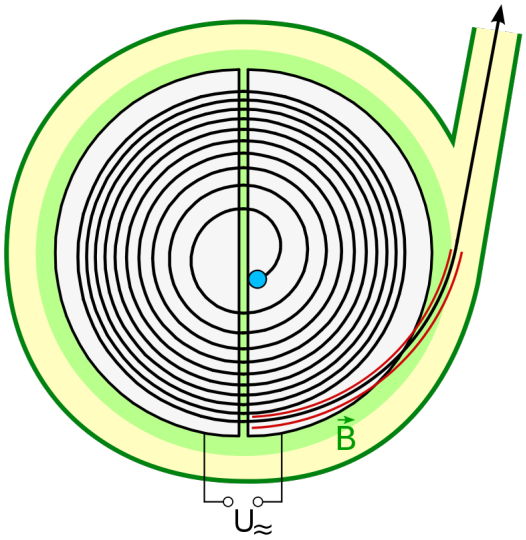


Figure 2.13: Schematic overview of a cyclotron. The blue dot is the entry point of charged particles. The red line is the exit path of the beam. $\mathbf{U}_{\approx}$ is the source of the alternating current, the $\vec{B}$ in the figure is a magnetic field used to collect the charged particles. [11]

In figure 2.13 an electric field will oscillate between the two hollow cylinders also called dees. A charged particle will be accelerated due to a force $F = q\vec{E}$ in the gap. Since a particle has mass, this will lead to an acceleration $a = q\vec{E}/m$ (Newton's second law). Due to the discs' shape, there is no electric field here; hence the particle will not accelerate in the dees. A magnetic field is perpendicular to the electric field, and the particle trajectory $\vec{B} \perp \vec{E}$ and the force acting upon the particle will lead it in a circular pattern, $F = qv \times \vec{B} = qv \sin\theta$.
When the particle is in one of the discs, the electric field will change polarity such that the particle accelerate with every pass through the gap.
Traditionally the cyclotron was used to do particle research and create medical isotopes to be used in e.g. PET and isotope therapy to treat cancer.

In the more modern era, the cyclotron has brought the ability to treat cancers by steering the beam directly to the target.

2.6 Biology

Our body is built by small building blocks called cells. The cells can take many forms and shapes. From meter-long nervous cells, running signals from and to the brain, to small red blood cells, transporting oxygen from the lungs to parts of the body that needs it. These cells are constantly working, doing their specific task. In doing their designated tasks, they can get worn out, damaged, and when this happens, it can result in a process called apoptosis, or programmed cell death. There are also various other ways the cell could die. Therefore cells need to be replaced, this happens through a replication process called the cell cycle.

2.6.1 The cell cycle

When replicating, the cell goes through 4 phases. G_1 is the first part of the cell cycle. Here the cell uses nutrients to grow and checks if there is any damage to the DNA (DNA will be discussed in section 2.6.2) before moving into the S phase of the cycle.

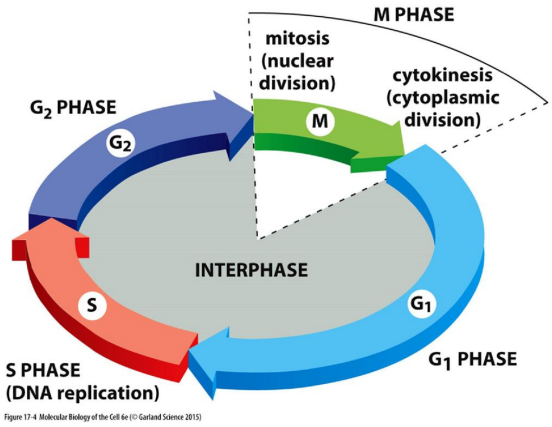


Figure 2.14: A schematic overview of the cell cycle with the four stages. [7]

In the S phase, the DNA is replicated in hopefully two identical parts. At the beginning of this phase, the cell is the most susceptible to radiation damage. The cell is more susceptible to damage due to the fact that early in S, there is only one replica of the DNA. The replication of DNA is a continuous process, and the cell becomes more resistant the further it is in S. When the DNA is replicated, it moves on to G_2. In G_2, the cell grows even more and checks if the DNA is replicated correctly. After G_2 the cell gets ready for the cell cycle's final phase. The M phase or mitosis is the final step of the cell cycle, in which one cell divides and becomes two, new daughter cells.

2.6.2 DNA and damage

DNA is an acronym for deoxynucleic acid. It is composed of 2 phosphate-sugar strands connected through four bases, adenine (A), guanine (G), cytosine (C) and thymine (T). Adenine is always connected to thymine via two hydrogen bonds and cytosine is always connected to guanine via 3 hydrogen bonds.
3 base pairs is the code for one amino acid, amino acids make up protein and what protein gets made is decided by the order of these bases.

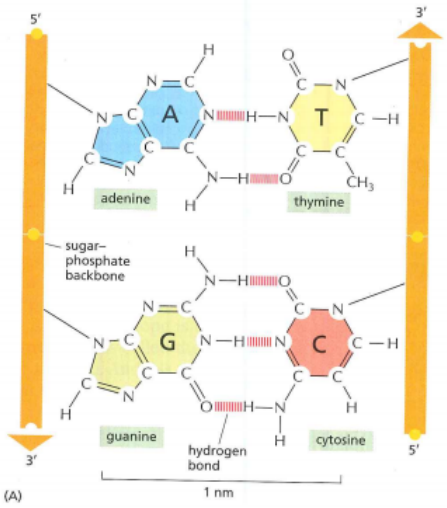


Figure 2.15: The structure of DNA visualised. Orange bands are the sugar phosphates, red dotted lines are the hydrogen bonds and the nitrogen, ogsygen, carhon and hydrogen structures are the bases.[7]

Factors that can induce damage to a cell's DNA are called carcinogens and is divided into three categories, chemical, oncogenic viruses (cancer-causing) and physical, ionizing radiation falls under the latter category of carcinogens. In addition to the carcinogens, time also play a factor, dividing cells can in some instances get a mutation to its DNA by chance which can result in the cell dividing uncontrollably. The definition of cancer is uncontrolled cell division. When ionizing radiation traverses through the cell nucleus, the liberated electrons can induce damage to the DNA by direct effects (directly by liberated electrons) or by the process of interacting with radicals (a byproduct of ionizing radiation), not enabling the cell to repair the DNA. For the first-mentioned, there are several types of damage.

BD

Base damage (BD) is a form of molecular damage related to one or more of the four bases.

SSB

One of the simplest forms of DNA damage that will be discussed is, the single-strand break. This is known as sub-lethal damage, meaning not lethal on its own. However, if several of these damages are accumulated in a small enough region in a short enough time frame, it can be lethal. The time frame part here is important because the cell's repair mechanism will repair an SSB as soon as it is discovered.

DSB

Double strand break (DSB) a is damage to both strands of the DNA molecule. The breakage of strands must happen within 10 base pairs and opposite strands to be considered a DSB.
There are several configurations of the 3 mentioned damages, as shown in figure 2.16.

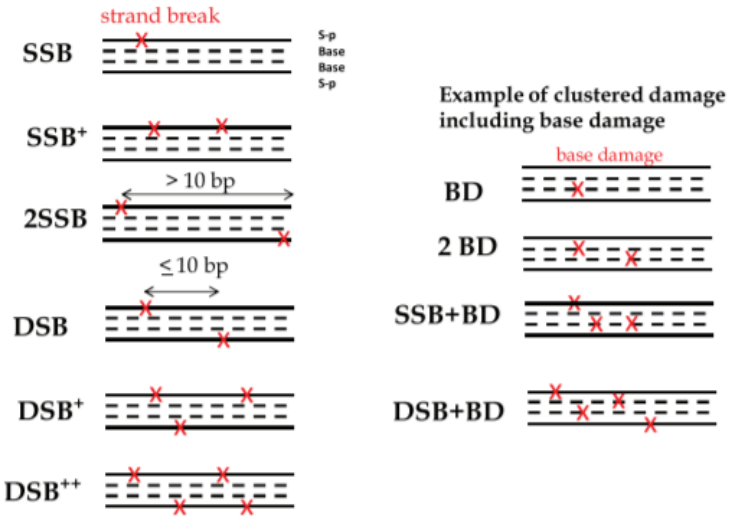


Figure 2.16: The categorisation of DNA damage. X marks the location of damage. Dotted lines are the bases and the lines are the alternating sugar phosphates strands. [30]

As seen in figure 2.16 the damage is not limited to BD, SSB and DSB, can be a a combination of these, resulting in increased severity of the clustered damage. When increasing the LET of a beam, the damage will most likely be more clustered, resulting in less probability for successful damage repair.

2.6.3 Response models

When measuring radiation response, an endpoint has to be chosen., Endpoints can be tumor stagnation, tumor shrinkage, healthy tissue response or cell death. In the latter case, cell death indicates the inability to proliferate.

Usually, a model called the linear-quadratic model is used.

$$S = e^{-\alpha D - \beta D^2} \tag{2.31}$$

S is the surviving fraction of cells and D is the dose. In a cell culture the surviving fraction is measured as a function of dose, when the linear-quadratic model is fitted to the data, α and β can determine and used as parameters to categorize the radiation response of a particular cell line. Often α and β are given in a ratio (α/β), which can used to radiation response after different fractionation schemes.

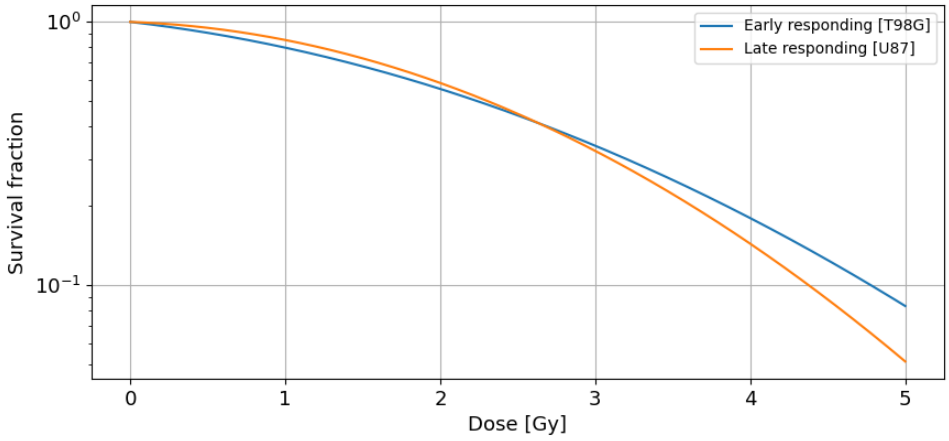


Figure 2.17: Response models for early and late responding cellines with a 1.7keV/μm LET. The y axis is survival fraction in $\log_{10}$ scale. The x-axis is the dose. Blue is the early responding celline. Orange is the late responding celline. (Program: A.19, Celline α and β [9])

As seen in figure 2.17 the late responding tissue ($\alpha/\beta > 8$) will have less response at lower doses overtake early responding tissue at higher doses. $\alpha/\beta < 3$ is carracteristic for early responding tissuey. More often than not cancers will be early responding and healthy tissue late responding, but it is not always the case.

The dose can be divided into smaller fractions, separating the individual fractions by some time interval. The main effect of this is a larger differentiation between the late and early responding tissue. This is a useful tool when irradiating cancers, since cancers are early responding, and normal tissue is late responding. Hence a larger effect on the tumor and a smaller effect on normal tissue, which is preferable.

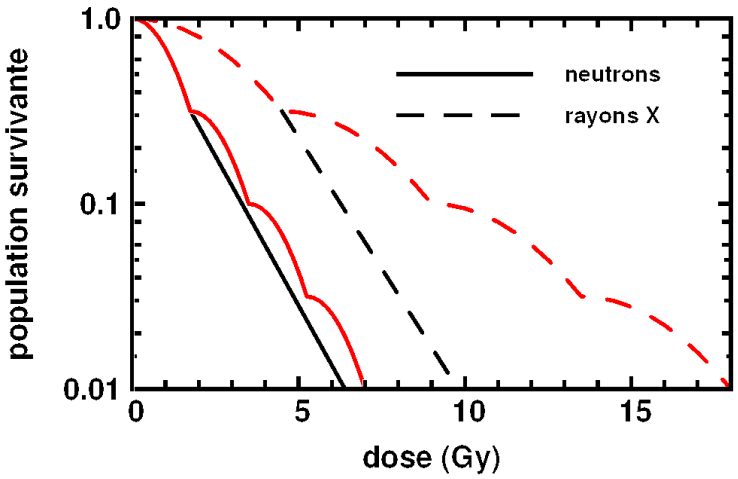


Figure 2.18: Survival fraction of cells when fractions are applied. Red lines are the fractionated doses and black are the same response with a continuous dose. The y-axis are survival fraction and x-axis are dose, neutrons and X-rays where used.[35]

Ionizing radiation with high LET does not gain a significant advantage from fractionation the dose as seen in figure 2.18.

RBE

Relative biological effectiveness (RBE) the relationship between some test radiation (D_T) has compared to the reference radiation (D_R) which give the same biological effect. For reference radiation, the most used is $Co^{60} - \gamma$-radiation due to the monoenergetic photons radiated and the wide use in medical applications. X-rays are also used but since it comes as a spectrum of energies, it has higher LET compared to the monoenergetic γ. It is therefor important to specify which reference radiation is used and which biological effect is measures.

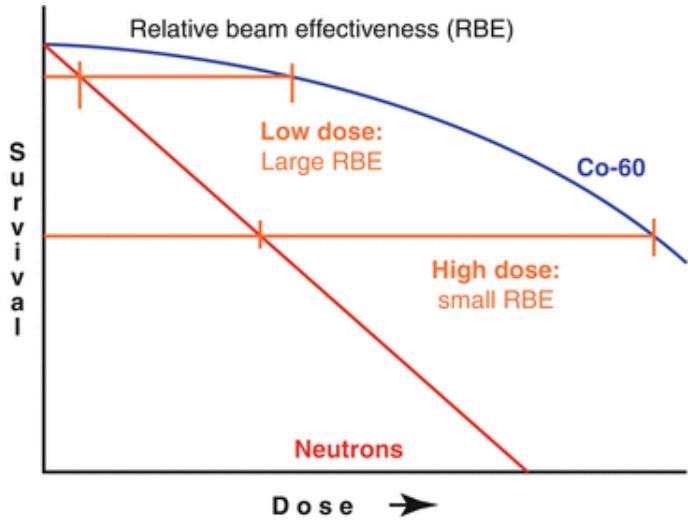


Figure 2.19: The plot shows survival on the y axis as a function of dose on the x axis. The blue graph is survival of cells irradiated with a γ-ray from a cobalt 60 source. The red graph is survival of cells irradiated with neutrons.[8]

The way it is calculated can impact the RBE, as seen in figure 2.19. It is calculated is through equation 2.32:

$$RBE = \frac{D_R}{D_T} \tag{2.32}$$

RBE and LET

RBE is highly dependant on LET as seen in figure 2.19, increasing LET results in increasing ionization density, and it thereby increasing the damage done to DNA both in amount and complexity. RBE for cells will reach a maximum at around $100keV/\mu m$. At this LET, the average distance between ionizations will be approximately $2nm$, which corresponds to the distance between the two DNA strands. An increase in LET will lead to a shorter distance between ionizations but the extreme dose deposited through, will not increase the cell kill. The effect will therefor decrease as seen in figure 2.20.

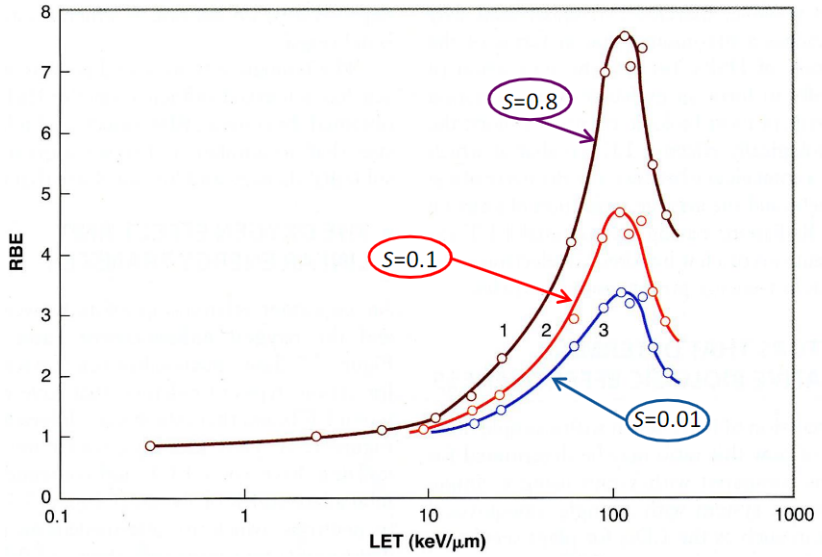


Figure 2.20: Calculated RBE depending on the LET with DNA as an endpoint. The different curves shows survival fraction. The survival fraction are 80%, 10% and 1% for brown, red and blue respectively.[19]

In figure 2.19 RBE is lower at higher doses; this can as well be seen in figure 2.20 when assuming a lower survival fraction is correlated to higher doses of the reference radiation.

2.7 External Beam Radiation Therapy

External radiotherapy is a method of delivering ionizing radiation to a patient's tumor while sparing the healthy tissue. The ionizing radiation delivered to a tumor can be X-rays (produced by the linac), electrons (also linac) and protons/ alpha-particles (accelerated with the cyclotron).

2.7.1 Clinical treatment planning

For treatment planning to proceed a tumor needs to be found and defined. For a tumor to be located, image modalities such as computed tomography (CT), magnetic resonance imaging (MRI), etc. are used. A tumor volume to irradiate is defined, accounting for invisible (in the image) cancer cells and tumor movement. In external radiotherapy, all treatments are planned in digital simulations systems. The simulation systems helps the planner set up the optimal radiation field, giving a tumor the prescribed dose while limiting the dose to healthy tissue. These calculations however are not done by MC simulations but with kernal-based methods (point, cone or pencil) which distribute radiation based on MC simulations data. Radiation transport algorithms applied in hospitals to treat tumors are optimized to work fast while not having a large error margin compared to the time intensive MC simulation method.

2.7.2 Proton therapy

Heavy-charged particle therapy utilizes the increasing LET for charged particles as they lose their velocity. Particles used in this type of radiation are protons, alpha-particles and carbon, the most common tough are protons. The charged particle will maximize the dose deposition at a point, this point is called the Bragg peak (BP) and illustrated for several energies in figure 2.21 (blue lines). Without modification, a monoenergetic pencil beam, a beam with one energy and is a point in a plane transversal to the beam, will be sharp and not enough to cover a tumor volume. To modify the depth of the BP, the beam energy is altered to superpositioned individual BP's resulting in a dose plateau called the spread out Bragg peak (SOBP). The abrupt end of the BP with the most energy (defining the end of the dose platou) will spare the tissue behind it. This feature is the main promoting reason to use protons over photons.

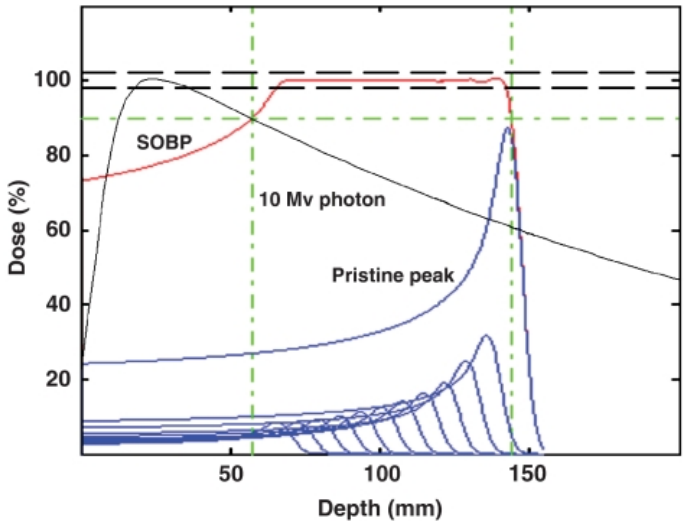


Figure 2.21: A dept dose curve for X-ray and proton irradiation. The gray line showing the dose depth of a traditional linac (energy 10MV). The blue lines are beams of protons at different energies, resulting in the SOBP (red line). The PTV are the green dotted lines. [24]

In figure 2.21 a dose depth is illustrated for monoenergetic protons, the SOBP an a photon beam.

Protons are delivered to patient's in beamlets which is proton beams with set intensity, energy, lateral spread and position. Each beamlet will spread out in the lateral direction when interacting with matter, most at the BP. This lateral spread alone are in most cases not enough to cover the whole tumor, so to cover the whole tumor alternating magnetic fields orthogonal to the beam moves the beam in x- and y-direction (if the beam travels in z-direction). This method is called pencil beam scanning.

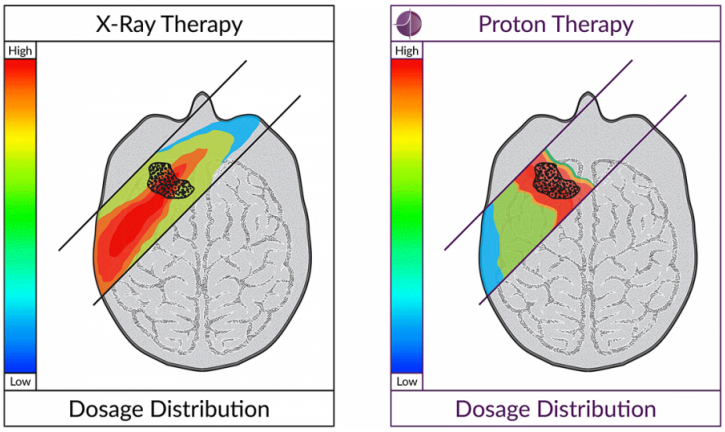


Figure 2.22: The dose plan will vary on what type of irradiation used [21]

In figure 2.22 the advantage of proton therapy are shown, sparing the healthy region behind the SOBP.

2.8 Statistics

2.8.1 Distributions

A normal distribution, also referred to as a Gaussian distribution, is a bell-shaped curve. Several observations and data in nature follow this pattern and its form is generally described by equation 2.33.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.33}$$

Here, μ is the mean of the distribution. In statistics, this value is also referred to as the expected value and is the sum of all data points ($\sum x_i$) divided by the total number of data points (N), $\mu = \sum_i x_i / N$. The σ in equation 2.33 is the standard deviation. It is a measure of expected deviation from the mean in a data set. Standard deviation can be found by taking the square root of variance (σ^2). The variance is found by summing the squared difference between each data point and the mean, divided by the total amount of data points.

$$\sigma^2 = \frac{\sum_i^N (x_i - \mu)^2}{N}$$

The normal distribution is a probability function. Another probability distribution function is the Poisson distribution. Poisson is often used when counting the occurrence of independent events within a certain rate.

$$P(k) = \frac{e^{-\mu}\mu^k}{k!} \tag{2.34}$$

The k in equation 2.34 is the number of independent events and is valid for $k \in \{0, 1, 2, ..., n\}$. The μ is the rate parameter, the number of expected number of events in a interval.

2.8.2 Spatial autocorrelation

"Everything is related to everything else, but near things are more related than distant things"-Waldo R Tobler [41]. Regular statistical analyses assume every event is independent. However, in spatial statistics, the underlying analysis is how phenomena and space are correlated. If there is positive spatial autocorrelation, the phenomena in question will be clustered together. Negative spatial autocorrelation results in a uniform pattern much like a checkerboard. No spatial autocorrelation will result in a random pattern. This can further be illustrated in figure 2.23.

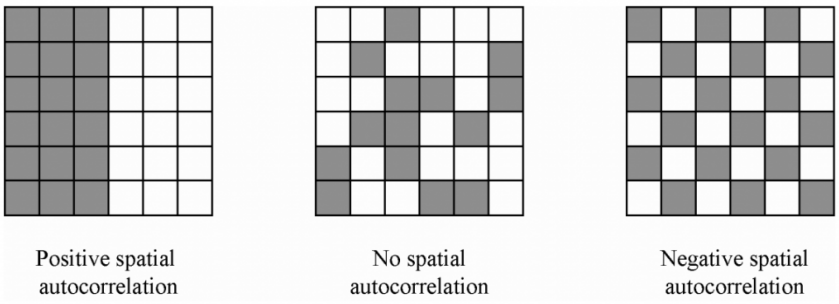


Figure 2.23: The 3 main outcomes for spatial autocorrelation in 2d.[23]

Morans's I

Morans's I measures the spatial autocorrelation with regards to the mean. It is sensitive to the global spatial autocorrelation and is defined as:

$$I = \frac{N}{W} \frac{\sum_i \sum_j w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2} \tag{2.35}$$

In equation 2.35 N is the number of spatial units indexed by i and j. x is the variable of interest and $\bar{x}$ is the mean of x. w_{ij} is the spatial weight matrix and W is the sum of w_{ij}. The weight matrix determines how impactful the phenomena in question are. Common solutions to the weight matrix are the inverted transposed of the distance matrix. Another common technique is to set a spatial parameter (p), if the distance between two separate points is larger than the parameter i.e. $abs(x_i^2 - x_j^2) > p$, $w_{ij} = 0$ and if the distance is smaller, the $w_{ij} = 1$. The weight matrix can have an impact on Moran's I. The values of I can be in the interval $I \in [-1, 1]$ where -1 is negative autocorrelation and 1 is positive autocorrelation. I=0 is no autocorrelation.

Geary's C

Geary's C is inversely correlated to Moran's I, it measures negative autocorrelation, but the two are not equal tests. Where Moran's I is sensitive to global spatial autocorrelation, Geary's C is more sensitive to local spatial autocorrelation on an event by event basis.

$$C = \frac{(N-1)\sum_i \sum_j w_{ij}(x_i - x_j)^2}{2W \sum_i (x_i - \bar{x})^2} \tag{2.36}$$

The variables for equation 2.36 are the same as 2.35. C is in the interval $C \in [0, \rightarrow\rangle$. $C > 1$ indicate negative spatial autocorrelation and $C < 1$ indicate a positive spatial autocorrelation. $C \approx 1$ indicate no spatial autocorrelation.

Chapter 3

Materials and methods

3.1 Geant4

Geant4 is an open-source, general-purpose Monte Carlo program. Open source is a term designated for programs with an open-source code. Developers and alike can go into the code and change it. This results in continuous development, adding to an already extensive program.
General-purpose Monte Carlo programs does not have a specific usage area. Geant4 can simulate medical applications, high energy impact (an example is LHC in Geneva), and irradiation of space instrumentation.

Geant4 is a powerful tool with the ability to simulate particles on an event-by-event basis, as well as through multiple scatter theory. It has a semi-visual interactive interface, where one can specify number of particles and their type (proton, electron, etc) and kinetic energy. This interactive interface is often used to confirm and visualize the experimental setup.
Being written in C++, Geant4 needs a main. The main (module 0) implements all of the modules needed to make Geant4 run and some more. There are a bare minimum of base modules/classes needed to be implemented for Geant4 to run.

3.1.1 Physics list

Physics list (module 1), the module where all of the physics processes are implemented into each specific simulation setup. Each physics process can have several models, e.g., Bhete-bloch is such a model, but there are several more models in use. What models are implemented in module 1 depends on what information is desired and is up to the user of Geant4 to decide. The validity of the models used is often empirically tested for one or several materials. If the physics processes are omitted from the physics list it won't be simulated. Example of physics processes are e.g., soft collision, pair production, etc.

3.1.2 DetectorConstruction

DetectorConstruction (module 2) is a module that needs to be implemented. In this module, the detector is constructed. The detector can be the physical detector (for example, ionization chamber) and the whole of the experimental setup that is requested to be simulated. DetectorConstruction is based on simple geometrical objects/volumes such as boxes, cylinders, and triangles. There is a strict hierarchy to follow when implementing these geometrical objects. The first order of the hierarchy is the world. The world determines whether a physical process will happen and when trajectories should be terminated. One can place the experimental setup of interest within the world, consisting of mother and daughter volumes. If the world where a cube (C_1) with the volume $V_{C_1} = 1m \times 1m \times 1m$ and another cube (C_2) with a volume $V_{C_2} = 0.1m \times 0.1m \times 0.1m$ where to placed in the center of the world. C_2 would be the daughter of C_1, and C_1 would be the mother and the world. If a smaller cube C_3 with $V_{C_2} > V_{C_2}$ where placed in C_2, C_2 would be the mother but not the world. In addition to the designation of daughter and mother, will the volumes need to be assigned a material. Most of the NIST database material can be implemented via a library included in Geant4. Though the code will always run if written correctly, physics model implemented to the region needs to be compatible with the material, more on this in 3.3. A volume consists of two parts, the physical volume containing the geometry (box, cylinder, etc.), size and name. The other part of a volume is the logical volume containing all physical information material, magnetic field, shape and size, The last two (shape and size) are inherited from the physical volume. Logical volumes of the same shape and size can share physical volumes.

3.1.3 PrimaryGeneratorAction

The last program needed is the PrimaryGeneratorAction (module 3). Here the user specifies beam position, energy, particle type, geometry of the beam and fluence rate.

Module 1, 2 and 3 are the base classes of any Geant4 program. These are kernel operations. Kernel operations are the most fundamental part of a computational system and are the classes which controls the other classes and allocate memory and CPU (Central Processing Unit) capacity.

3.1.4 Information gathering

Additional modules are required to sample and store data from tracks and impact on a given target. The module used to track a particle is called SteppingAction (module 4) by convention. Here one can sample data from each interaction in a scoring volume. A scoring volume is essentially the same as a regular volume but module 4 (though it can be any information gathering module) is told to gather information from the scoring volume. Information gathered can be any physical property of the particle such as;

track structure, energy deposited in an event, etc. Tracking action is another way to sample information; one can quantify the total energy depositions in the scoring volume.

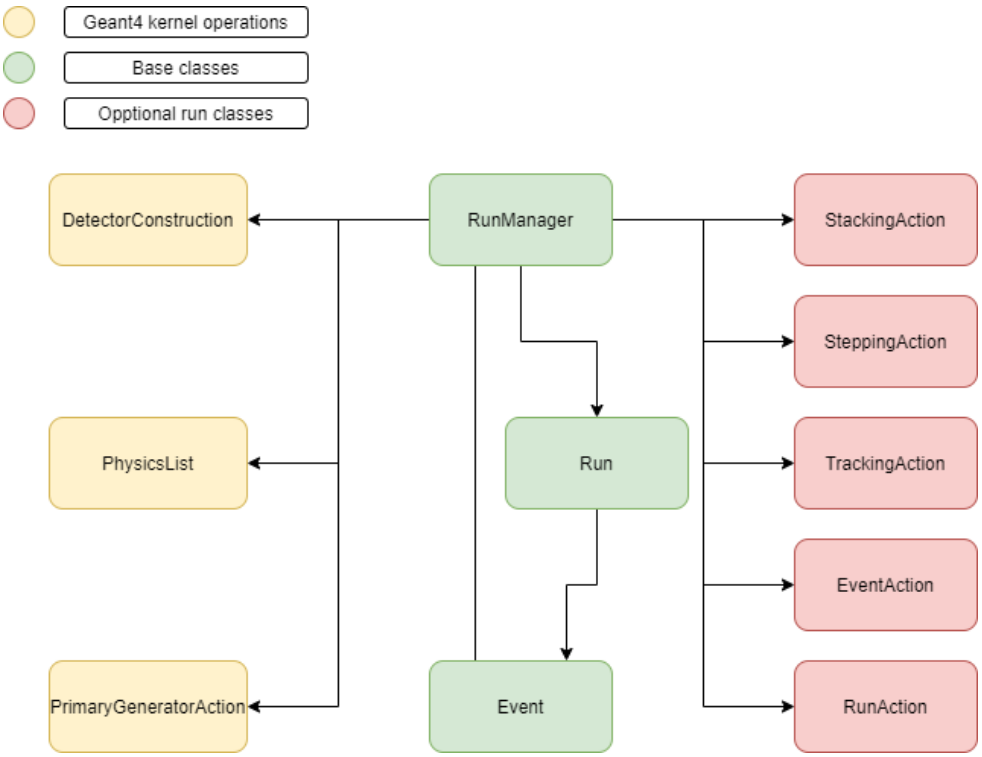


Figure 3.1: A flowchart for a generic Geant4 setup. Kernel operations is essential (module 1, 2 and 3), as well as the base classes (implemented in the main).

