Monte Carlo simulations of neutron based in-vivo range verification in proton therapy – optimization of detector dimensions and positioning

Janne Therese Syltøy

Supervisors: Kristian Smeland Ytre-Hauge and Ilker Meric

Master thesis in medical physics and technology

Department of Physics and Technology

University of Bergen

November 2019

Acknowledgements

First of all, I would like to thank my supervisors Dr. Kristian Ytre-Hauge and Dr. Ilker Meric for your help and guidance during this project, it has been invaluable.

Thank you, Kristian, for presenting me with the interesting topic of my project, and the excellent guidance and feedback I have received throughout the project. Thank you Ilker, for giving me thorough feedback to improve my reflections and decisions during the processing of my results. I would also like to thank Ph.D. candidate Lars Fredrik Fjæra for assisting me with installation of necessary software and other technical issues I have encountered during the past year.

A special thanks to my fellow classmate, Eidi, for the good companionship and teamwork during the past five and a half years of studies. You have made these years fun and exciting and helped me become a better student.

Thanks to my family, for all your support and encouragement during this project. Finally, I would like to thank Daniel for the love and positivity you have shown me during this project, it has meant a lot to me.

Bergen, November 2019
Janne Therese Syltøy

Abstract

Proton therapy is a cancer treatment modality expanding in clinical applications around the world. Due to the finite range of a monoenergetic proton beam, it is possible to spare more of the healthy tissue surrounding the tumour compared with using traditional radiation therapy with photons. However, this finite range will also make the treatment plan more susceptible to density changes, which will affect the dose delivered to the tumour. To reduce the need for large safety margins in particle therapy and enable proton therapy treatments of patient groups where motion is an issue, there is a substantial ongoing research effort in so-called range verification techniques. Range estimates can be performed in-vivo through the detection of secondary radiation species emitted in nuclear interactions between the incident protons and the nuclei in the patient, such as $\beta+$ emitters, prompt gamma-rays, charged fragments and secondary fast neutrons.

The objective of this project was to optimize detector dimensions and positioning of an existing detector concept for neutron-based range verifications using Monte Carlo simulations. The MC simulation package FLUKA was used for simulation of four monoenergetic proton beams of typical clinical energies, 100, 160, 200 and 230 MeV, entering a water phantom, and subsequent tracking of the secondary fast neutrons produced in the phantom. In addition, a patient treatment plan for prostate cancer was equivalently evaluated where the primary beam energy ranges from 93 to 197 MeV. The results were processed using python, and the python libraries Matplotlib and NumPy. In order to characterize the position and size of the detector, the neutron detection rate was evaluated for a range of detector sizes and positions.

The results showed that the neutron detection rate differed considerably with varying detector position and size, both for the water phantom and the patient treatment plan simulation. As a function of the position of the detector, the rate increased gradually until a peak was reached, followed by an almost symmetric decrease. In relation to the proton beam Bragg peak, the neutron rate peak was generally located close to or distal to the Bragg peak. When the size of the detector was considered, the neutron detection rate increased almost linearly with increasing detector area.

The results in this thesis show that the optimal placement of the detector will depend on the specific treatment plan that is to be delivered and should therefore be evaluated for the particular case. With a detector size of 20x20 cm^2, using a single position for a broad range of proton energies would lead to a clear reduction in neutron detection rate compared to using multiple positions. The feasibility of moving the detector during treatment should therefore be evaluated for treatment plans with large tumours and target volumes. Alternatively, a larger detector could be applied in order to achieve sufficient neutron detection rate for the energy levels included in the treatment plan.

Contents

List of abbreviations

BP	**B**ragg **P**eak
CPU	**C**entral **P**rocessing **U**nit
CTV	**C**linical **T**arget **V**olume
CT	**C**omputed **T**omography
DDF	**D**istal **D**ose **F**all-off
DICOM	**D**igital **I**maging and **C**ommunications in **M**edicine
FLUKA	**FLU**ktuierende **KA**skade (fluctuating cascade)
HU	**H**ounsfield **U**nit
IMPT	**I**ntensity-**M**odulated **P**article **T**herapy
LET	**L**inear **E**nergy **T**ransfer
LINAC	**LIN**ear **AC**celerator
MCS	**M**ultiple **C**oulomb **S**cattering
MC	**M**onte **C**arlo
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
NOVO	**N**eutr**O**n detection for real-time range **V**erificati**O**n
OAR	**O**rgan **A**t **R**isk
PBS	**P**encil **B**eam **S**canning
PET	**P**ositron **E**mission **T**omography
PGI	**P**rompt **G**amma **I**maging
PTV	**P**lanning **T**arget **V**olume
RBE	**R**elative **B**iological **E**ffectiveness
RT	**R**adiation **T**herapy
SOBP	**S**pread-**O**ut **B**ragg **P**eak
TPS	**T**reatment **P**lanning **S**ystem

1. Introduction

Cancer is the term for a large group of diseases characterized by abnormal cell growth with the possibility of spreading to other tissues or organs. This is one of the main causes of death worldwide and is estimated to reach 10.6 million deaths per year by 2020 [1]. But as the technology and methods used for cancer treatment develops, the survival rates increase. This results in a higher number of people living with cancer. In 2017 about 273 000 Norwegians lived with a cancer diagnosis. This represent more than 5% of our population [2].

The method used for cancer treatment varies e.g. depending on the type of cancer. The different modalities are chemotherapy, surgery, immunotherapy and radiotherapy, and these are often combined to constitute an ideal treatment plan. Radiotherapy is a treatment modality where the treatment dose delivered to the tumour is given by ionizing radiation. The overall aim of radiotherapy is to reduce the dose to the surrounding tissue while maintaining the prescribed dose in the tumour volume.

1.1 Radiotherapy – past and present

Radiotherapy has been used to treat cancer patients for more than a century. Before the use of ionizing radiation, there were few options in curing oncologic patients. This changed when Wilhelm Conrad Röntgen discovered X-rays in 1895 [3]. The following year Emil Herman Grubbe used the discovery to treat a breast cancer patient. This happened at the same time Antoine Henri Bequerel began researching radioactivity and different natural sources of radiation. Only a few years later, in 1898, Marie Sklodowska-Curie and Pierre Curie discovered radium as a natural source of radiation [3].

In the 1950s and 1960s there was a development in machinery with higher energies, in addition to an increase in popularity of using Cobalt-60 for treatment. This led to a reduced use of the conventional kilovoltage machines where the generated voltage was below 300 kVp [4]. The new high-energy megavoltage machines are e.g. the Van de Graaff generator and linear accelerator. Today, the most used particle accelerator for radiotherapy is the linear accelerator, also known as the LINAC.

Another important discovery was when Godfrey Newbold Hounsfield invented Computed Tomography (CT) scans in 1971, which in the 1980s was introduced to the clinics. This led to a shift from 2D to 3D treatment planning in radiation delivery. Performing treatment planning based on CT images resulted in better radiation dose distributions, and as new inventions such as multileaf collimators appeared, radiotherapy was rapidly revolutionized [5].

1.2 Proton therapy

In 1946, Dr. Robert Wilson stated the advantages of using proton beam therapy for cancer treatment, because of their improved depth-dose distribution as seen in Figure 1 [6]. He explained how the protons emit a low dose in the beginning of their trajectory, with an increasing dose towards the end of the particle range. This endpoint is called the Bragg peak, and this can be estimated and used to maximise the damage in the tumour and minimise the damage to surrounding tissue. In order to provide dose coverage to the entire tumour, multiple proton beam energies are used to create a so-called spread-out Bragg peak (SOBP). The advantages of using proton therapy compared to traditional radiation therapy with photons, is the possibility to reduce the dose delivered to healthy tissue surrounding the tumour and reduce the side effects of radiation.

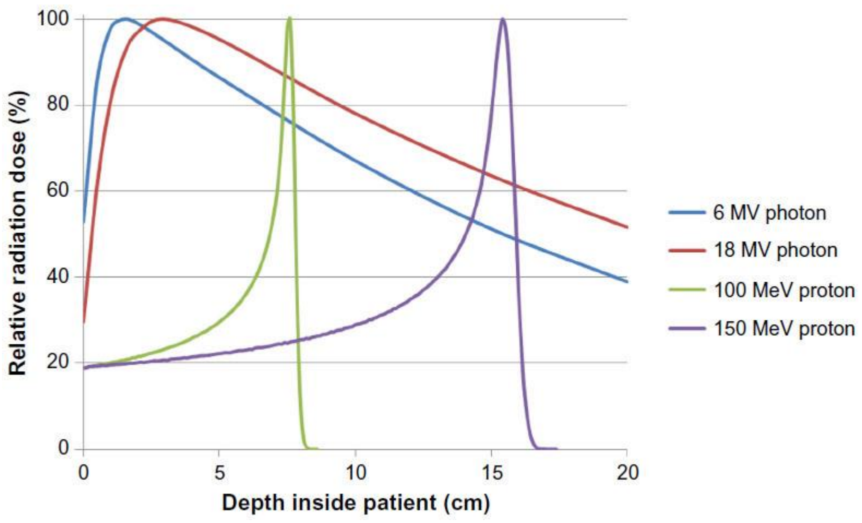

Figure 1: Simple illustration showing the depth dose curves for photons and protons of two different energies [7].

The first proton facilities were primarily used for research, and even though the first clinical use of proton beams is dated 1954, it wasn't until the late seventies computer-assisted proton accelerators were successfully used to treat various kinds of tumours [3]. Proton therapy is now used to treat several different cancer types, such as prostate, lung, paediatric and more. There are 86 particle therapy facilities worldwide currently in operation, where six of them use carbon, five use both carbon and protons, and the rest use protons [8].

1.3 Range uncertainties and verification

Since protons have a finite range in tissue, it is possible to spare more of the healthy tissue surrounding the tumour using proton therapy than using traditional radiation therapy with photons. However, this finite range will also make the treatment more susceptible to density changes, which will affect the dose delivered to the tumour. As seen in Figure 2, this could result in underdosage of the tumour and a large dose being deposited in healthy tissue either before or after the tumour. Due to sensitivity of charged particle ranges to even the smallest density changes along the radiological path in the patient, it is challenging to treat cancer in certain areas with a lot of motion and irregular density, e.g. in the lungs [9].

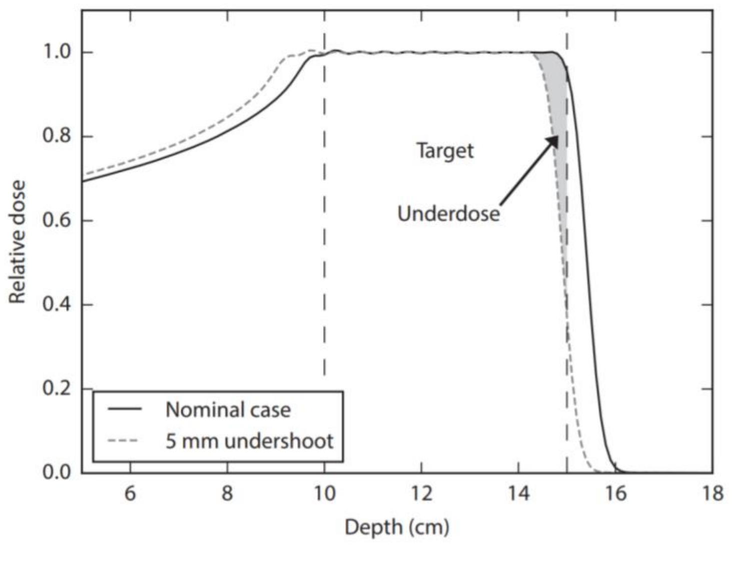

Figure 2: The spread-out Bragg peak illustrated with the planned depth, and with 5 mm undershoot as a result of patient motion or differences in anatomy [10].

Another factor that increases the uncertainty in proton beam range is the fact that the range is calculated based on attenuation of X-rays in the patient CT-scan. The stopping power ratios of the protons are then derived from the CT-scans through the use of calibration curves, but this recalculation is flawed and leads to additional errors. Other uncertainties related to CT-scans could be image noise, calibration, beam hardening and spatial resolution [10].

Traditionally, motion and variable density uncertainties are handled via margins. The clinical target volume (CTV) that contains cancerous cells, is expanded to a planning target volume (PTV) in order to ensure full dose delivery to the tumour. But in cases of heterogeneous tissue it is not always enough to use conventional PTV planning to get a robust plan, especially for particle therapy planning. Therefore, other robust optimization methods have been developed that account for the range and setup errors that could lead to dose degradation and misalignment of dose contributions from different beams during proton therapy [11].

The problem with using robust treatment plans, is that the radiation dose to healthy tissue increases, which potentially degrades the benefits of using protons compared to photons. Instead of using enlarged safety margins, one could use in-vivo range verification during treatment to provide an accurate dose delivery with smaller treatment margins and minimize dose to healthy tissue.

1.4 Project objectives

To reduce the need for large safety margins in proton therapy and enable proton therapy treatments of patient groups where e.g. motion is an issue, there is a substantial ongoing research effort in so-called range verification techniques. Range estimates can be performed in-vivo through the detection of secondary radiation species emitted in nuclear interactions between the incident protons and the nuclei in the patient, such as $\beta+$ emitters, prompt gamma-rays and charged fragments [12].

During pencil beam scanning (PBS) proton therapy, another type of secondary radiation is also produced, i.e. secondary fast neutrons. Until recently, this has been an unexplored option for in-vivo range verification. There is a correlation between the neutron production and the range of protons in the patient. This means that these secondary neutrons can, in principle, be used to verify the range of the particle beam in the patient [13].

The objective of this project was to optimize detector dimensions and positioning of an existing detector concept for neutron-based range verifications using MC (Monte Carlo) simulations. The detector has a converter made of a hydrogen-rich material, where secondary neutrons from the patient undergo elastic and inelastic interactions and produce protons. These protons are then detected in two tracking detectors, and the depth distribution of the neutron production is calculated with a reconstruction algorithm. The resulting neutron production distribution then contains information on the primary proton beam range.