In figure 3.1 the yellow boxes (module 1, 2 and 3) and the optional run classes are implemented in a RunManager in the main. The optional run classes can in addition to gathering information, specify how the simulation shall take place, postponing a particle, killing a track, etc. The RunManager starts of by running one or several particles in batches in the Run class, events are simulated and the information are gathered. When information sampling is done, RunManager will start the process all over until the whole simulation is done. The information from these batches can be stored in several ways but usually in a histogram or dose map are used. Batches are used since running the whole simulation in one run wold not be the best use of computational resources.

3.2 The experimental setup

In current work, the aims, is to do MC simulations mimicking the setup used to irradiate cancer cells with proton at the Oslo Cyclotron Laboratory (OCL). The cells are grown in a cell disc (diameter=60mm) and placed in a heated chamber down the beam line. Dose to the cells are determined by fluence and energy, these two variables are measured by the transmission chamber. After irradiation, the cells are stored, colonies are counted after cells had a chance to proliferate. As mentioned in section 2.6.3 cell survival can be measured in the ability to proliferate, therefor colonies are an indication that cells survived the irradiation. Cells not able to proliferate are considered dead. Cell survival can then be deduced by dividing colonies by assumed cells seeded, this gives an estimation of survival for the cell culture. The whole experimental setup for cell experiments at OCL can be seen in figure 3.2:

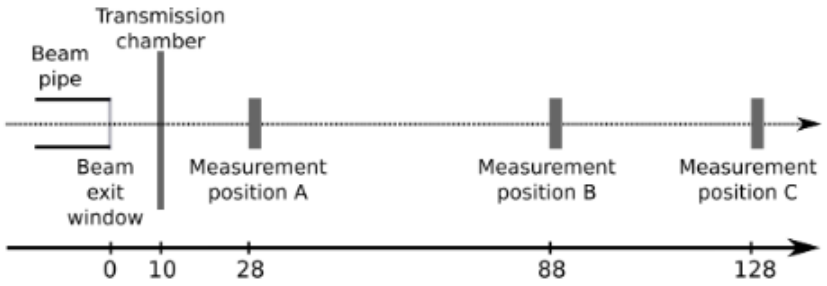


Figure 3.2: The experimental setup for the cellular experiments at the OCL. The beam exit window is a tungsten beam degrader. And the energies is measured at the different positions.[12]

The tungsten beam degrader in figure 3.2 is measured to be $52 \pm 1 \mu m$ thick, and is used to spread out the beam to assure a uniform proton distribution over the cell disc. In the experiment at OCL, dose average LET (LET_d) where estimated with MC simulations (done in FLUKA) at 3 different energies for the 3 different positions in figure 3.2.

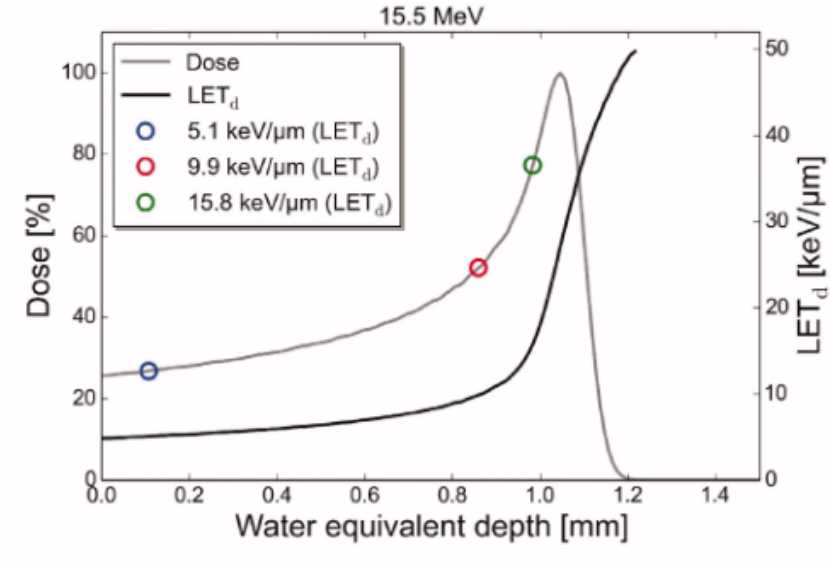


Figure 3.3: The Bragg peak for the experimental setup seen in figure 3.2 measured in water equivalent depth. The simulation estimated LET_d at 3 positions in water at different depths. [12]

In figure 3.3 there are no estimates at the peak or distal end of the peak, so these positions are interesting to investigate. It should be noted that Dahle et al. [12] simulated dose average LET and is not directly comparable to LET.
In simulations done in Geant4-DNA the cell culture disc is placed 80cm from the beam window, getting hit with protons from the frontal, mid and distal end of the Bragg peak, as well as the raw energy output from the cyclotron. The beam energy is moderated experimentally with a water-equivalent plastic bath placed in front of the cell culture.

3.3 Geant4-DNA

In current work, it is of interest to investigate the low energy domain of proton irradiation $< 2MeV$ and map single ionizations and excitations in the region of the Bragg peak. In the low energy spectrum (bellow keV), the options for MC tools are plenty; KURBUC, TRAC and NOREC are codes used to study energy depositions in the microdosimetric range. The only problem is that, these codes are closed to the public and cannot be used unless special access is granted.
Therefore, the logical choice is a sub package of Geant4 called Geant4-DNA. Included in Geant4-DNA are several physics models for low energy interactions. The models used for the physics processes in Geant4-DNA are often semi-empirical. Semi-empirical, a term used for analytical models fitted to empirical data.

Physicslist

First, most of the models used in this implementation of Geant4-DNA are semi-empirical models for liquid water to include it's dielectric effects. Water is abundant in eukaryotic cells and the molecular composition of cells can often be approximated to water. In contrast to gold and some special tissue substitutes, water is easy to get ahold of, this makes it a good candidate to experiment on. The Geant4-DNA is a analog MC simulation tool-kit and will simulate every discrete process.

Standard Bethe-Bloch formalism (equation 2.17) is used outside the target or scoring volume. The reason for this is simply calculation time. The Bethe-block formalism is also used when the simulated particle has an energy $> 0.5MeV$ in the target volume for all charged particles. Models used to simulate energy depositions for electrons in the microdosimetric range are tabulated in 3.1. The literature does not specify any uncertainty for the models used. However, the cross sections used for the models discussed in section 5.3.

Process	Inside Target Volume	Outside Target Volume
Ionization (inelastic)	G4DNABorn model [15] (11eV-0.99..MeV)	Bethe-Bloch (1MeV)
Electronic excitation (inelastic)	G4DNABorn model [15] (9eV-0.99..MeV)	Bethe-Bloch (1MeV)
Elastic scattering (elastic)	Partial wave model [15] (7.4eV-100eV)	n/a
Vibrational excitation (inelastic subexcitation)	Sanche data [28] (2eV-100eV)	n/a
Attachment (inelastic subexcitation)	Melton data [27] (4eV-13eV)	n/a

Table 3.1: Electron interaction models and applicability of the models outside and inside target volume. The models inside the Target volume are only valid for liquid water.

For protons, Born and Bethe-Bloch theories for excitation and ionization are used when the proton energy is above 500keV. If T<500keV and the particle is inside the target volume, the models used can be seen in the table 3.2.

Process	Inside Target Volume	Outside Target Volume
Ionization	Bethe Bloch for T>500keV Rudd semi-empirical approach T<500keV	Bethe-Bloch
Excitation	Bethe Bloch for T>500keV Miller & Green speed scaling T<500keV	Bethe-Bloch
Charge Change	Analytical parameterization by M. Dingfelder et al.	n/a
Nuclear Scattering	Classical approach	n/a

Table 3.2: Proton interaction with matter outside and inside target region. Models for T<500keV are only valid for liquid water.

The code where the models are implemented can be found in A.1.1 and A.1.2.

Primary generator action

This class define the primary beam characteristic, one needs to specify the particle type (proton, electron, photon, etc.). In addition to this, the beam's starting position, direction, and energy are required. From the publications on the proton beam at OCL [12], it can be found that the beam FWHM is 2mm and the initial energy at 15.5MeV with a Gaussian energy distribution of 0.2%. When the tungsten degrader and distance to the target volume is taken into account, it can be assumed that the difference in proton distribution between beam characteristic measurements at OCL and a pencil monoenergetic beam is negligible. Therefore a pencil monoenergetic (15.5MeV) beam has been used to model this experiment.
The code can be found in A.1.3.

DetectorConstruction

The world is a box with the volume $V_{world} = 1m \times 1m \times 1m$. At the OCL laboratory the cyclotron and beam guide is under a vacuum, hence the beam will be under a vacuum until it interact with the degrader, which acts as a border between the vacuum and air in addition to degrade the pencil beam. Within the world a air filled volume is placed, and the $52\mu m$ degrader is placed at the edge of this volume. 80cm down the beam line from the degrader the target volume is placed, a cylinder with $height = 300\mu m$ filled with water. The target volume is used to sample proton data from the Geant4-DNA simulation, and has a diameter of 60mm, which is the same as the diameter of the cell discs used at the OCL experiment.

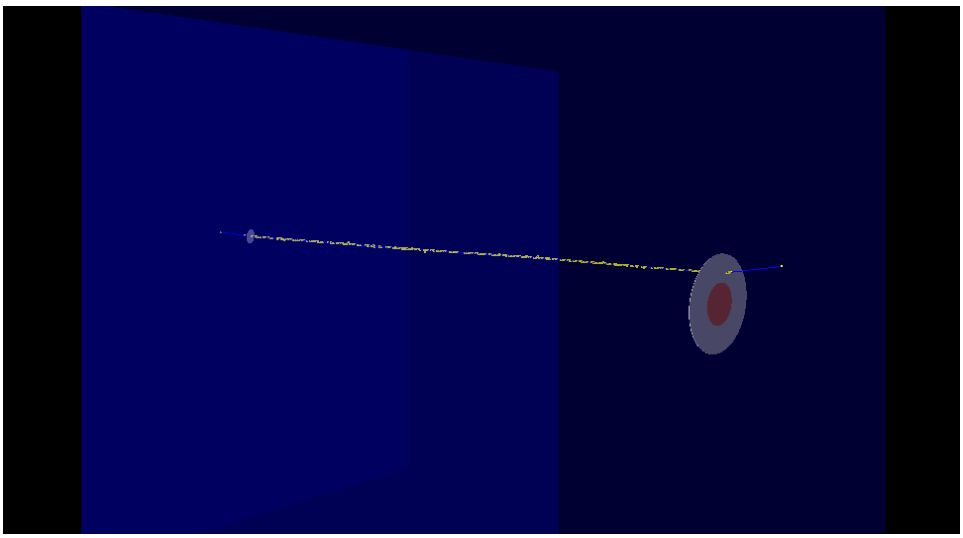


Figure 3.4: This figure is an overview for this particular setup for 1 proton. The red circle marks the cell disc and the gray cylinder right behind marks the water ring used to vary the energy. The gray circle on the other side is the tungsten foil. The blue area is the box within the world and is filled with air. The blue line is the proton and yellow dots are ionizations.

From figure 3.4 the experimental setup can be seen through the interactive

interface in Geant4. A cylindrical volume containing water (gray in figure 3.4) is placed 2mm in front of the target region to regulate energies similar to what is done experimentally. The water volume's length used to vary the beam energy is set to 0.495mm, 0.485mm and 0.47mm. Protons can, when interacting with matter scatter in any direction. Therefor the radius is of the energy variable ring's radius (70mm) is larger than the targets. This is so if any protons with a potential trajectory to hit outside the target radius, they will have the opportunity to scatter back to the target. Energies of the proton beam of interest in the frontal, center and the distal end of the Bragg peak and as the cyclotron's raw energy output. Source code can be found at A.1.4.

Stepping action

There are several methods of tracking energy depositions by particles. Since the Monte Carlo method used is a analog method, we can gather information about every event down to a certain energy threshold. This energy threshold is dependant on the applicability of the models used in the simulation. For the models referenced in table 3.1 and 3.2 this is down to eV. The information gathered from a single proton track are the phase space coordinate (E, r, Ω), step length, particle type, the type of interaction taking place (proton ionization, excitation, etc) and the energy transfer in the interaction as well as ID for every process, particle and step. Information gathered from the simulation requires storage space. The information about a $10\mu m$ long proton track and its interaction in water requires about 4Gb of storage space. Therefore the code implemented has been restricted to sample information only in the target region. Code can be found in A.1.5.

3.4 Modeling

The objective is to simulate clinical doses ($> 1Gy$) in cellular volumes with analog Monte Carlo, but there are two problems. Solved by implementing proton track data simulated in Geant4-DNA in a MC algorithm written in Python (will be discussed in 3.4.2).
In low energy regions (<100eV), the elastic cross section for electrons increase drastically [22]. This is the case for protons as well below 100keV [20]. Since every process needs to be simulated, among other reasons, the analog MC simulations will take longer time than condensed history MC simulations. If a proton hits the target region where Geant4-DNA physics are active it will take 1 minute to simulate the track, a clinical dose (1Gy-10Gy) consists of $10^8 - 10^{10}$ protons in this particular setup. So if a clinical dose of 1Gy for 1.2MeV protons where only to be simulated in Geant4-DNA it would take a few years. This exceeds preferable timing for this thesis. So another approach is required to get good results within preferable timing, mapping ionization and excitation for different energies at different doses. The solution is an approximation, simulating 20000

protons for four different energies (1.2MeV, 1.4MeV, 1.8MeV and 8.7MeV) in Geant4-DNA and implementing them in another MC algorithm.
Geant4 has no ability to differentiate between two proton tracks. If 10 protons where to be simulated in Geant4 these will be catalogued as the same particle. The Python implementation uses a radial distribution model (more on that in 3.4.1), simulating points on a disc and importing single proton tracks to each point. If Geant4 cannot differentiate between these protons, the Python implementation would not work. The way around this is to run Geant4-DNA for one proton, saving the information in a file, 20000 times. Then the single simulation can be stored as 1,2,3,...,20000, differentiating each proton.
Furthermore, the MC algorithm needs the LET from each of the proton energies. This can be done by summation of the energy deposition along the proton track, averaging over all protons for each of the energies, $LET = Energydeposition/Tracklength$. LET is calculated by Analasys.py (A.3).

3.4.1 The spatial distribution of protons

Since protons tracks from Geant4-DNA simulations will be implemented in another MC algorithm it is necessary to extract exactly where the protons initially hit on the target (cell dish) so a distribution model can be made. We can assume cylinder symmetry due to the experiment's cylindrical nature and the absence of a magnetic field. Since cylinder symmetry applies, the only information needed to make a distribution model is the radial distance where the protons hit from the centre. Radial distance (r_{dist}) for each protons initial position on target can be calculated via Pythagoras theorem ($r_{dist} = \sqrt{x^2 + y^2}$). The radial distance is calculated by Analasys.py (A.3). When the radial distance for each proton are calculated, a linear approximation is made, which gives a probability distribution, continuous in the interval [0cm,30mm]. This distribution is perfect for inverce transform MC so equation 2.29 can be applied to the linear approximation. The inverted CPD is used to calculate the initial proton position in the MC algorithm (explained in 3.4.2)

3.4.2 Cell irradiation

The MC algorithm aims to simulate clinical doses to a cell culture with proton data from analog MC simulations done in Geant4-DNA while minimizing time spent on the simulations. Figure 3.5 explains in essence how the MC algorithm flows.

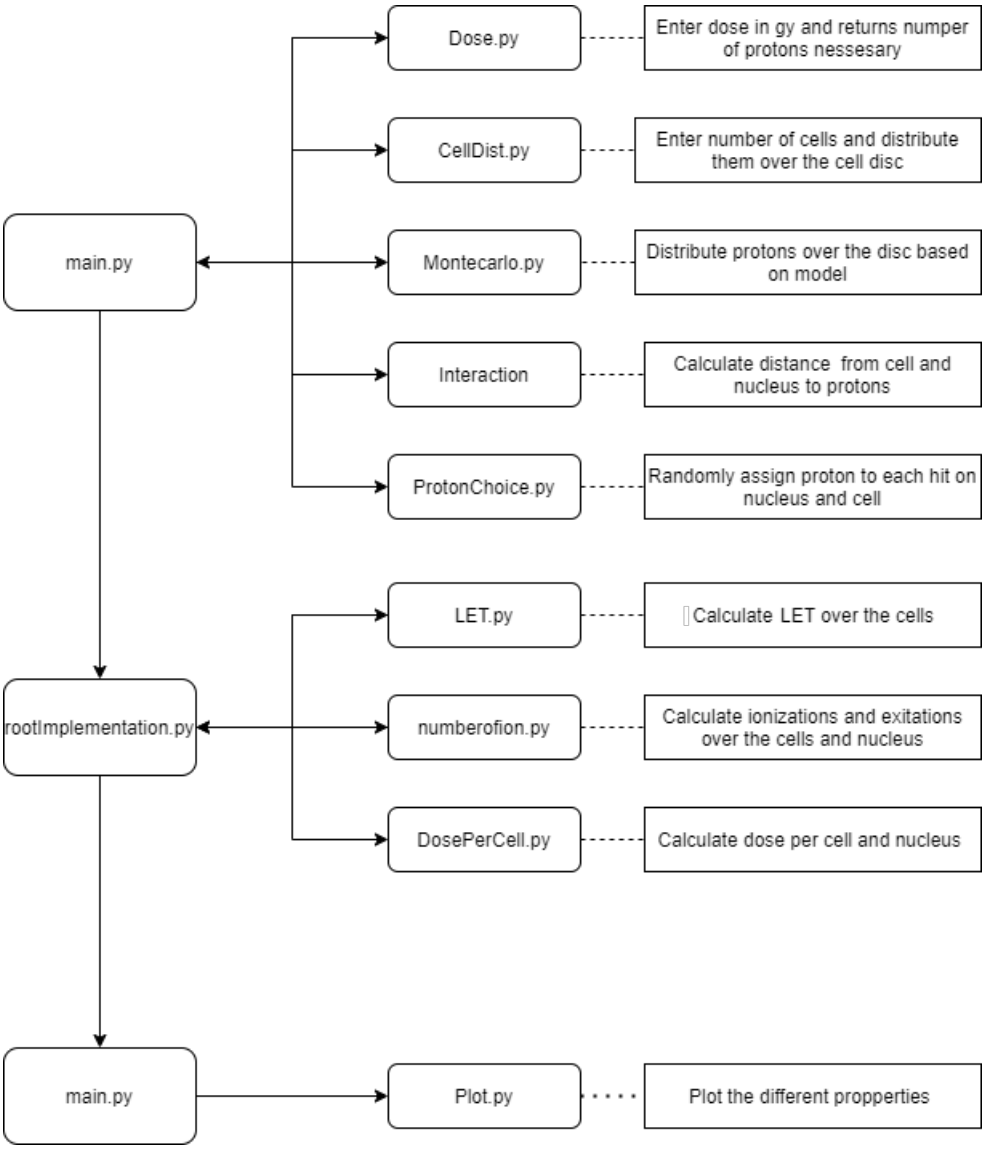


Figure 3.5: A flowchart explaining in simple terms how the analysis works.

The main.py (A.9) in figure 3.5 is the main routine of the setup. The user gives values for dose (D), mean energy ($\bar{T}$) of the proton data where four energies can be requested, corresponding to the four depths discussed in the Bragg peak. The main routine will also request cell height (h_{cell}) and radius (r_{cell}) in addition to the radius (r_{nuc}) of the nucleus, where the height of the nucleus is assumed to be the same as the cell. The cell and nucleus are approximated to be cylinders with the radius r_{cell} and r_{nuc} respectively, the height for both are h_{cell}.
The main program will start to calculate the number of protons needed to reach the dose requested. Dose requested in this instance is the dose to the whole cell disc, a cylinder with the diameter=60mm and height=4μm, the height corresponds to the cell and nucleus height. The dose to the cell disc will be calculated by dose.py, by multiplying the dose needed by the mass of the cylinder and divide by LET, this gives us number of protons needed

to achieve a dose, as seen in equation 3.1:

$$\#protons = D\frac{mass}{LET} \tag{3.1}$$

In eq 3.1, mass is the mass of the cell disc. The dose.py A.4 program returns the number of protons needed based on the LET calculations done in Analysis.py over $4\mu m$ (doses in the simulations consists 9×10^8 to 4×10^{10} protons).
CellDist.py (A.5, cell distribution) randomly distribute points uniformly (x_{cell} and y_{cell}) on a disc with d=60mm. These points will be used as centers for the cells and nuclei. Montecarlo.py (A.6, proton distribution) distributes points (x_{prot} and y_{prot}) using inverse transform MC according to the spatial distribution described in section 3.4.1. These points will be used as initial proton position. The process of distributing protons are done in several iterations. To demonstrate why several iterations are needed, the dose of 2Gy at 8MeV protons will be used. This will result in $\sim 10^{10}$ protons over the disc. If we use a number storage form called float64 to not compromise in accuracy, it takes 2×8 bytes of storage for a single proton to store both the x_{prot} and y_{prot}, it will require one 1/2*terabyte of fast storage, a.k.a ram. Therefore the easy way of getting around this problem is to loop the Montecarlo.py program in main.py several times and check if the proton initial position is in the vicinity of the cell and nuclear radius given by the user and thus interact.
The interaction calculated is the distance between the cell center and a proton incident position. Incident protons not within the radius will be deleted, and protons within will be included. This can further be seen in figure 3.6.

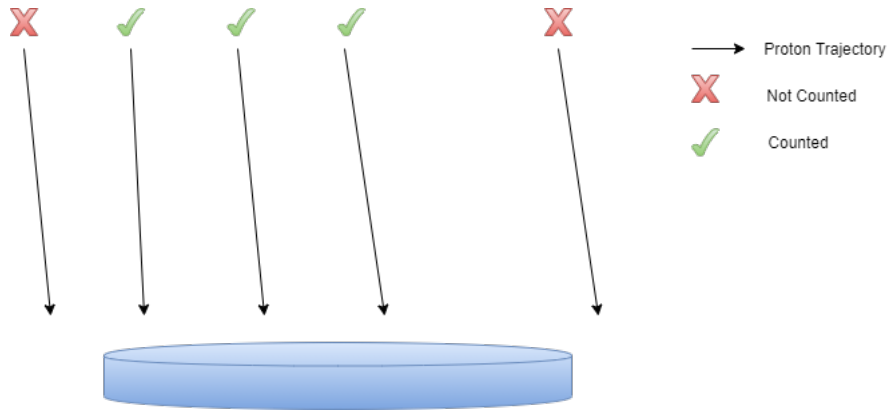


Figure 3.6: Illustration of how protons interacting with cells are counted. The blue cylinder is the approximation of a cell. Not to scale.

The distance (d) from the cell center to the proton is calculated with Pythagoras in euclidean space as in equation 3.2.

$$d(p_{prot}, p_{cell}) = \sqrt{(x_{prot} - x_{cell})^2 + (y_{prot} - y_{cell})^2} \tag{3.2}$$

If $d < r_{cell}$, the proton's position will be saved to the individual cell. This is the case for $d < r_{nucleus}$ as well. No matter how the code is implemented,

it will be slow (several days). Several iterations were tested to maximize the efficiency of the code used. The most efficient single-core method was achieved with a python library named jit by numba ([33]). Jit compiles the routine to machine code, making the code about 230% more efficient. Code with this implementation can be found in A.7. Several attempts were tested to make the code more efficient, both array implementation which is an efficient way to write code in with NumPy and cuda, cuda is a way to run code on the graphics processing unit, but to no prevail. If a proton position interact with the cell ($d < r_{cell}$) it will be assigned one of the protons simulated in Geant4-DNA for the given $\bar{T}$. ProtonChoice.py (A.8) designates a random proton from the protons data (with the energy given by the user) to the hits within a cell and its nucleus.
The rootimplementation.py (A.10) analyzes the track data. For each unique proton, the program opens the proton track files and searches for the necessary information. The whole track for the given height (h_{cell}) is used if it hits within the cell/nucleus (more on this in 5.1.1 and 5.1.2).
The numberofion.py (A.12) program will collect the number of ionisations and excitations to each cell/nucleus. The DosePerCell.py (A.11) collects the total dose deposited within each cell and nucleus.
The information is designated to each cell and sent back to the main for further analysis.
The information gathered in rootimplementation is sent to Plot.py (A.13) to be plotted and the data is saved as .csv files to be further analyzed. For a full overview of how this process works, see figure 3.7.

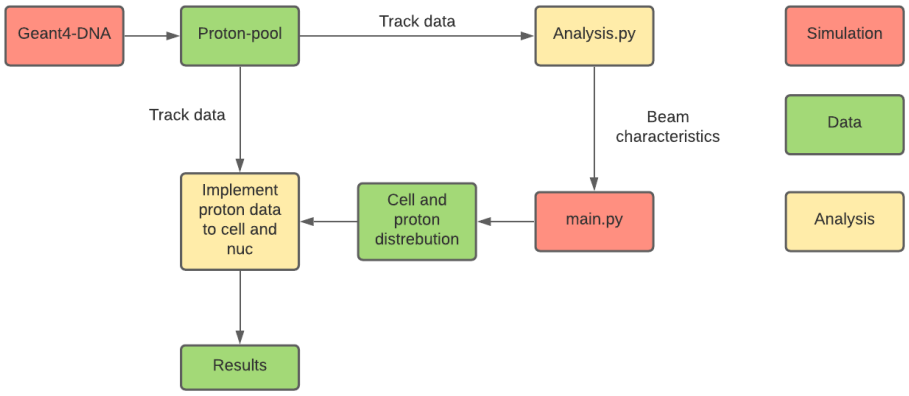


Figure 3.7: This flow diagram shows the total workings of simulating. Red blocks are the simulations itself. Green squares are data from the simulations. Yellow blocks are the analytics of the data simulated.

Figure 3.7 shows the total setup for the simulations. Simulating the protons in Geant4-DNA and analyzing their initial radial distribution, and using the radial distribution to simulate protons in the main routine.

3.4.3 Cell and nucleus dose analytics

All of the ionizations and excitations are stored for the individual cells. The program ionizationAnalasys.py gives insight into the dose distribution as well as ionization and excitation distribution for the cell culture by making histograms for each dose and the four different energies (1.2MeV, 1.5MeV, 1.8MeV and 8.7MeV) for both cells and nuclei.

3.4.4 Spatial analysis

In an attempt to characterise proton track structures, spatial autocorrelation is applied. Moran's I (2.35) and Geary's C (2.36) and an inter-track analysis are implemented in a smaller version of the main.py program. A random distribution of points on a flat circle ($r_{nuc} = 6\mu m$) and protons are assigned to each point.
A regular dose in a nucleus consists of 10-500 protons, each proton contains up to 10k data points for a proton track of height $h_{cell} = h_{nuc} = 4\mu m$. This huge amount of data points restricts the scope of the analysis. Therefore the analysis has been limited to a single nucleus, and the dose consists of the mean dose obtained through the dose.py in figure 3.5 with the cell disc volume replaced with a nucleus.
The moransi.py (A.14) program distributes proton tracks in a random pattern on a cylinder with a $6\mu m$ diameter and a height of $4\mu m$. The ionizations and excitations are extracted from the designated proton file and their x, y and z positions stored in an array P.
To check for global autocorrelation, the nucleus is divided into a grid of small boxes x*nm in height, width and length, named voxel. The amount of events is counted inside each box, giving each box a number B, $B_i \in [0, \rightarrow)$. This is highly based on figure 2.9. Moran's I is calculated layer by layer along the cylinder axis of the nucleus, giving a value layer matrix w_j for $j \in [0, n]$, n depending on the voxel size. The applied algorithm does not have support for 3D space yet. The weight matrix in equation 2.35 is calculated by inversely transposing the value layer matrix for the layer j calculated. The weight matrix accounts for the proximity of the different voxels in the vicinity to each other.
gearysc.py (A.15) inherits the P array and computes the local spatial autocorrelation by equation 2.36. The weight matrix element w_{ij} is either 1 or 0 depending on each event's distance determined by a length parameter. The x_i and x_j coordinate is the i'th and j'th event and $(x_i - x_j)^2$ is the squared distance between the points.
intertrack.py (A.16) and intratrack.py (A.20) algorithms are used to determine the average distance to another proton track or ionization/excitation within a proton track respectively. The intra-track algorithm measures the average distance in nm between each event in a proton track for a given energy. The inter-track algorithm measures the average distance between each of the proton tracks' mean event position, which is the average position for all events. This will fluctuate throughout every proton track due to inconsistency in every proton's initial energy and stopping power will

lead to a higher clustering near the end of a path. Both intra-track and inter-track uses equation 3.3.

$$Intratrack = \frac{\sum_j \sum_i (x_i - x_j)^2}{N^2} \tag{3.3}$$

The x_i and x_j are each mean event points for inter-track. In intra-track, the x_i and x_j will be counted as events.
To visualize how a spatial autocorrelation algorithm would work, see figure 3.8.

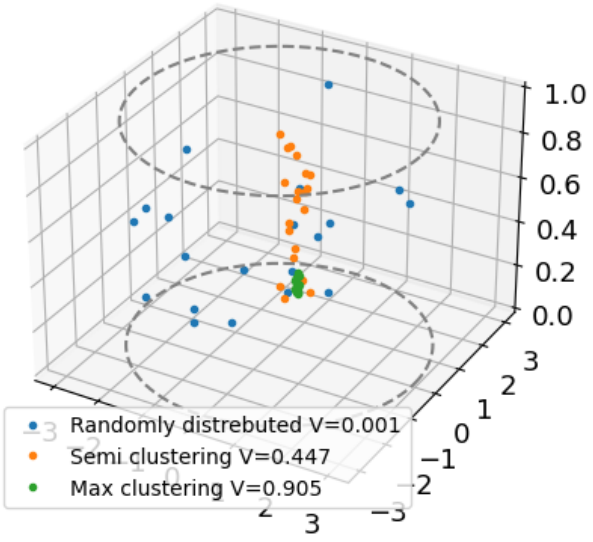


Figure 3.8: Clustering value using a Geary's C index, cut parameter set to 0.3.