The positioning of the detector relative to the patient may have a great impact on the detected signal and the precision of the final proton range estimate. Additionally, the size of the detector may impact the detection rates and achievable statistics. The objectives of this work were therefore (1) to use MC simulations to investigate the optimal placement of the detector and (2) to investigate the impact of different detector sizes on the achievable neutron detection rates.

2. Physics of proton therapy

Since it was first proposed in 1946, the technology of proton therapy has progressed significantly. The basic physics that makes it possible to use protons for radiation therapy, can be explained by how the protons interact when they traverse a medium.

2.1 Proton interactions

Charged particles, such as protons, can interact with an atom or nucleus in four ways: inelastic coulomb scattering with atomic electrons, elastic coulomb scattering with atomic nucleus, nuclear reactions and Bremsstrahlung [14]. The relevant types of interactions are displayed in Figure 3.

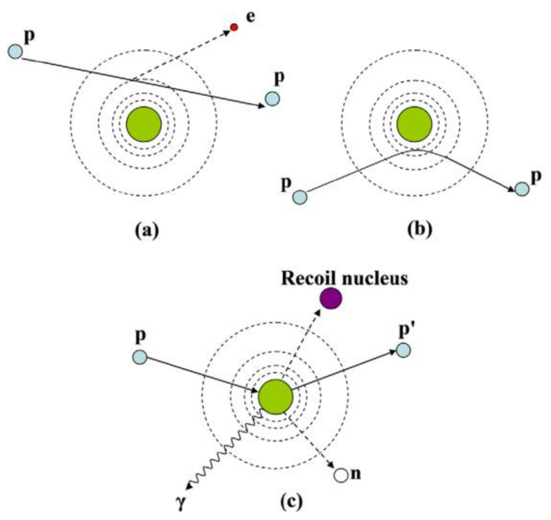

Figure 3: Illustration displaying the different proton interaction mechanisms relevant for proton therapy: (a) energy loss via inelastic Coulomb scattering with atomic electrons, (b) alteration of proton trajectory as a result of the repulsive Coulomb force (elastic scattering) with atomic nucleus, (c) formation of secondary particles by non-elastic nuclear interaction (p: proton, e: electron, n: neutron, γ: gamma-rays [14].

2.1.1 Stopping power

Protons traversing a material will interact with the atomic electrons by electrical (Coulomb) forces, and the proton continuously loses energy by ionizing and exciting the material [12]. This interaction is what determines the range of the proton in matter.

A particle with charge number Z_p and velocity β relative to the speed of light, moves in a material with atomic number Z_t and density ρ. The mean ionization loss, also known as stopping power, can be described by the Bethe-Bloch equation [12]:

$$\frac{dE}{dx} = K\rho \frac{Z_p^2}{\beta^2} \frac{Z_t}{A_t} \left[\frac{1}{2} ln \left(\frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I_e^2} \right) - \beta^2 - \frac{\delta}{2} - \frac{C}{Z_t} \right] \tag{2.1}$$

Where $K = 4\pi N_A r_e^2 m_e c^2$, N_A is Avogadro's number, r_e and m_e are the radius and mass of the electron, A_t is the material molar mass, $\gamma = \frac{1}{\sqrt{1-{}^{1}/_{\beta^2}}}$, I_e is the mean ionization energy for the material, δ is the density correction factor and C is the shell correction factor [12].

As shown in (2.1, the stopping power is inversely proportional to the particle velocity squared. This implies that as the proton speed decreases, the stopping power and energy loss increases towards the end-of-range for the proton, giving rise to the so-called Bragg peak. The stopping power is also proportional to the ion charge squared, hence a greater ion charge (e.g. carbon compared to proton) gives a larger energy loss. This equation also implies that the material of the absorber has a strong impact on the stopping power, since the stopping power is proportional to the density of the traversed material. This density can vary significantly considering a patient, with three orders of magnitude difference from air in the lungs to cortical bone [14].

Figure 4 shows a sample graph of mass collision stopping power of water against proton kinetic energy obtained from National Institute of Standards and Technology (NIST).

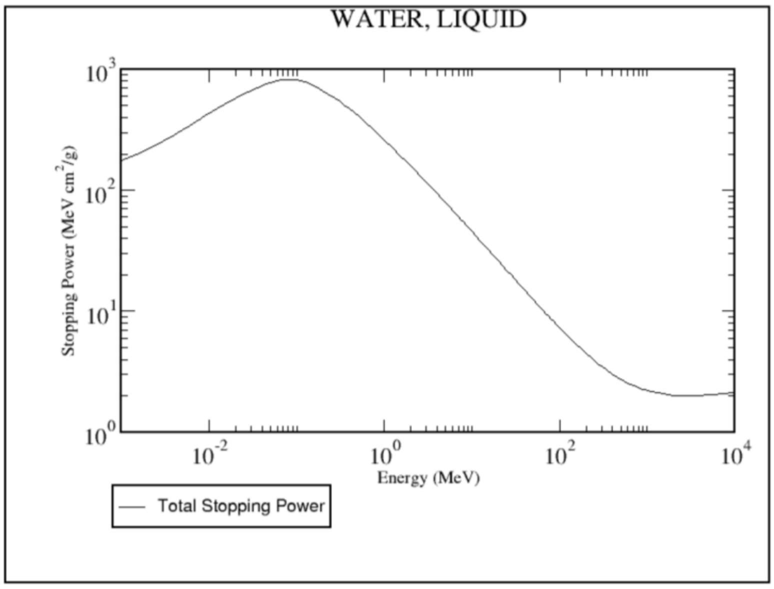

Figure 4: Stopping power of a proton in water as a function of proton energy [15].

2.1.2 Deflection and nuclear interaction

Because of the big difference in rest mass of a proton compared to an electron, the interactions with the atomic electrons will not affect the direction of the protons. However, when the protons approach a nucleus, a repulsive Coulomb force causes the proton to change direction from its straight-line path. This is called Multiple Coulomb Scattering (MCS), and this determines the lateral penumbral sharpness of the dose distribution.

Non-elastic nuclear reactions are less common but have a bigger impact once it happens regarding the outcome for an individual proton. Here, the nucleus will emit secondary particles, such as a proton, deuteron, triton, or heavier ion, or neutrons [14]. The effects of these nuclear reactions in proton therapy is a reduced fluence and dose due to the primary protons that are absorbed in the nucleus during these reactions. However, this is also somewhat compensated by the creation of secondary protons and other heavier ions that contribute to the delivered dose.

When the incident proton interacts with a nucleus in a non-elastic nuclear reaction, the nucleus may be permanently transformed as the proton is absorbed and a neutron is ejected (symbolized by (p,n)). With a 250 MeV proton beam about 20% of incident protons undergo non-elastic interactions with nuclei in the target material and create secondary particles [16].

The charged secondary particles have limited range and are absorbed locally, while non-charged particles such as neutrons and gamma-rays may travel further without being absorbed. A great number of neutrons are produced when protons undergo nuclear interactions, and since they are very penetrating, they can increase the risk of late effects. Therefore, the potential effects of secondary neutrons should be considered during planning and performing of proton therapy. This large particle range also enables detection of these secondary neutrons outside the patient.

Bremsstrahlung is also a possibility for a proton approaching a nucleus, but with the energies used in proton therapy, this is a negligible interaction.

2.1.3 Linear energy transfer

Linear energy transfer is a term that is similar to stopping power, but instead of considering how much energy that is lost by the radiation, the focus is on the absorbing media and how much energy it absorbs. Linear energy transfer is the linear rate of energy absorption in the medium as the incident particles ionize it along their path, and is defined by the following equation [17]:

$$LET = \frac{dE_L}{dl} \tag{2.2}$$

Where dE_L is the energy transferred to the material due to a charged particle travelling a distance dl. This value is also known as unrestricted linear energy transfer, LET_∞, and equals the stopping power in the medium. In order to focus solely on the energy deposited locally, which is deposited close to the particle's track, a new term called restricted linear energy transfer, LET_Δ, is introduced.

$$LET_\Delta = \frac{dE_\Delta}{dl} \tag{2.3}$$

Here dE_Δ is the energy loss caused by electronic collisions except the loss due to delta electrons with kinetic energy larger than Δ, and dl is the travel distance for the ionizing particle [18].

LET is usually expressed in units of keV/µm [17].

2.2 Interactions of secondary particles

In nuclear collisions of proton with tissue, various secondary particles are produced. Some of these with sufficient energy to penetrate larger depths can be of relevance for range verification studies. These are neutrons, photons, positron emitters and secondary ions including protons. In the following, a very brief account of neutron and photon interaction mechanisms will be given.

2.2.1 Neutron interactions

The probability for different neutron interactions depends strongly on the neutron's energy. Neutrons with high energy (larger than 1 MeV) are so-called fast neutrons, while low energy neutrons (lower than 100 MeV) are called thermal neutrons [17] . Since neutrons are neutral particles, they will not be affected by Coulomb repulsive or attractive forces. However, when a neutron is close to a target nucleus the short range attractive nuclear potential can cause a nuclear reaction [19]. Neutrons can interact with the nuclei in five different ways: elastic scattering, inelastic scattering, neutron capture, nuclear spallation and nuclear fission. The cross section, or probability, of each process depends on the kinetic energy of the neutron as well as the physical properties of the nuclei in the target material.

Elastic scattering

As fast neutrons traverse a material, they lose their energy primarily by elastic interactions with the nuclei of the material [20]. The struck nucleus recoils as some of the neutron energy is transferred to the nucleus. This energy is then dissipated by ionization, excitation and elastic collisions with other atoms in the material. In an elastic interaction, the kinetic energy and momentum are conserved, meaning the energy lost by the neutron equals the energy of the recoil nucleus.

Elastic scattering is the collision mechanism used for moderating fast neutrons to lower, thermal energies. Materials with low Z value (atomic number), such as water, are ideal moderators.

Inelastic scattering

Inelastic scattering can occur for neutrons with energies above a few MeV, but is not particularly relevant for energies below 10 MeV [20]. Inelastic scattering is a specific type of non-elastic collision, meaning that the kinetic energy is not conserved. Here the final nucleus is the same as the initial nucleus struck by the incoming neutron.

The nucleus captures the neutron and re-emits it with lower energy and a new direction. Then the nucleus will be left in an excited state and de-excites by emitting γ-ray with high energy.

Thermal neutron capture

Neutron capture is defined as a nuclear reaction where a thermal neutron is absorbed by a nucleus, and secondary radiation is emitted in the form of a proton or γ-ray [19]. This is the most relevant reaction for low energy neutrons, and product nuclei from this reaction will usually be radioactive β- and γ-emitters [21].

Spallation

Spallation is a term describing when a target nucleus breaks apart into several smaller fragments because of a collision or stress applied to the target. Hence, when neutrons (or also an ion beam) are sent towards a target nucleus it could separate into different smaller components, such as α particles and nucleons. The heavier fragments produced in such a reaction will carry most of the excess energy, and deposit this in close approximation to the location of impact. This contrasts with smaller fragments (e.g. neutrons) and de-excitation γ rays, where the energy is carried further away before its deposited [19].

Fission

Lastly, fission is when neutrons are sent towards target nuclei of high atomic number ($Z \geq 92$) which separate into two lighter daughter nuclei. In addition to these two fragments, there are also some fast neutrons produced in the process.

In proton therapy spallation and fission are not relevant, and the most important neutron processes are inelastic and elastic scattering, leading to neutron capture [17].

2.2.2 Photon interactions

Photons traversing a material have no finite range, and the intensity is attenuated through the medium. In order to characterize the ray of photons penetrating an absorbing material, the linear attenuation coefficient μ [cm^{-1}] is used. μ is defined as the probability per unit path length that the photon will interact with the absorber, and this depends on the photon energy (hν) and the absorber atomic number (Z) [22]. This can also be seen in correlation with the thickness (x) of the absorber and intensity (I) of the attenuated photon beam, according to the Beer-Lambert attenuation law [22].

$$I(x) = I_0 e^{-\mu x} \tag{2.4}$$

Where I_0 is the initial beam intensity before hitting the absorber.

The three main interactions for photons are the photoelectric effect, Compton scattering and pair production. Photoelectric effect is when the photon interacts with a tightly bound orbital electron and is completely absorbed in the process. Its energy is transferred to the electron which is then ejected from the atom. Compton scattering is an interaction between a photon and a loosely bound orbital electron. When this happens, a scattered photon with energy lower than the incident photon is produced, and the electron is ejected from the atom and receives the rest of the energy from the incoming photon. When the energy of the incident photon exceeds 1.022 MeV, which equals the combined rest mass of the electron-positron pair, so-called pair production is possible. The incident photon is absorbed and an electron-positron pair is created. Further, the created positron will promptly disappear by reconversion into photons during annihilation with an adjacent electron [22].

2.3 Dosimetry and depth dose curves

2.3.1 Absorbed dose

Absorbed dose is a measure of the physical dose delivered by ionizing radiation. [4]. It is defined as

$$D = \frac{d \Subset}{dm} \tag{2.5}$$

Where d⋐ is the mean energy transferred to a material with mass dm as a result of ionizing radiation. Absorbed dose in the international system of units (SI system) uses Gray (Gy), where 1 Gy = 1 J/kg.

Relative biological effectiveness (RBE) is a term used for describing the ratio of the doses needed by two different types of ionizing radiation to cause the same damage. It is defined as:

$$RBE = \frac{D_X}{D_R} \tag{2.6}$$

Where D_X is the reference value for an absorbed dose of type X radiation (often X-rays), and D_R is the absorbed dose of the radiation type considered (here protons) that causes the same biological effect. In clinical proton beam therapy, a generic RBE value of 1.1 is normally applied [23]. The biological dose for proton therapy is called RBE-weighted dose (D×RBE, unit [Gy(RBE)]), which is considered the most important measure for determining dose prescription in particle therapy [24].