The data points and distance in figure 3.8 are totally arbitrary, used to show how these cluster algorithms can be assesed.

	P=0.3	P=0.7	P=10
C	0.905	1.000	1.000
SC	0.447	0.716	1.000
RD	0.001	0.008	1.000

Table 3.3: Clustered (C), semi clustered (SC) and random distribution (RD) with different parameters set for cluster assessment algorithm.

In table 3.3, 3 parameters have been set for the three data sets. As mentioned, photons and charged particles will interact differently in any material, figure 3.8 is a thought experiment on how such algorithms as Geary's C and Moran's I can be applied. Figure 3.8 is a simplified model for a scenario with $clustering \propto LET$. It should be mentioned that for larger clustering Geary's C will in this example tend to 1, this is the opposite of what is expected but it might be since there is only one cluster for each cluster density.

Chapter 4

Results

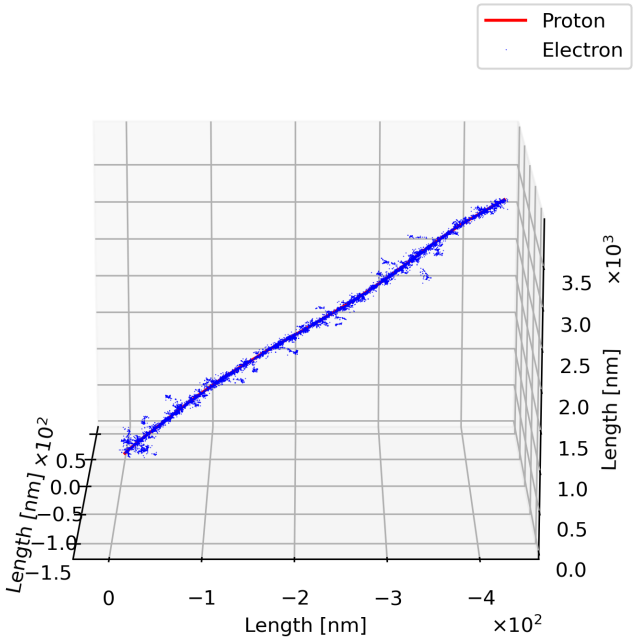
4.1 Computational simulations with the Geant4-DNA simulation toolkit

Simulations mimicking the the experimental setup at OCL as shown in 3.2 was done in Geant4-DNA. The four energies simulated where $\bar{T}_1 \approx 8.7MeV$, $\bar{T}_2 \approx 1.8MeV$, $\bar{T}_3 \approx 1.5MeV$ and $\bar{T}_4 \approx 1.2MeV$. The cell and nucleus height throughout the simulations done are set to $4\mu m$. When the track length is restricted to a height of 4μm the average ionization and excitation count for one proton track is shown in table 4.1:

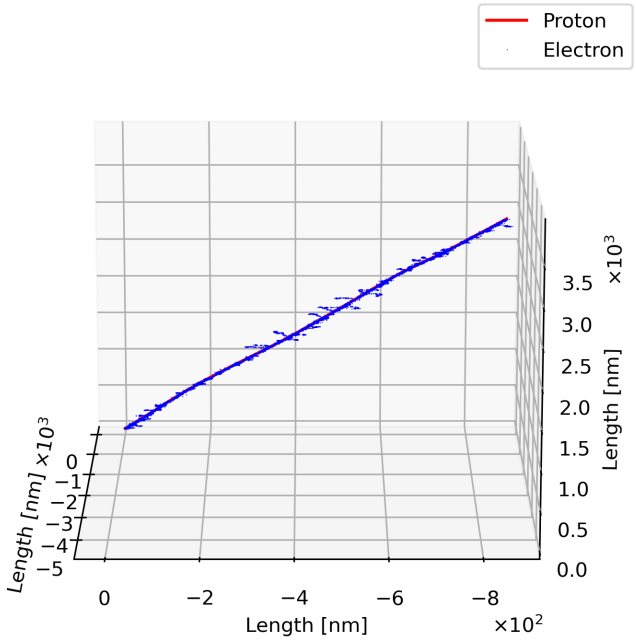
	Excitations	Ionizations	Total processes
1.2MeV	1000	7200	17000
1.5MeV	650	4400	9300
1.8MeV	400	2500	5200
8.7MeV	150	1000	2200

Table 4.1: The different event averages for protons with different energies. The total processes column includes ionizations and excitations, as well as processes such as elastic coalitions, vibrational excitations, etc.

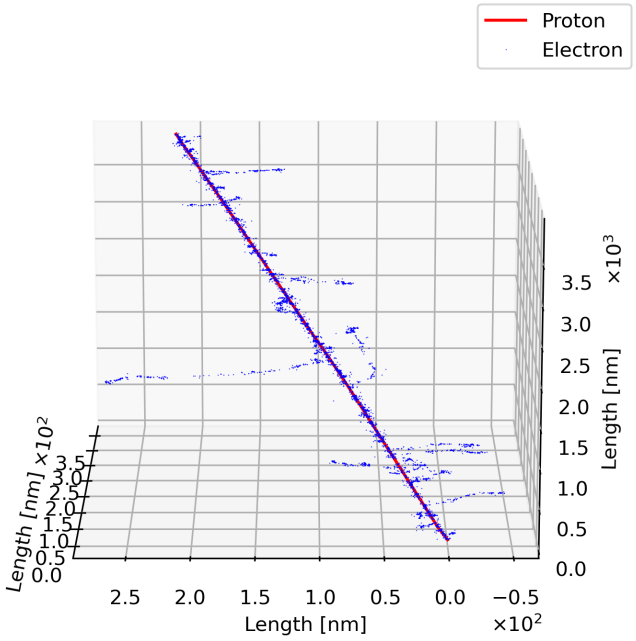
As expected, since the stopping power increases with decreasing kinetic energy, shown in table 4.1, protons with lower kinetic energy will have a larger amount of excitations and ionizations. The Total processes will roughly be double the amount of ionizations and excitations combined for all the energies.
Proton track structures will be different for different energies. Proton track structure for a single proton is shown in figure 4.2 for the 4 energies used.



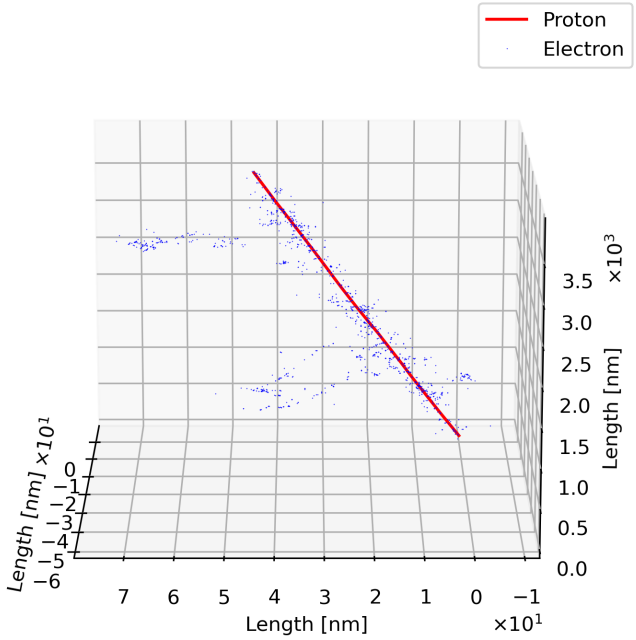
(a) 1.2MeV.



(b) 1.5MeV.



(c) 1.8MeV.



(d) 8.7MeV.

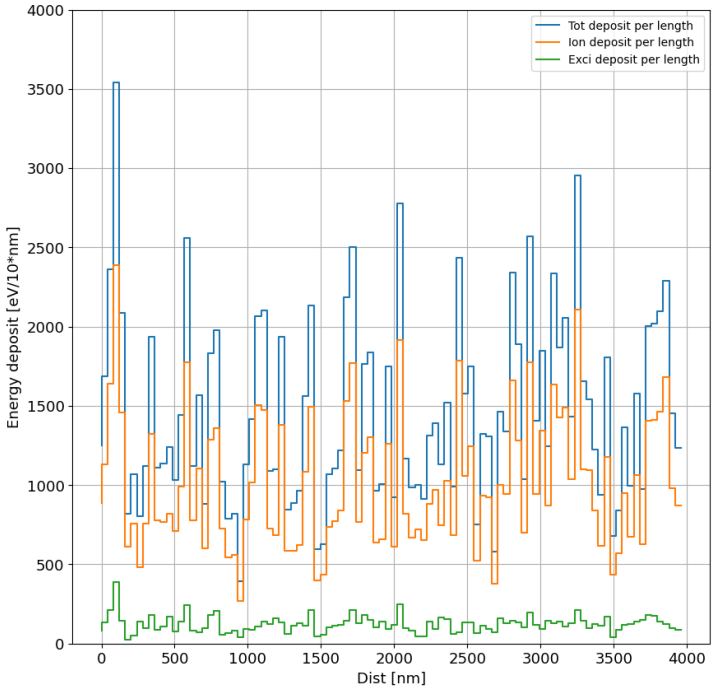
Figure 4.1: Proton tracks (full red line) in $4\mu m$ of liquid water. Electron events marked with blue dots. Units are noted on the axis.

In figure 4.1 the LET is made visible, for four different energies. In figure 4.1a the proton beam is barely visible, covert by secondary electrons. Figure 4.1d with a $8.7MeV$ proton is totally visible and the ionized electrons stray further away from the proton, in contrast to the $1.2MeV$ proton. The further electron range is due to a larger energy transfer when the proton has higher energy in the hard collisions. A trend can be seen in figure 4.1, protons simulated in Geant4-DNA will not stay on a straight path. It is therefor futile to calculate the total lateral deflection for a proton track from it's entry point over $4\mu m$. It should be mentioned, figure 4.1 exaggerate the lateral deflection behavior of the protons. The height of the plot is 4000nm while the lateral deflection for the protons in figure 4.1a-4.1d is only deviating a few hundred nm. The lateral deflection calculations are done through TrackDeviation.py (A.1); the lateral deflection for different proton energies can be seen in table 4.2:

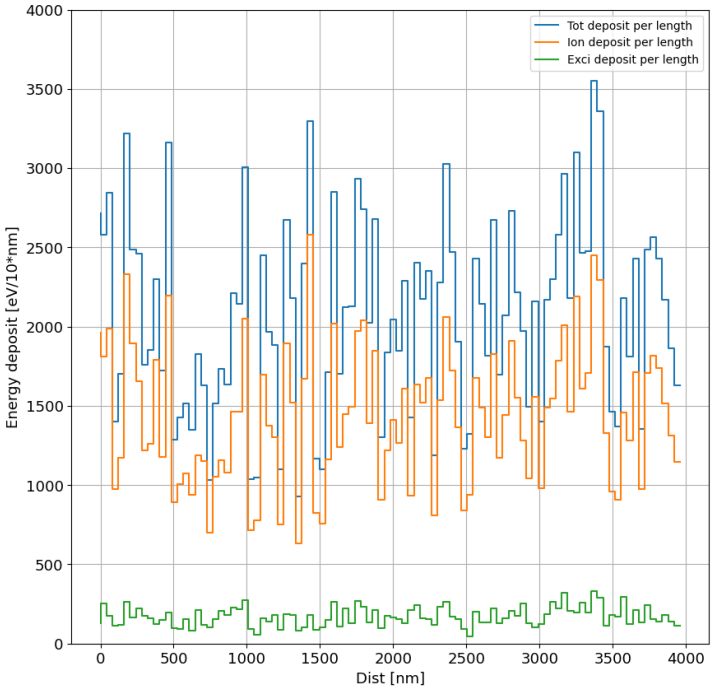
	Deviation [nm]	SD [nm]
8.7MeV	236	200
1.8MeV	250	231
1.5MeV	271	443
1.2MeV	305	628

Table 4.2: Mean track lateral deflection for each energy.

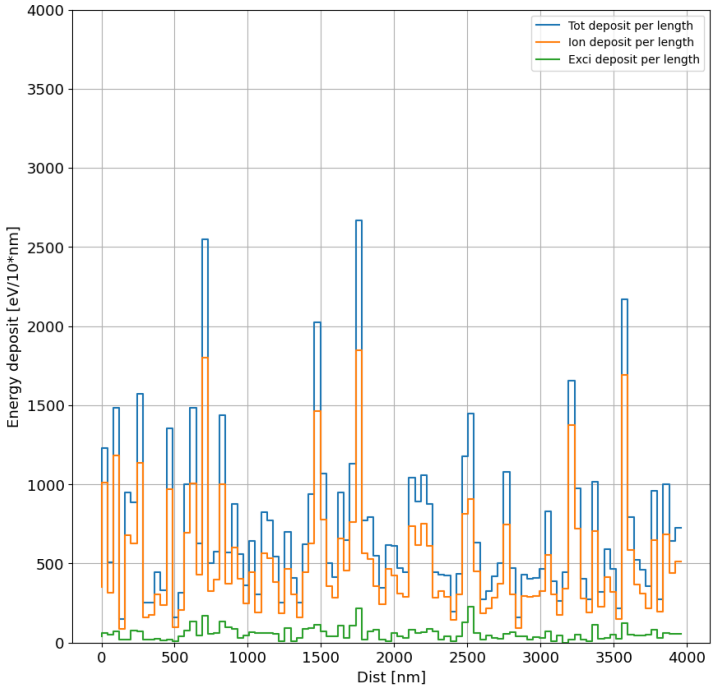
In table 4.2, a not so surprising trend shows itself. The protons with less kinetic energy tends to deflect more. This can be related to events in table 4.1 as more interactions can accumulate to a larger lateral deflection.
As mentioned, in figure 4.1 LET can to some degree be made visible but it would be more informative to compartmentalize energy deposited into bins of $10nm$ along the protons trajectory for the four energies. Furthermore, separate the ionizations, excitations and total events to gain better insight in proton behavior, this can be seen in figure 4.2.



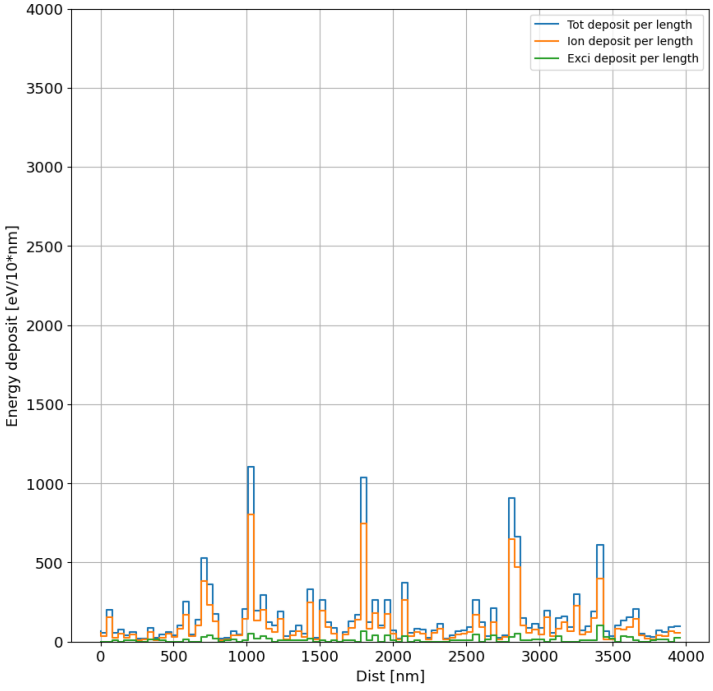
(a) 1.2MeV



(b) 1.5MeV



(c) 1.8MeV



(d) 8.7MeV

Figure 4.2: Protons and their energy depositions through 4μm of liquid water. The lines in figure 4.2a, 4.2b, 4.2c, 4.2d are divided into 3 categories.The green and orange lines are excitations and ionizations respectively.The blue line is the total energy deposited, including not only ionizations and excitations but every interaction such as elastic collisions. Displayed are the distance traversed (x-axis) and the energy released from the primary proton (y-axis), distance per bin is 10nm.

All of the figures (figure 4.2a, 4.2b, 4.2c and 4.2d) summarizes the respective energy depositions over a small delta (10nm) and compartmentalizes them into bins. These four are in some sense a display of the uncertainty of a singular proton, and its energy deposits along it's trajectory. When considering table 4.1, it can be expected that ionizations will be more plenty than excitations, this can be confirmed in figure 4.2. Predicted through theory, if a proton has lower energy, its LET will increase. This can be seen as well in figure 4.2 through the sum of energy deposited per bin. The program fluencekinetic.py considers the equation 2.22, we can rewrite the equation; $\Phi = D/\frac{S}{\rho}$. The main routine estimates the fluence of protons with LET and the total mass of the cell disc. LET is calculated by summing ionizations and excitations over $4\mu m$. Fluence from the main routine and fluence from the equation ($\Phi = D/\frac{S}{\rho}$) are comparable by getting mass stopping power for liquid water from the PSTAR library [31]. The PSTAR library is a library containing calculated mass stopping power for protons at different energies and materials. The comparison can be seen in figure 4.3:

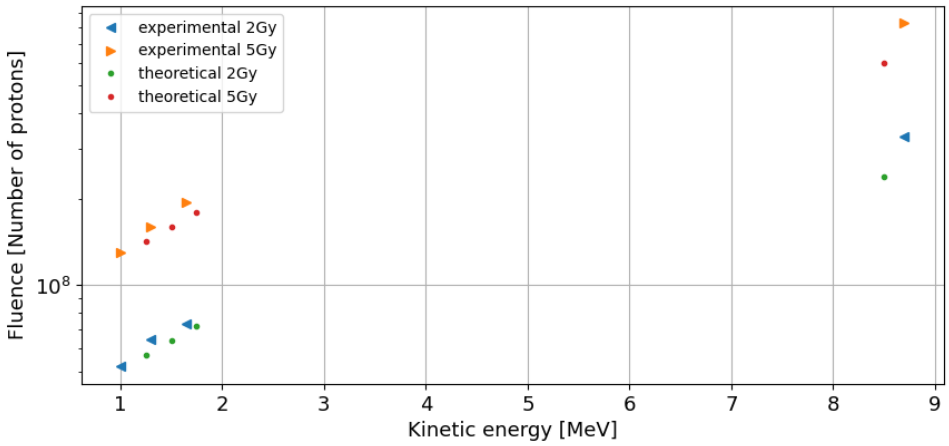


Figure 4.3: Kinetic energy of the protons on the x-axis and number of needed to achieve a dose on the y-axis (in log form). The experimental number are calculated with the main routine, theoretical numbers are derived with the mass stopping power tabulated in P-star [31]

Only ionizations and excitations was used, this is because it was the most alike the theoretical fluence. At lower energies, the fluence difference between the theoretical and experimental is relatively low. For higher energies the simulated ($T_{ex} = 8.7MeV$) and the theoretical from PSTAR ($T_{Theo} = 8.5MeV$) the fluence difference are larger.
As explained in section 3.4.1, a model for the spatial distribution for protons is needed. This is done by calculating distance (r) from the incident proton position (for each proton) to tho cell disc center via Pythagorean distance in a 2D plane. Distance the incident protons hit from the center of the cell disc, the frequency and cumulative probability can be seen in figure 4.4:

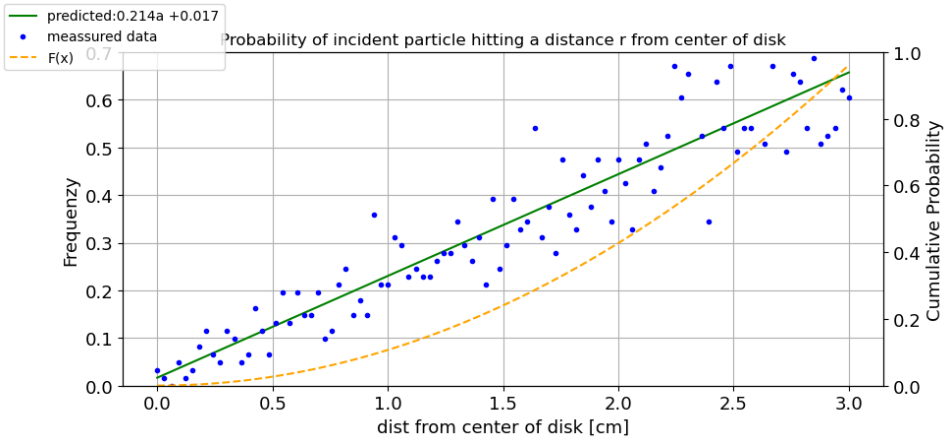


Figure 4.4: The simulated data is the amount of protons at a distance from the centre of the disc in Geant4 (blue dots, left y-axis). The predicted data is a linear fit to the Geant4 model (green line, left y-axis). The function F(x) is the integral of the linear fit model (orange line, right y-axis). On the y-axis The data and linear model is measured in % of total. F(x) is the cumulative probability at a distance from the centre. Program found in A.2

The radial distance from the centre of the cell disc is found by Pythagorean distance from eq 4.1.

$$Dist = \sqrt{x^2 + y^2} \tag{4.1}$$

Initial protons radial distance to the cell disc are calculated with equation 4.1. A linear model can be applied to the radial distance calculated, the linear approximation can be seen in equation 4.2.

$$f(r) = 0.214r + 0.017 \tag{4.2}$$

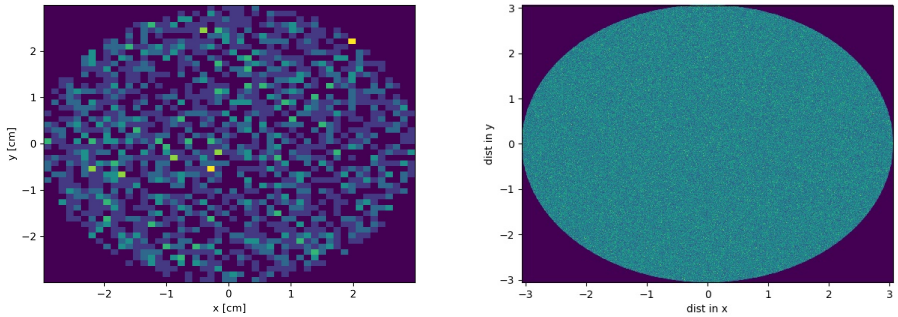
The linear approximation has an $p - value = 2.3 \times 10^{-45}$, a $p - value < 0.05$ indicate the null hypothesis can be rejected, the null hypothesis in this instance is that the slope of the linear regression is flat. The model has a correlation r-value=0.93 and a std-error of 0.008 with r as the radial distance. A linear model is thus a good approximation. Since equation 4.2 is continuous in the interval $r \in [0, 3]$, inverse transform sampling Monte Carlo can be applied by integrating equation 4.2, F(r) (integrated f(r)) is shown by equation 4.3:

$$F(r) = \frac{0.214r^2}{2} + 0.017r \tag{4.3}$$

The inverse function needed to model the beam's proton distribution can be seen in equation 4.4:

$$Y = \frac{\sqrt{2 * 0.214y + 0.017^2}}{0.214} \tag{4.4}$$

Equation 4.4 can be used in inverted transform sampling Monte Carlo and is the method used to distribute protons in the main routine for $y \in [0, 1]$. A visual comparison of incident proton position from the simulations done in Geant4-DNA to the ones in the main routine can be seen in figure 4.5:



(a) Proton distribution on the Geant4 side of the experiment.

(b) Incident proton distribution based on model 4.4. 10^7 protons was used.

Figure 4.5: Both figures displays histograms, counting every incident proton position. $d = 60mm$ for both discs. Brighter spots indicates a higher accumulation of protons in that particular bin.

Figure 4.5 is a display of the bridging between the proton data from Geant4-DNA to the model used in the main routine. The distribution in Geant4-DNA (figure 4.5a) are limited to the protons simulated and therefor the resolution are lower.

4.2 Cell irradiation

Four proton energies was used in the simulation associated with the three points in the Bragg peak as well as the raw output of the cyclotron. The proton data used for the experiments consisted of 20000 protons for each energy, of witch $10\% - 15\%$ hit the cell disc with $\bar{T}_1 \approx 8.7MeV$, $\bar{T}_2 \approx 1.8MeV$, $\bar{T}_3 \approx 1.5MeV$ and $\bar{T}_4 \approx 1.2MeV$. As mentioned, proton interaction with the cell are based on calculating the distance from the initial proton position's distance to the cell center. For the cells a $r_{cell} = 23\mu m$ and $h_{cell} = 4\mu m$ where used, the nucleus parameters where set to $r_{nuc} = 6\mu m$ and $h_{nuc} = h_{cell} = 4\mu m$.

4.2.1 Cell irradiation with 8.7MeV protons

At OCL, the maximum energy delivered from the cyclotron is 15.5MeV, as measured by Tordis J.D. et al., [12]. At the point of irradiation, situated 80cm down the beamline from the wolfram degrader, the mean energy of the protons at the first interaction is approximate $\bar{T}_1 \approx 8.71MeV$ with an $LET = 3.8keV/\mu m$. Even though the raw output from the cyclotron is 15.5MeV, some energy loss is expected in the distance traveled by the protons to the cell disc. The consensus is that the nucleus is the most sensitive target within a cell, from section 2.6.2. Therefore it is interesting to investigate the dose and events to nuclei as well as the cells. The distribution of ionizations and excitations for a 3Gy dose in cells and nuclei are plotted in figure 4.6.

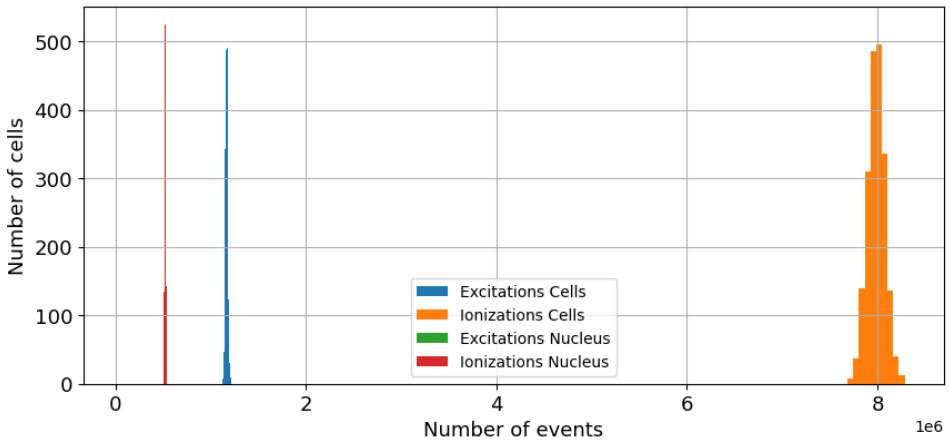


Figure 4.6: Histogram of the events within a cell and nucleus. Each group in the legend has 10 bins each in the histogram. Kinetic energy of the protons used was 8.7MeV. The dose simulated was 3Gy.

Excitations in nucleus are not visible for 8.7MeV due to the small number of events it occupies in figure 4.6. The occurrence of ionizations in figure 4.6 are 6.8 times more prominent than excitations for both nucleus and cells. Mean number of excitations and ionizations per cell and nucleus can further be seen in table 4.3 for the nuclei and table 4.4 for the cells.

Dose	Exci μ [10^3]	Exci σ	Ion μ [10^6]	Ion σ	Nuclei
1Gy	26	493	0.179	3780	1000
2Gy	50	653	0.343	5036	1000
3Gy	76	834	0.523	6425	2000
5Gy	134	1112	0.893	8292	1000
8Gy	216	1350	1,5	10000	1000
10Gy	268	1557	1,8	11000	1000

Table 4.3: Number of excitations and ionizations delivered to the nuclei when a dose is prescribed to the disc. σ is the standard deviation. The highlighted row are the data displayed in figure 4.6

Dose	Exci μ [10^6]	Exci σ [10^3]	Ion μ [10^6]	Ion σ [10^3]	Cells
1Gy	0.378	7	2.6	51	1000
2Gy	0.756	13	5.2	92	1000
3Gy	1.2	13	7.9	91	2000
5Gy	2	16	13.3	128	1000
8Gy	3.1	21	21.3	148	1000
10Gy	3.8	24	26.6	170	1000

Table 4.4: Number of excitations and ionizations delivered to the cells when a dose is prescribed to the disc. σ is the standard deviation. The highlighted row are the data displayed in figure 4.6

Events occur 14.53 times more in the cells versus nuclei; this is within the

expected value since $r^2_{cell}/r^2_{nuclei} = 16$.

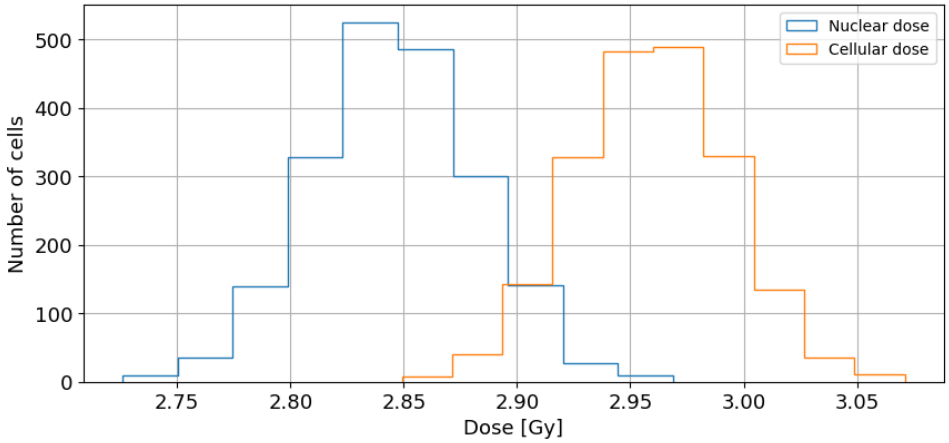


Figure 4.7: Dose distribution for cells and nuclei for T=8.7MeV at 3Gy.

In figure 4.7 a difference in dose can be seen from cells to nuclei. The difference in dose delivered to the cell and nucleus is 0.113Gy. The σ are approximately equal for cell and nucleus in figure 4.7. For the $\bar{T}_1$ beam, the dose distributions for all doses can be seen in figure 4.8.

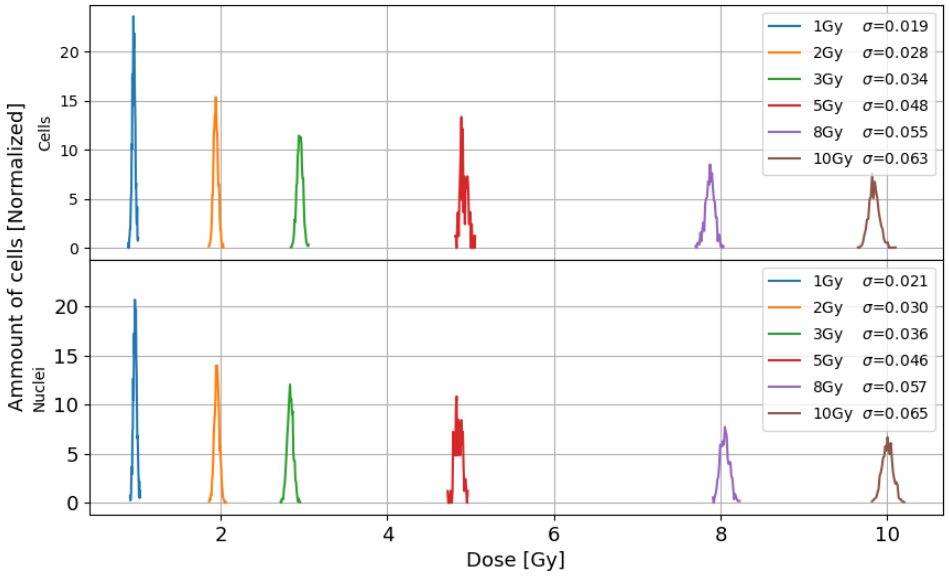


Figure 4.8: Dose distribution for the cells (upper plot) and nuclei (lower plot) simulated. Kinetic energy of the protons used was $\bar{T}_1$. The area under each curve equivalents to 1.

As seen in figure 4.8 the dose given to the cell disc by the user does not mean that each cell will get that dose. For $\bar{T}_1$, the dose to cells and nuclei is different from the average dose to the disc. The standard deviation seems to be equivalent with a constant times the dose squared, $\sigma \propto k\sqrt{D}$. This will be further investigated later.

4.2.2 Cell irradiation with 1.8MeV protons

To achieve an average $\bar{T}_2 \sim 1.814 MeV$ a 0.47mm thick water cylinder was placed in front of the cell disc. The average LET was calculated to be $14.49 keV/\mu m$. As with he $\bar{T}_1$, the amount of excitations and ionizations for cells and nuclei is plotted for $\bar{T}_2$ in figure 4.9

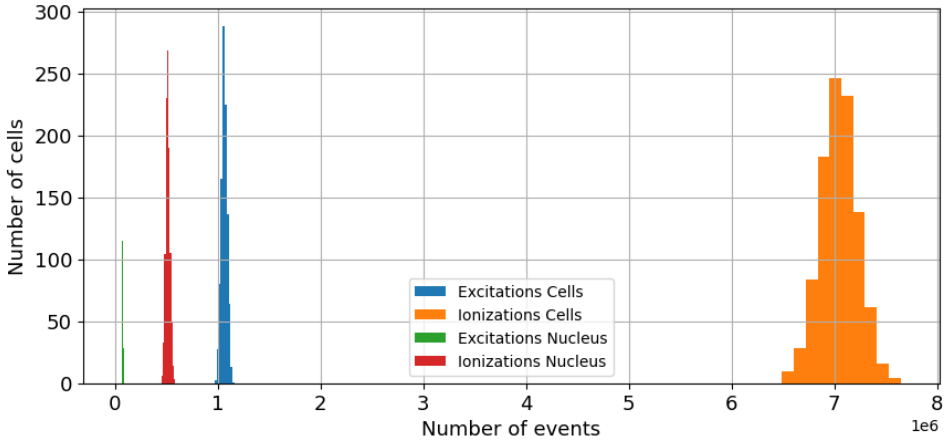


Figure 4.9: Histogram of the events within cells and nuclei when the $\bar{T}_2$ proton beam is applied. Dose used for the simulation was 3Gy.Number of events are on the x-axis and cells on the y-axis.

The difference in ionizations versus excitations with a beam of 1.8MeV is 6.6. Comparing events from the $\bar{T}_1$ beam to the $\bar{T}_2$ beam will result in 0.2 times more excitations. The full overview of mean excitations and ionizations for the $\bar{T}_2$ beam can be seen in table 4.5 for the nuclei and table 4.6 for the cells.

Dose	Exci μ [10^3]	Exci σ	Ion μ [10^6]	Ion σ [10^3]	Nuc
1Gy	24	1826	0.157	11	1000
2Gy	58	3018	0.381	18	2000
3Gy	88	3738	0.516	19	2000
5Gy	148	4800	0.972	29	3000
8Gy	224	5814	1.4	35	3000
10Gy	279	6448	1.7	39	3000

Table 4.5: Number of excitations and ionizations delivered to the nuclei when a certain dose is prescribed. σ is the standard deviation. The highlighted row are the data displayed in figure 4.9

Dose	Exci μ [10^6]	Exci σ [10^3]	Ion μ [10^6]	Ion σ [10^3]	Cell
1Gy	0.409	18	2.7	119	1000
2Gy	0.817	26	5.4	171	2000
3Gy	1.2	32	7.0	188	2000
5Gy	2	42	13.4	276	3000
8Gy	3.3	52	21.4	339	3000
10Gy	4.1	83	26.7	550	3000

Table 4.6: Number of excitations and ionizations delivered to the cells when a certain dose is prescribed. σ is the standard deviation. The highlighted row are the data displayed in figure 4.9

The difference here in the cell and nucleus events is on average 15.51. It can be interesting to look at dose distributions for a single dose of 3Gy for cells and nuclei. This can be seen in figure 4.10.

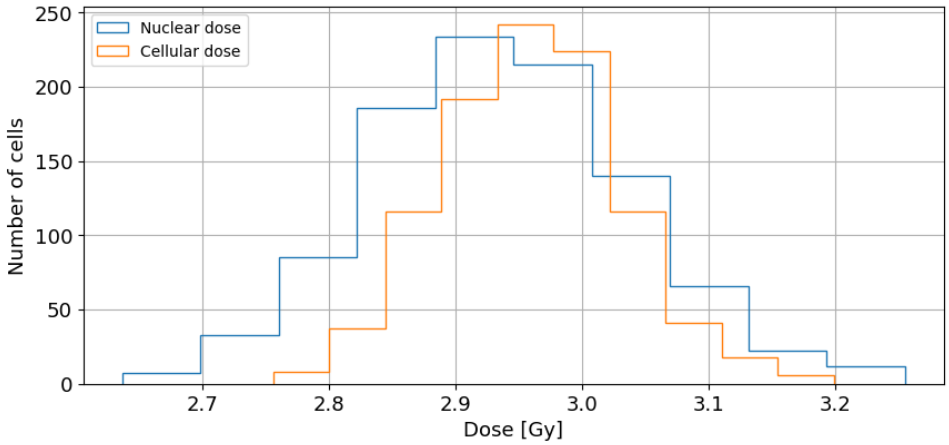


Figure 4.10: Dose distribution for the cells and nuclei when a dose of 3Gy is applied. Proton energy in the simulation was $\bar{T}_2$.

It can be seen in figure 4.10, compared to 4.7, that the mean dose difference between cells and nuclei has decreased slightly but not not pronounced. Compared to $\bar{T}_1$, the standard deviation is larger both for cell and nucleus. Though for $\bar{T}_2$, standard deviation for the nucleus has increased slightly more. The total dose distributions for cells and nuclei can be seen in figure 4.11.

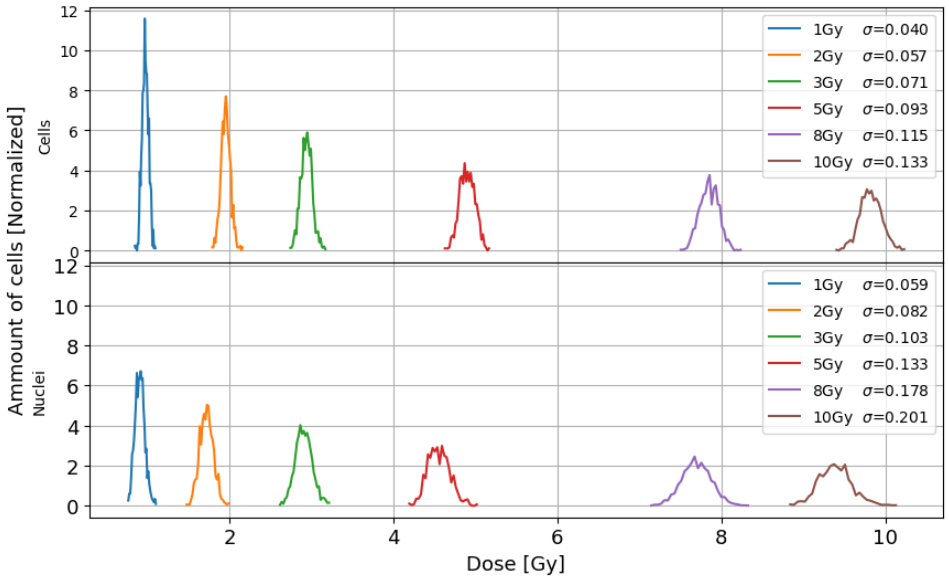


Figure 4.11: Dose distribution for the cells (upper plot) and nuclei (lower plot) with σ as the standard deviation. Doses displayed on the x-axis and normalized cell and nucleus count on the y-axis. Proton energy used was $\bar{T}_2$.

In figure 4.11 the dose is deviating from the prescribed average dose. This is true for both nucleus and cell. Though the nucleus tends to have a larger deficit in dose than the cell. The standard deviation still seems to follow the pattern $\sigma = k\sqrt{D}$.

4.2.3 Cell irradiation with 1.5MeV protons

To achieve an average $\bar{T}_3 \sim 1.514MeV$ a 0.485mm thick water cylinder was placed in front of the cell disc. The average LET was calculated to be $17.3keV/\mu m$. As with he $\bar{T}_1$, the amount of excitations and ionizations for cells and nuclei is plotted for $\bar{T}_2$ in figure 4.12:

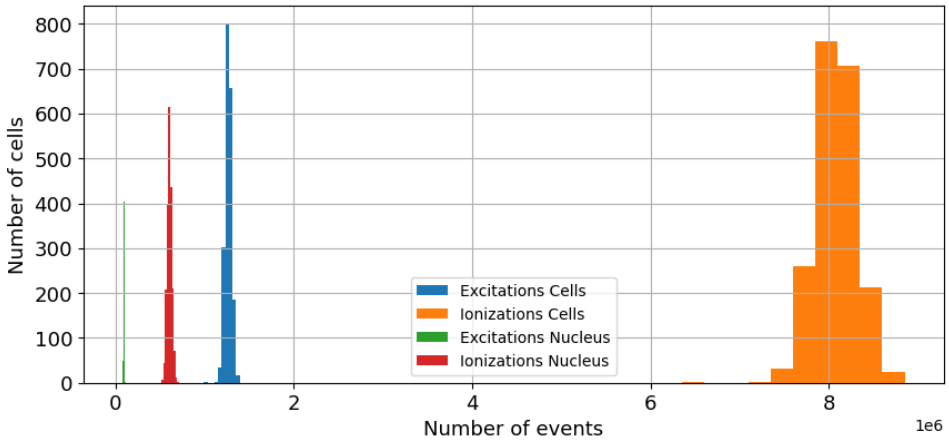


Figure 4.12: Ionizations and excitations distribution for cells and nuclei with 3Gy for the $\bar{T}_3$ beam.

The mean difference in ionizations and excitations was calculated to be 6.6, the same as with the $\bar{T}_2$ beam. Furthermore, all data on mean ionizations and excitations for cells and nuclei can be seen in table 4.8 and 4.7.

Dose	Exci μ [10^3]	Exi σ	Ion μ [10^6]	Ion σ [10^3]	Nuc
1Gy	31	2636	0.201	16	1000
2Gy	54	3315	0.247	20	1000
3Gy	95	4243	0.607	26	2000
5Gy	142	5142	0.911	32	1000
8Gy	232	6224	1.4	34	1000
10Gy	296	6879	1.8	43	1000

Table 4.7: Mean number of excitations and ionizations of the nuclei for different doses, energy used was $\bar{T}_3$. σ is the standard deviation. The outlined row is the data set used in figure 4.12.

Dose	Exci μ [10^6]	Exi σ [10^3]	Ion μ [10^6]	Ion σ [10^3]	Nuc
1Gy	0.442	20	2.6	134	1000
2Gy	0.853	29	5.5	188	1000
3Gy	1.3	35	8.1	229	2000
5Gy	2.1	48	13.7	307	1000
8Gy	3.4	57	21.7	366	1000
10Gy	4.3	66	27.4	442	1000

Table 4.8: Mean number of excitations and ionizations of the cells for different doses, energy used was $\bar{T}_3$. σ is the standard deviation. The outlined row is the data set used in figure 4.12.

From the data in table 4.8 and 4.7 a difference of 15.71 was calculated in regards to the ionizations. As previously it can be interesting to look at the dose distribution for cells and nuclei; a simple example for 3Gy can be seen in figure 4.13:

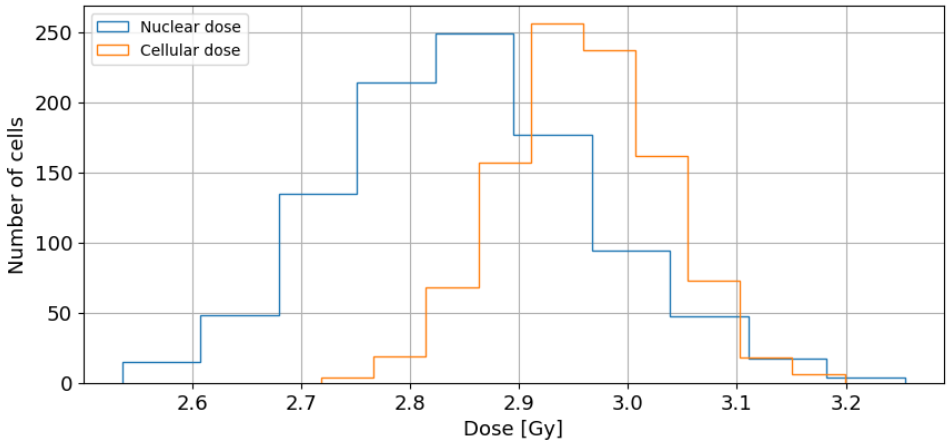


Figure 4.13: Dose distribution for cells and nuclei for the highlighted data in table 4.8 and table 4.7. The dose used was 3Gy and energy used was $\bar{T}_3$.

In figure 4.13 it can again be seen that the mean dose of the cells is higher than the nuclei. The mean difference between the two are $0.2Gy$, which is a larger difference than for the $\bar{T}_1$ beam. The standard deviation for cells follows the same trend as for the $\bar{T}_2$ beam. The nuclei standard deviation are larger than $\bar{T}_2$ but this can be due to stochastic fluctuation.
The total dose distribution for cells and nuclei can be seen in figure 4.14.

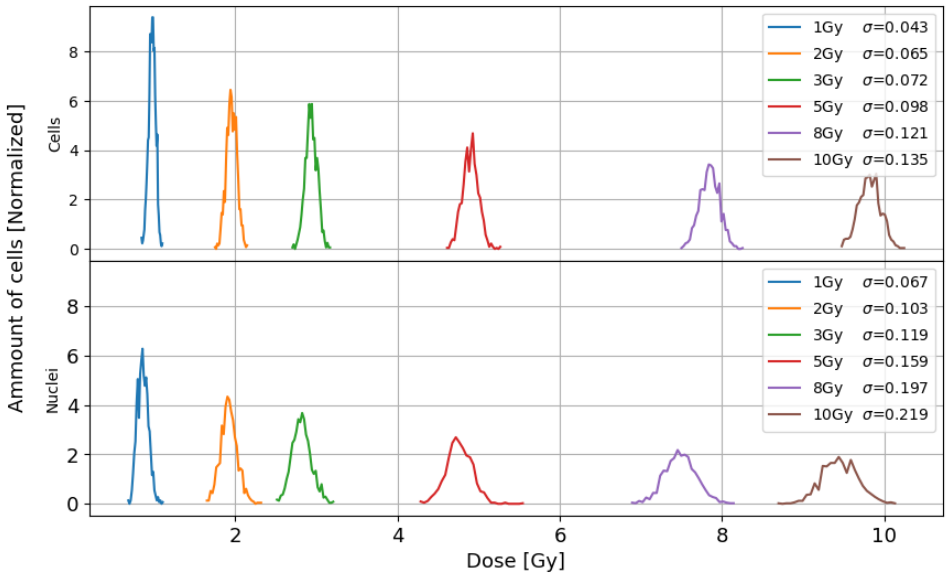


Figure 4.14: Dose distributions for the cells and nuclei irradiated with the $\bar{T}_3$ beam. The upper graph represent the cells and the graph under are the nuclei. Area under the curves are equal to 1.

The standard deviation for the nuclei is higher for the nuclei than the cells. From figure 4.14 it can be seen that the standard deviation still follows the pattern seen for higher energies, $\sigma = k\sqrt{D}$.

4.2.4 Cell irradiation with 1.2MeV protons

In the last energy state of the protons with $\bar{T}_4 \approx 1.21MeV$ a 0.49mm thick water cylinder was used. LET calculated was $18.8keV/\mu m$. Event distributions for cells and nuclei can be seen in figure 4.15:

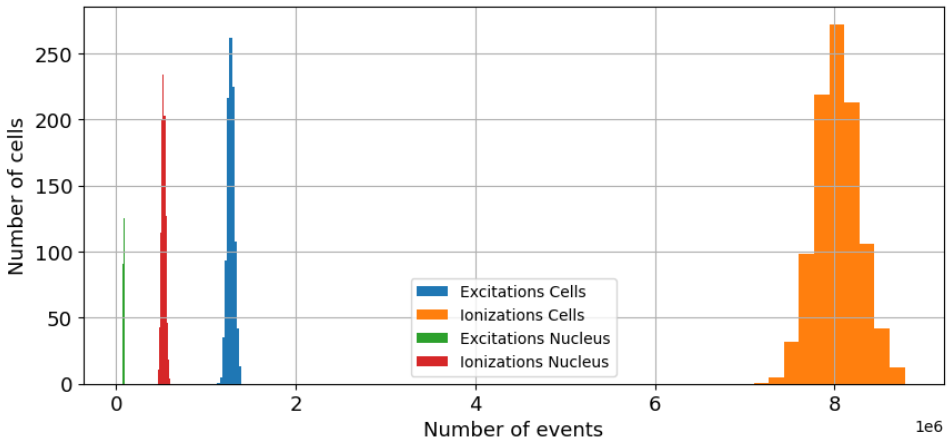


Figure 4.15: Ionizations and excitations distribution for cells and nuclei for 3Gy, energy used was $\bar{T}_4$.

The mean difference in ionizations and excitations was calculated to be 6.6. All of the mean ionizations and excitations can be seen in table 4.10 for cells and 4.9 for nuclei:

Dose	Exci μ [10^3]	Exci σ	Ion μ [10^6]	Ion σ [10^3]	Nuc
1Gy	21	2214	0.168	13	1000
2Gy	83	3314	0.326	18	1000
3Gy	83	4175	0.526	23	1000
5Gy	132	5126	0.830	28	1000
8Gy	223	6865	1.4	38	1000
10Gy	284	7526	1.8	45	1000

Table 4.9: Mean number of excitations and ionizations for nuclei when a beam of 1.2MeV protons are applied. σ is the standard deviation for a normal distribution. The highlighted row are data from figure 4.15.

Dose	Exci μ [10^6]	Exci σ [10^3]	Ion μ [10^6]	Ion σ [10^3]	Cells
1Gy	0.430	21	2.8	124	1000
2Gy	0.851	32	5.3	202	1000
3Gy	1.2	39	8.0	245	1000
5Gy	2.1	49	13.4	308	1000
8Gy	3.4	65	21.4	404	1000
10Gy	4.2	64	27.6	414	1000

Table 4.10: Mean number of excitations and ionizations for Cells when a beam of 1.2MeV protons are applied. σ is the standard deviation for a normal distribution. The highlighted row are data from figure 4.15.

On average, there are about 15.5 times more events in the cell versus the nucleus. As for the previous energies dose for cells and nucleus are plotted in figure 4.16.

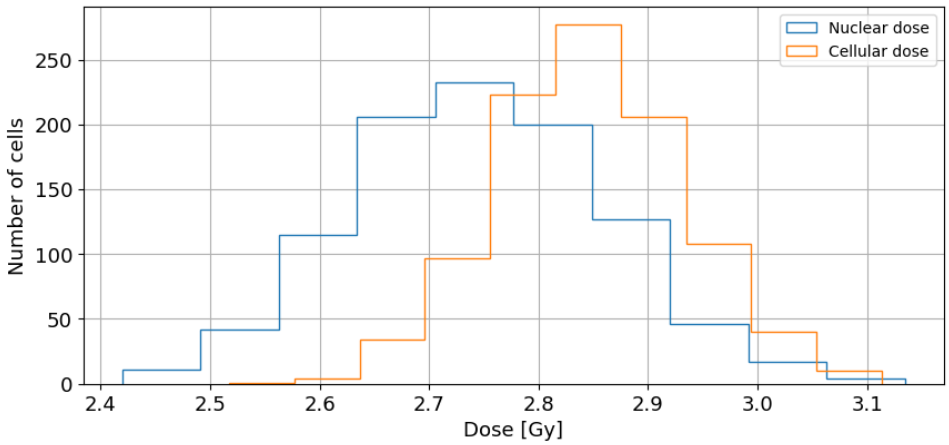


Figure 4.16: Dose distribution for cells and nuclei. The dose applied was 3Gy and proton energy $\bar{T}_4$.

The mean dose difference from cell to the nucleus is calculated to be 0.10Gy. For $\bar{T}_4$, the standard deviation for the nucleus is prominently larger, at least for 3Gy. The total dose distribution can be seen in figure 4.17.

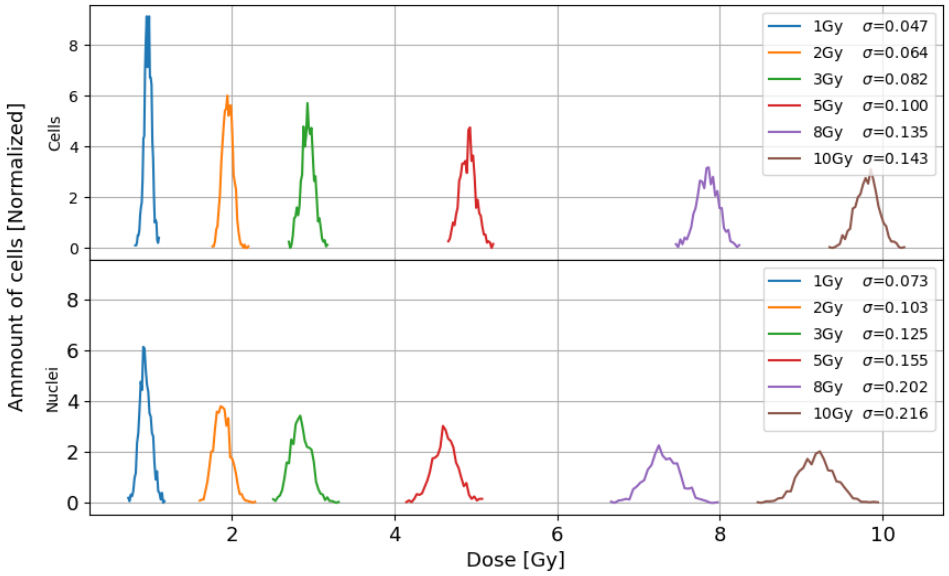


Figure 4.17: Dose distribution for cells (upper plot) and nuclei (lower plot) for a 1.2MeV beam.

Seen in 4.17 the $\bar{T}_4$ beam has the highest standard deviation for all of the beams as well as the largest difference in prescribed dose to the disc and simulated mean dose to cells and nuclei. The standard deviation still shows a pattern in standard deviation, as seen in previous simulations.

4.2.5 Comparing doses at different energies

For the relatively high energy ($\bar{T}_1$) it can be seen in figure 4.8 that the difference in the prescribed dose to the cell disc and nuclei are not too different. When comparing this to the low energy protons, for example $\bar{T}_4$ there is a difference in dose for the nuclei and the prescribed dose to the cell disc as seen in figure 4.17. Therefor it can be of interest to compare the dose to nuclei for all of the energies used for 1Gy, 5Gy and 10Gy. The reason these were chosen was because it gives broad range of doses. The plots for dose distribution to nuclei for 1Gy, 5Gy and 10Gy can be seen in 4.18, 4.19 and 4.20 respectively.

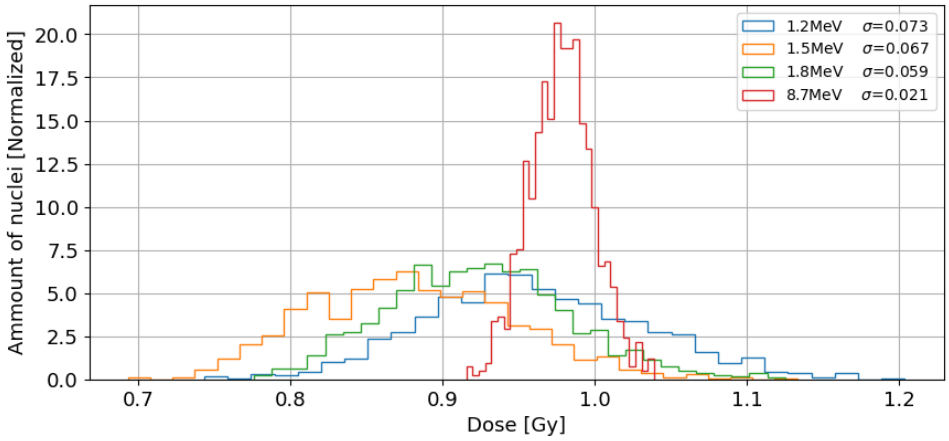


Figure 4.18: Dose distribution for different energies, prescribed dose to the cell disc are 1Gy. Curves are normalized so area under curve=1.

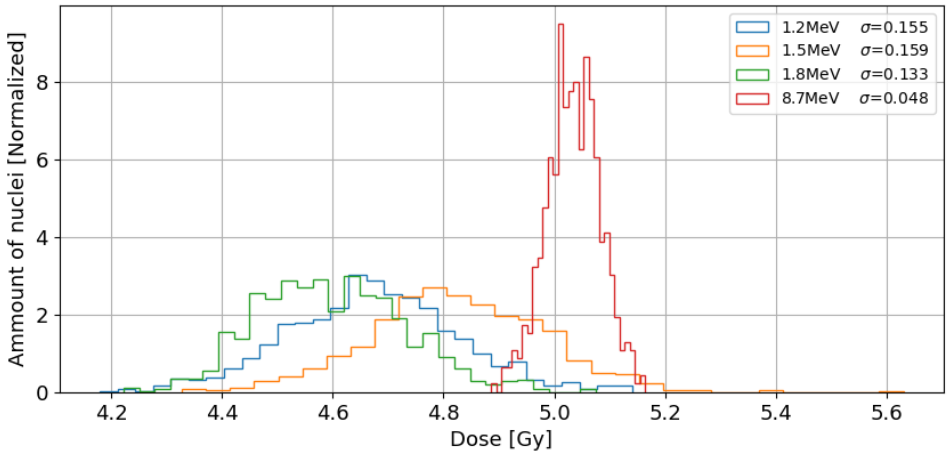


Figure 4.19: Dose distribution for different energies, prescribed dose to the cell disc are 5Gy. Curves are normalized so area under curve=1.

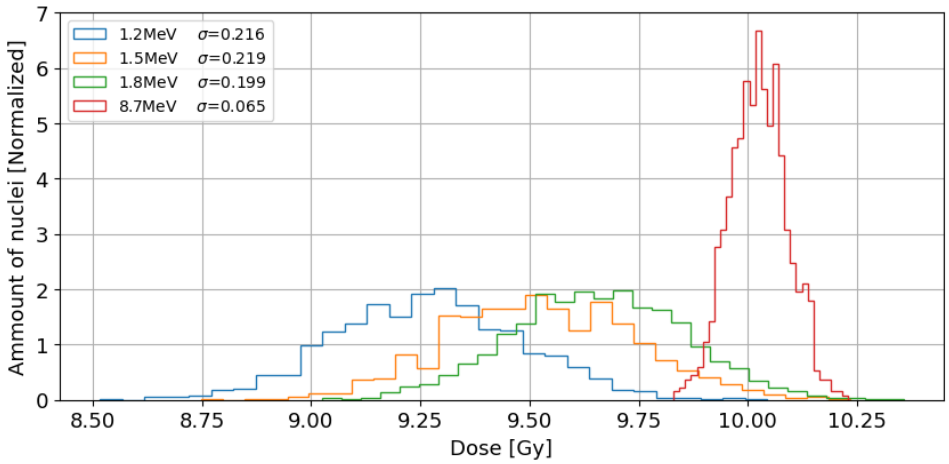


Figure 4.20: Dose distribution for different energies, prescribed dose to the cell disc are 10Gy. Curves are normalized so area under curve=1.

In figure 4.18, 4.19 and 4.20 the simulated dose to the nuclei with $\bar{T}_1$ are close to the prescribed dose for every dose. The $\bar{T}_2$, $\bar{T}_3$ and $\bar{T}_4$ simulated doses to nuclei are lower than the prescribed dose by 10% for 1Gy and 5% for 10Gy. The lower dose to nuclei will be further examined in 4.3.

4.3 Analysis of the modeling

To check if the main routine was stable with consistent results, 3 simulations were performed. 3 individual simulations with 1.8MeV protons and a dose of 3Gy were simulated and dose histogram for nuclei can be seen in figure 4.21.

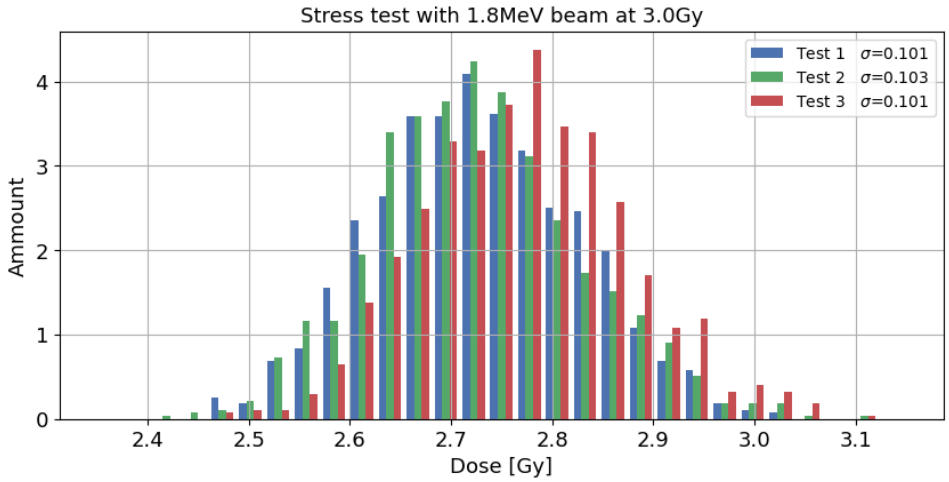


Figure 4.21: A histogram over the stress test preformed. Each of the curves are normalized and the area under the cuve are 1.

In the stress test, each simulation contained 1000 cells. For the same dose and energy the stress test simulated a mean dose of $\mu = 2.73Gy$, $\mu = 2.73Gy$

and $\mu = 2.77Gy$.
When all of the simulations are done, it is possible to compare dose differences for dose delivered to cells and nuclei. In figure 4.22 the dose difference from cell to nucleus for each simulation can be seen.