2.3.2 Dose deposition and the spread-out Bragg peak

As shown in Figure 5, there is a significant difference in the way the dose is deposited for photons compared to protons. For photons the largest dose deposition is near the surface of the tissue they are traversing and decreases exponentially according to the Beer-Lambert exponential attenuation law ((2.4). For protons on the other hand, the dose deposited starts out lower and increases until the Bragg peak is reached, followed by a sharp decrease. Beyond the Bragg peak, there is essentially no dose deposition from incident protons. Because of this narrow Bragg peak, it is not suitable to use a monoenergetic proton beam for treatment. The Bragg peak is instead “spread out” (SOBP) to cover the whole depth of the target volume and provide a uniform dose over the entire target volume [25].

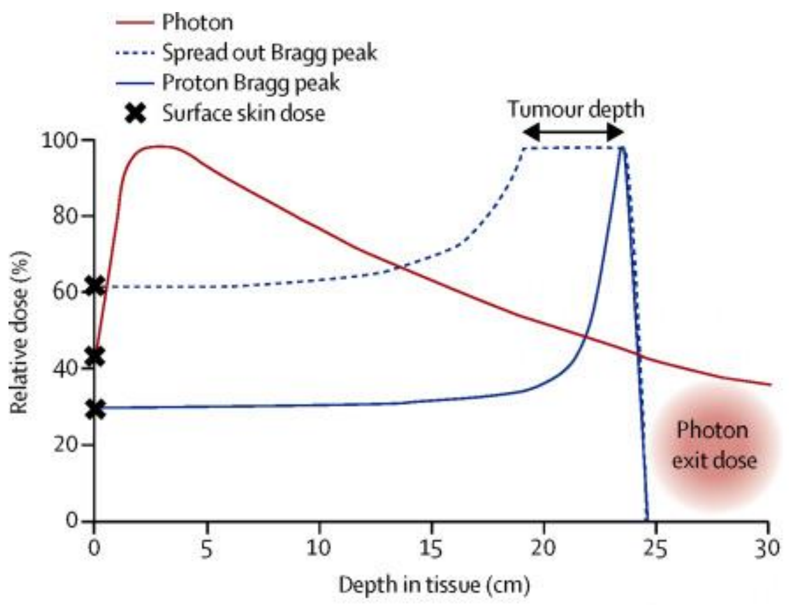

Figure 5: Illustration of depth-dose curves for proton and photon beams [26].

3. In-vivo range verification in proton therapy

Because of small variations in the energy loss for the individual particles (called range straggling), the range of a monoenergetic proton beam is often defined as the depth in a medium where half of the traversing protons have been absorbed and stopped by the medium, which is illustrated in Figure 6 [14]. This implies that the range is an average quantity, defined for the whole beam and not for individual protons.

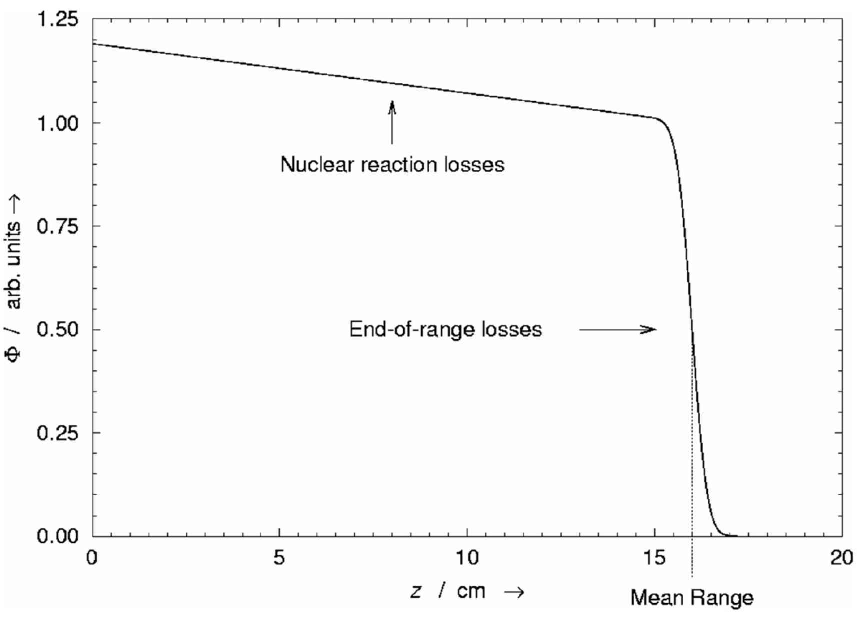

Figure 6: Range-number curve. Illustrates the relative fraction of the proton fluence Φ remaining as a function of depth z in water. Nuclear reactions in the medium causes the gradual reduction in number of protons. Near the end of range, the protons have lost all their energy and are absorbed by the medium. Because of range straggling and stochastic variations in how the protons lose energy, the distal falloff has a sigmoid shape [14].

For the range of a clinical beam (i.e. a SOBP) other definitions of range could be applied, often at a distal dose percentage of 90 (d90) as shown in Figure 7. This is located at the distal depth where the dose delivered is 90% of the prescribed dose.

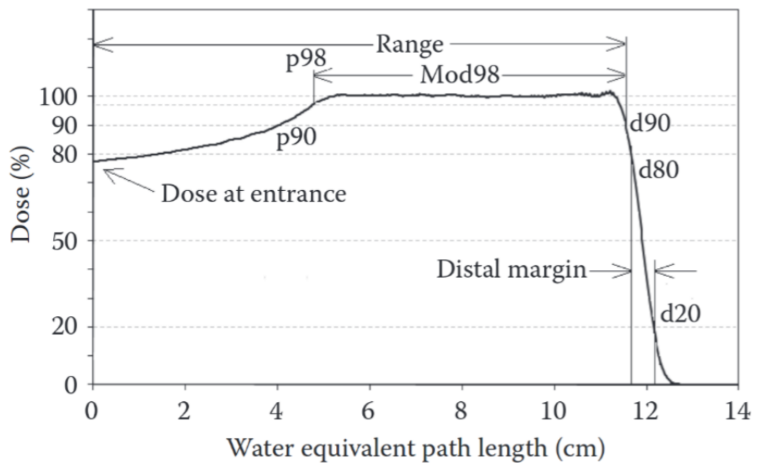

Figure 7: Illustration of the parameters used to describe a spread-out Bragg peak (SOBP). d refers to distal while p is proximal. d80–d20 is called the distal margin (or DDF - distal dose fall-off), ranging from a dose percentage of 80 to 20 of the prescribed dose. Here the range is defined as d90, but sometimes d80 is referred to as Mean projected range. Modulation width is the distance between p90/80 and d90 [27].

3.1 Range uncertainty

The Bragg peak and the general shape of the depth-dose curve of protons indicate that protons are well suited for cancer treatment, since protons could deliver a high dose to the tumour while minimizing the deposited dose to the surrounding healthy tissue [10]. On the other hand, there is an uncertainty regarding the range of the protons that is important to consider while preparing a proton treatment plan. If not considered, this will lead to a risk for underdosage of the tumour and overdosage of the distal healthy tissue, as illustrated in Figure 8.

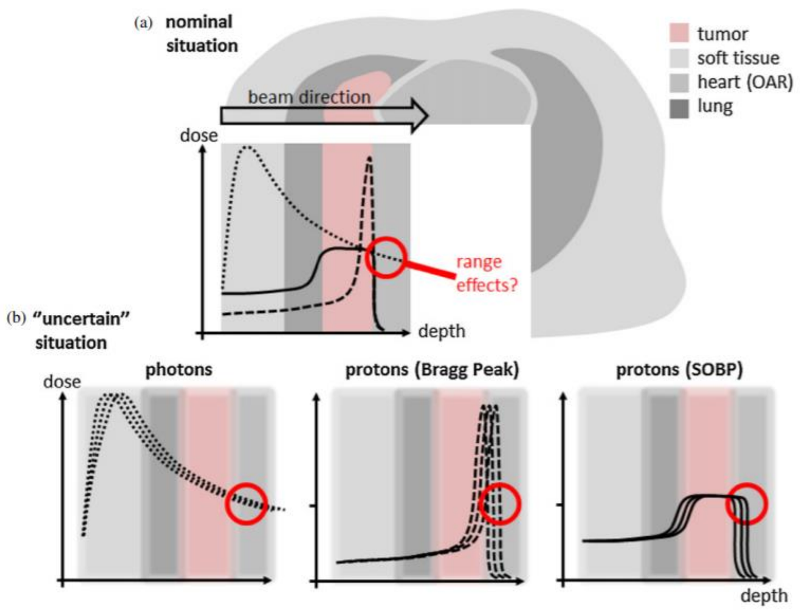

Figure 8: Photon and proton depth dose curves. (a) Reduced distal dose deposition for proton therapy compared to photon (dotted line is photon curve, dashed line is mono-energetic proton curve, and solid line is the proton spread-out Bragg peak curve (SOBP)). (b) The figures illustrate that uncertainties in the proton treatment plan have greater effects on the received dose in the target and the distal dose to the healthy tissue, compared with photons [28].

Range uncertainties have several different origins, where many of them are related to the fact that the proton range in tissue is estimated from attenuation of X-rays in the patient CT scan. Deriving the stopping power ratio of the protons from CT scans is not an exact method, so certain assumptions about the tissue composition and the ionization potential are needed [10]. Also, a certain X-ray attenuation value in a given image voxel could correlate to various stopping power values. There could also be uncertainties related to image noise, calibration and the spatial resolution of the patient CT scan, however these factors will have smaller effect on the derived stopping power ratios [10].

Another possible source of uncertainty is the algorithm used for calculating the proton range. This will have a greater impact for inhomogeneous tissue, such as the interface between different tissues in the patient [10]. There is also a possible uncertainty in the treatment plan if there is a difference between the patient anatomy at the time of planning and treatment, such as a change in weight or tumour size. Organ motion is also an important factor to consider, especially if the tumour is located near or inside a moving organ such as a lung.

Uncertainties in proton beam range are accounted for by making the treatment plan robust against these uncertainties. For treatment plans with uniform dose delivery from each field, a robust plan is in general obtained by including margins around the tumour, creating a larger target called planning target volume (PTV). In this way, it is made sure that the clinical target volume (CTV) will receive the planned dose if the proton beam range is within the margin. In addition, there are mathematical optimization techniques used to optimize the planned treatment. This mathematical optimization is based on a function *f(d)*, of the dose distribution *d*. For a good treatment plan, this function is minimal, which implies that the algorithms used try to find the beam intensities that minimize *f(d)* [29].

$$\begin{aligned} &Minimize_x \; f(d) \\ &Subject\ to\ d_i \; = \sum_j D_{ji} x_j \end{aligned} \qquad (3.1)$$

$$x_j \geq 0$$

Where d_i is the dose in voxel i, x_j is the pencil beam j fluence and D_{ij} is the dose contribution of pencil beam j to voxel i.

If the treatment plan is based on intensity modulated beams (IMPT) this traditional method is not accurate enough, since the range of the protons is different within the beam. Therefore, robust mathematical optimization methods have been developed where the uncertainties are directly incorporated in the matrix $D_{ij.}$ This is done by assuming multiple possible dose matrices, e.g. by viewing an undershoot and overshoot scenario [29].

Regardless of the exact characterization of patient position, anatomy, and tissue stopping power properties before treatment, there will still be uncertainties in the actual application of the treatment dose. Additional methods such as in-vivo assessment of the proton beam range or, ideally, dose delivery before, during, or shortly after treatment will help further reduce these uncertainties. This will also allow for more ideal arrangements of irradiation fields where organs at risk (OAR) are in close proximity to the target.

3.2 *In-vivo Range verification in proton therapy*

The different methods used for range verification can be separated into two categories; direct and indirect techniques [28]. The direct techniques imply that the range is found by measuring the dose or fluence of the proton beam itself, while indirect techniques measure the different secondary products from the protons interactions to determine the range of the initial beam.

3.2.1 Direct techniqes

Proton transmission imaging: radiography and tomography

Radiographic (i.e., from one direction) or tomographic (i.e., from multiple directions) transmission of proton beams though a patient makes it possible to create images in treatment position with the same radiation quality as for treatment [30]. The study of proton radiography started in the late 1960s. Here a high energy proton beam is sent through the patient and detected on the other side, and the residual range is measured directly. One of the main attributes of proton radiography early on was the high contrast [28]. It was also determined in the 1990s that the imaging dose was considerably reduced with protons compared to conventional imaging using X-rays [28]. In addition, if proton radiography is used the stopping power is directly measured, and the uncertainties from the derivation of stopping power ratios from X-ray attenuation is no longer an issue. So, if proton radiography is used it is possible to do in-vivo measurements of the range of the proton beams.

Proton transmission imaging is also used in a 3D tomography mode. This means that it is possible to perform range verification in 3D as well, although the main focus at the time in proton tomography is related to accurate measurements of stopping power for both range and dose calculations during treatment planning [28].

3.2.2 Indirect tecnique

Prompt gamma imaging (PGI)

Protons sent through a patient will experience nuclear interactions with the tissue, and some of these interactions will lead to excitations of nuclei. In the immediate decay to its ground state, the nuclei may emit a γ-ray (photon), also referred to as prompt γ-rays. These inelastic interactions between proton and nuclei occur along the entire path of the proton beam up to 2-3 mm proximal to the Bragg peak. Here the interaction cross section drops and the energy of products from the nuclear reactions decreases [28]. Since there is a correlation between the range of the protons and the emission of prompt gamma-rays as shown in Figure 9, these can be used to determine the range of proton beams in patients during treatment.

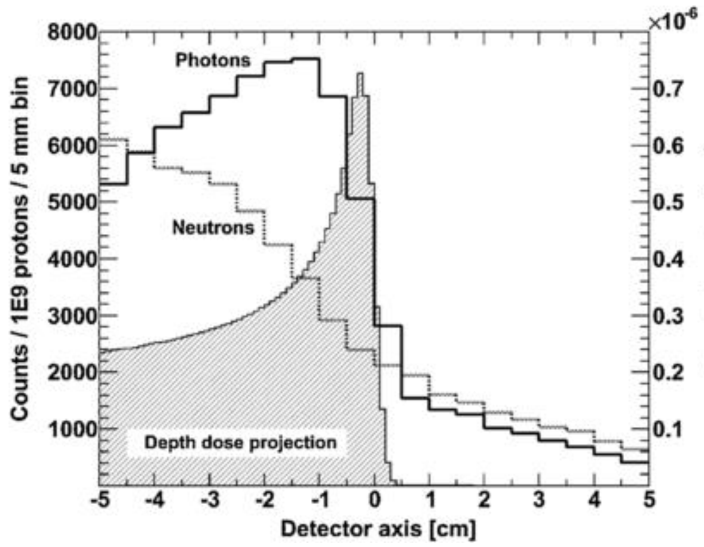

Figure 9: Illustration showing the correlation between prompt gamma emission and proton range. Photon and neutron detection profiles obtained from simulation with a perfect scintillator and collimator, i.e. with infinite density. The grey-shaded curve illustrates the depth dose curve and is given as a reference for the proton beam range depth. The coordinates along the beam and detector axes are set based on the expected beam range in the target: 15.2 cm in PMMA (para-Methoxy-N-methylamphetamine) at 160 MeV [31].

The PGI technique was first introduced in clinical proton therapy in 2015 at OncoRay in Dresden Germany, using a knife-edge slit camera, illustrated in Figure 10 [32].

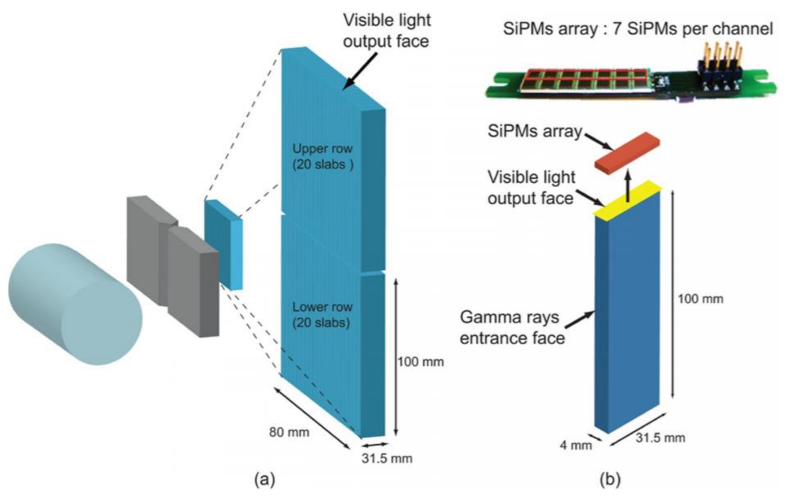

Figure 10: Design of knife-edge slit camera. a: Grey tungsten collimators and a photon detector consisting of LYSO (lutetium-yttrium oxyorthosilicate) scintillation crystals, red out by silicon photomultipliers. It consists of 40 crystals in total, organized in two rows with 20 crystals each. b: Illustration of the readout from the crystals [33].

PET imaging

In 1969, Maccabee et al was the first to suggest using PET imaging for verification of hadron beam therapy [34]. Coincident gamma-rays produced by the annihilation process resulting from positron emission from β^+ decay of radioactive isotopes are exploited. During an ion beam irradiation, some of the ions propagating through the patient will, in the nuclear interactions, create positron emitting isotopes that can later be detected by a PET camera. When the isotopes undergo β^+-decay, a positron is emitted, and this positron will annihilate with a nearby electron. The annihilation process results in two coincident gamma photons emitted back-to-back, each with energy of 511 keV.

For heavier ion such as 12C and 16O beams, the fragmentation can happen both for the projectile and target, but for a proton beam the only possible outcome is target fragmentation. The isotopes most relevant for soft tissues are 11C, 13N and 15O. Since tissue usually has a high density of oxygen, and 15O have a short half-life and consequently high decay constant, 15O usually dominates as contributor in the beginning of these PET measurements. But because of its short half-life, 11C takes over as dominant nuclide after a while. In Figure 11 the contributions from these three isotopes as a function of time are displayed. The differences in decay time for the radionuclides used in PET imaging, makes the verification process very sensitive to the time course of data acquisition, and is considered a disadvantage for range verification. Diffusion causes a lack of accuracy in the correlation between the location of the gamma-ray emission and the nuclear reaction. Depending on the tumour location, this will reduce the precision of range verification [35].

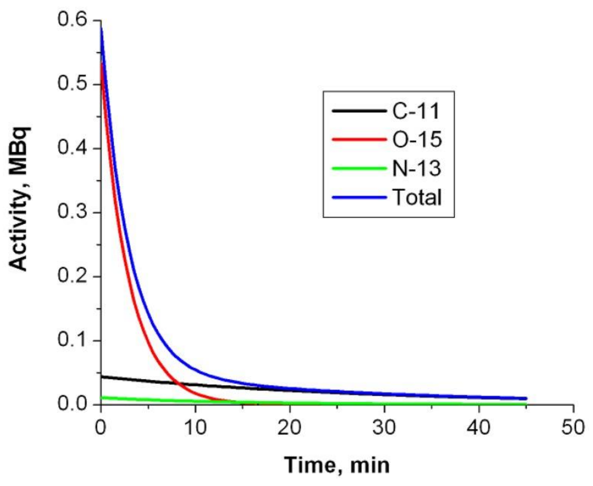

Figure 11: Radionuclide relative contributions in activity as a function of time [34].

3.3 Neutron detection for verification of proton beam range

Another possible solution for verifying the range of the proton beam in real-time, is to measure the secondary fast neutrons produced by nuclear interactions along the proton beam path. Marafini et. al. recently explored the possibility of detecting these neutrons for estimations of the additional dose due to neutrons in the patient [36]. Neutron-based range verification has later been studied in the NOVO (NeutrOn detection for real-time range VerificatiOn) project, with the use of FLUKA Monte Carlo simulations with monoenergetic proton beams impinging on regular a water phantom [13]. The current MSc thesis is also a part of the ongoing NOVO project.

Neutron production as a function of depth in the water phantom was found to be stable in the entrance region and decreasing until just proximal to the Bragg peak, as seen in Figure 12a. As expected, the neutron production rates increased with increasing primary proton beam energies. Figure 12b shows that the dominating neutron energies were in the order of 10 - 100 MeV for all three initial energies, and that the maximum energy for the neutrons increased as the proton beam energy increased. In Figure 12c it is shown that the neutrons were mainly emitted in the forward direction, i.e. along the direction of the primary proton beam, and that the angular distributions of the neutrons were symmetric across the primary proton beam.

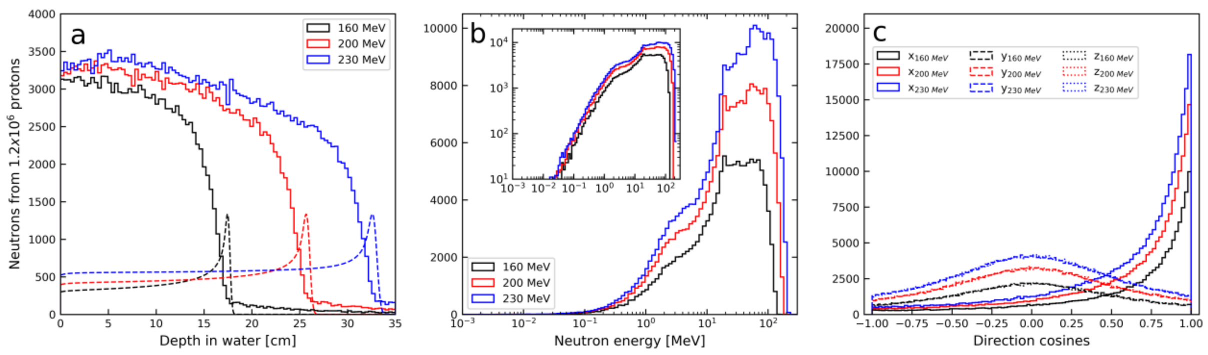

Figure 12: Illustrations of the neutron production in a water phantom for three different proton beam energies. a: Neutron production as a function of depth in the water phantom, shown with relative depth doses (dashed lines). b: Initial energy distribution for the neutrons produced. c: Angular distribution of the neutrons produced [13].

4. Materials and methods

4.1 Neutron detector concept

The conceptual detector design explored in this thesis was first introduced in the NOVO project [13]. The principle behind the detection system is seen in Figure 13. It consists of a converter made of a hydrogen-rich organic scintillator material called EJ309, with atomic ratio of 1.25 for hydrogen/carbon. Here, the neutrons produced in the patient may undergo elastic and inelastic interactions and produce secondary protons. These protons are then detected in two position sensitive detectors, and the depth distribution of the neutron production can be calculated with an ad-hoc reconstruction algorithm. This result can finally be used to estimate the primary proton beam range.

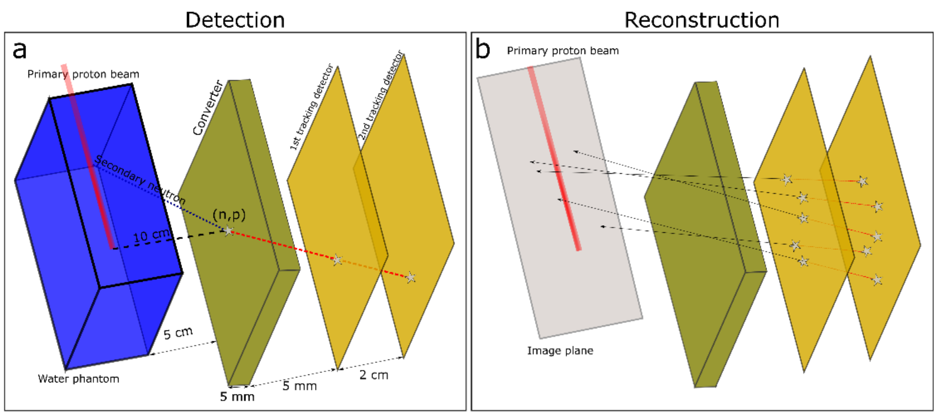

Figure 13: The detector concept. a: The detector design used in Monte Carlo simulations, not to scale. Neutrons produced in the water phantom are converted to protons in the converter. b: Secondary protons produced in the converter material traverse two position sensitive detectors and the positional information is used for reconstruction of the track and production coordinates of the initial neutron [13].

In this thesis, the dimensions and position of the converter and detectors have been evaluated first with the use of a water phantom, and later with a clinical proton treatment plan for prostate cancer. In simulation ideal detectors have been used, i.e. an abstract detection layer used for scoring of the parameters of the protons created in the converter.

4.1.1 Irradiation set-up for water phantom simulations

In the present study a 190x200 cm^2 converter and two equally large tracking detectors were defined in simulation, located parallel to a 35x10x10 cm^3 water phantom (see Figure 14). The unrealistically large size of the detectors and converter made it possible to explore many different detector positions and dimensions along this plane using a single Monte Carlo simulation.

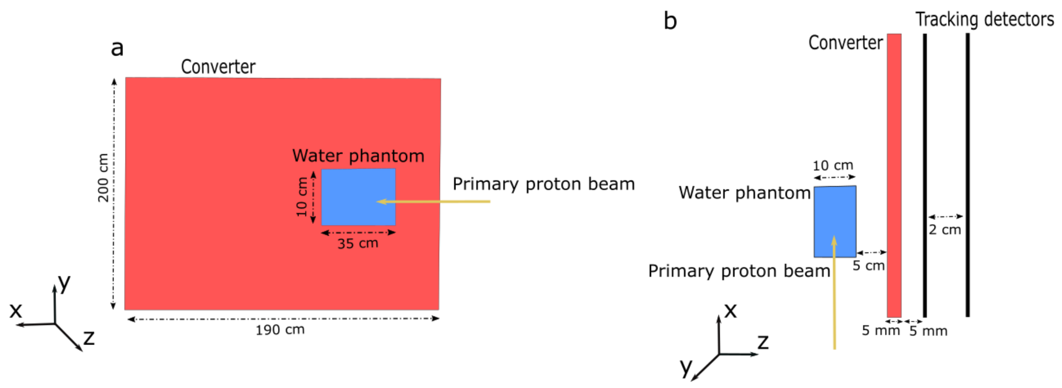

Figure 14: Illustration showing the scale of the converter and detectors relative to the water phantom used in the FLUKA simulations. a: Two-dimensional illustration from the side b: Two-dimensional illustration from above.

Monte Carlo simulations were performed at four different proton energies: 100 MeV, 160 MeV, 200 MeV and 230 MeV, covering typical ranges of therapeutic proton beams. The primary proton beams had spatial Gaussian profiles of 10 mm full width at half maximum.

4.1.2 Irradiation set-up for patient treatment plan simulations

The patient treatment plan is designed with proton beam energies ranging from 93 MeV to 197 MeV. The plan is developed at Haukeland university hospital for a prostate cancer patient, and it consists of two opposing lateral fields. Both fields contribute to the prostate PTV prescribed 67.5 Gy(RBE), while each of the fields separately irradiates lymph nodes on their respective sides (55 Gy(RBE) prescribed). One of the fields were simulated in this thesis, as seen in Figure 15. Also here, a large converter with equally large tracking detectors, based on the detector concept shown in Figure 12, was used. The detection system was placed next to the patient to detect the neutrons produced in the patient during treatment.

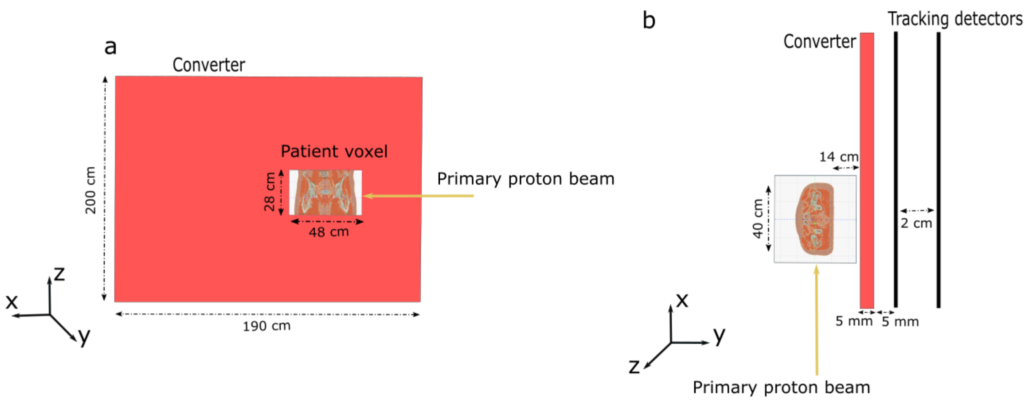

Figure 15: FLUKA geometry for patient treatment plan simulations. The beam enters from x-direction, and neutrons produced in the patient are detected in the adjacent tracking detectors.