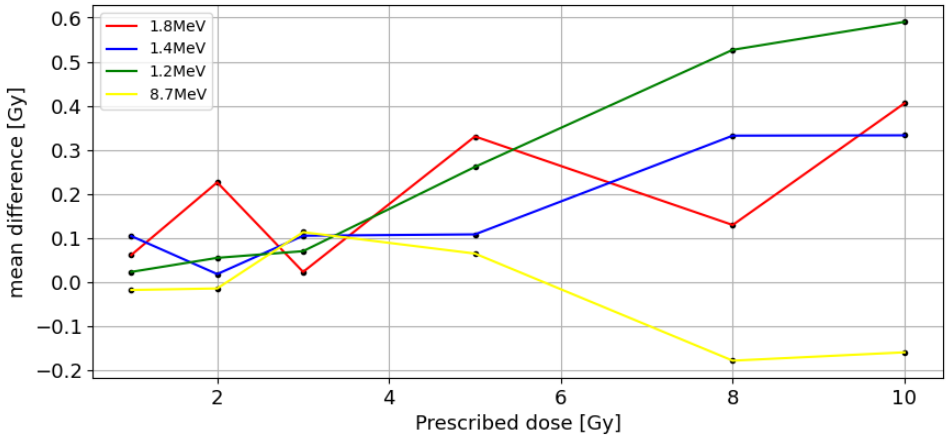


Figure 4.22: Mean difference in doses between the cell and nucleus. Negative values on the y-axis means a larger dose to the nucleus. The x-axis is the prescribed dose.

In figure 4.22 the average difference tends to be larger at large doses. The only two consistent throughout dose differences are $\bar{T}_3$ and $\bar{T}_4$ as the difference increases consistently when dose increases. The $\bar{T}_1$ beam has a tendency to fluctuate around zero. The $\bar{T}_2$ beam have a much more stochastic fluctuation than $\bar{T}_3$ and $\bar{T}_4$. The Stress test in figure 4.21 indicate a dose should not fluctuate with more than 0.05Gy for a 3Gy dose. There is some tendency for the dose to fluctuate from cells to the nucleus, though this fluctuation will always be positive for lower energy protons ($\bar{T}_2$-$\bar{T}_4$).

The standard deviation (SD) for the dose distribution, as mentioned, has a pattern related to the square root of the dose and a constant dependant on the proton energy. In figure 4.23, this assumption can be visualised By plotting SD for each energy at every dose simulated:

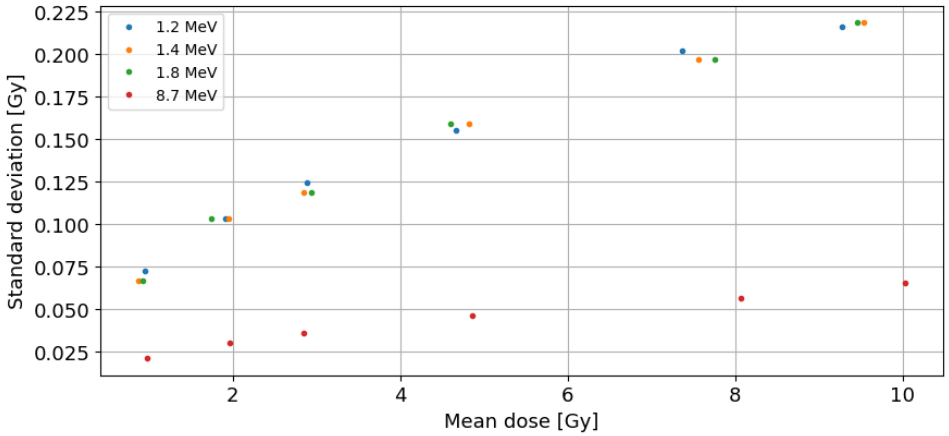


Figure 4.23: The standard deviation (y-axis) for each energy as a function of dose (x-axis). The standard deviation used for the nuclei.

In figure 4.23 the dose and SD are approximated from a normal distribution fit model applied to the data from the simulations. Several multi variable fit models were tested to check the relationship between dose, energy and the SD. Among the models tested was model 4.5*a*, 4.5*b* and 4.5*c*:

$$\sigma(\bar{T}, D) \approx \beta \frac{\sqrt{D}}{\sqrt{\bar{T}}} + \beta_0 \tag{4.5a}$$

$$\sigma(\bar{T}, D) \approx \beta_1 \frac{\sqrt{D}}{\sqrt{\bar{T}}} - \beta_2 \frac{1}{\sqrt{\bar{T}}} + \beta_0 \tag{4.5b}$$

$$\sigma(\bar{T}, D) \approx \beta_1 \frac{\sqrt{D}}{\bar{T}^2} - \beta_2 \sqrt{D} + \beta_0 \tag{4.5c}$$

$\bar{T}$ is the mean energy of the beam, D is the dose and β_n are the fit parameters. Model 4.5a has a adjusted $R^2 = 0.92$ and a p-value$= 8.39 \times 10^{-14}$. Model 4.5b ends up with a adjusted $R^2 = 0.98$ and a $p - value < 2.2 \times 10^{-16}$. Model 4.5c has a adjusted $R^2 = 0.88$ and a p-value$= 1.7 \times 10^{-9}$ and is the worst model of the three, and can be discarded. The full analysis for model 4.5a and model 4.5b can be found in table 4.11 and 4.12 respectively.

	Estimate	Std. Error	t-value	p-value (>|t|)
β_0	-0.04	0.01	-4.95	0.55
β_1	0.17	0.01	16.39	8.14×10^{-14}

Table 4.11: Estimates for model 4.5a along with the error, t-test value and p-value for the t-test.

	Estimate	Std. Error	t-value	p-value (>|t|)
β_0	0.019	0.033	0.59	0.55
β_1	-0.17	0.045	-3.9	0.000881
β_2	0.27	0.032	8.4	5.51×10^{-8}

Table 4.12: Estimates for model 4.5b along with values for error, t-test and p-values for the t-test.

In table 4.12 it can be seen that the intercept (β_0) for model 4.5b, does not have a significant effect on the model, and will not be included in the final model. Model 4.5a tends to undershoot for lower doses and overshoot for higher doses. Model 4.5b does have a higher accuracy on lower doses but overshoots more than model 4.5a for higher doses. In the end, model 4.5a is used to approximate SD for different doses to nucleus when different energies are applied. In figure 4.24 model 4.5a can be seen applied over the estimates from all of the simulations seen in figure 4.23.

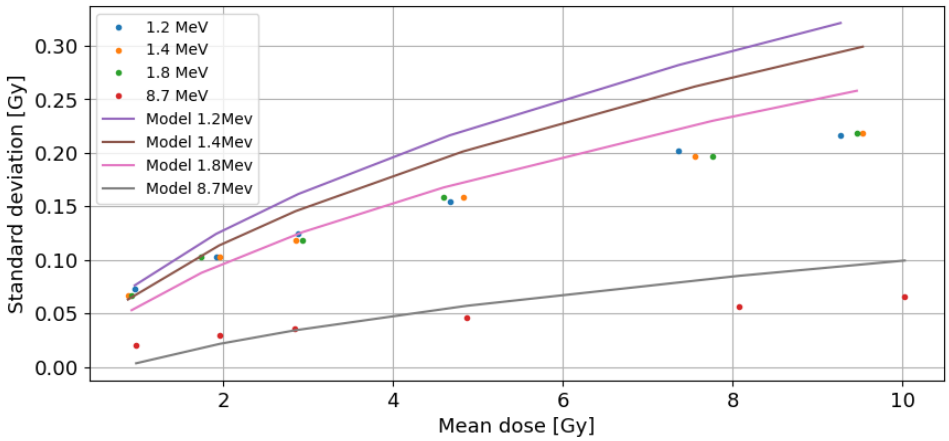


Figure 4.24: Model 4.5a plotted versus data points from figure 4.23.

Even though the model 4.5b is the best on paper, it overshoots the data by a larger margin than model 4.5a even though adjusted R^2 is accounted for. Model 4.5a also has an advantage in being simpler than model 4.5b.

4.4 Spatial analyses

The prescribed dose in the spatial statistical analysis are based on the mean expected protons to hit a nucleus. The algorithm used to simulate proton distribution for all of the statistical analysis are based on a random distribution over the nucleus. This is calculated by modifying the program which calculates the total number of protons over the cell disc, changing the irradiated volume from the cell disc to a nucleus with $r_{nucleus} = 6\mu m$ and a height of $4\mu m$. The expected number of protons to hit a nucleus to achieve a prescribed dose can be found in table 4.13:

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	47	12	10	9
2Gy	94	24	20	18
3Gy	140	36	30	28
5Gy	234	60	51	46
8Gy	374	97	81	74
10Gy	468	121	102	93

Table 4.13: The mean expected number of protons to hit the nucleus of a cell for a given dose and a given beam energy.

To visualize how the protons might look from a beam's eye view, a quick simulation of only the incident position of protons was performed, as can be seen in figure 4.25.

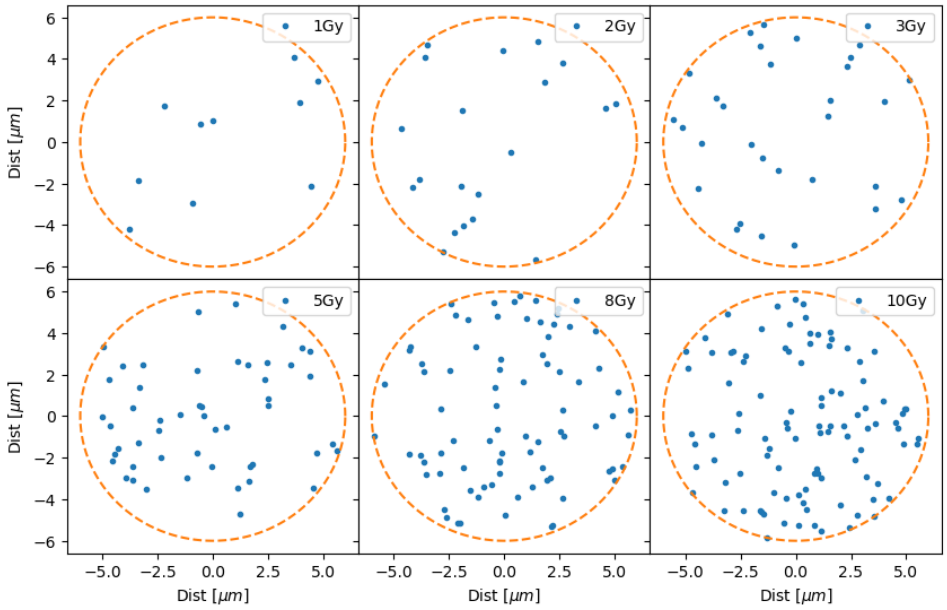
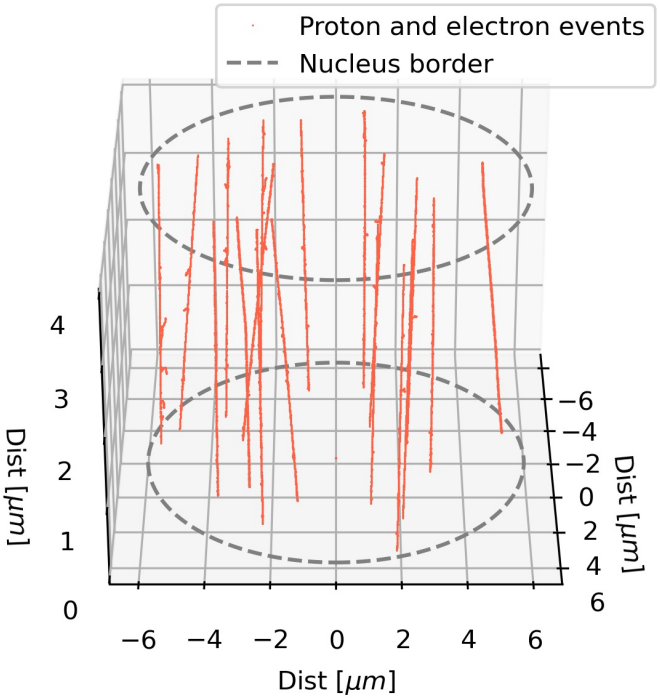
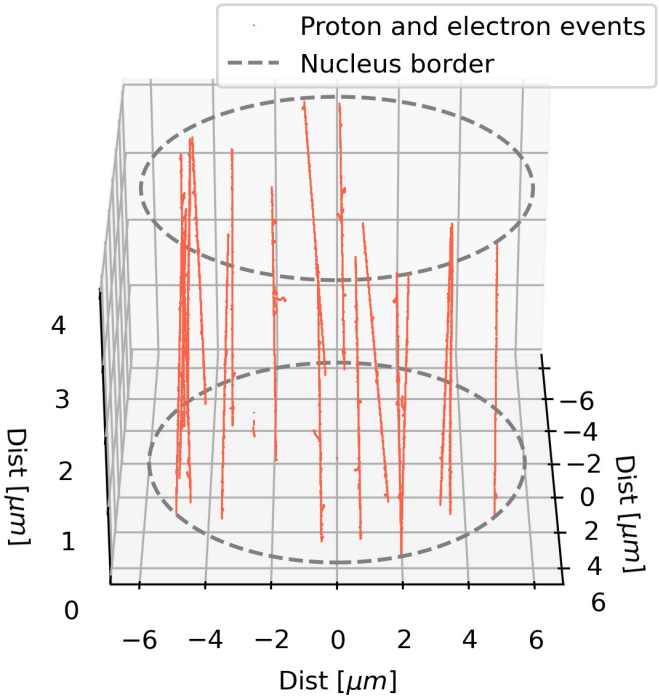


Figure 4.25: Simulation of incident proton position in a nucleus. Number of protons used are from the 1.5MeV protons in table 4.13. The dose to the nucleus is in the upper right corner of every sub figure. Every dot (blue) is a proton, and the dashed line (orange) is the nucleus border.

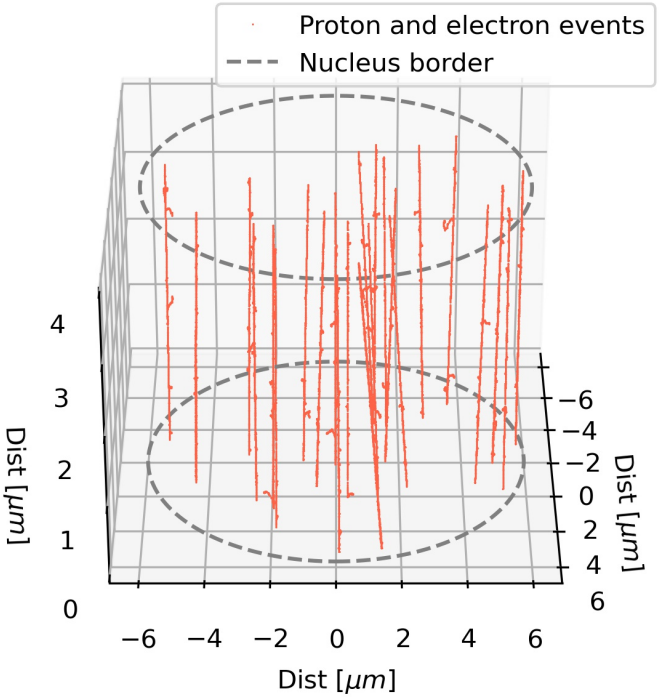
Protons in nuclei can also be visualized in 3D space for different energies as seen in figure 4.26:



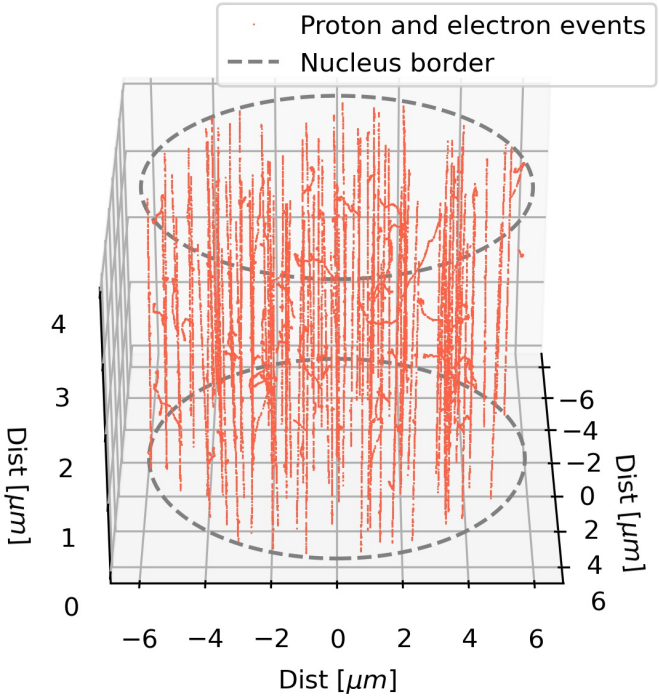
(a) $\bar{T}_4$ protons.



(b) $\bar{T}_3$ protons.



(c) $\bar{T}_2$ protons.



(d) $\bar{T}_1$ protons.

Figure 4.26: 2Gy prescribed dose for the four energies used in the analysis, number of protons from table 4.13. Dashed gray lines are the nucleus border. Red dots illustrates ionizations and excitations from both protons and electrons.

4.4.1 Moran's I

As mentioned the nucleus is divided into voxels where hits are registered within a each voxel. A good size to approximate damage to DNA would be to have voxels which are $2nmx2nmx2nm$ since 2nm, is the diameter of a DNA strand. Moran's I full calculations can be seen in table 4.14, 4.15 and 4.16 for a voxel size of 15nm, 50nm and 100nm respectively. The analysis with the voxel size of 15nm are limited to 1Gy and 2Gy.

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	0.086	0.087	0.098	0.098
2Gy	0.097	0.090	0.099	0.085

Table 4.14: Mean Moran's I for different energies at different doses. Grid set at $15nm \times 15nm \times 15nm$.

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	0.078	0.064	0.061	0.050
2Gy	0.077	0.062	0.053	0.043
3Gy	0.085	0.046	0.056	0.043
5Gy	0.081	0.060	0.053	0.067
8Gy	0.083	0.062	0.049	0.049
10Gy	0.082	0.060	0.049	0.045

Table 4.15: Mean Moran's I for different energies at different doses. Grid set at $50nm \times 50nm \times 50nm$.

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	0.056	0.056	0.034	0.035
2Gy	0.076	0.062	0.030	0.036
3Gy	0.067	0.053	0.037	0.028
5Gy	0.081	0.037	0.035	0.041
8Gy	0.080	0.040	0.037	0.036
10Gy	0.081	0.043	0.036	0.028

Table 4.16: Mean Moran's I for different energies at different doses. Grid set at $100nm \times 100nm \times 100nm$.

For the voxel size equal to $50nm$ (table 4.15) is showing a trend, Moran's I will decrease but is dose independent. Table 4.16 for the $100nm$ voxels shows the same trend as table 4.15, decreasing Moran's I with decreasing energy but dose independent.

4.4.2 Geary's C

The weight matrix element w_{ij} in the Geary's C (formula 2.36) are either 0 or 1 depending on the distance between the two events calculated in each iteration. If the distance is larger than the cut parameter, the weight matrix element is set to 0, and if the distance is smaller than the cut parameter it is set to 1. Geary's C's full calculations can be seen in table 4.17, 4.18 and 4.19 for a cut value of 15nm, 50nm and 100nm respectively. Geary's C was calculated with 3 different parameters due to unforeseen results.

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	5.25×10^{-7}	1.22×10^{-7}	4.77×10^{-7}	1.98×10^{-6}
2Gy	1.5×10^{-6}	2.87×10^{-7}	3.26×10^{-7}	1.81×10^{-7}
3Gy	1.31×10^{-6}	5.83×10^{-7}	8.22×10^{-7}	1.81×10^{-7}
5Gy	1.4×10^{-7}	5.1×10^{-7}	3.07×10^{-7}	2.25×10^{-7}
8Gy	1.83×10^{-7}	2.7×10^{-7}	2.8×10^{-7}	2.9×10^{-7}
10Gy	1.4×10^{-10}	3.44×10^{-7}	3.1×10^{-7}	2.9×10^{-7}

Table 4.17: Geary's C for different energies at different doses. Parameter set at 15nm.

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	0.00013	9.24×10^{-5}	7.79×10^{-5}	6.34×10^{-5}
2Gy	0.00013	3.18×10^{-5}	8.23×10^{-5}	3.47×10^{-5}
3Gy	0.00021	8.53×10^{-5}	5.12×10^{-5}	5.55×10^{-5}
5Gy	0.00055	8.08×10^{-5}	6.54×10^{-5}	4.79×10^{-5}
8Gy	0.00027	7.12×10^{-5}	3.67×10^{-5}	5.83×10^{-5}
10Gy	0.00022	7.65×10^{-5}	5.46×10^{-5}	4.35×10^{-5}

Table 4.18: Geary's C for different energies at different doses when parameter set at 50nm.

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	0.0012	9.24×10^{-5}	7.79×10^{-5}	6.34×10^{-5}
2Gy	0.0007	3.18×10^{-5}	8.23×10^{-5}	3.47×10^{-5}
3Gy	0.0009	8.53×10^{-5}	5.12×10^{-5}	5.55×10^{-5}
5Gy	0.0018	8.08×10^{-5}	6.54×10^{-5}	4.79×10^{-5}
8Gy	0.0025	7.12×10^{-5}	3.67×10^{-5}	0.00020
10Gy	0.0009	7.44×10^{-5}	0.00056	3.01×10^{-5}

Table 4.19: Geary's C for different energies at different doses when parameter set at 100nm.

For a cut parameter set to 50nm and 100nm, there will be a difference in Geary's C for different energies.

4.4.3 Inter-Track

The intertrack algorithm finds the average distance between the mean of each proton to every other proton in a nucleus. Results can be seen in table 4.20:

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	5.6	5.6	5.8	5.5
2Gy	5.8	5.6	6.7	5.7
3Gy	5.6	5.5	5.7	5.4
5Gy	5.5	5.6	5.6	5.5
8Gy	5.5	5.7	5.3	5.8
10Gy	5.5	5.4	5.4	5.6

Table 4.20: Inter-track value for different doses with different energies. Units in the table given in μm.

Table 4.20 shows that the mean distance between each track will be around $5.6\mu m$.

4.4.4 Intra-track

Intra-track, in contrast to the other analysis was done with one proton over several iterations to get a mean since a singular proton tends to be unpredictable. The intra-track finds the average distance between each event and the results for each energy can be seen in table 4.21:

	8.7MeV	1.8MeV	1.5MeV	1.2MeV
1Gy	1.3±0.2	1.3±0.1	1.4±0.1	1.3±0.2

Table 4.21: Intra-track values gathered for a group of 20 protons. Values for the intra-track are in μm.

Table 4.21 shows that the mean distance between two events will be around $1.3\mu m$. The deviation is fluctuating around $0.15\mu m$.

Chapter 5

Discussion

5.1 Aspects of Monte Carlo simulations

Monte Carlo simulations done by other research groups, to the knowledge of the author, have not done analog (microscopic) Monte Carlo on a cell culture. First of all, the condensed history Monte Carlo simulations are mostly applied to large-scale simulations as in the RMC code [37]. This is a program simulating a reactor core. The other aspect of a radiation Monte Carlo code is done on a microscopic level. The Geant4-DNA project has several applications similar to current study. Microbeam is one such example, constructing a cellular volume in the 4 respective stages of the cell cycle (section 2.6) where the geometry is slightly changed. The cell is divided into small voxels, and hits are scored in each voxel. But Microbeam is mainly to show what the Geant4-DNA code is capable of. A more quantitative approach was done in the three-dimensional nanodosimetric characterisation of proton track structure [6]. The authors builds on the recently developed mathematical construct of nanodosimetry by looking at the energy depositions in bins at radial distances from the proton trajectory.

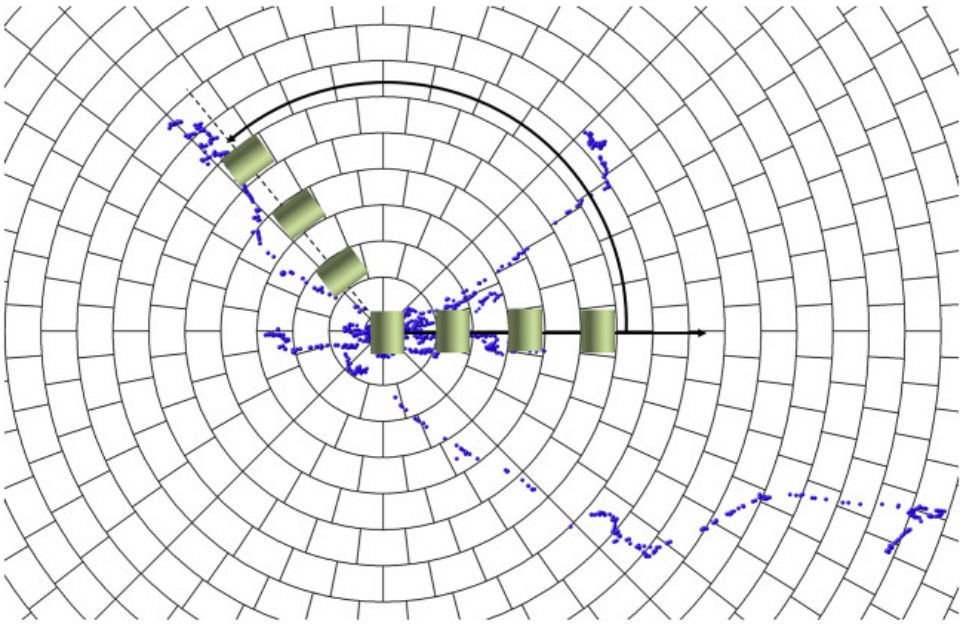


Figure 5.1: The methodology of calculating deposition in a radial distance from the proton trajectory. The propagation of protons is z direction (perpendicular to the figure). The segments are volumes used to calculate dose. Blue dots are events. [6]

Figure 5.1 is an example of how most analog Monte Carlo is used. This is due to the computational constraint when simulating energy depositions down to the eV scale. A proton track simulated with analog MC will take time and space (as discussed), therefor analog MC is more often than not used as an research tool for small setups. In addition, analog MC is used to examine the nature of radiation when interacting with matter. This is what is done, to a degree in the first part for the current study when analyzing the proton tracks.
In between the larger-scale simulations done with condensed history technique and the nm scale, we have current cell structures (micrometer scale). The simulations done with the main routine are an attempt at bridging this gap. For the present work there are concerns when simulating proton entry points on a disc, calculating their distance to the center of a cell and terminating their involvement based on whether their incident position is on the disc. This will be discussed extensively in section 5.1.1-5.2.1.

5.1.1 Radial distribution for electrons

In the current simulations, one of the main concerns is the border between the cell and nucleus and inaccuracies in the estimated dose deposition in this region. The issue is how protons and all of its cascading effects are included/excluded in the dose to the cell/nucleus based solely on it's initial position. A proton can hit near an edge (border proton), and some amount of the liberated electrons will traverse outside of the nucleus. In reality, these liberated electrons will not contribute to the dose in the nucleus. In these simulations however, they will result in an increased dose

in the cells/nuclei. Electrons liberated outside of these borders, traversing in to the volume in question are not accounted for as well, on the other hand, the difference in liberated electrons from both side of the borders needs to be substantially different for this to be a problem. In a paper on three-dimensional nanodosimetry ([6]), Braunroth et al. investigated the propagation of events and their radial dependency from the proton track. The frequency of events will decrease 10 times within 1nm and 1000 times within 10nm. Since the nucleus and cell are on the scale of micrometers, a reasonable assumption is to neglect the contribution of the secondary and tertiary electrons from border protons.

5.1.2 Divergence of protons

Though the contribution from secondary and tertiary electrons is negligible, the divergent trajectory of protons needs to be discussed. Few protons will travel in a straight path, and if the divergence had been in the range 10nm, such as the ionized electrons, this would not cause a problem. As shown in table 4.2 the proton will, on average, deviate laterally up to 300nm over traversing a cell of height $4\mu m$. A deviation on this scale is not a problem for the cell volume due to the scale of the cell relative to the divergent proton path. A 300nm deviation might be a problem for the nucleus, though. The whole track of a proton that initially hit within the border of the nucleus will be included in the dose calculations, regardless of whether it traverses the border or not. Protons with an initial hit outside the border of the nucleus will, in a similar manner, not be taken into account. For this to be a problem, protons hitting near the border within the nucleus and outside the nucleus need to give different energy contributions. Two energy terms are hypothesised; $T_{in \Rightarrow out}$ is the sum of energy deposited outside (T_{out}) the nucleus from protons hitting within the nucleus (from $0 - 6\mu m$). $T_{out \Rightarrow in}$ is the sum of energy absorbed within (T_{in}) the nucleus from protons hitting outside the nucleus (from $6\mu m - \infty$). These two terms can be derived from two simple sums, (see also figure 5.2).

Figure 5.2: Simple nucleus diagram, with a $6\mu m$ radius.

$$T_{in \Rightarrow out} = \sum T_{out} \quad \text{and} \quad T_{out \Rightarrow in} = \sum T_{in} \tag{5.1}$$

In reality the contribution from protons to $T_{in \Rightarrow out}$ hitting in the center (0) of the nucleus are minimal, this is also true for protons at ∞. If the value for $T_{in \Rightarrow out}$ is comparable to $T_{out \Rightarrow in}$ i.e $T_{in \Rightarrow out} - T_{out \Rightarrow in} \approx 0$, the

assumption that border protons will not contribute to a dose deficit and can be neglected. Border protons and electrons might be a area for further research.
Furthermore, figure 4.26 can be used to as a indicator, as protons travel in a somewhat straight line in the nucleus. Protons divergence shown in figure 4.1 are exaggerated as mentioned, with a truncated z-axis and stretched x and y-axis. Just by examine these two figures it is fair to assume the contribution from crossing border protons are minimal.

5.1.3 CPE validity

If charged particle equilibrium was, in fact true, Fano's theorem 2.23 could be used to prove that the number of charged particles within some border are the same as outside. However, the presence of CPE can easily be disproved due to the uncertainty in the dose calculations. If CPE if in fact were present, the dose would be approximately homogeneous throughout and a large SD would not be expected. A viable option to the simulations performed to this date is a full-scale nanodosimetric simulation of irradiation of a cellular culture. However, this would require time on a supercomputer or cloud computing. If a full scale nanodosimetric analog MC simulation was to be performed, the geometry cell and nucleus would impact the simulations, so, cylinders would not be the best approximation (further discussed in section 5.2).

5.1.4 LET

LET is an expected value for a charged particle with a certain velocity and charge in a given material, and its estimate has some inherent uncertainties. Comparing the current work and the benchmark ([31]) will lead to insight in proton energy loss calculations.

Beam energy	PSTAR	Thesis
1.21MeV	22.9keV/μm	18.8kev/μm
1.51MeV	19.1keV/μm	17.3keV/μm
1.81MeV	17.1keV/μm	14.5keV/μm

Table 5.1: LET values for beams at different mean kinetic energies.