4.2 Monte Carlo simulations and data analysis

In order to determine the ideal detector position and area in this thesis, the neutron detection rate was evaluated using Monte Carlo simulations.

Monte Carlo simulation is a technique used to model the probability of different outcomes in processes when these are difficult to predict analytically, due to the stochastic nature of these processes [37]. This makes MC simulations a good method for solving problems related to particle physics, such as in this thesis. FLUKA (FLUktuierende Kaskade or Fluctuating Cascade) [38], [39], is a Monte Carlo simulation package for particle physics. It can be used for calculations of particle transport and interactions with matter, including a great variety of applications such as accelerator shielding, calorimetry, dosimetry, detector design, radiotherapy etc. [40].

In this study FLUKA is used to simulate monoenergetic proton beams entering a water phantom, and the production, and subsequent tracking of secondary fast neutrons. FLUKA is also used to simulate a realistic patient treatment plan with protons, as well as to track the secondary fast neutrons produced in the patient. Even though the particles of interest in this thesis are neutrons and protons, other particles (e.g. electrons and alpha particles) will also be created during simulation. Various settings in FLUKA can be implemented to exclude some of these irrelevant particles, in order to reduce simulation time.

4.2.1 Water phantom simulations

24 parallel simulations, each with 5.0x10^7 primaries, were performed in order to simulate a total of 1.2x10^9 primary protons per energy. The total CPU time for the simulations were 2.7x10^4 s (7.5 hours), 3x10^4 s (8.3 hours), 1.2x10^5 s (33.3 hours) and 1.36x10^5 s (37.8 hours) for 100, 160, 200 and 230 MeV, respectively.

Prior to performing the simulations, the simulation settings in FLUKA were evaluated. The main goal was to perform simulations with good statistics and enable production and tracking of secondary particles. The FLUKA input card called DEFAULT can be used to set various default choices, reducing the number of cards needed in the input file. The default was set to PRECISIOn, which is a default suitable for simulations requiring high accuracy and tracking of secondary particles. This setting includes several effects, but most relevant in this thesis is that low energy neutron transport is turned on down to thermal energies, and that the particle transport threshold is set at 100 keV for all particle types, except neutrons (10^{-5} eV).

The physics cards in FLUKA which allow overriding the defaults for physics processes, were set to COALESCEnce and EVAPORATion. COALESCEnce is used to activate coalescence mechanism in the simulations, while EVAPORATion is used for evaporation, to use a new evaporation model with heavy fragment evaporation. Both these cards are recommended for precise particle production calculations [41].

Transport of ions was activated in the simulations with the card IONTRANS, which was set to HEAVYION. This implies that transport of all light and heavy ions is activated.

The USRBIN option, which is the standard card for volume scoring in FLUKA, was used for scoring of the primary proton beam dose delivery. This was further utilized for creation of depth dose plots used for determining the Bragg peak depth of the different primary proton beams. This was defined as the depth of maximum dose.

Two FLUKA user routines (tracking codes) were implemented to gather relevant neutron and proton information at the water phantom, converter and tracking detectors, in addition to extracting information on the trajectory of the particles. These two codes, and a table presenting the information these extract during the simulations, can be found in Appendix A.

The first code, called BXDRAW, is used to check whether a proton crossing the first tracking detector was created in the converter. Further, it is controlled whether the same proton crosses the second tracking detector. If so, the kinetic energy of the detected proton, the primary particle number, the position, energy and direction of the neutron at production point, and information describing the position and direction of the secondary proton crossing both tracker planes, is stored. The second code, STUPRF, is used to check if the detected neutron was created in the water phantom, and if so, determine which reaction happened in the converter. Possible interactions were either inelastic, elastic or low energy neutron scattering ($E < 20$ MeV). All detected neutrons were included in the analysis, also low energy neutrons.

Information is gathered from the water phantom as well, independently of the interactions that take place in the detection planes. This includes the position, direction and energy of each secondary neutron produced in the water phantom.

Simulation parameters for the water phantom simulations are gathered in Table 4-1.

Table 4-1: Simulation parameters from FLUKA, for simulations with water phantom.

PARAMETERS	**DESCRIPTION**
Proton beam energy	100 MeV, 160 MeV, 200 MeV and 230 MeV
Defaults	**PRECISIOn** (defaults for precision simulations)
Physics processes	**COALESCEnce** on (activates coalescence mechanism), **EVAPORATion** New Evap with heavy frag (new evaporation model, with heavy fragmentation)
Transport	**IONTRANS** heavyion (Transport of all light and heavy ions)
Scoring	**USERDUMP** complete, all (activates calls to the user routine BXDRAW), **USRBIN** dose (needed for depth dose plotting)

4.2.2 Patient treatment plan simulations

24 parallel simulations, each with 5.0x10^7 primaries, were performed in order to simulate a total of 1.2x10^9 primary protons. The total CPU time for the simulation of 24 independent jobs was 2.1x10^5 seconds, which equals 59.5 hours, about 2,5 days.

Before running simulations with FLUKA, the patient data needed to be transformed to a format that could be implemented in FLUKA. This process is illustrated in Figure 16 and is based on a method developed by Fjæra [42]. From the treatment planning system, the DICOM (Digital Imaging and Communications in Medicine, the standard used for storing and managing medical imaging data) files were exported and sorted using the script dicomSort.py, providing the necessary input for the FLUKA simulation, both beam characteristics and CT images. The script set_HU.py was run in order to set the Hounsfield Units outside the patient to vacuum, and further import the CT images to the FLUKA simulation.

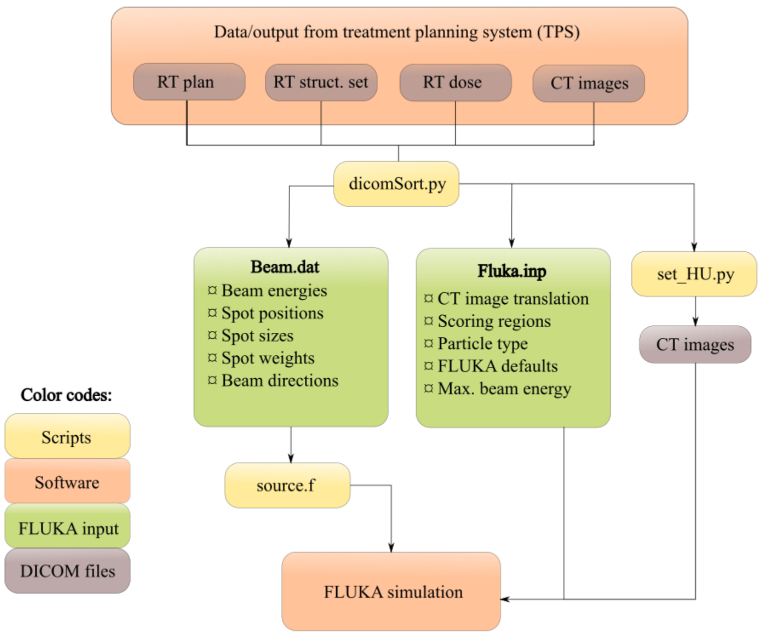

Figure 16: Illustration of the workflow for implementing patient treatment plans in FLUKA simulations. The DICOM files contain the important information regarding dose deposition and anatomy, which is transformed to a format readable in FLUKA (green boxes). The scripts (yellow boxes) are custom made by Fjæra [42].

The default in FLUKA was set to HADROTHErapy, which is a default created for hadron therapy calculations, including proton therapy. This default was set in the process described in Figure 16, and differs from the water phantom default setting in some ways. However, the most important settings decided by the default card are the same for both HADROTHErapy and PRECISIOn, i.e., that low energy neutron transport is turned on down to thermal energies, and that the particle transport threshold is set at 100 keV for all particle types, except neutrons (10^{-5} eV). It is therefore assumed that the differences in default setting will not affect the results in this thesis.

The physics cards in FLUKA were the same as for the water phantom, but the transport setting was different. In addition to the card IONTRANS which determines the transport of ions, another card implemented for transport was EMFCUT. This card can be used to set the energy thresholds for electron and photon production and transport cut-offs in the selected materials/regions. All regions were included in the cut, and it was set to 6 MeV. EMFCUT was also applied as a production cut, to set an energy limit for the electrons and photons created in the materials in simulation (6 MeV). All materials were included in the cut-off. This was utilized to reduce the CPU time for the simulations and could have been implemented for water phantom simulations as well, without affecting the results.

Three different user routines were applied in the simulations. The two tracking codes used for water phantom simulations, BXDRAW and STUPRF, were altered to record the neutron production in the patient. The last user routine is found in Appendix C, and is the source.f file as described in Figure 16. This is used for incorporation of beam information (energy, position, direction and beam spots) for the different primary proton beams used in the treatment field.

The upper and lower energy limits for the spread-out Bragg peak (SOBP) was found by running two separate simulations for the upper and lower beam energy, 93 and 197 MeV. Each simulation consisted of four parallel simulations with $1x10^5$ primary protons each. The depth dose curve in the patient was created for each energy, enabling localisation of the Bragg peaks defined at maximum dose deposition. These plots can be found in Appendix D.

Simulation parameters in FLUKA for the patient treatment plan simulations are given in Table 4-2.

Table 4-2: Simulation parameters from FLUKA, for simulations with patient treatment plan.

PARAMETERS	DESCRIPTION
Proton beam energy	26 different energies ranging from 93 MeV - 197 MeV. Number of beams per energy varies from 4 - 288.
Defaults	**HADROTHErapy** (defaults for hadron therapy calculations)
Physics processes	**COALESCEnce** on (activates coalescence mechanism), **EVAPORATion** New Evap with heavy frag (new evaporation model, with heavy fragmentation)
Transport cut	**IONTRANS** heavyion (full transport of heavy ions), **EMFCUT** all regions
Production cut	**EMFCUT** all materials
Scoring	**USERDUMP** complete, all (activates calls to the user routine BXDRAW), **USRBIN** dose/all parts (used for 2d dose plot)

4.2.3 Processing of simulation results

Python scripts were written to organize and process the information gathered from FLUKA. All scripts can be found in Appendix B. Matplotlib[1], a python 2D/3D plotting library, and NumPy[2], a python scientific computing library were used.

In order to obtain the optimal placement for the detector based on the neutron detection rate, a 20x20 cm^2 detector was defined in the data analysis and moved stepwise 1 cm along the

[1] https://matplotlib.org

[2] https://numpy.org

depth direction exploiting information from the large detection plane. Neutrons produced closer to the Bragg peak could be considered more useful in order to perform in-vivo range verification, since they hold more information about the end of range for the initial proton beam. Therefore, two additional analyses were implemented, considering the optimal detector position for the water phantom simulations. Here, only the neutrons produced in the last half and last quarter of the proton beam range were included in the neutron detection rate estimates. The optimal detector position was defined as the point where the neutron detection rate was highest. The statistical uncertainty in neutron detection rate was calculated as $\sqrt{N},$ where N is the number of detected neutrons.

For the water phantom simulation, a common location for 160 and 200 MeV was explored, considering that clinical applications of proton therapy include a range of energies and that moving the detector during treatment may be challenging. The neutron detection rate for 160 and 200 MeV were summed up and a new peak was located. The uncertainties in the peaks for 160 and 200 MeV were also compared individually to see if the uncertainty range for each energy had an overlap.

The area of the detector was considered by increasing the detector dimensions quadratically while placed in the optimal position (for each energy), obtained in this thesis. The thickness of the converter was, however, kept constant at 5 mm in all cases.

5. Results

5.1 Water phantom simulations

5.1.1 Neutron production rates and detection characteristics

The neutron production and detection rate, predicted by MC simulations at different primary proton energies, are displayed in Table 5-1. Both rates increased with increasing primary proton energy, and the difference is greater for increasing energies. The neutron detection rate was larger by a factor of 12 for 230 MeV protons compared to the lowest energy of 100 MeV protons, and by a factor of 1.4 for 230 MeV protons compared with 200 MeV protons.

Table 5-1: Total neutron production and detection rates per primary proton for the 190x200 cm^2 detector, for the four simulated energies. The uncertainties are calculated as the standard deviation of the 24 spawns used in simulation.

Proton beam energy [MeV]	**Neutron production rate [produced neutrons/primary proton]**	**Neutron detection rate [detected neutrons/primary proton]**
100	0.03 ± 0.1 %	2.4 x 10^{-5} ± 2.2 %
160	0.08 ± 0.05 %	1.1 x 10^{-4} ± 1.0%
200	0.13 ± 0.04 %	2.1 x 10^{-4} ± 1.0 %
230	0.18 ± 0.03 %	2.9 x 10^{-4} ± 0.7 %

The relative dose delivered in the water phantom is illustrated in Figure 17, as a function of the depth in water. The Bragg peak depths are located at 7.6 cm, 17.5 cm, 25.7 cm and 32.6 cm for a primary proton beam energy of 100, 160, 200 and 230 MeV, respectively. Considering the small bin sizes used in simulation (1 mm step) the uncertainty in Bragg peak position is negligible.

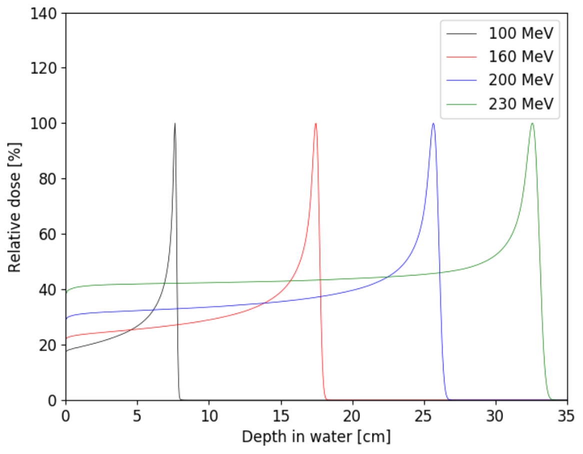

Figure 17: Relative dose delivered in the water phantom by the primary proton beam as a function of the depth in water, for each simulated energy.

The secondary neutron production distribution in the water phantom is illustrated in Figure 18a, and as seen neutron production is high in the entrance of the water phantom and gradually falls-off towards the primary proton Bragg peak. The secondary neutron production distribution for only the detected neutrons is illustrated in Figure 18b. Compared to Figure 18a showing all neutrons produced, the falloff towards the Bragg peak is gentler and starts earlier.

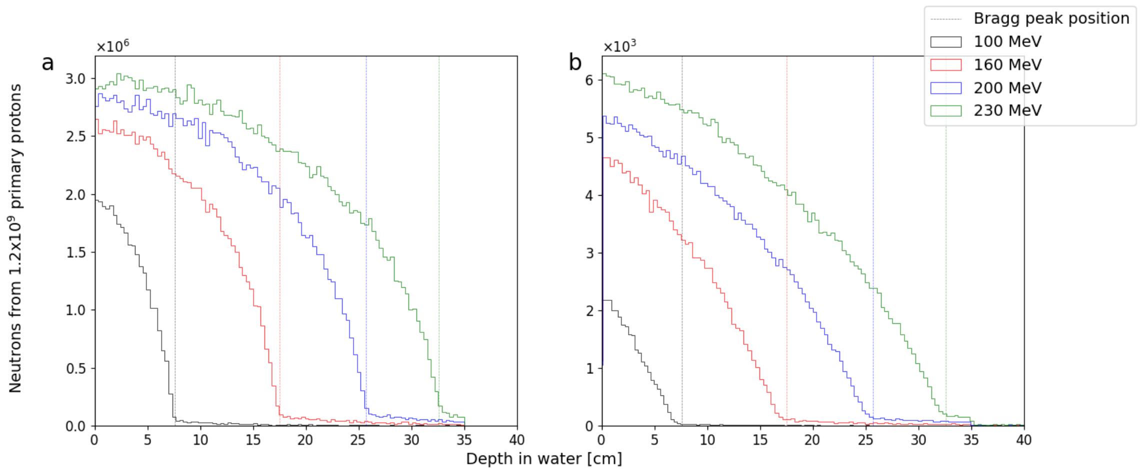

Figure 18: a: The neutron production as a function of depth in the water phantom. b: Production depth distribution for the detected neutrons from the set-up with water phantom.

Figure 19a shows the initial energy distribution for the neutrons produced in the water phantom. Neutrons with energies above 1 MeV dominated, and the magnitude of the maximum neutron energy increased with increasing primary beam energy. The initial energy distribution for detected neutrons is illustrated in Figure 19b, and as seen, exceedingly few neutrons with initial energies below 10 MeV were detected compared with the higher energy neutrons. The general shape of the distributions was similar for the different energies, mainly increasing in magnitude.

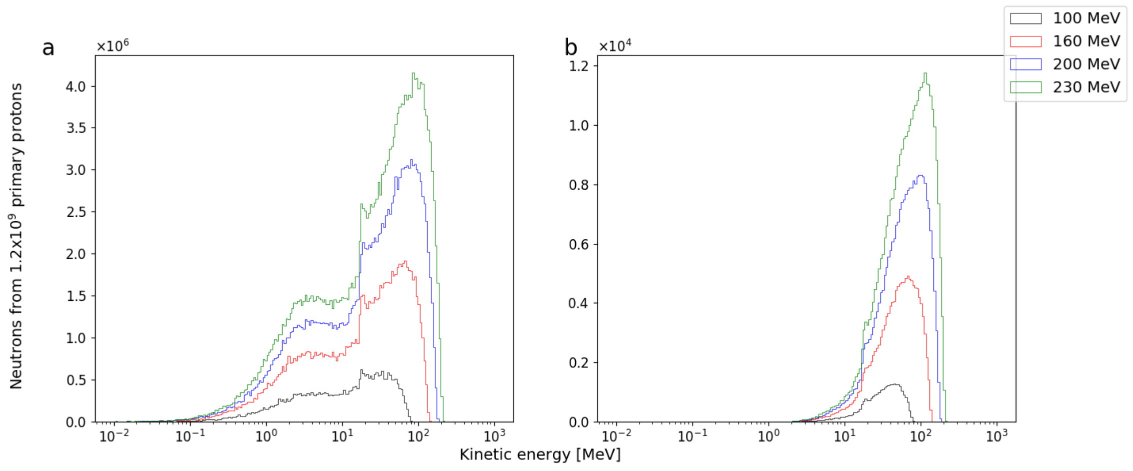

Figure 19: a: Distribution of initial kinetic energy for the neutrons produced in the water phantom. b: Distribution of initial kinetic energy for detected neutrons. The neutron spectrum is shown with logarithmic bins and with linear ordinate axis.

The energy and production positions inside the water phantom for 1000 detected neutrons are illustrated in Figure 20. We can see that, as expected, the neutrons were primarily produced along the axis of the primary proton beam. It is also observed that the average neutron energy at shallow depths was higher than for neutrons produced closer to the end of range for the primary proton beam.

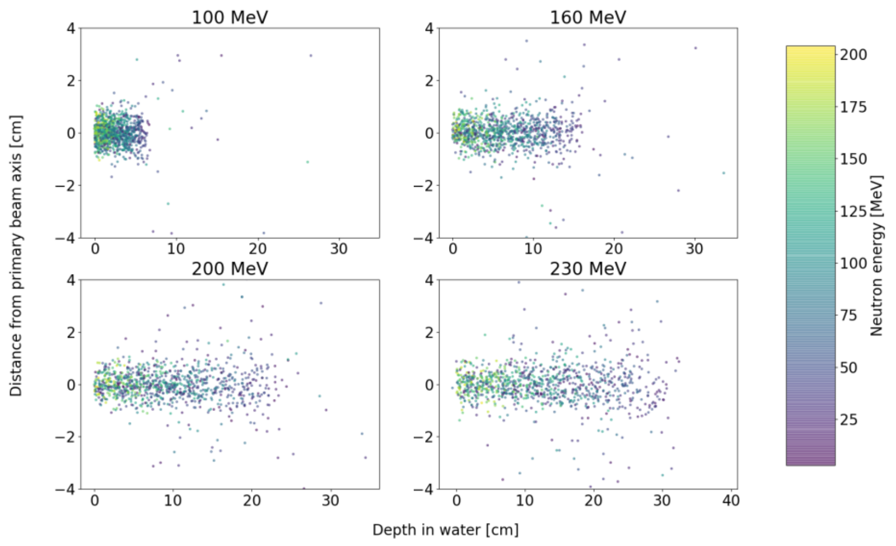

Figure 20: Neutron production shown as a function of distance from the beam axis (y-axis) and depth in water phantom (x-axis) for four proton energies. The kinetic energies of the neutrons at the time of production are indicated by the color bar.

The production positions inside the water phantom can also be displayed with a two-dimensional histogram as shown in Figure 21. While Figure 20 shows the discrete neutron positions, Figure 21 shows the distribution of neutrons in the water phantom.

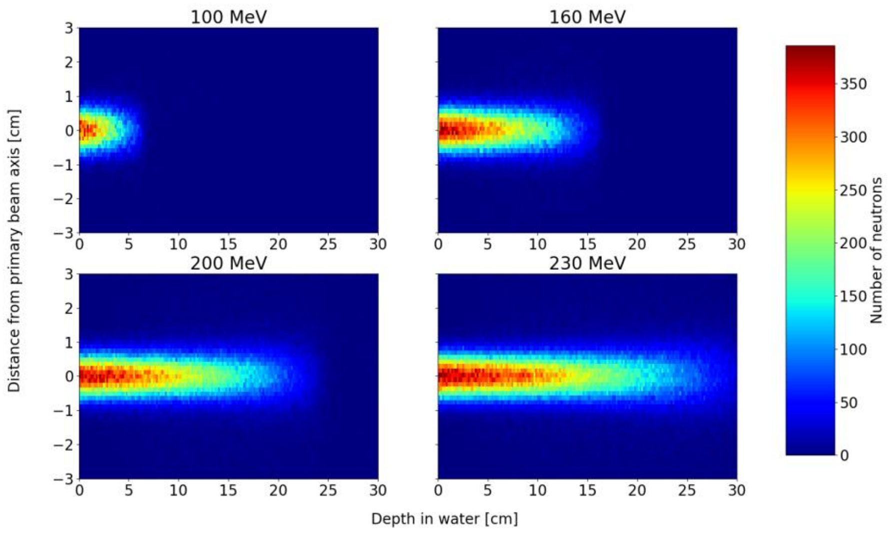

Figure 21: Two-dimensional histogram illustrating the production of all neutrons inside the water phantom, from 1.2x10^9 primary protons.

The distribution of the detected secondary protons from (n,p) reactions on the two tracking detectors rapidly increased at the entrance point of the water phantom (at x=0) until it reached a peak, with a following gradual decrease as the depth increased, as shown in Figure 22. The histograms for the four different energies look similar but are shifted deeper into the water phantom as the energy increases. In addition, a greater number of neutrons were detected for higher energies, which corresponds to the increased neutron production rate as shown in the previous figures.

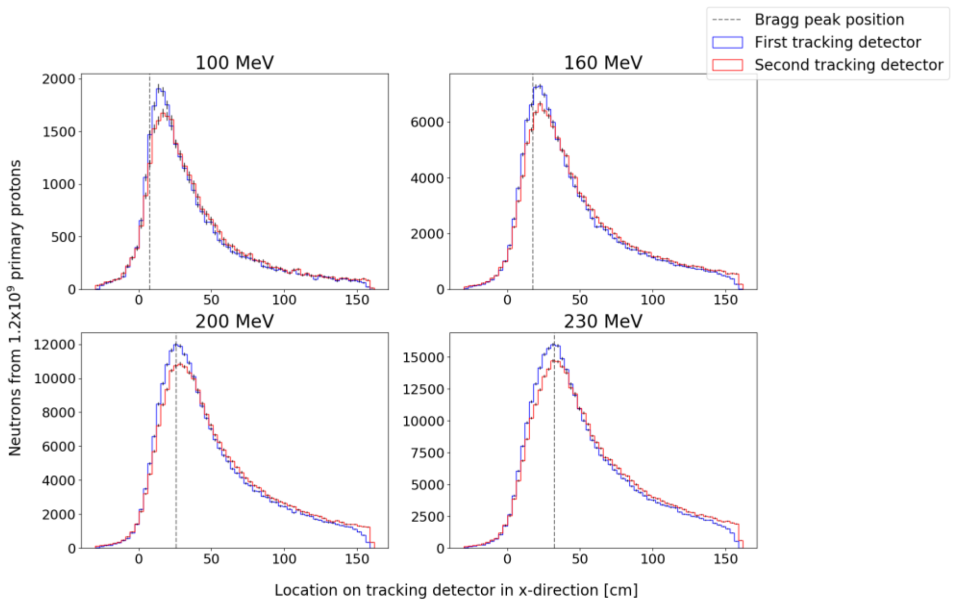

Figure 22: Neutron detection distribution illustrating the correlation between Bragg peak position and neutron detection rate on the two large tracking detectors. The statistical uncertainties are indicated by the error bars for each bin (about 1-2%).

There is also a correlation between the location of the Bragg peak in the water phantom and the location where the neutron detection rate is highest. For all energies, the location with the most detected neutrons was located around Bragg peak depth. Distal for all energies, but gradually closer for increasing energy. The maximum neutron detection rates from Figure 22 are given in Table 5-2.

Table 5-2: Positions of maximum neutron detection rate on the two detection planes. The number in the parentheses gives the location relative to the Bragg peak position (positive number indicate peak distal to Bragg peak), with Bragg peaks located at 7.6 cm, 17.5 cm, 25.7 cm and 32.6 cm for protons of 100, 160, 200 and 230 MeV energy, respectively.

Energy [MeV]	**Tracking plane nr.**	**Location [cm]**
100	1	15.0 ± 3.0 (7.4 ± 3.0)
100	2	18.0 ± 6.0 (10.4 ± 6.0)
160	1	21.0 ± 3.0 (3.5 ± 3.0)
160	2	22.5 ± 1.5 (5.0 ± 1.5)
200	1	27.0 ± 3.0 (1.3 ± 3.0)
200	2	28.5 ± 4.5 (2.8 ± 4.5)
230	1	33.0 ± 3.0 (0.4 ± 3.0)
230	2	33.0 ± 3.0 (0.4 ± 3.0)

5.1.2 Neutron detection rates as function of detector position

In Figure 23 the total number of detected neutrons is displayed for different positions of a 20x20 cm^2 detector within the large detection plane used in the simulation. The neutron detection rate is illustrated as a function of the centre point of the various detector positions. The reference point was set at 0 cm at the entrance of the water phantom. In general, we see that the positioning of the detector strongly impacts the detection rates which increases as the detector was moved in depth direction, to a certain point where a peak was reached, followed by an almost symmetric decrease. The detector position that provides maximal detection rate for a 20x20 cm^2 detector is centred in 15 ± 2 cm, 20.5 ± 1.5 cm, 24.5 ± 1.5 cm and 28 ± 2 cm, for 100, 160, 200 and 230 MeV, respectively. For 100 MeV this results in a maximum sensitivity position of 7.4 ± 2 cm distal to the Bragg peak position, while for 230 MeV it is at 4.6 ± 2 cm proximal to the Bragg peak. This shows that there is a shift in the ideal detector position relative to the Bragg peak depth, depending on the initial energy of the primary proton beam.