In table 5.1, the PSTAR values are interpolated from data since there are not stopping power data in the PSTAR library for $1.21MeV$, $1.51MeV$ and $1.81MeV$. There is a discrepancy in the LET, for example: $2.6keV/\mu m$ and $1.8keV/\mu m$ for the $1.8M1eV$ and $1.5M1eV$ beam respectively, with beam energies from PSTAR to the current work. However, PSTAR calculate the stopping power for monoenergetic protons at optimal conditions. In contrast, in this thesis the protons have travels through 80cm air and a water bath. It is a fact that the beam will not be monoenergetic and the protons will have different energies as they are entering the cell disc. Some protons will not even have the energy to traverse the whole cell

height ($4\mu m$). These protons will have low kinetic energy and reduce the mean kinetic energy of the beam ($\bar{T}$) while not contributing to LET calculation across the cell. This will reduce the beam's total LET compared the theoretical value.
In figure 4.3 there was a discrepancy in the simulated fluence versus the theoretical. If all of the processes where included (elastic scattering, vibrational excitations, etc.) the simulated fluence would be up to 13% over the tabulated. So only including ionizations and excitations in the simulations gave the best correspondence with tabulated LET's.

5.2 Cell and nucleus geometry

To say that all cells are a perfect cylinder with $23\mu m$ radius and a $4\mu m$ height would be naively wrong. In reality, the cells will differentiate in size and shape. Cells approaching the M phase will be larger than a newly divided cell going into G_1. When in S phase, the cell will resemble a blob and in G_1 it will resemble a sphere. However, cylinders are a fair approximation to the mean geometry of a cell adherent to the bottom of a cell dish. The nuclei, like the cells, are approximated with cylinders at the same height and a radius of $6\mu m$.
For further research, implementing a cell shape library containing cells and nuclei in the various phases of cell cycle would be relevant. Though the gain from accurately simulating the cell shapes on such a scale is questionable. Generally, Monte Carlo simulations, is mostly an approximation to the real world, often volumes are approximated with boxes, triangles and cylinders. The cell in the Microbeam example in Geant4-DNA is approximated with several boxes, accumulated to a cell like structure. So by implementing cells in various stages of the cell cycle, one could solve the geometrical problem with approximating cells/ nuclei to cylinders.
Geometry, though, is not the only problem when modeling a cellular culture. Cells come in various sizes depending on the cell line used. The relationship between the cell and nucleus volume will also be different for each cell line and stage in the cell cycle. The volume used in these particular simulations are not representative of all cells.

5.2.1 The interface effect

Even though the cells and nuclei consist mostly of water, it needs to be acknowledged that besides water, the cells contain sugar, phosphates and other molecules. This changes the effective atomic number of the material the proton traverses, thus; changing its interaction. A change in path, energy deposition and so on can result in a difference between the true dose and the calculated dose. The true dose being the dose cells actually will receive in a real experiment.
To see if water is a good substitute for the cellular matrix, a comparison needs to be made between liquid water and tissue equivalent materials,

MS20 (epoxy with filler elements, mimicking human tissue) and skeletal muscle tissue are two candidates for comparison. These materials are choosen due to simelar molecular atomic composition (Z_{edd}) as human tissue. A proton stopping power between these materials is shown in in figure 5.3.

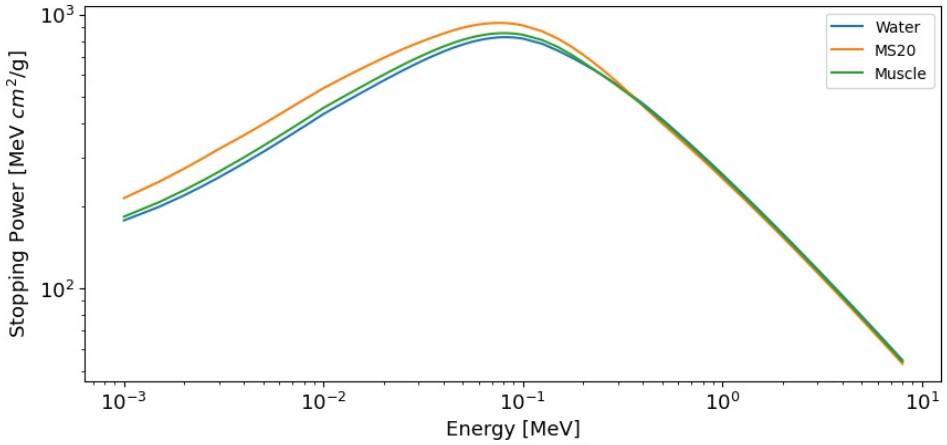


Figure 5.3: Mass stopping power for protons from 10keV to 8MeV for 3 materials. Both axis are displayed in a logarithmic scale. Stopping powers are gathered from the PSTAR library [31].

From figure 5.3 it can be seen that there is not a huge difference in stopping power between water and the two other materials, with the energies used in the current simulation. A difference can however be seen for $T < 100keV$, where water has a lower mass stopping power than the comparison materials. This might be due to shell corrections as this is affected for protons with low energy. An increase in stopping power will perhaps lead to a larger dose difference in cells and nuclei. Figure 4.22 can indicate a scenario where the dose difference between cell and nucleus will increase for lower energies, a increase in stopping power will lead to higher LET and thereby increasing the difference since $T \propto (LET)^{-1}$. From model 4.5a it can be assumed that an increase in LET will also lead to an increase in standard deviation in the dose distribution. From the PSTAR library, [31] the mean ionization potential can be found for these 3 materials. Water and MS20 have similar attributes with $\rho_W = 1g/cm^3$, $I_W = 75eV$ and $\rho_{MS20} = 1g/cm^3$, $I_{MS20} = 75.1eV$ so there is no large deviance between the tissue substitute and water. Muscle tissue, on the other hand, has a larger density at $\rho_{muscle} = 1.04g/cm^3$. The mean ionization value at $I_{Muscle} = 75.3eV$. So water compared to the two control materials is a good substitute for simulating heavy charged particles.

For electrons, the NIST data goes down to 10keV, as for the proton data for stopping power, which is above the maximal energy transfer from proton to electron. Though it is desirable to have electron stopping power data in the low energy region, it is still helpful to compare the three materials already discussed for protons (MS20, muscle and water).

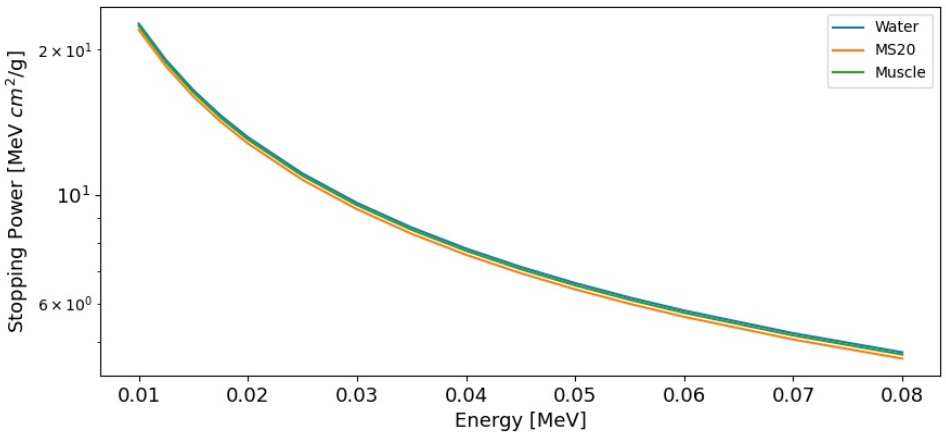


Figure 5.4: Mass stopping power for electrons from 10keV to 0.8MeV for 3 materials. The y-scale are in log scale and the x-scale is conventional. Stopping powers are gathered from the ESTAR library [32].

By inspecting figure 5.4 it can be concluded that if the proton interface effect is negligible, it is so for electrons are as well.

5.3 Cross section models in Geant4-DNA

Cross section models in the Geant4-DNA toolkit are semi-empirical, derived from cross section data published by the International Commission on Radiation Units Measurements (ICRU), but the data can also be published by another independent organisation. This creates a plethora of data, all with some degree of uncertainty which will carry over to the models implemented.

5.3.1 Electron cross section

Electron cross section data to this day is only available for water vapor. It is important to mention this due to the increase of polarization in condensed media for charged particles. Emfietzoglou and Nikjoo [15] discuss this to a fair degree. They found that if the polarization is not corrected for, the electron cross section for 100 eV electrons in water vapour (corrected for density) is 1.7 times larger than the measured outcome in liquid water. As mentioned in section 2.1.6 the polarization effect decreases stopping power as the atoms gets closer. In M.A. Bernal et al., [5] it is mentioned that all inelastic scattering cross sections today are based on the Emfietzoglou model [14] using the dielectric response function (DR), a method based on the theoretical framework of the plane-wave Born approximation. The dielectric approach is currently state-of-the-art in inelastic electron transport Monte Carlo algorithms [13]. One assumption made with the dielectric approach and the use of optical reflectance data are that the charged particle have a greater velocity than the shell electrons. In Geant4-DNA, there are a few models to choose from, as seen in figure 5.5 for excitations and 5.6 for ionizations.

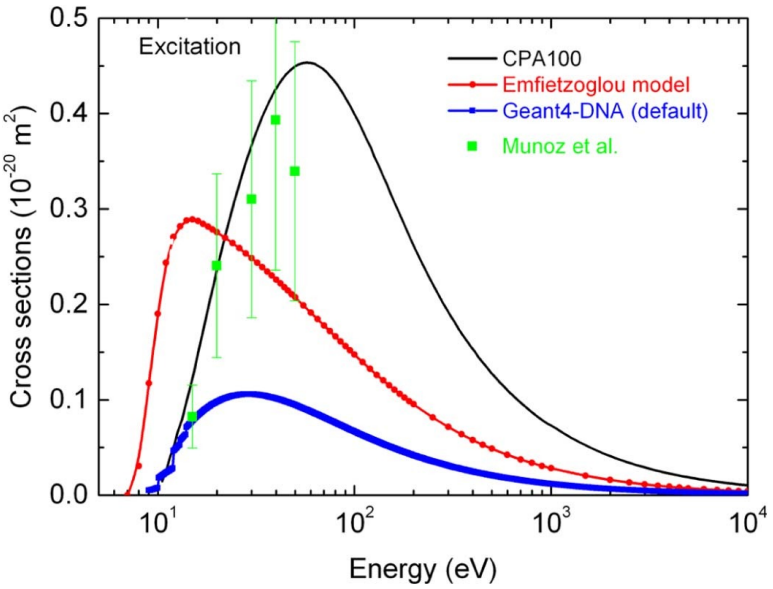


Figure 5.5: Three different models for total electron impact excitation cross section (marked as lines), with the CPA100 as black, G4DNABronModel/Geant4-DNA as blue and Emfietzoglou as red. Munoz et al are data for electron excitation cross section in water vapor.[5]

The three cross sections in figure 5.5 shows some of the excitation models which can be chosen when working with the Geant4-DNA toolkit. They differ greatly in size and the outcome of a simulation can be altered by using another model. The CPA100 model seems to be the best the best when comparing to the only cross section data, but the data from Munzo et al [29] are for electron tracks bellow 100eV in water vapor. Thus, the CPA100 model does not account for the polarization effect, and can be discarded for excitations. In the simulations, G4DNABronmodel where used. This is state-of-the-art for modeling low energy loss for low energy electrons [5] and was included in the 10.2 beta for Geant4. Inclusion of this will increase the simulations' the accuracy compared electron tracks in liquid water.

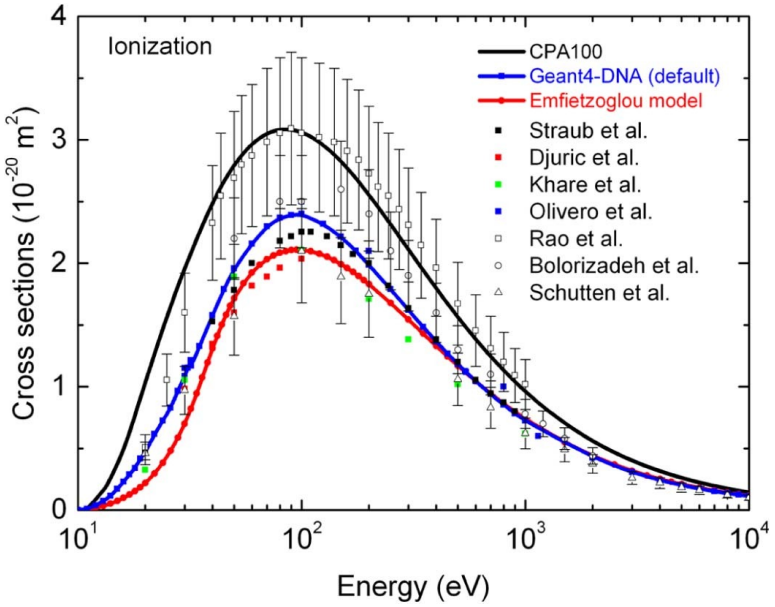


Figure 5.6: The same three models as in figure 5.5 but for total electron impact excitation cross section (marked as lines), with the CPA100 as black, G4DNABronModel/ Geant4-DNA as blue and Emfietzoglou as red. Plotted in the same figure are data points for several experiments [5].

The total electron ionization cross section data are not as different from one another as the excitation cross sections. Geant4-DNA and Emfietzoglou model are the two main contenders. CPA100 is not as stable for low energies ([14]) but includes some extra materials in addition to liquid water. Since liquid water is a good substitute for the biological matter (section 5.2.1), the CPA100 model is discarded. Furthermore, CPA100 is not in total agreement with data from liquid water as CPA100 uses cross section data from water vapor.
In essence, there are no cross section data for electrons in liquid water; that is true. However the DR mentioned above uses optical data from liquid water applied both to the Born model (blue line in figure 5.6) and Emfietzoglou-model (red line in figure 5.6) with data form Heller et al. [16]. Both models have been altered for the low energy domain to comply with experimental data. The difference between these two models (Born and Emfietzoglou) are minimal, especially when applied in the manner of the simulations done, so each could be picked as the model. For the purpose of these simulations, the Born model was chosen.

5.3.2 Proton cross section

Protons cross section data is necessary to get accurate dose calculations.
The published cross section data for protons, both for elastic and non elastic interactions are not as variable as for electrons, so choosing a model in the current frame will not have a major impact on the results of the simulations.

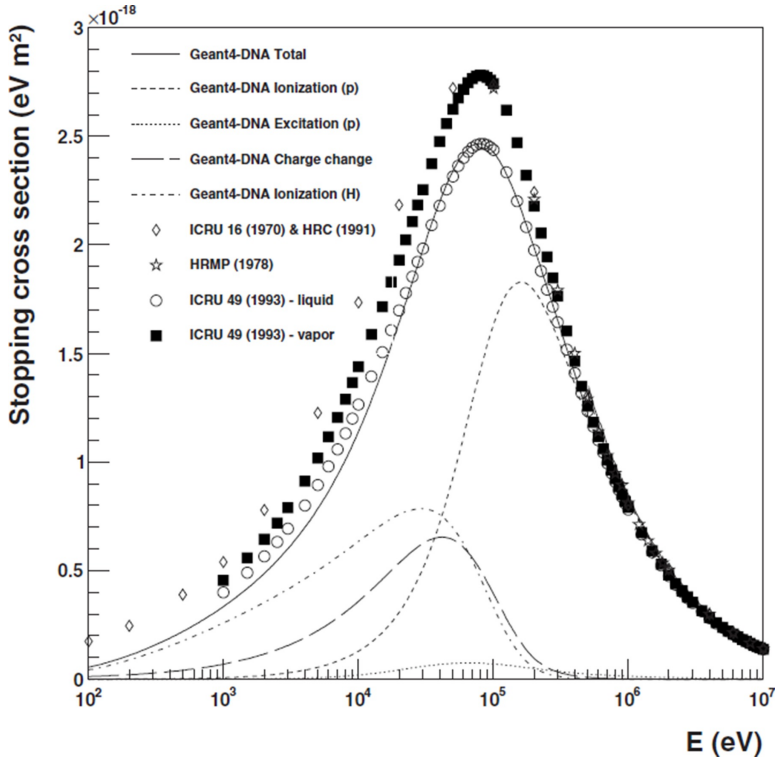


Figure 5.7: Total proton cross section model (solid line) and total ionization cross section (dashed line). Points are measurements in liquid (circles) and vapor (dots). Total ionization cross section marked as dotted line and excitation as semi dotted. [43]

Figure 5.7 shows Geant4-DNA implementation of the total proton cross section for water. The experimental data from ICRU [4] (circles) shows good agreement with the Geant4-DNA total cross section. Since a increase in cross section will increase probability for interaction, it is expected that there are some similarities between the proton stopping power for liquid water (figure 5.3) and cross section data for protons in liquid water (figure 5.7), this is true, at least for data down to $100keV$. Measuring the total cross section for protons does not suffer the same problems as for electrons as the data are collected in liquid water and does not need to have a correction for the polarization effect. This is much due to the research done by M. J. Berger at el. [4] at the ICRU. The ionization cross section in figure 5.7 are dominant for all energies down to about $30keV$, where the excitation cross section is dominant til the proton is captured. All of the results from the simulations involves protons in the energy range $1.2MeV - 8.7MeV$ with a dominant ionization outcome. When examining figure 5.7 it could be interesting to examine the impact of excitations for low energy protons in further research.

5.4 Dose analysis

When assessing irradiated cell cultures experimentally, one does not know the accurate dose. The dose in cell cultures is an estimate calculated from

fluence estimates and energy. Cell survival is based on counting the cells which have proliferated to form a colony after irradiation.
The Monte Carlo simulations done show a accurate dose estimate for cells and nuclei for the dimensions used. One result which is of interest is the dose difference from cell to nucleus, examined in figure 4.22. This might be a sort of geometric problem or statistical phenomena not quite understood. A thought experiment, if the number of particles decreases but each particle has a higher impact on dose one would expect a dose to stay the same for a decreasing volume. As the volume decreases it is however expected that the SD would get larger as each hit has a higher impact totally. The dose difference is not totally understood and is a topic for further research.

5.5 Dose variations

The standard deviation in dose to the nucleus and to a certain extent the cell is an important topic to discuss. Some of the nuclei irradiated with low energy protons will receive, in the extreme case 15% less dose than expected. In cell survival studies this can result in large errors for the low energy protons. Fact of the matter is that it could be interesting to have a model for expected local deviance within the tumor. While this might be implemented in the future there are some physical aspect one cannot predict while working with protons as they are stochastic by nature. One would need to do analog MC for every patient for a relatively large volume such as a tumor, the value from doing this would be most likely minimal. Analog MC is however a good method to understand the nature of radiation or testing implementation of a new system as for the cell experiments at OCL.

Model 4.5a in figure 4.24 is the best approximation for standard deviation with equation 4.5a as the model. No matter if model 4.5a, 4.5b or 4.5c are chosen the standard deviation tends to approach 0 with higher energies (see figure 5.8).

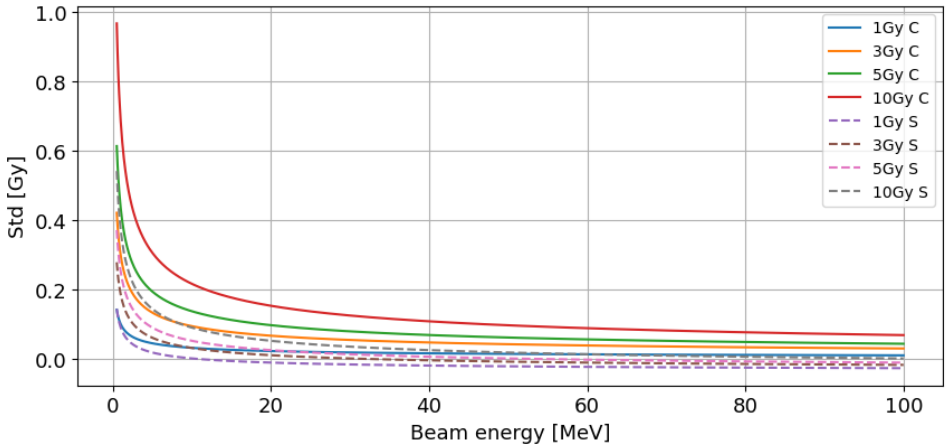


Figure 5.8: A extended range for the standard deviation as a function of energy for different doses. In the legend model 4.5b is denoted C for complex and model 4.5a is denoted S for simple.

A limitation of model 4.5a is it's predicted value for low doses and high energies where it predicts a negative SD. On the other hand, high dose and low energy predicts a standard deviation of up to 10% of the total dose for model 4.5b. While this is in the area of the Bragg peak and protons tends to deviate sufficiently, it needs to be considered if this model is more accurate. But for the energy ranges used in these simulations, model 4.5a is the best.

5.6 Track analysis

Perhaps it was naive to assume discovering meaningful patterns and trends out of the simulations performed on the nucleus by only applying a single statistical model such as Geary's C, Moran's I, Inter-track and Intra-track. Every simulation involving more than a single proton track was only preformed once for their respective voxel size. The reason several voxel sizes where used is the hope of discovering some meaning full pattern when varying the sizes. The Intra-track value are based on 20 protons for each respective proton energy. Ideally, one would do several simulations for each dosage, energy and cut value/voxel size to check if there is any variations with the simulations. The goal was to be able to predict clustering of ionizations within the nucleus, and thus the extent of complex damage to the DNA. Experiments involving protons at different energies and x-rays shows that LET and survivability are closely related [17]. As mentioned the leading theory behind this is that a higher LET will result in a more complex damage to DNA in the nucleus, prohibiting the cell's ability to repair the damage.
Moran's I will measure global spatial autocorrelation, over the whole area in question (the nucleus). We expect to see a higher value (greater autocorrelation) for higher energies due to that an increase of proton paths within the nucleus would lead to a increase in global autocorrelation. Though in table 4.14 to 4.16 the difference in autocorrelation between energies are

minimal. When looking at the expected number of tracks for a given dose (table 4.13) one would expect that Moran's I would be about the same for T=8.7MeV at 2Gy with 94 proton tracks and T=1.2MeV at 10Gy with 93 proton tracks if Moran's I are dependant on number of tracks. But as seen in table 4.15 the values are 0.077 and 0.045 for 8.7MeV and 1.2MeV respectively. The same is true for a voxel size of 100nm, this can be seen in table 4.16; where again the 8.7MeV are about twice that of 1.2MeV. Another explanation is to assume a proton will "overwhelm" the algorithm to the point where the amount of protons don't have an impact on the outcome. If the outcome is only dependant on LET the expected outcome would be a higher autocorrelation with higher clustering.

As a reminder, Geary's C is a local autocorrelation inversely correlated to Moran's I and is in the interval 0 to infinity where 0 is total spatial autocorrelation. Analysis done (table 4.18 and 4.19) shows a decreasing Geary's C with increasing LET which thus is expected. In addition to this, the C index seems to stabilize with increasing dose. Thus, there is a higher degree of spatial autocorrelation with decreasing energy but not with increasing dose.

Inter-track simulations shown in table 4.20 shows on average that a proton will be $5.5\mu m$ from another proton no matter how many in a nucleus with a radius $6\mu m$. This is more or less what is expected. Wolfram MathWorld [26] that if two points where placed randomly on a disc with $r \in [0,1]$ and $\theta \in [0,2\pi]$, they would be, on average $128/45\pi$ apart. $128/45\pi = 0.905$ and $5.5\mu m/6\mu m = 0.91$ so there are good agreement with theory for the Inter-track. Intra-track results shown in table 4.20 are within what is expected. Events happen on average $1.35\mu m$ from every other event within a given proton track.

Another way to approach clustering within a nucleus, cell or cellular culture for future analysis are a method using polyhedron approach discussed in the article *"An Extended Density based Clustering Algorithm for Large Spatial 3D Data using Polyhedron Approach"* [3]. In this paper the author uses a polyhedron to estimate the clustering and spatial correlation on large data sets as shown in figure 5.9.

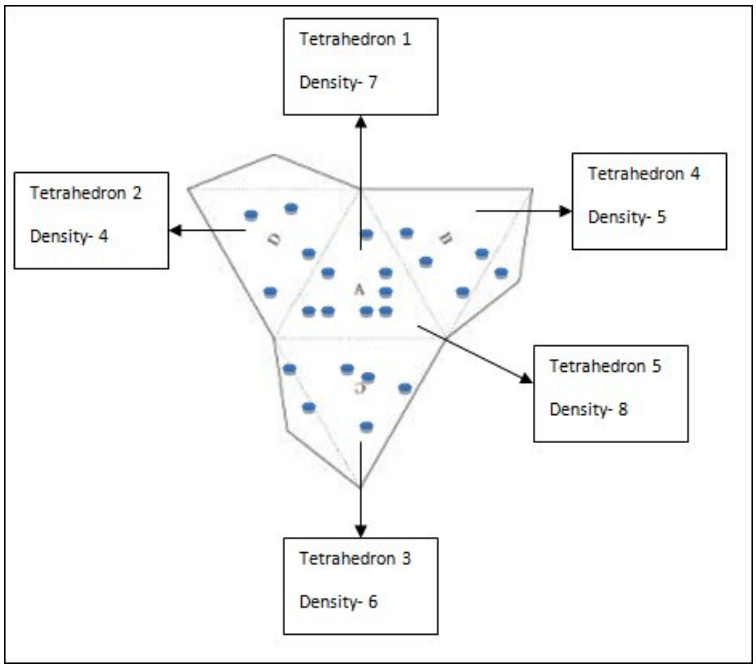


Figure 5.9: The polyhedron approach projected in 2D plane. [3]

In figure 5.9 it can be seen how they approach spatial autocorrelation. The algorithm makes tetrahedrons and summarizes events within each tetrahedron. By clustering events together in tetrahedrons, there is no need to calculate individual distances between every event, reducing the computational time. In a way it is the same approach as the Moran's I algorithm but more efficient. The reason they use a polyhedron is due to the fact that this is the smallest 3D shape and therefore the computationally most efficient. The author also claims that this tool-kit can be used for several dimensions such as time, energy and type of event. Instead of the calculations taking days when calculating Geary's C or Moran's I the tool-kit will take seconds. In addition to being a overall faster code, one can specify complexity of clustering and polygon size which makes it a powerful tool.

5.7 Temporal aspects

One part of simulations in cellular and nuclear volumes is the spatial aspects of the simulations, another aspect is the temporal aspect. As mentioned in section 2.6.2, some DNA damage only leads to cell inactivation if the damage is close enough in time and space (typically minutes). The radiation process involving a photon or a charged particle are almost instantaneous ($10^{-16}s - 10^{-11}s$). In this time frame the DNA don't got any time to repair, however to deliver a full clinical dose for any circumstance takes significantly longer than this. This gives the DNA time to repair sublethal damage.
A theory for further spatial analysis is that protons in the low energy do-

main will not be impacted by time. This is due to a higher LET will increase the amount of lethal damages to the DNA (see section 2.6.3) which maces the time aspect of when protons hit less significant. Of course, the DNA will have some sub-lethal damage but the theory is that the complex damage outweighs this. The temporal aspects could also be examined by the polyhedron approach, in addition to the spatial aspects by adding a time dimension for the events simulated.

Chapter 6

Conclusion

The goal of this thesis was to do nanodosimetric simulations on a cell culture. It can been concluded that the methodology used in the dose simulations to cell and nucleus are not perfect, but they seems to provide good estimate. The most interesting finding is the dose discrepancy between cell and nucleus for the low energy protons. This means that in the region of the Bragg peak, the most sensitive volume of the cell will not receive the prescribed dose. Moreover, a standard deviation in dose to the cell inversely proportional to the energy was found. Understandably, this is to be expected as systems in nature usually follows a Gaussian or Poisson distribution. The larger standard deviation for lower energy also make sense, when a lower amount of protons hit the nucleus but each hit counts more, there will be a larger variations.
Spatial statistics might be the way forward to examine how events are distributed within a nucleus, both spatially and temporally. Moran's I and Geary's C measures a global and local spatial autocorrelation in a data set. The hope was to see a correlation with energy as well as dose but the only indication for correlation with energy for these two algorithms for voxels/cut values above $50nm$. A cut value and voxel size of $15nm$ resulted in estimating noise. Intra and inter track results were more or less expected; if a proton track traverses $4\mu m$ one would expect events are on average $\approx 2\mu m$ from one another, which was true for inter-track as well.

Appendix A

Code

A.1 Geant4-DNA

All links are clickable.

Main Geant4-DNA repository:
https://github.com/JohannesTjeltaMaster/Geant4-DNA

A.1.1 Physicslist.cc

PhysicsList.cc:
https://github.com/JohannesTjeltaMaster/Geant4-DNA/blob/main/src/PhysicsList.cc

A.1.2 ElectronCapture.cc

ElectronCapture.cc:
https://github.com/JohannesTjeltaMaster/Geant4-DNA/blob/main/src/G4ElectronCapture.cc

A.1.3 PrimaryGeneratorAction.cc

BoxPrimaryGeneratorAction.cc:
https://github.com/JohannesTjeltaMaster/Geant4-DNA/blob/main/src/BoxPrimaryGeneratorAction.cc

A.1.4 DetectorConstruction.cc

BoxDetectorConstruction.cc:
https://github.com/JohannesTjeltaMaster/Geant4-DNA/blob/main/src/BoxDetectorConstruction.cc

A.1.5 SteppingAction.cc

SteppingAction.cc:
https://github.com/JohannesTjeltaMaster/Geant4-DNA/blob/main/src

/SteppingAction.cc

A.2 Python

All of the code written for this thesis, including plot code not included in the appendix can be fount in https://github.com/JohannesTjeltaMaster

A.2.1 TrajectoryDivert.py

```
import numpy as np
import uproot
import os

def numberOfFiles(path):
    root_files=os.listdir(path)
    if 'Res'in root_files:
        root_files.remove('Res')
    if 'analasis' in root_files:
        root_files.remove('analasis')

    return(len(root_files))

def TrackDiv(Path):
    ammount=0
    tot_diff=np.zeros(3000)
    file_of_interest=list(open('Data1_8/Res/FilesOfInterest.txt
    ','r'))
    for i,str in enumerate(file_of_interest):

        file_of_interest[i]=str.replace('\n','')

    for i,str in enumerate(file_of_interest):
        file=uproot.open(path+'/'+str)['microdosimetry']  #
    import root file as data set
        x = file['x'].array()  # import position from data set
    to numpy array
        if len(x)<2:
            continue
        elif Path=='Data1_2' and i==invalid_protons1_2[np.where
    (invalid_protons1_2==i)]:
            print(123)
            continue
        else:
            ammount+=1

        y = file['y'].array()
        z = file['z'].array()
        particle=file['flagParticle'].array()
        z=z-z[0]

        x=x[np.where(z<4000)];x=x-x[0]
        y=y[np.where(z<4000)];y=y-y[0]
        z=z[np.where(z<4000)]
        if np.amax(x)>6000:# or np.amax(y)>6000:

            x[np.where(x>6000)] = 0
```

```
            y[np.where(y>6000)] = 0
        if np.amax(y)>6000:# or np.amax(y)>6000:
            print(np.amax(y))
            x[np.where(x>6000)] = 0
            y[np.where(y>6000)] = 0        #print(np.where(z==
    np.amax(z)))
        particle=particle[np.where(z<4000)]
        #print(np.sqrt((x[0]-x[np.where(x==np.amax(x))])**2+(y
    [0]-y[np.where(x==np.amax(x))])**2))
        temp1=(x[0]-x[np.where(x==np.amax(x))])**2
        temp2=(y[0]-y[np.where(x==np.amax(x))])**2
        #print(temp1,temp2)
        temp3=np.sqrt(temp1+temp2)
        if len(temp3)>1:
            temp3=temp3[0]

        tot_diff[i]=temp3
        print(sum(tot_diff[tot_diff!=0])/ammount,np.std(
    tot_diff[tot_diff!=0]),ammount)


path="Data1_8"
#N=numberOfFiles(path)
TrackDiv(path)
```

Listing A.1: Code for the track deviation

A.2.2 LinearModel.py

```
import numpy as np;import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.stats import linregress

prob_dist = np.loadtxt('initialPosDist.txt')
r=np.linspace(0,3,100)

def func(a,x,b):
    return a*x+b
slope, intercept, r_value, p_value, std_err = linregress(r,
    prob_dist)
print(slope,'\n',intercept,'\n',r_value,'\n',p_value)
popt,pcov=curve_fit(func,r,prob_dist)
print(popt,pcov)

def plot_regression_line(x, y, b):
    y_pred = b[1] + b[0]*x
    plt.plot(x, y_pred, color = "g")

plot_regression_line(r,prob_dist,popt)
plt.plot(r,prob_dist,'.')
plt.legend(('predicted:'+str(round(slope,3))+'a +'+str(round(
    intercept,3)),'meassured data'),loc='upper left')
plt.xlabel('dist from center of disk [cm]')
plt.ylabel('probability ')
plt.title('Probability of incident particle hitting a distance
    r from center of disk')
plt.grid()
plt.savefig('linearModelProbDistDisc.PNG')
```

```
plt.show()
```

Listing A.2: Code for the linear model plott

A.2.3 Analasys.py

```
import uproot; import numpy as np
import matplotlib.pyplot as plt; import collections
from matplotlib.colors import LogNorm
import os

save_results_to = 'Data1_2/Res/'
root = ".root"
path='Data1_2/' # Data8,Data1_2
N=20000  # simulations
initialPos=np.zeros(N)
totEnergyDeposit=np.zeros(N)
ionEnergyDeposit=np.zeros(N)
x1=np.zeros(N)
y1=np.zeros(N)
mean_e=np.zeros(N)
FilesOfInterest=[]
dz=4*1e3
for i in range(N):
    file=uproot.open(path+str(i+1)+root)['microdosimetry']  # import root file as data set
    x = file['x'].array()  # import position from data set to numpy array
    y = file['y'].array()
    z = file['z'].array()
    en =file['totalEnergyDeposit'].array()
    pros = file['flagProcess'].array()
    kin = file['kineticEnergy'].array()
    par=file['flagParticle'].array()
    if len(x)<1:
        initialPos[i]=0  # zero is an indication of: did not hit
        totEnergyDeposit[i]=0
    else:
        initialPos[i]=np.sqrt(x[0]**2+y[0]**2)/10**7  # initial distance from the centre of the disk
        totEnergyDeposit[i]=np.sum(en[np.where((z>z[0])&(z<z[0]+dz))])
        ionEnergyDeposit[i]=np.sum(en[np.where((z>z[0])&(z<z[0]+dz)&((pros==13)|(pros==12)|(pros==22)|(pros==23)))])

        x1[i]=x[0];y1[i]=y[0]
        FilesOfInterest.append(str(i+1)+root)

        m=kin[np.where(par==2)]
        print(i)
        mean_e[i]=m[0]
        print(mean_e[i])

        #print(len(par[np.where(parentID!=0) and np.where(par==4)]))
mean_e=sum(mean_e)/len(mean_e[mean_e!=0])/1e6
print(mean_e)
amo=len(initialPos[initialPos==0])
```

```
probOutside=float(amo)/N
eee=sum(totEnergyDeposit)/len(totEnergyDeposit[totEnergyDeposit
    !=0])
iii=sum(ionEnergyDeposit)/len(ionEnergyDeposit[ionEnergyDeposit
    !=0])
print('\ntot energy deposit:',eee)
print('\nion energy deposit:',iii)
print('\ndifferanse mellom kun ion og total energi:',eee-iii)
plt.hist(initialPos[initialPos !=0],alpha=0.7,density=True,bins
    =60)#,normed=True)
plt.xlabel('Distance from centre [cm]')
plt.ylabel('Ammount [Normalized]')
plt.title('Radial distance the where the protons hits the
    target')
plt.grid()
plt.savefig(save_results_to + 'probabilitydist.png', dpi = 300)


plt.figure()
plt.hist(totEnergyDeposit[totEnergyDeposit!=0],alpha=0.7,
    density=True,bins=30)#,normed=True)
plt.xlabel('Total energy deposit per proton [MeV]')
plt.ylabel('Ammount [Normalized]')
plt.grid()
plt.title('Energy deposition for each proton')
plt.savefig(save_results_to + 'totenergydepo.png', dpi = 300)


plt.figure()
plt.hist2d(x1[x1!=0],y1[y1!=0],bins=50)
plt.title('2D distrebution of incident particle')
plt.xlabel('x [nm]')
plt.ylabel('y [nm]')
plt.savefig(save_results_to + '2dhist.png', dpi = 300)

hist,bins=np.histogram(initialPos[initialPos!=0],bins=100)#,
    normed=True)
output_file=open(save_results_to+"initialPosDist.txt","w")
np.savetxt(output_file,hist)
output_file.close()

output_file1=open(save_results_to+"FilesOfInterest.txt","w")
for stuff in FilesOfInterest:
    output_file1.write('%s\n' % stuff)
```

Listing A.3: Code for LET

A.2.4 Dose.py

```
import numpy as np

def dose(Gy,mean_energy_dep):
    eV=1.60218e-19  # ev to Joule
    volume =  1.161e-5#1.161e-5  # mass of volume in kg ->
    1.161e-5:  cell nucleus volume -> 1.131e-13
    deposit2Joule = eV*mean_energy_dep
    numberProt = Gy*volume/deposit2Joule
    print(numberProt)
    return int(numberProt)
```

```
10
11
12 if __name__=='__main__':  # test function
13     print(dose(np.array([1,2,3,5,8,10]),75369))
```

Listing A.4: Code to convert dose to number of protons

A.2.5 CellDist.py

```
1  import numpy as np
2  import random
3  import matplotlib.pyplot as plt
4  def celldist(NumberCell):
5      x_cell=np.zeros(NumberCell)
6      y_cell=np.zeros(NumberCell)
7      i=0
8      while i<NumberCell:
9          x_temp=random.uniform(-3,3)
10         y_temp=random.uniform(-3,3)
11         d =  np.sqrt(x_temp*x_temp+y_temp*y_temp)
12         if d> 3:
13             None
14         elif d<3:
15             x_cell[i]= x_temp
16             y_cell[i]=y_temp
17             i+=1
18         if i==NumberCell:
19             x_nucleus=x_cell
20             y_nucleus=y_cell
21             break
22
23     return x_cell,y_cell,x_nucleus,y_nucleus
24
25 if __name__=='__main__':
26     cellMatrix=celldist(100000)
27     #print(cellMatrix)
28     plt.plot(cellMatrix[0],cellMatrix[1],'.')
29
30     radiel=np.zeros(len(cellMatrix[1]))
31
32     radiel=np.sqrt(cellMatrix[1]**2+cellMatrix[0]**2)
33     plt.figure()
34     plt.hist(radiel,density=True)
35     plt.show()
```

Listing A.5: Program distributing points in a randome pattern on a disc with d

A.2.6 MonteCarlo.py

4.4.4.4.

```
1 import numpy as np
2 def MC(numberProt):
3     n='n'
4     y='y'
5     def invert_int_func(a,y,b):  # inverted cpd
6         return np.sqrt(2*a*y+b**2)/a
7
8     slope = 0.2135106133113824  # slope for the model
9     intercept = 0.00  # intercept for the model
```

```
10
11      Y = np.random.rand(numberProt).astype('float64')  # Check
        cell dist to se explenation
12      r = invert_int_func(slope,Y,intercept)
13      deg = np.random.rand(numberProt).astype('float64')*2*np.pi
14
15      def pol2cart(rho, phi):
16          x = rho * np.cos(phi)
17          y = rho * np.sin(phi)
18          return(x, y)
19
20      x_dist,y_dist=pol2cart(r,deg)
21      return x_dist,y_dist
```

Listing A.6: Code to distribute protons on a disc with d

A.2.7 Interaction.py

```
1  import numpy as np
2  from numba import njit
3  @njit
4  def interaction(r,r_nucleus,x,y,x_cell,y_cell,x_matrix,y_matrix
       ,hitsCell,x_nucleus_matrix,y_nucleus_matrix,hitsNucleus,
       x_nucleus,y_nucleus):
5      r=r*1e-4
6      r_nucleus=r_nucleus*1e-4
7      # from cm to um
8      #max_hit = 10000 # integer to make hit matrix
9      cell_count = len(x_cell)  # rows in matrix corresponds co
       cells
10     for i in range(len(x_cell)):  # for loop for cell
11         dist = np.sqrt((x-x_cell[i])**2+(y-y_cell[i])**2)  #
       distance between cell center and protons
12
13         inCell=np.where(dist<r)  # check if proton is whithin a
        cell
14
15         numHits = len(inCell[0])  # number of hits per cell
16         first_zero_value_index=np.where(x_matrix[i,:]==0)
17         first_zero_value=first_zero_value_index[0]
18
19         x_matrix[i,first_zero_value[0]:numHits+first_zero_value
       [0]]=x[inCell]
20         y_matrix[i,first_zero_value[0]:numHits+first_zero_value
       [0]]=y[inCell]
21
22         hitsCell[i]+=numHits  # number of hits per cell
       corresponding to row in matrix
23
24     #for i in range(len(x_cell)):  # for loop for nucleus
25         #dist_nucleus = np.sqrt((x-x_nucleus[i])**2+(y-
       y_nucleus[i])**2)  # distance between nucleus center and
       protons
26
27         inCell_nucleus=np.where(dist<r_nucleus)  # check if
       proton is whithin a nucleus
28         numHits_nucleus = len(inCell_nucleus[0])  # number of
       hits per nucleus
```

```
29
30          first_zero_value_index_nucleus=np.where(
    x_nucleus_matrix[i,:]==0)
31          first_zero_value_nucleus=first_zero_value_index_nucleus
    [0]
32
33          x_nucleus_matrix[i,first_zero_value_nucleus[0]:
    numHits_nucleus+first_zero_value_nucleus[0]]=x[
    inCell_nucleus]
34          y_nucleus_matrix[i,first_zero_value_nucleus[0]:
    numHits_nucleus+first_zero_value_nucleus[0]]=y[
    inCell_nucleus]
35
36          hitsNucleus[i]+=numHits_nucleus
37
38      return x_matrix,y_matrix,hitsCell,x_nucleus_matrix,
    y_nucleus_matrix,hitsNucleus
```

Listing A.7: Code to calculate euclidean distance.

A.2.8 ProtonChoice.py

```
1  import numpy as np
2  from numpy import random
3
4  def chooseProton(hitCount,hitsNucleus,pathFiles,max_hit):
5      cell_count=len(hitCount)  # the hitcount is a matrix with
       rows for cells
6      file = open(pathFiles,'r')  # import the text file for the
       protons of interest
7      Tracks=file.readlines()
8
9      for i in range(len(Tracks)):
10             # make the strings into floats for easy handeling
11             Tracks[i]=float(Tracks[i].replace('.root\n', ''))
12
13     #max_hit=35000#max(hitCount)  # assumed maximum hits per
       cell
14     hitMartix=np.zeros((cell_count,max_hit))  # a row is a cell
        and #collom is for each cell
15     hitMatrixNucleus=np.zeros((cell_count,max_hit))
16     for i in range(cell_count):
17         hitMartix[i,0:int(hitCount[i])]=random.choice(Tracks,
       size=int(hitCount[i]))
18         # the hit matrix is the same dim as the cell matrix.
       gives each hit its own proton
19         hitMatrixNucleus[i,0:int(hitsNucleus[i])]=hitMartix[i
       ,0:int(hitsNucleus[i])]#random.choice(Tracks,size=int(
       hitsNucleus[i]))
20
21     return hitMartix,hitMatrixNucleus
22
23
24 if __name__=='__main__':
25     hitCount= [6,2,3,4,5,6,7,8]
26     hitMartix=chooseProton(hitCount)
27     for i,x in enumerate(hitMartix):
28         print(x,i)
```

Listing A.8: Code sampling random protons to the accepted proton points.

A.2.9 main.py

```
#Python libraries
import numpy as np
import matplotlib.pyplot as plt
from numba import jit, cuda
from timeit import default_timer as timer

from src.Files import FilesOfInterest
from src.Dose import dose
from src.Montecarlo import MC
from src.CellDist import celldist
from src.Interaction import interaction
from src.protonChoice import chooseProton
from src.rootImplementation import TingTarTid
from src.Plot import PlotNumIonPerCell,PlotLET, PlotDoseCell,
    Dosedist,PlotNumIonPerNuc,PlotDoseNuc

y='y'
n='n'

print('\n\n
    ------------------------------------------------------------------------
    ')
Enter_energy = float(input('Enter proton energy [MeV]
    (8.0=1,1.8=2,1.4=3,1.2=4):'))
print('
    ------------------------------------------------------------------------
    ')
"""
First coupple of lines is a if test to declare the energy
which shall be used and the location of the protons of interest
, furthermore the mean ernergy deposit per proton per 4um is
    declared
"""
if Enter_energy==1:
    pathFiles='Data8/Res/FilesOfInterest.txt'
    pathData='Data8'
    name = '8MeV'
    mean_energy_dep=15078.96   # LET keV/4micro m
elif Enter_energy==2:
    pathFiles='Data1_8/Res/FilesOfInterest.txt'
    pathData='Data1_8'
    name='1_8MeV'
    mean_energy_dep=57960.49666363634  # LET keV/4micro m
elif Enter_energy==3: #1.46
    pathFiles='Data1_4/Res/FilesOfInterest.txt'
    pathData='Data1_4'
    name='1_4MeV'
    mean_energy_dep=69045.56688785045  # LET keV/4micro m
elif Enter_energy==4:
    pathFiles='Data1_2/Res/FilesOfInterest.txt'
    pathData='Data1_2'
    name='1_2MeV'
    mean_energy_dep=75369.00045882349   # LET keV/4micro m

print('\n\n
    ------------------------------------------------------------------------
    ')
```

```
49 d = float(input('Enter dose in Gy:'))  # user inputs the dose
       in Gy
50 print('
       ------------------------------------------------------------------------
       ')
51 numberProt = dose(d,mean_energy_dep)  # Calculate fluence
52 #numberProt = 2*1e6 # used for testing
53
54 print('\n\n
       ------------------------------------------------------------------------
       ')
55 NumberCell = int(input('Enter cell count:'))  # input cell
       ammount
56 print('
       ------------------------------------------------------------------------
       ')
57 x_cell,y_cell,x_nucleus,y_nucleus=celldist(NumberCell)  # cell
       distrebution func
58
59
60 print('\n\n
       ------------------------------------------------------------------------
       ')
61 print('The cell is aproximated with a sylinder with a height
       and a radius')
62 print('
       ------------------------------------------------------------------------
       ')
63 r = float(input('\nEnter cell radius [um]:'))
64 h = float(input('Enter cell height [um]:'))
65
66 if h>6:
67     print('maximum cell heigt is 5.9um, try agen')
68     exit()
69
70 print('\n\n
       ------------------------------------------------------------------------
       ')
71 print('You will now get a few choises on what you want.\nThis
       will take time so grab a coffe or take a nap :')
72 print('\n
       ------------------------------------------------------------------------
       ')
73 LET = input('\nDo you want to calculate LET of the protons? (y/
       n)')
74 NumIon = input('\nDo you want to calculate the number of
       ionisations per cell? (y/n)')
75 DoseCell = input('\nDo you want to calculate the dose per cell?
        (y/n)')
76
77
78
79 r_nucleus=6  # implemantet at a later stage, therfor hardcoded
       in the program
80
81 max_hit = 35000 # integer to make hit matrix
82 x_matrix=np.zeros((NumberCell,max_hit))  # x position for each
83 y_matrix=np.zeros((NumberCell,max_hit))
84 x_nucleus_matrix = np.zeros((NumberCell,max_hit))
```

```
y_nucleus_matrix = np.zeros((NumberCell,max_hit))
max_prot_per = 10**5
iterations=int(float(numberProt)/max_prot_per)
print(iterations)
hitsCell= np.zeros(len(x_cell))
hitsNucleus= np.zeros(len(x_cell))


for i in range(iterations):  # loop to iradiate cells
    start = timer()
    x,y = MC(max_prot_per)  # montecarlo for protons
    """
    Interaction takes a cell/nucleus hit matrix and returns the same matrix with
    the hits added to the matrix
    """
    x_matrix,y_matrix,hitsCell,x_nucleus_matrix,y_nucleus_matrix,hitsNucleus=interaction(r,r_nucleus,x,y,x_cell,y_cell,x_matrix,y_matrix,hitsCell,x_nucleus_matrix,y_nucleus_matrix,hitsNucleus,x_nucleus,y_nucleus)
    x=None;y=None  # free up memory

    print('progress Cell hit reg: {}%'.format(int(float(100)/iterations*i))+"\ntime taken:", timer()-start)

np.savetxt('filesxy/'+name+str(NumberCell)+'cellss'+str(int(d))+'GyCellx.txt',(x_matrix))
np.savetxt('filesxy/'+name+str(NumberCell)+'cellss'+str(int(d))+'GyNucx.txt',(x_nucleus_matrix))
np.savetxt('filesxy/'+name+str(NumberCell)+'cellss'+str(int(d))+'GyCelly.txt',(y_matrix))
np.savetxt('filesxy/'+name+str(NumberCell)+'cellss'+str(int(d))+'GyNucy.txt',(y_nucleus_matrix))

# chooseProton assign protons from the proton-pool to each cell hit
proton_randoom_draw_matrix,proton_randoom_draw_matrix_nucleus = chooseProton(hitsCell,hitsNucleus,pathFiles,max_hit)


# Ting tar tid opens the proton data and reads it into the cell/nuc
totIonPerCell,totExiPerCell,LET_array,totDoseCellM=TingTarTid(pathData,x_matrix,y_matrix,hitsCell,h,proton_randoom_draw_matrix,LET,NumIon,DoseCell)

totIonPerNucleus,totExiPerNucleus,LET_array_nuc,totDoseNucM=TingTarTid(pathData,x_nucleus_matrix,y_nucleus_matrix,hitsNucleus,h,proton_randoom_draw_matrix_nucleus,LET,NumIon,DoseCell)

if NumIon=='y':
    PlotNumIonPerCell(totIonPerCell,totExiPerCell,name,NumberCell,d)
    PlotNumIonPerNuc(totIonPerNucleus,totExiPerNucleus,name,NumberCell,d)
```

```
    np.savetxt('DataOut/Ion{}Gy{}cells'.format(d,NumberCell)+
    name+'.txt',totIonPerCell)
    np.savetxt('DataOut/Exi{}Gy{}cells'.format(d,NumberCell)+
    name+'.txt',totExiPerCell)
    np.savetxt('DataOut/Ion{}Gy{}cellsNuc'.format(d,NumberCell)
    +name+'.txt',totIonPerNucleus)
    np.savetxt('DataOut/Exi{}Gy{}cellsNuc'.format(d,NumberCell)
    +name+'.txt',totExiPerNucleus)

if LET=='y':
    print(LET_array)
    PlotLET(LET_array,name,h,NumberCell)

if DoseCell=='y':
    vol=3.1415*(r*1e-5)**2*h*1e-5
    eV=1.60218e-19
    totDoseCellM=totDoseCellM*eV/vol
    volN=3.1415*(r_nucleus*1e-5)**2*h*1e-5
    totDoseNucM=totDoseNucM*eV/volN
    print('Mean dose per cell: {}Gy'.format(np.mean(
    totDoseCellM)))
    print('Mean dose per nuc: {}Gy'.format(np.mean(totDoseNucM)
    ))
    np.savetxt('DataOut/Dose{}Gy{}cells'.format(d,NumberCell)+
    name+'.txt',totDoseCellM)
    np.savetxt('DataOut/Dose{}Gy{}cellsNuc'.format(d,NumberCell
    )+name+'.txt',totDoseNucM)
    PlotDoseCell(totDoseCellM,name,NumberCell,d)
    PlotDoseNuc(totDoseNucM,name,NumberCell,d)
```

Listing A.9: The main program.

A.2.10 rootImplementation.py

```
import numpy as np; import uproot

from src.numberofion import NumberOfIon
from src.Plot import PlotNumIonPerCell
from src.LET import LETfunc
from src.DosePerCell import DosePerCell
def TingTarTid(pathData,x_rad,y_rad,hitMatrix,h_cell,ProtDrawM,
    LET,NumIon,DoseCell):
    # start of with declaring some varriables
    root='.root'
    path=pathData
    unique=np.unique(ProtDrawM)  # check for unique elements in
     the proton matrix
    unique=unique[unique!=0].astype(int)  # no proton is named
    0 and make the array int to work
    h_cell=h_cell*1e3
    """
    to get info from root files
    ['flagParticle', 'flagProcess',
     'x', 'y', 'z', 'totalEnergyDeposit',
     'stepLength', 'kineticEnergyDifference',
     'kineticEnergy', 'cosTheta', 'eventID',
     'trackID', 'parentID', 'stepID']
    """
    # define the different arrays for data accumulation
```

```
    PerCellArrayIon=np.zeros_like(ProtDrawM)
    PerCellArrayExi=np.zeros_like(ProtDrawM)
    LET_array=np.zeros(5)
    DoseCellM=np.zeros_like(ProtDrawM)
    for i,x in enumerate(unique):  # opens each unique proton so every proton does not have to be opend twice
        specific_cell_index=np.where(ProtDrawM==x)  # where is the proton in question
        file=uproot.open(path+'/'+str(x)+root)['microdosimetry'] # open the specific file.root for that specific proton
        z=file['z'].array()  # distance traveld
        #print(z[0],h_cell)
        #h_cell=h_cell*10**3+z[0]  # height of cell pluss incident proton position
        h_index=np.where(z<h_cell)  # exclude the interactions larger than the cell, h-index is the index for these interactions
        if LET=='y':
            energy=file['totalEnergyDeposit'].array()
            process=file['flagProcess'].array()
            tempLET=LETfunc(energy,z,h_cell,process)
            LET_array=LET_array+tempLET
        if NumIon=='y':  # extracts number of events
            process=file['flagProcess'].array()
            PerCellArrayIon,PerCellArrayExi=NumberOfIon(z,h_cell,process,PerCellArrayIon,

    PerCellArrayExi,specific_cell_index)
        if DoseCell=='y':  # extract dose deposited to cell/nuc
            energy=file['totalEnergyDeposit'].array()
            process=file['flagProcess'].array()
            DoseCellM=DosePerCell(z,h_cell,process,energy,DoseCellM,
                                  specific_cell_index)



        print('progress: {}%'.format(int(float(100)/len(unique)*i)))


    totIonPerCell=np.sum(PerCellArrayIon,axis=1)
    totExiPerCell=np.sum(PerCellArrayExi,axis=1)
    totDoseCellM = np.sum(DoseCellM,axis=1)
    LET_array=LET_array/len(unique)
    return totIonPerCell,totExiPerCell,LET_array,totDoseCellM
```

Listing A.10: Code for reading and analysing proton tracks

A.2.11 DosePerCell.py

```
import numpy as np

def DosePerCell(z,dz,process,energy,DoseCellM,
    specific_cell_index):
    DoseCellM[specific_cell_index]=sum(energy[np.where((z>z[0])&(z<z[0]+dz)&((process==12)|(process==13)|(process==22)|(process==23)))])
```

```
    return DoseCellM
```

Listing A.11: Calculate total sum of dose deposited within a cell and a nucleus.

A.2.12 numberofion.py

```
import numpy as np

def NumberOfIon(z,dz,process,PerCellArrayIon,PerCellArrayExi,
    specific_cell_index):
    PerCellArrayIon[specific_cell_index]=len(process[np.where((
    z>z[0])&(z<z[0]+dz)&((process==13)|(process==23)))])
    PerCellArrayExi[specific_cell_index]=len(process[np.where((
    z>z[0])&(z<z[0]+dz)&((process==12)|(process==22)))])

    return PerCellArrayIon,PerCellArrayExi
```

Listing A.12: Number of ionizations per cell and nucleus.

A.2.13 Plot.py

```
import matplotlib.pyplot as plt
import numpy as np



def Dosedist(x,y,d):
    plt.figure()
    plt.hist2d(x,y,bins=800)
    plt.xlabel('donno')
    plt.ylabel('donno')
    plt.title('2D Histogram of distrebution of {} Gy '.format(d
    ))
    plt.savefig('Plots/2dhistprotondist.PNG')

def PlotNumIonPerCell(totIonPerCell,totExiPerCell,name,
    NumberCell,d):
    plt.figure()
    plt.hist(totIonPerCell,alpha=0.5,bins=40,color='blue',label
    ='Ion')
    plt.hist(totExiPerCell,alpha=0.5,bins=15,color='red',label=
    'Exi')
    plt.legend()
    plt.title('total ion and exi per cell for {} cells at {}Gy'
    .format(len(totIonPerCell),d))
    plt.ylabel('cell count')
    plt.xlabel('Events')
    plt.grid()
    plt.savefig('Plots/totExiPerCell'+name+'{}cells{}Gy.PNG'.
    format(NumberCell,d))
    print('\nPlot for ionisations per cell has been saved to
    Plots as totIonPerCell.PNG')

def PlotNumIonPerNuc(totIonPerCell,totExiPerCell,name,
    NumberCell,d):
    plt.figure()
    plt.hist(totIonPerCell,alpha=0.5,bins=40,color='blue',label
    ='Ion')
    plt.hist(totExiPerCell,alpha=0.5,bins=15,color='red',label=
    'Exi')
```

```
    plt.legend()
    plt.title('total ion and exi per nucleus for {} cells {}Gy'
    .format(len(totIonPerCell),d))
    plt.ylabel('cell count')
    plt.xlabel('Events')
    plt.grid()
    plt.savefig('Plots/totExiPerNuc'+name+'{}cells{}Gy.PNG'.
    format(NumberCell,d))
    print('\nPlot for ionisations per cell has been saved to
    Plots as totIonPerNuc.PNG')

def PlotLET(LET,name,h,NumberCell):
    plt.figure()
    plt.plot(np.linspace(0,h,len(LET)-1),LET[:-1],'.')
    plt.grid()
    plt.xlabel('um')
    plt.ylabel('keV/um')
    plt.title('LET')
    plt.savefig('Plots/LET'+name+'{}cells.PNG'.format(
    NumberCell))
    print('\nLet plot has been saved as Plots/LET'+name+'{}
    cells.PNG'.format(NumberCell))

def PlotDoseCell(totDoseCellM,name,NumberCell,d):
    plt.figure()
    plt.hist(totDoseCellM,alpha=0.7,bins=30,color='blue')
    plt.ylabel('cell count')
    plt.xlabel('Total dose per cell [Gy]')
    plt.grid()
    plt.savefig('Plots/DoseForCells'+name+'{}cells{}Gy.PNG'.
    format(NumberCell,d))
    print('\nPlot for dose per cell has been saved to Plots as
    Plots/DoseForCells'+name+'{}cells.PNG'.format(NumberCell))

def PlotDoseNuc(totDoseCellM,name,NumberCell,d):
    plt.figure()
    plt.hist(totDoseCellM,alpha=0.7,bins=30,color='blue')
    plt.ylabel('cell count')
    plt.xlabel('Total dose per nucleus [Gy]')
    plt.grid()
    plt.savefig('Plots/DoseForNuc'+name+'{}cells{}Gy.PNG'.
    format(NumberCell,d))
    print('\nPlot for dose per cell has been saved to Plots as
    Plots/DoseForCells'+name+'{}cells.PNG'.format(NumberCell))
```

Listing A.13: Plot.py is part of the code visualizing the total dose deposited to cells and nuclei.