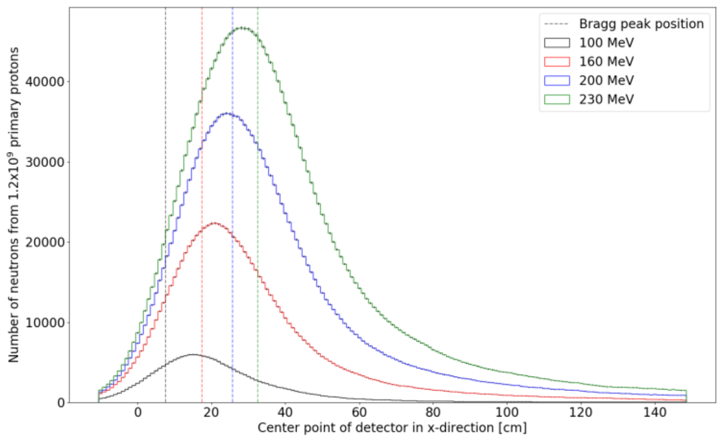

Figure 23: Neutron detection rate as a function of the center position of the different 20x20 cm^2 detectors. x = 0 cm at the entrance point of the water phantom.

Analyses were also performed only for those neutrons that were produced in depths corresponding to the last half and last quarter part of the range of the primary proton beams. The difference in neutron detection rate is significant when comparing the different analyses, and a change can be seen in the optimal detector placement (Figure 24). Compared to the analysis for all neutrons, the peaks are still almost symmetric, and the shift is still energy dependent, but the distance relative to the Bragg peak is greater. The difference is greater for greater energies, and all peaks are here located distal to the Bragg peak positions.

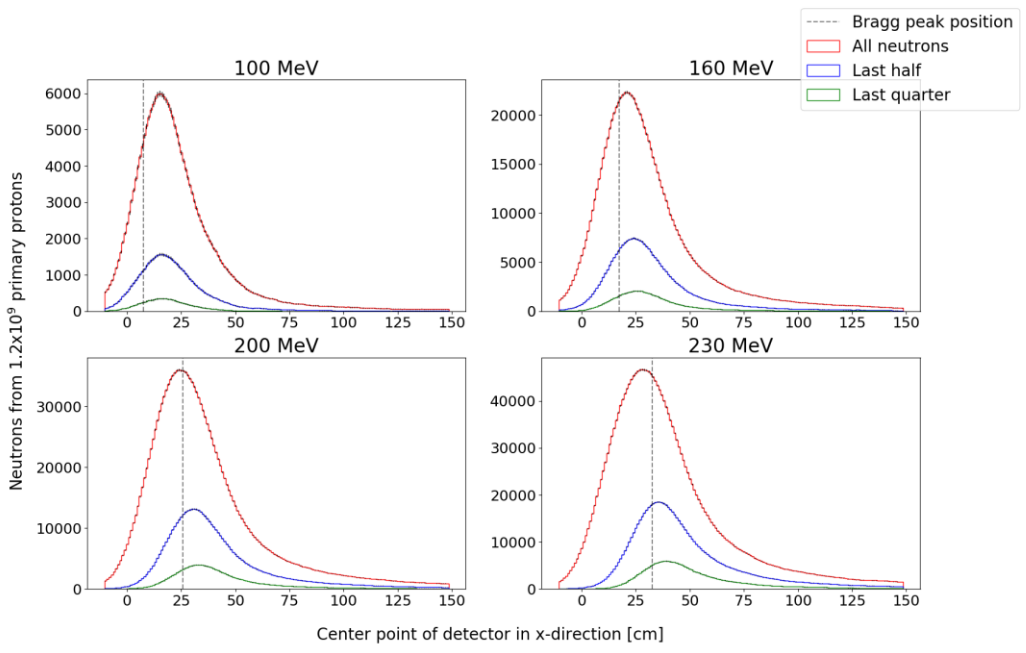

Figure 24: Neutron detection rate as a function of the center position of the different 20x20 cm^2 detectors. The detection rate is illustrated for each energy, and for each analysis with produced neutrons from all, half and last quarter part of range for the primary proton beam.

The positions of maximum neutron detection rate found in Figure 24, including Bragg peak positions and error bars, are illustrated in Figure 25. The exact numbers can be found in Appendix E.

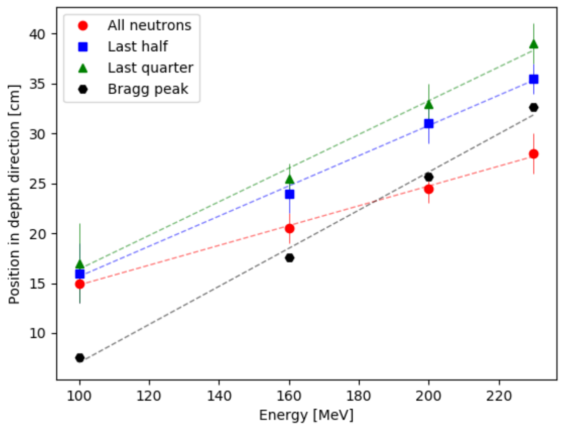

Figure 25: Position of maximum neutron detection rate (including error bars) for the four different energies used in simulation, and for each analysis with produced neutrons from all, half and last quarter part of range for the primary proton beam. Bragg peak positions are included for comparison.

The neutron detection rates for the 20x20 cm^2 detectors in the position of maximum detection rate in Figure 24, are gathered in Table 5-3.

Table 5-3: Total neutron detection rates per primary proton for the 20x20 cm^2 detector, for the four simulated energies. The statistical uncertainties are calculated as $\sqrt{N}$/P where N is the number of detected neutrons and P is the number of primary protons.

Proton beam energy [MeV]	**Neutron detection rate in ideal position [detected neutrons/primary proton]**		
	All neutrons	**Neutrons detected last half prior to Bragg peak**	**Neutrons detected last quarter prior to Bragg peak**
100	5.0 x 10^{-6} ± 1.3 %	1.3 x 10^{-6} ± 2.5 %	2.9 x 10^{-7} ± 5.3 %
160	1.9 x 10^{-5} ± 0.7 %	6.2 x 10^{-6} ± 1.2 %	1.7 x 10^{-6} ± 2.2 %
200	3.0 x 10^{-5} ± 0.5 %	1.1 x 10^{-5} ± 0.9 %	3.3 x 10^{-6} ± 1.6 %
230	3.9 x 10^{-5} ± 0.5 %	1.5 x 10^{-5} ± 0.7 %	5.0 x 10^{-6} ± 1.3 %

In clinical proton treatments the energies are changed rapidly, and it is not always expedient to move the detector when delivering the different energies. Therefore, a common location for a 160 and 200 MeV proton beam has been evaluated. In Figure 26 all neutrons produced are included, and the combination of 160 and 200 MeV gave a peak neutron detection rate at a position of 23 ± 1 cm. When compared to the results of 160 and 200 MeV, 20.5 ± 1.5 cm and 24.5 ± 1.5 cm, we see that the uncertainty limits of the individual peaks do not overlap, but the magnitude of the difference is only 1 cm. However, both peaks are within the limits of the uncertainties for the summed histogram.

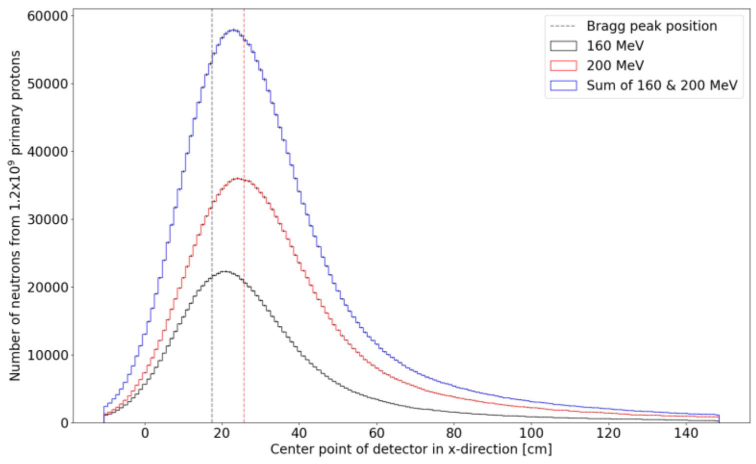

Figure 26: Neutron detection rate for primary proton beams of 160 and 200 MeV, including a rate for the combined results. All neutrons were included in this analysis. The statistical uncertainties are indicated by the error bars for each bin.

5.1.3 Neutron detection rates as function of detector size

In Figure 27 the neutron detection rates are displayed for different detector sizes placed in the ideal position at each primary proton beam energy (see Appendix E). Linear fits were included in order to evaluate the slope of the histograms. In general, we see that the rate increases as the detector size increases, and that the rate strongly depends on energy. For all energies the detection rate increases continuously as the area increases, but with higher energies it rises faster. In conclusion, the neutron detection rate increases almost linearly with increasing detector size.

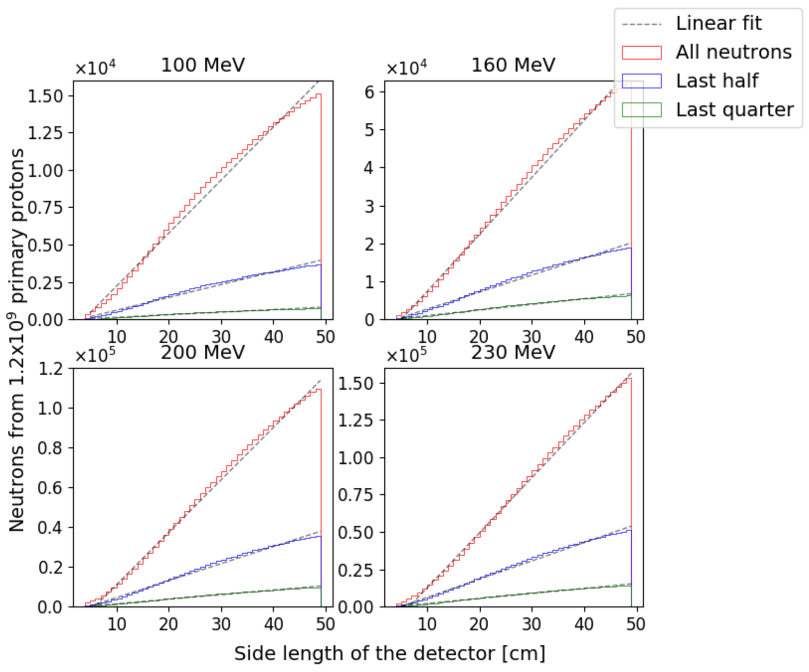

Figure 27: Neutron detection rate as a function of the side length of the quadratic detector, for the different proton beam energies and analyses done in this thesis.

5.2 Patient treatment plan simulations

5.2.1 Neutron production rates and detection characteristics

The total neutron production and detection rate for the large detection plane, predicted by MC simulations were 0.09 ± 0.05% neutrons produced/primary proton and 7.0x10^{-5} ± 1.1% neutron detected/primary proton. Compared to the water phantom simulation this result is, on average, approximately three times bigger than for the primary proton beam of 100 MeV, and almost identical as the result from the 160 MeV proton beam. This corresponds well with the energy range for the patient treatment plan (93-197 MeV).

The neutron production and detection distribution in the patient is illustrated in Figure 28a, and as seen neutron production rate rapidly increases as the beam enters the patient, remains stable, and starts to gradually decrease at the lower energy limit of the SOBP. This contrasts with results for single energies but can be explained by the large energy interval and resulting SOBP. The neutron detection distribution appears analogous as the distribution for the produced neutrons except that it drops earlier. In Figure 28b, the initial kinetic energy of the neutrons produced and detected in the patient is illustrated. As seen, the dominating energies for produced neutrons were between 1 and 100 MeV, similar to the results from the water phantom simulation. The energy distribution for detected neutrons shows that exceedingly few neutrons with energies below 10 MeV were detected compared with the higher energy neutrons. The general shape of the distributions also corresponds well with the results from the water phantom simulation.

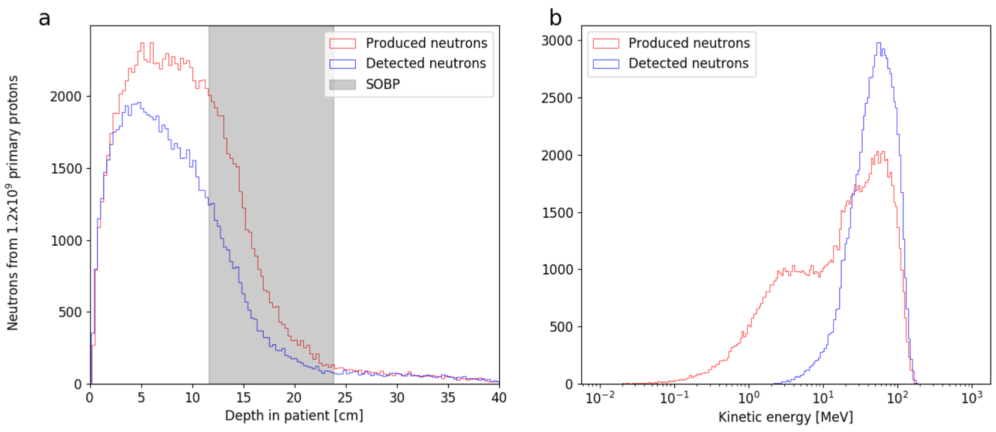

Figure 28: Comparison of characteristic for produced and detected neutrons. Only 1 of 1000 produced neutrons are illustrated. a: Neutron production and detection distribution as a function of depth in the patient. The Bragg peak positions are estimated at x = 11.6 cm for the lower energy limit, and x = 23.8 cm for the upper energy limit. b: Distribution of initial kinetic energy for neutrons produced and detected in the patient.

The relative dose deposited in the patient is illustrated in Figure 29. This gives a good depiction of where the tumour is located, and the dose delivered to surrounding tissue.