A.2.14 moransi.py

```

#import packages
import numpy as np
import matplotlib.pyplot as plt
from numba import jit
import uproot
import random
import os
from timeit import default_timer as timer
```

```
from matplotlib.ticker import ScalarFormatter

from gearysC import gearysC
from intraTrack import intratrack
from interTrack import intertrack

# the libraries from line 12-15 needs to be installed.
# pysal espesially needs the conda enviroment to be installed
# to avoid headake drop these librarries and comment out

#import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import pysal.explore as ps
import libpysal

print('whawt do you want to calculate?')
print('Moran\'s I=0,Geary\'s C=1, intraTrack=2, interTrack=3')
Method =int(input())
def prot_pos(N):
    """
    This function is distrebuting protons within the
    cellular core, N is the number of protons based on
    simulations done in the main.py program.
    """
    i = 0
    x_prot =np.zeros(N)
    y_prot=np.zeros(N)
    while i<N:
        x_temp=random.uniform(-6,6)  # edge values is for a 6um
     diameter cell
        y_temp=random.uniform(-6,6)
        dist_temp=np.sqrt(x_temp**2+y_temp**2)
        if dist_temp>6:
            x_temp=None
            y_temp=None
        else:
            x_prot[i]=x_temp
            y_prot[i]=y_temp
            x_temp=None
            y_temp=None
            i+=1

    return x_prot,y_prot

def protonChoice(N,path):
    """
    Extract the proton tracs from the proton tracks generated
    in Geant4 for different energies, the path is the directory
    containing the proton data.
    """
    root_list=[]
    root_files=os.listdir(path)
    if 'Res'in root_files:
        root_files.remove('Res')
    if 'analasis' in root_files:
        root_files.remove('analasis')
    i=0
    while i<N:
        root_temp=np.random.choice(root_files)
```

```
        file = uproot.open(path+'/'+root_temp)['microdosimetry']
        x=file['x'].array()
        if len(x)==0:
            None
        else:
            root_list.append(root_temp)
            i+=1
    return root_list

def ion_pos(N,path,rootfiles,x_prot,y_prot):
    """
    extract info from the proton files and assign the x and y
    possition from prot pos to the proton from the root files
    """
    dz=4000

    x_prot=x_prot*1e3
    y_prot=y_prot*1e3
    max_ion=14000  # assumed maximal ion per proton
    ion_tot=np.zeros((N,3,max_ion))
    for i in range(N):
        file = uproot.open(path+'/'+rootfiles[i])['microdosimetry']
        x=file['x'].array()
        y=file['y'].array()
        z=file['z'].array()
        type = file['flagProcess'].array()
        if len(z)==0:
            print('you did a poopy')
        else:
            x=x[np.where((z>z[0])&(z<z[0]+dz)&((type==13)|(type==23)))]
            y=y[np.where((z>z[0])&(z<z[0]+dz)&((type==13)|(type==23)))]
            z=z[np.where((z>z[0])&(z<z[0]+dz)&((type==13)|(type==23)))]
            x0=x[0]
            y0=y[0]
            z0=z[0]
            x=x-x0
            y=y-y0
            z=z-z0

            x=x+x_prot[i]
            y=y+y_prot[i]
            if np.amax(x)>6000:
                x[np.where(x>6000)]=0
                y[np.where(x>6000)]=0
                z[np.where(x>6000)]=0
            elif np.min(x)< -6000:
                x[np.where(x<-6000)]=0
                y[np.where(x<-6000)]=0
                z[np.where(x<-6000)]=0
            elif np.amax(y)>6000:
                x[np.where(y>6000)]=0
                y[np.where(y>6000)]=0
                z[np.where(y>6000)]=0
            elif np.min(y)< -6000:
```

```
                x[np.where(y<-6000)]=0
                y[np.where(y<-6000)]=0
                z[np.where(y<-6000)]=0
            if np.mean(x)>6000 or np.mean(x)<-6000 or np.mean(y
    )>6000 or np.mean(y)<-6000:

                x[np.where(y<-6000)]=0
                y[np.where(y<-6000)]=0
                z[np.where(y<-6000)]=0
                x[np.where(y>6000)]=0
                y[np.where(y>6000)]=0
                z[np.where(y>6000)]=0
            ion_tot[i,0,0:len(x)]=x
            ion_tot[i,1,0:len(y)]=y
            ion_tot[i,2,0:len(z)]=z
    return ion_tot,max_ion


def SpatialDataMining(ion_tot,grid_size,voxel_size):
    """
    SpatialDataMining count the number of ionizations in a
    voxel and sums
    the total ionizations in that voxel
    """
    x_grid=np.arange(-6000,6000,step=voxel_size)  # define grid
     in x 0.002
    y_grid=np.arange(-6000,6000,step=voxel_size)   # define
    grid in y
    print(len(x_grid))
    z_grid=np.arange(0,4000,step=voxel_size)  # define grid in
    z
    hit_matrix=np.zeros((len(x_grid),len(y_grid),len(z_grid)))
    ion_tot_x = 2
    for i in range(len(x_grid)-1):
        if i==0:
            start = timer()
        for j in range(len(y_grid)-1):
            #print(j)
            for k in range(len(z_grid)-1):
                hits=np.where(((ion_tot[:,0,:]>x_grid[i])&(
    ion_tot[:,0,:]<x_grid[i+1])&(ion_tot[:,1,:]>y_grid[j])&(
    ion_tot[:,1,:]<y_grid[j+1])&(ion_tot[:,2,:]>z_grid[k])&(
    ion_tot[:,2,:]<z_grid[k+1])))
                hit_matrix[i,j,k]=len(hits[0])
        if i==0:
            print('Time in s for one layer to be calculated:',
    timer()-start)
            print('Total time will approximatly be:',(timer()-
    start)*len(x_grid),'s')
    return hit_matrix




if __name__=='__main__':
    InputName=input('Enter energy of protons:')
    InputPath=input('Enter folder:')
    #InputName='8.7MeV'
    #InputPath='Data8'
```

```
    print('Enter 6 values for protons:')
    N_array=np.array([10,20,30,51,81,102])
    N_array=np.array([int(input('N1:')),int(input('N2:')),int(input('N3:')),
                      int(input('N4:')),int(input('N5:')),int(input('N6:'))])
    #print('First value for Geary\'s C is a test to compile the code via njit.')

    if Method==1:
        parameter_limit=int(input('Input cut-value [nm]:'))
    if Method==0:
        voxel=int(input('input voxelsize [nm]:'))
    meanint=np.zeros(len(N_array))
    temp=0
    #fig, axs = plt.subplots(2,3, sharex=True, sharey=True)
    #fig = plt.figure(figsize=(10,6.3))
    #gs = fig.add_gridspec(2, 3, hspace=0, wspace=0)
    #(ax1, ax2, ax3), (ax4, ax5, ax6) = gs.subplots(sharex='col', sharey='row')
    for lol,i in enumerate(N_array):
        N=i
        path='../'+InputPath
        e = InputName
        x_prot,y_prot=prot_pos(N)
        rootfiles=protonChoice(N,path)


        ion_tot,max_ion=ion_pos(N,path,rootfiles,x_prot,y_prot)
        IonMatrix=ion_tot.reshape(3,N*max_ion)

        if Method==2:
            print(np.shape(ion_tot))
            intratrackvalue=intratrack(ion_tot)
            print('The intratrackvalue is {}'.format(intratrackvalue))
            meanint[temp]=intratrackvalue
            print(np.mean(meanint),np.std(meanint),meanint)
            temp+=1
        elif Method==1:
            print('Beginning calculating geary\'s C. This might take some time')
            C=gearysC(IonMatrix,parameter_limit)
            print('Geary\'s C for',N,e+' protons is:',C)
            print('This is with the parameter for the weight matrix set at {}nm'.format(parameter_limit))

        elif Method==0:



            hit_matrix=SpatialDataMining(ion_tot,3,voxel)
            Mean_I=np.zeros(len(hit_matrix[1,1,:]))
            for ii in range(len(hit_matrix[1,1,:])):
                w = libpysal.weights.lat2W(hit_matrix.shape[0], hit_matrix.shape[1])
                mi = ps.esda.Moran(hit_matrix[:,:,ii],w)
                #print('\nMoran\'s I for the {}\'th layer:'.format(i),mi.I)
```

```
                #print('And standard deviation for the {}\'th layer:'.format(i),mi.p_norm)
                Mean_I[ii]=mi.I

            print('Mean Moran=',np.mean(Mean_I[:-1]),'for {} protons'.format(i))
            #print('Moran\'s I for the jth layer:',mi.I)
            #np.save('export_celle_kjerne/test{}nm'.format(voxel),matlab_hit_matrix)
        elif Method==3:
            interTrackvalue=intertrack(ion_tot)
            print('The intertrackvalue is {} for {} Protons'.format(interTrackvalue,N))
        elif Method==9:
            if lol==0:
                ax1.plot(x_prot,y_prot,'.',label='1Gy')
                ax1.legend(loc='upper right')
                ax1.plot(np.cos(np.linspace(0,2*np.pi,1000))*6,np.sin(np.linspace(0,2*np.pi,1000))*6,'--')
            elif lol==1:
                ax2.plot(x_prot,y_prot,'.',label='2Gy')
                ax2.legend(loc='upper right')
                ax2.plot(np.cos(np.linspace(0,2*np.pi,1000))*6,np.sin(np.linspace(0,2*np.pi,1000))*6,'--')
            elif lol==2:
                ax3.plot(x_prot,y_prot,'.',label='3Gy')
                ax3.legend(loc='upper right')
                ax3.plot(np.cos(np.linspace(0,2*np.pi,1000))*6,np.sin(np.linspace(0,2*np.pi,1000))*6,'--')
            elif lol==3:
                ax4.plot(x_prot,y_prot,'.',label='5Gy')
                ax4.legend(loc='upper right')
                ax4.plot(np.cos(np.linspace(0,2*np.pi,1000))*6,np.sin(np.linspace(0,2*np.pi,1000))*6,'--')
            elif lol==4:
                ax5.plot(x_prot,y_prot,'.',label='8Gy')
                ax5.legend(loc='upper right')
                ax5.plot(np.cos(np.linspace(0,2*np.pi,1000))*6,np.sin(np.linspace(0,2*np.pi,1000))*6,'--')
            elif lol==5:
                ax6.plot(x_prot,y_prot,'.',label='10Gy')
                ax6.legend(loc='upper right')
                ax6.plot(np.cos(np.linspace(0,2*np.pi,1000))*6,np.sin(np.linspace(0,2*np.pi,1000))*6,'--')
        elif Method==6:
            #Doselist=['1Gy','2Gy','3Gy','Gy','8Gy','10Gy']
            if InputPath=='Data1_2':
                EnergyName='1.2MeV Protons'
                savename='1_2mevprotons'
            elif InputPath=='Data1_4':
                EnergyName='1.5MeV Protons'
                savename='1_5mevprotons'
            elif InputPath=='Data1_8':
                EnergyName='1.8MeV Protons'
                savename='1_8mevprotons'
            elif InputPath=='Data8':
                EnergyName='8.7MeV Protons'
                savename='8_7mevprotons'
            nuc_h=4
```

```
270             def plot_circle():
271                 n=2000
272                 whole_pi=np.linspace(0,2*np.pi,n)
273                 c1=np.zeros(n)
274                 c2=np.cos(whole_pi)*6*1
275                 c3=np.sin(whole_pi)*6*1
276                 return c1,c2,c3
277             qwe=plot_circle()
278             ax = plt.axes(projection='3d')
279             for i in range(np.shape(ion_tot)[0]):
280                 ax.plot3D(ion_tot[i,0,:]/1000,ion_tot[i
    ,1,:]/1000,ion_tot[i,2,:]/1000,color='tomato',
281                           marker='.',ms=450./300, mew=0,
    linestyle="", lw=0)
282                 if i==0:
283
284                     ax.plot3D(qwe[1],qwe[2],qwe[0],'--',color='
    gray')
285                     ax.plot3D(qwe[1],qwe[2],np.ones(2000)*nuc_h
    ,'--',color='gray')
286                     ax.legend(['Proton and electron events','
    Nucleus border'])
287             #for label in (ax.get_xticklabels() + ax.
    get_yticklabels() +ax.get_zticklabels()):
288           #       label.set_fontsize(13)
289             #ax.legend()
290
291             ax.view_init(30, 0)
292             from matplotlib import ticker
293             formatter = ticker.ScalarFormatter(useMathText=True
    )
294             formatter.set_scientific(True)
295             formatter.set_powerlimits((-1,1))
296             ax.yaxis.set_major_formatter(formatter)
297             ax.xaxis.set_major_formatter(formatter)
298             ax.zaxis.set_major_formatter(formatter)
299             ax.set_xlabel('Dist [$\mu m$]')
300             ax.set_ylabel('Dist [$\mu m$]')
301             ax.set_zlabel('Dist [$\mu m$]')
302
303             #plt.title('Nucleus irradiated with {}, {}'.format(
    EnergyName,Doselist[lol]))
304             plt.savefig('../Plots/CellDosePlot/Protontrack{}_
    {}.jpeg'.format(str(N),savename),bbox_inches='tight',dpi
    =300)
305             #plt.show()
306     #ax1.set(ylabel='Dist [$\mu m$]')
307     #fig.xlabel('Dist [$\mu m$]')
308     #fig.ylabel('Dist [$\mu m$]')
309     #ax4.set(xlabel='Dist [$\mu m$]', ylabel='Dist [$\mu m$]')
310     #ax5.set(xlabel='Dist [$\mu m$]')
311     #ax6.set(xlabel='Dist [$\mu m$]')
312     #plt.savefig('../Plots/incidentprotonexample.png',
    bbox_inches='tight')
```

Listing A.14: The code moransi.py simulate proton tracks on a nuclei and computes Moran's I.

A.2.15 gearysC.py

```
import numpy as np
import matplotlib.pyplot as plt
import random
def gearysC(IonMatrix,parameter):


    x_mean=np.array([np.mean(IonMatrix[0,IonMatrix[0,:]!=0]),np
    .mean(IonMatrix[1,IonMatrix[1,:]!=0]),np.mean(IonMatrix[2,
    IonMatrix[2,:]!=0])])
    W=0
    mean_sum=0
    sum=0
    N=len(IonMatrix[0,IonMatrix[0,:]!=0])
    print('N=',N)

    for i in range(N):
        if IonMatrix[2,i]==0:
            continue
        x_i=np.array([IonMatrix[0,i],IonMatrix[1,i],IonMatrix
    [2,i]])
        mean_sum+=(np.sqrt((x_i[0]-x_mean[0])**2+(x_i[1]-x_mean
    [1])**2+(x_i[2]-x_mean[2])**2))**2
        for j in range(N):
            if i>j:
                continue
            x_j=np.array([IonMatrix[0,j],IonMatrix[1,j],
    IonMatrix[2,j]])
            dist=(np.sqrt((x_i[0]-x_j[0])**2+(x_i[1]-x_j[1])
    **2+(x_i[2]-x_j[2])**2))**2
            dist2=np.sqrt(dist)
            if dist>parameter:
                #print('lol')#,dist2,x_i[0],x_j[0],x_i[1],x_j
    [1],x_i[2],x_j[2])
                w_ij=0

            elif dist<parameter:
                 sum+=dist
                 W+=1
            else:
                print('no')
    C=(N-1)*sum/(2*W*mean_sum)

    print(C,W)
    return C


if __name__=='__main__':
    parameterArray=np.array((0.3,0.7,10))
    for parameter in parameterArray:
        points =20
        #parameter=10
        points3D=points*3
        #rm=random matrix. rmc= random matrix clusterd
        rm=np.random.uniform(0,1,points3D).reshape(3,points)
        rmc=np.random.uniform(0.3,1,points3D).reshape(3,points)
        rmc[1]=rmc[1]*-1
        rmc2=np.random.uniform(0.3,0.2,points3D).reshape(3,
    points)
        def xy(n):
```

```
            x=np.ones(n)
            y=np.ones(n)
            i=0
            while i<n:
                x_temp = random.uniform(-3,3)
                y_temp = random.uniform(-3,3)
                dist=np.sqrt(x_temp**2+y_temp**2)
                if dist>3:
                    continue
                else:
                    x[i]=x_temp
                    y[i]=y_temp
                    i+=1
            return x,y
        def plot_circle():
            n=2000
            whole_pi=np.linspace(0,2*np.pi,n)
            c1=np.zeros(n)
            c2=np.cos(whole_pi)*3
            c3=np.sin(whole_pi)*3
            return c1,c2,c3
        qwe=plot_circle()
        xy=xy(points)

        rm[0]=xy[0];rm[1]=xy[1]
        c1=gearysC(rmc2,parameter)
        c2=gearysC(rmc,parameter)
        c3=gearysC(rm,parameter)
        fig = plt.figure()
        ax = plt.axes(projection='3d')
        #ax=fig.add_subplot(211,projection='3d')
        ax.plot3D(rm[0],rm[1],rm[2],'.',label='Randomly
    distrebuted V={:1.3f}'.format(c3))
        ax.plot3D(rmc[0],rmc[1],rmc[2],'.',label='Semi
    clustering V={:1.3f}'.format(c2))
        ax.plot3D(rmc2[0],rmc2[1],rmc2[2],'.',label='Max
    clustering V={:1.3f}'.format(c1))
        #ax.plot3D(qwe[0],qwe[1],qwe[2],'--',color='gray')
        ax.plot3D(qwe[1],qwe[2],qwe[0],'--',color='gray')
        ax.plot3D(qwe[1],qwe[2],np.ones(2000),'--',color='gray'
    )
        #ax.plot3D(qwe[2],qwe[0],qwe[1])
        #plt.title('Geary\'s C with a cut parameter={}'.format(
    parameter))
        for label in (ax.get_xticklabels() + ax.get_yticklabels
    () +ax.get_zticklabels()):
            label.set_fontsize(13)
        ax.legend()
        plt.savefig('Plots/qualitativeC{}.png'.format(parameter
    ),bbox_inches='tight')
        plt.show()
```

Listing A.15: Code computing Geary's C.

A.2.16 intertrack.py

```

#import packages
import numpy as np
```

```
import matplotlib.pyplot as plt
from numba import jit
import uproot
import random
import os


def intertrack(TrackMatrix,parameter):
        mean_sum=0
        sum=0
        N=len(TrackMatrix[0,TrackMatrix[0,:]!=0])
        print('N =',N,'resulting in ',N**N,' calculations')
        for i in range(N):
            if TrackMatrix[2,i]==0:
                continue
            x_i=np.array([TrackMatrix[0,i],TrackMatrix[1,i],
    TrackMatrix[2,i]])
            #mean_sum+=np.sqrt((x_i[0]-x_mean[0])**2+(x_i[1]-
    x_mean[1])**2+(x_i[2]-x_mean[2])**2)
            for j in range(N):
                if j==i:
                    continue
                x_j=np.array([TrackMatrix[0,j],TrackMatrix[1,j
    ],TrackMatrix[2,j]])
                dist_squared=(np.sqrt((x_i[0]-x_j[0])**2+(x_i
    [1]-x_j[1])**2+(x_i[2]-x_j[2])**2))**2

                if dist_squared>parameter:
                    w_ij=0
                elif dist_squared<parameter:
                     sum+=dist_squared
                else:
                    print('poop')
        intertrackvalue=sum/(N**2)
        return intertrackvalue
```

Listing A.16: Code computing intertrack autocorrelation.

A.2.17 PDF.py

```
import numpy as np; import matplotlib.pyplot as plt; import
    scipy as sci

def f(alpha,x):
    return alpha*np.exp(-alpha*x)

def F(alpha,x):
    return 1-np.exp(-alpha*x)


x = np.arange(0,3,0.01)
alpha = 1

f = f(alpha,x)
F = F(alpha,x)

plt.figure()
plt.plot(x,f,'--')
plt.plot(x,F)
```

```
plt.legend(('f(x)','F(x)'))
plt.xlabel('x')
plt.ylabel('f(x)/F(x)')
plt.grid()

plt.savefig('PDF')
```

Listing A.17: Plott for montecarlo.

A.2.18 reject.py

```
import numpy as np; import matplotlib.pyplot as plt


n=100000000
N=100
pi = np.zeros(N)

X=np.random.random(size=n)
Y=np.random.random(size=n)
square = np.sqrt(X**2+Y**2)

x_in=X
y_in=Y
x_in[1<square]=0
y_in[1<square]=0

X=np.random.random(size=n)
Y=np.random.random(size=n)
square = np.sqrt(X**2+Y**2)
x_out=X
y_out=Y
x_out[1>square]=0
y_out[1>square]=0
pi_num=(x_in!=0).sum()/n*4

print((x_in!=0).sum()/n*4)
plt.plot(x_in,y_in,'.')
plt.plot(x_out,y_out,'.')
plt.title('Calculation of pi')
plt.xlabel('x')
plt.ylabel('y')
#plt.show()
#plt.savefig('rejectofpi')
```

Listing A.18: Code for the rejection montecarlo simulation

A.2.19 Survival.py

```
import numpy as np
import matplotlib.pyplot as plt

def survival(D,alpha,beta):
    return np.exp(-alpha*D-beta*D*D)

alpha1 = 0.157
beta1 = 0.068

alpha2 = 0.049
```

```
beta2 = 0.109

Dose = np.arange(0,5,0.0001)

survival1 = survival(Dose,alpha1,beta1)
survival2 = survival(Dose,alpha2,beta2)

plt.plot(Dose,survival1,label='Early responding [T98G]')
plt.plot(Dose,survival2,label='Late responding [U87]')
plt.grid()
plt.legend()
plt.xlabel('Dose [Gy]')
plt.ylabel('Survival fraction')
plt.yscale('log')
plt.show()
```

Listing A.19: A simple program for the α β model example

A.2.20 intraTrack.py

```

#import packages
import numpy as np
import matplotlib.pyplot as plt
from numba import jit
import uproot
import random
import os
#One proton
def intratrack(TrackMatrix):
        mean_sum=0
        sum=0
        TrackMatrix=np.reshape(TrackMatrix,newshape=(3,14000))
        N=len(TrackMatrix[0,TrackMatrix[0,:]!=0])
        #print('N =',N,'resulting in ',N**N,' calculations')
        for i in range(N):
            #if TrackMatrix[2,i]==0:
            #    continue
            x_i=np.array([TrackMatrix[0,i],TrackMatrix[1,i],
    TrackMatrix[2,i]])
            #mean_sum+=np.sqrt((x_i[0]-x_mean[0])**2+(x_i[1]-
    x_mean[1])**2+(x_i[2]-x_mean[2])**2)
            for j in range(N):
                if j==i:
                    continue
                x_j=np.array([TrackMatrix[0,j],TrackMatrix[1,j
    ],TrackMatrix[2,j]])
                dist_squared=np.sqrt((x_i[0]-x_j[0])**2+(x_i
    [1]-x_j[1])**2+(x_i[2]-x_j[2])**2)
                sum+=dist_squared

        intertrackvalue=sum/(N**2)
        return intertrackvalue
```

Listing A.20: intraTrack code

Bibliography

[1] URL: https://commons.wikimedia.org/wiki/File:Mean_Excitation_Potential.png.

[2] Dr. Khalid KhanAbdul JabbarAbdul JabbarP. AkhterP. AkhterShow all 5 authorsHasan Mahmood KhanHasan Mahmood Khan. 'Assessment of Primordial Radionuclides in Pakistani Red Bricks and Associated Radiation Doses'. In: (Mar. 2010). DOI: 10.1088/0256-307X/27/3/039301.

[3] Hrishav Barua and Saurav Sarmah. 'An Extended Density based Clustering Algorithm for Large Spatial 3D Data using Polyhedron Approach'. In: *International Journal of Computer Applications* 58 (Nov. 2012), pp. 4–15. DOI: 10.5120/9252-3418.

[4] M. J. Berger et al. 'Report 49'. In: *Journal of the International Commission on Radiation Units and Measurements* os25.2 (Apr. 2016), NP–NP. ISSN: 1473-6691. DOI: 10.1093/jicru/os25.2.Report49. eprint: https://academic.oup.com/jicru/article-pdf/os25/2/NP/9587198/jicruos25-NP.pdf. URL: https://doi.org/10.1093/jicru/os25.2.Report49.

[5] M.A. Bernal et al. 'Track structure modeling in liquid water: A review of the Geant4-DNA very low energy extension of the Geant4 Monte Carlo simulation toolkit'. In: *Physica Medica* 31.8 (2015), pp. 861–874. ISSN: 1120-1797. DOI: https://doi.org/10.1016/j.ejmp.2015.10.087. URL: https://www.sciencedirect.com/science/article/pii/S1120179715010042.

[6] Thomas Braunroth et al. 'Three-dimensional nanodosimetric characterisation of proton track structure'. In: *Radiation Physics and Chemistry* 176 (2020), p. 109066. ISSN: 0969-806X. DOI: https://doi.org/10.1016/j.radphyschem.2020.109066. URL: https://www.sciencedirect.com/science/article/pii/S0969806X1931415X.

[7] Martin Raff Keith Roberts Bruce Alberts Alexander Johnsen Julian Lewis David Morgan and Peter Walter. *Molecular biology of THE CELL, Sixth Edition*. Garland Science, Tylor Francis Group, 2015, pp. xxxiv+1342. ISBN: 978-0-08153-4464-3.

[8] David S. Chang et al. 'Oxygen Effect, Relative Biological Effectiveness and Linear Energy Transfer'. In: *Basic Radiotherapy Physics and Biology*. Cham: Springer International Publishing, 2014, pp. 235–240. ISBN: 978-3-319-06841-1. DOI: 10.1007/978-3-319-06841-1_22. URL: https://doi.org/10.1007/978-3-319-06841-1_22.

[9] Ming Chew et al. 'Potential lethal damage repair in glioblastoma cells irradiated with ion beams of various types and levels of linear energy transfer'. In: *Journal of radiation research* 60 (Nov. 2018). DOI: 10.1093/jrr/rry081.

[10] Wikipedia communs. URL: https://commons.wikimedia.org/wiki/Category:Particle_accelerator_schemas#/media/File:Akcelerator_liniowy_z_fal%5C%C4%5C%85_stoj%5C%C4%5C%85c%5C%C4%5C%85.svg.

[11] Wikipedia communs. URL: https://commons.wikimedia.org/wiki/Category:Particle_accelerator_schemas#/media/File:Zyklotron_Prinzipskizze02.svg.

[12] Tordis J. Dahle et al. 'Monte Carlo simulations of a low energy proton beamline for radiobiological experiments'. In: *Acta Oncologica* 56.6 (2017). PMID: 28464743, pp. 779–786. DOI: 10.1080/0284186X.2017.1289239. eprint: https://doi.org/10.1080/0284186X.2017.1289239. URL: https://doi.org/10.1080/0284186X.2017.1289239.

[13] Michael Dingfelder. 'Updated model for dielectric response function of liquid water'. In: *Applied Radiation and Isotopes* 83 (2014). Quantum scattering codes and Monte Carlo simulations to model dynamical processes in biosystems, pp. 142–147. ISSN: 0969-8043. DOI: https://doi.org/10.1016/j.apradiso.2013.01.016. URL: https://www.sciencedirect.com/science/article/pii/S0969804313000171.

[14] D. Emfietzoglou. 'Inelastic cross-sections for electron transport in liquid water: a comparison of dielectric models'. In: *Radiation Physics and Chemistry* 66.6 (2003), pp. 373–385. ISSN: 0969-806X. DOI: https://doi.org/10.1016/S0969-806X(02)00504-2. URL: https://www.sciencedirect.com/science/article/pii/S0969806X02005042.

[15] Dimitris Emfietzoglou and Hooshang Nikjoo. 'The Effect of Model Approximations on Single-Collision Distributions of Low-Energy Electrons in Liquid Water'. In: *Radiation Research* 163.1 (2005), pp. 98–111. DOI: 10.1667/rr3281. URL: https://pubmed.ncbi.nlm.nih.gov/15606313/.

[16] Nikjoo H. Emfietzoglou D. 'The effect of model approximations on single-collision distributions of low-energy electrons in liquid water.' In: *Radiat Res.* (2005 Jan). DOI: 10.1667/rr3281.PMID:15606313..

[17] A. Görgen et al. 'The Oslo Cyclotron Laboratory'. In: *The European Physical Journal Plus volume* 136.181 (2021). DOI: https://doi.org/10.1140/epjp/s13360-021-01150-3. URL: https://link.springer.com/article/10.1140/epjp/s13360-021-01150-3#citeas.

[18] Attix F. H. *Introduction to Radiological Physics and Radiation Dosimetry*. WILEY-VCH Verlag GmbH Co. KGaA, 2004, pp. xvii+599. ISBN: 13:978-0-471-01146-0.

[19] Eric J. Hall and Amato J. Giaccia. *Radiobioligy for the Radiologist Eighth Edition*. Wolters Kluwer, 2019, pp. vii+564. ISBN: 9781496335418.

[20] Sebastien Incerti et al. 'The Geant4-DNA project'. In: (Oct. 2009). DOI: 10.1142/S1793962310000122.

[21] ProTom International. URL: https://www.protominternational.com/proton-therapy/proton-therapy-treatment/.

[22] Yukikazu Itikawa and Nigel Mason. 'Cross Sections for Electron Collisions with Water Molecules'. In: *Journal of Physical and Chemical Reference Data* 34.1 (Mar. 2005), pp. 1–22. DOI: 10.1063/1.1799251.

[23] Emil O. W. Kirkegaard. *Some methods for measuring and correcting for spatial autocorrelation*. [Online; accessed 27-January-2021]. URL: https://emilkirkegaard.dk/en/2015/10/some-methods-for-measuring-and-correcting-for-spatial-autocorrelation/.

[24] W.P Levin H. Kooy J.S. Loefffler and T.F. DeLaney. In: *British journal of cance* (Sept. 2005). DOI: 10.1038/sj.bjc.6602754.

[25] Rossi H.H. M.Zaider. *Microdosimetry and Its Applications*. Springer-Verlag, Berlin-Heidelberg, 1994, p. 317. ISBN: 13:978-3-642-85186-5.

[26] wolfram mathworld. *Disk Line Picking*. 2021. URL: https://mathworld.wolfram.com/DiskLinePicking.html.

[27] Charles E. Melton. 'Cross Sections and Interpretation of Dissociative Attachment Reactions Producing OH, O, and H in H2O'. In: *AIP* (1972). URL: https://doi-org.ezproxy.uio.no/10.1063/1.1678051.

[28] M. Michaud, A. Wen and L. Sanche. 'Cross Sections for Low-Energy (1-100 eV) Electron Elastic and Inelastic Scattering in Amorphous Ice'. In: *Radiation Research* 159.1 (2003), pp. 3–22. ISSN: 00337587, 19385404. URL: http://www.jstor.org/stable/3580746.

[29] A. Muñoz et al. 'Single electron tracks in water vapour for energies below 100eV'. In: *International Journal of Mass Spectrometry* 277.1 (2008). Electron-induced atomic and molecular processes: A special issue honoring Eugen Illenberger on his 65th birthday, pp. 175–179. ISSN: 1387-3806. DOI: https://doi.org/10.1016/j.ijms.2008.04.028. URL: https://www.sciencedirect.com/science/article/pii/S1387380608001723.

[30] H Nikjoo et al. 'Radiation track, DNA damage and response—a review'. In: *Reports on Progress in Physics* 79.11 (Sept. 2016), p. 116601. DOI: 10.1088/0034-4885/79/11/116601.. URL: https://iopscience.iop.org/article/10.1088/0034-4885/79/11/116601.

[31] NIST. URL: https://physics.nist.gov/PhysRefData/Star/Text/PSTAR.html.

[32] NIST. URL: https://physics.nist.gov/PhysRefData/Star/Text/ESTAR.html.

[33] Numba. URL: https://numba.pydata.org/.

[34] H. Paganetti. *Proton Therapy Physics*. Taylor Francis, 2011.

[35] Régis Lachaume Robert N. Cherry Jr. *Encyclopédie de Sécurité et de Santé au travail, 3è édition*.

[36] Pedro Andreo David T. Burns Alan E. Nahum Jan Seuntjens and Frank H. Attix. *Fundamentals of Ionizing Radiation Dosimetry*. WILEY-VCH Verlag GmbH Co. KGaA, 2017, pp. xlii+945. ISBN: 13:978-3-527-40921-1.

[37] Kan Wang et al. 'RMC – A Monte Carlo code for reactor core analysis'. In: *Annals of Nuclear Energy* 82 (2015). Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, SNA + MC 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms, pp. 121–129. ISSN: 0306-4549. DOI: https://doi.org/10.1016/j.anucene.2014.08.048. URL: https://www.sciencedirect.com/science/article/pii/S0306454914004484.

[38] wikimedia. URL: https://commons.wikimedia.org/wiki/File:Differential_cross_section.svg.

[39] Wikipedia contributors. *Geary's C*. Mar. 2021. URL: https://en.wikipedia.org/wiki/Geary%5C%27s_C.

[40] Wikipedia contributors. *Moran's I*. Feb. 2021. URL: https://en.wikipedia.org/wiki/Moran%5C%27s_I.

[41] Wikipedia contributors. *Waldo R. Tobler*. [Online; accessed 26-January-2021]. 2021. URL: https://en.wikipedia.org/wiki/Waldo_R._Tobler.

[42] H. O. Wyckoff et al. 'II. Definitions'. In: *Reports of the International Commission on Radiation Units and Measurements* os-17.2 (1980), pp. 7–16. DOI: 10.1093/jicru_os17.2.7. eprint: https://doi.org/10.1093/jicru_os17.2.7. URL: https://doi.org/10.1093/jicru_os17.2.7.

[43] S. Incerti A. Ivanchenko M. Karamitros A. Mantero P. Moretto H. N. Tran B. Mascialino C. Champion V. N. Ivanchenko M. A. Bernal Z. Francis C. Villagrasa G. Baldacchino P. Guèye R. Capra P. Nieminen C. Zacharatou. 'Comparison of GEANT4 very low energy cross section models with experimental data in water'. In: *Medical Physics* (17 August 2010). DOI: 10.1118/1.3476457.