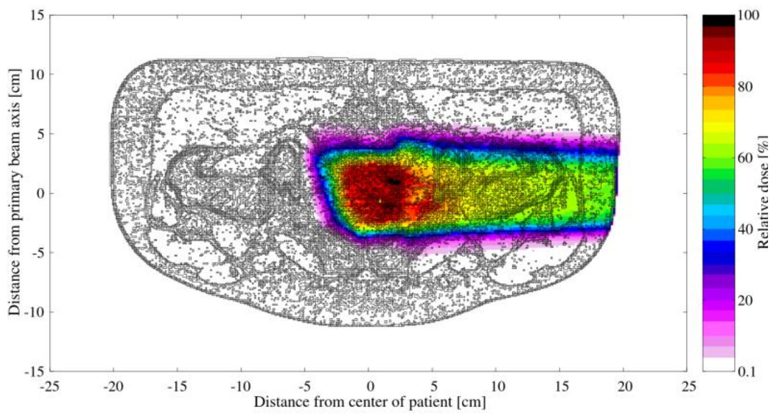

Figure 29: Two-dimensional dose deposition delivered in the patient, presented as the relative dose.

The detection rate for secondary protons on the two tracking detectors rapidly increases at the detector surfaces located parallel to the entrance of the patient (x=0) until a peak is reached, with a following gradual decrease further distal to the SOBP, as seen in Figure 30. The neutron detection rate peak is located in 36 ± 6 cm for the first tracking detector, and 36 ± 9 cm for the second tracking detector. This gives a distance of 12.2 ± 6 cm and 12.2 ± 9 cm distal to the SOBP for the first and second tracking plane, respectively. This result corresponds well with the results from the water phantom, with the peak being located distal to the primary proton beam Bragg peak.

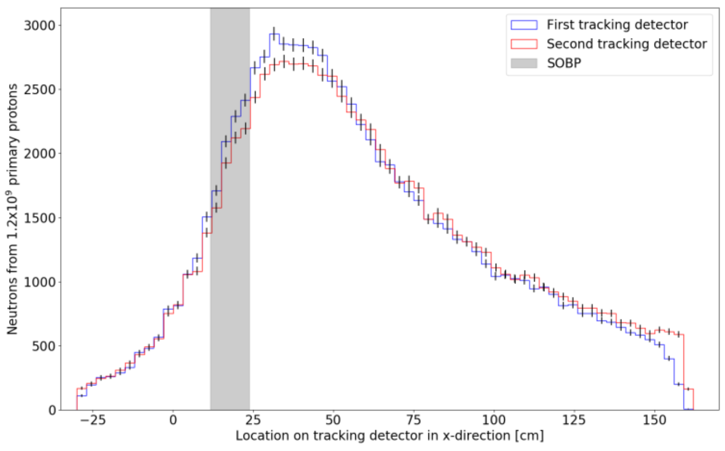

Figure 30: Neutron detection distribution on the two tracking detectors. x = 0 at the entrance of the patient. The statistical uncertainty is indicated by the error bars for each bin.

5.2.2 Neutron detection rates as function of detector position

In Figure 31 the total number of detected neutrons is displayed for different positions of a 20x20 cm^2 detector, within the large detection plane used in the simulation. The neutron detection rate is illustrated as a function of the centre position of the various detector placements. In general, we see that the rate increases as the detector is moved along the depth direction, to a certain point distal to the SOBP where a peak is reached, followed by a gradual decrease. Compared to the results from the water phantom simulations, the general shape and location of the peak is very similar, considering the peak is, also here, located distal to the Bragg peak. The maximum detection rate is found at 31 ± 5 cm, which corresponds to a location 7.2 ± 5 cm distal to the SOBP. The neutron detection rate in the bin with maximum detection rate is 4.6 x 10^{-6} ± 1.4 %.

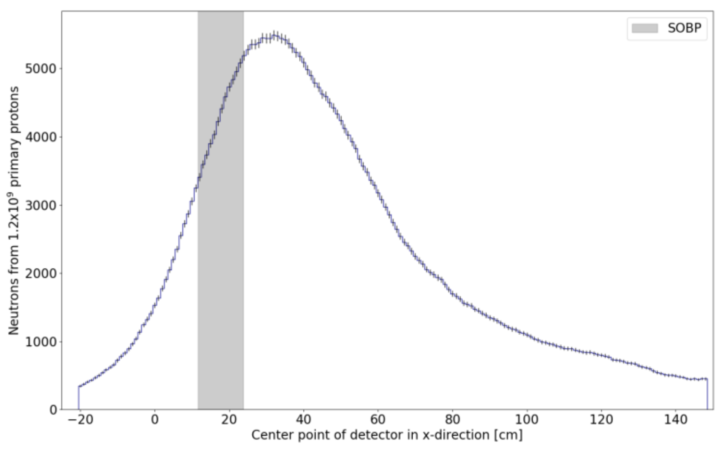

Figure 31: Neutron detection rate as a function of the center point of a 20x20 cm^2 detector. x = 0 cm at the entrance of the patient. The minimum and maximum values for the SOBP are 11.6 and 23.8 cm. The statistical uncertainty is indicated by the error bars for each bin.

5.2.3 Neutron detection rates as function of detector size

The total number of detected neutrons are displayed for different detector sizes, placed in the ideal position (as found in Figure 31), within the large detection plane used in the simulation. The neutron detection rate is illustrated as a function of the area of the detector (see Figure 32). A linear fit was included in order to evaluate the rise of the histogram. In general, we see that the rate increases continuously as the detector size increases. In conclusion, the neutron detection rate increases almost linearly with increasing detector size.

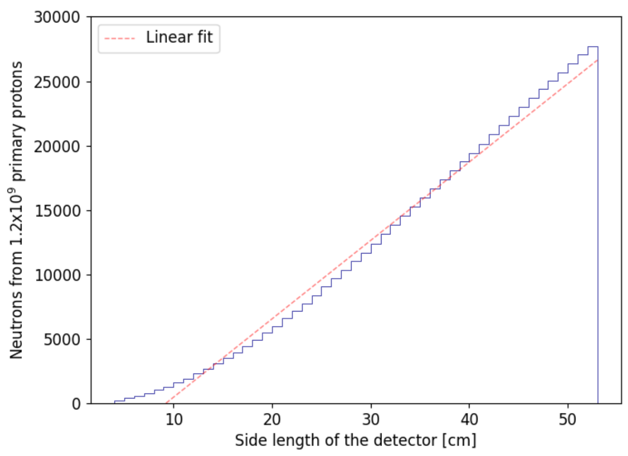

Figure 32: Neutron detection rate as a function of the side length of the quadratic detector, for the position of maximum neutron detection rate.

6. Discussion and conclusion

The objective of this project was to optimize detector dimensions and positioning of an existing detector concept developed for neutron-based real-time range verifications in proton therapy. The results of this work indicate that the positioning of the detector relative to the patient may have a great impact on the detected signal and the precision of the final proton range estimate. Additionally, the size of the detector may impact the detection rates and achievable counting statistics. The objective of this project was pursued using FLUKA Monte Carlo simulations. An unrealistically large detector was applied in simulations in order to evaluate the neutron detection rate for various smaller areas within the large detection plane. Simulations were performed both for a water phantom and a realistic patient treatment plan. Water phantom simulations included four different primary proton energies: 100, 160, 200 and 230 MeV, covering typical ranges of therapeutic proton beams. Further, a 20x20 cm^2 detector was defined in the analysis and moved stepwise 1 cm along the primary beam direction, in order to locate the position with the maximum neutron detection rate. Analyses separating the neutrons by production location in the water phantom were also implemented, in order to evaluate the neutrons produced closer to the primary proton beam Bragg peak.

The performance of the detector was seen to vary significantly depending on the detector size and positioning, both for the water phantom and patient treatment plan simulations. This shows that the detector placement and dimensions are important for achieving an acceptable neutron detection rate. Neutrons produced in the entrance region where the primary proton beam energy is still high had, on average, higher initial energy than neutrons produced later in the primary proton beam trajectory. Secondary neutrons were mostly produced close to the primary beam axis.

For the water phantom simulations (Figure 23), it is seen that for almost all energies and setups, the neutron detection rate peak is located distal to the Bragg peak. If only the neutrons produced just proximal to the Bragg peak are considered, these peaks shift further distal to the Bragg peak, as seen in Figure 24. However, which neutrons one should focus on is debatable. Even though the neutrons created adjacent the proton beam end of range could give a closer indication of the Bragg peak location, these neutrons have a lower average energy than neutrons produced earlier in the proton beam trajectory. This could make them more difficult to detect in clinical application, because a low energy neutron will produce a secondary proton in the converter with even lower energy, and the particle energy affects the particle range. There are also fewer neutrons produced closer to the end of range for the proton beam (as seen in Figure 20) which gives inferior statistics for the neutron detection rate.

In principle, it would be best to use all available neutrons for the range estimates, however it might be possible to weight information from neutrons created in the Bragg peak region more in the range reconstruction if these are considered to be better correlated to the primary beam range. As seen in Appendix E the uncertainty in neutron detection rate is not particularly affected by the reduced number of neutrons, except for the 100 MeV proton beam where the uncertainty increases by ± 1 cm. However, low uncertainty in neutron detection rate does not necessarily mean that the uncertainty in range estimates will be low as well. Furthermore, in a clinical situation, the number of protons in a particular beam spot may be significantly lower than in the MC simulations considered in this thesis and therefore statistical uncertainties may be higher and more relevant in clinical scenarios.

Another observation from Figure 23 is that the neutron detection rates depend strongly on the primary proton beam energy. Further, when the beam energy increases, the detection rate peak is shifted deeper into the water phantom. For a 20x20 cm^2 detector, maximum detection rate is achieved when centred in 15 ± 2 cm, 20.5 ± 1.5 cm, 24.5 ± 1.5 cm and 28 ± 2 cm, for 100, 160, 200 and 230 MeV, respectively. The Bragg peak positions for the different beam energies have a greater shift than the neutron detection rate peak. This is seen e.g. for 100 MeV where the maximum sensitivity position is 7.4 ± 2 cm distal to the Bragg peak position, while for 230 MeV it is at 4.6 ± 2 cm proximal to the Bragg peak. This means that the beam energy must be considered when determining the detector placement.

In clinical applications, the delivery of the prescribed dose to the planning target volume will be based upon using multiple proton beams covering a range of energies and intensities, resulting in a SOBP. It could be considered impractical and challenging to change the location of the detector during clinical application of proton therapy, due to the rapid energy modulation when delivering beams at different energy layers. Therefore, as a work around, a possible common position for 160 and 200 MeV beams was investigated. These two energies were chosen since they are within a common energy range for therapeutic proton beams, and the energy gap is relatively small. A small energy gap is common for clinical proton therapy treatment plans, although this depends strongly on the target volume size. This resulted in a new peak between the two original peaks for 160 and 200 MeV (20.5 ± 1.5 cm, 24.5 ± 1.5 cm), located at 23 ± 1 cm. When the uncertainty limits were considered, this peak was located within the peaks for the two individual beams. Since the results from both 160 and 200 MeV were included, the overall counting statistics were good, and the statistical uncertainty in neutron detection rate was reduced compared with the individual results. Nevertheless, a separate evaluation of the placement should be done considering the energy range for the specific case, since not all proton beam patient treatment plans range from 160 to 200 MeV.

For the patient treatment plan the results regarding the detector position correspond well with the results from the water phantom simulation. The peak for the neutron detection rate is also here located slightly (7.2 ± 5 cm) distal to the (spread-out) Bragg peak, and the neutron detection rate varies greatly with different detector positions. The energy range for the patient treatment plan is large which makes it difficult to compare the details of the results, but since the range for the patient treatment plan is from 93 to 197 MeV, the results could be compared with the water phantom beams of 100 and 160 MeV.

When comparing the results from the water phantom simulations in Figure 23 with the patient treatment plan simulations in Figure 31, it is clear that the statistics are poorer for the patient plan, resulting in lower detection rate and a larger statistical uncertainty. A possible explanation is the large energy range and SOBP in the patient treatment plan. It is also worth mentioning that the distance from the water phantom/patient to the converter is 14 cm for the patient plan simulations and 5 cm for the water phantom simulations. The distance for the patient treatment plan simulations is more realistic for clinical applications but may partially explain why the patient plan results are inferior.

The neutron detection rates accomplished in this thesis, when a 20x20 cm^2 detector is placed in the ideal position, were 5.0 x 10^{-6} ± 1.3 %, 1.9 x 10^{-5} ± 0.7 %, 3.0 x 10^{-5} ± 0.5 %, 3.9 x 10^{-5} ± 0.5 % for 100, 160, 200, 230 MeV, respectively. The corresponding value for the results in the patient treatment plan simulations, was 4.6 x 10^{-6} ± 1.4 %. This is comparable to detection rates achieved with existing prompt-gamma imaging techniques, i.e. 10^{-5} – 5.6x10^{-5} gamma counts per primary proton, reported by Krimmer et. al. [43]. These neutron detection rates are also comparable with previous studies in neutron-based range verifications, such as the results from Ytre-Hauge et. al. [13]. However, it is noticeable that the neutron detection rate from the patient treatment plan is considerably low compared to the water phantom simulations: four times smaller than for 160 MeV and almost identical as for the 100 MeV primary proton beam. This contrasts with the neutron detection rate for the entire 190x200 cm^2 detector, where the result was, on average, approximately three times bigger than for the primary proton beam of 100 MeV, and almost identical as the result from the 160 MeV proton beam. This shows that the neutrons created in the patient are spread over a larger area than for the water phantom simulations, which can be explained by the large energy range of the patient plan and the difference in distance to the converter. Consequently, the results show that moving the detector during treatment is beneficial in order to obtain an acceptable neutron detection rate when a large energy range is applied in the treatment plan.

An ideal size of the detector is a trade-off between the need for good statistics and the cost and inconvenience of a larger detector. Based on the neutron detection rate, we can draw the conclusion that the rate increases almost linearly with increasing detector area. A bulky detector is impractical in clinical applications, thus the distance between the converter surface and patient should be considered. Shorter distance enables a reduced detector area, since it leads to a greater solid angle for a given detector size. This could enable the use of a smaller detector area without reducing its performance. To further quantify the required size of the detector to achieve acceptable low uncertainties in the range estimates, a full reconstruction of the neutron production distribution must be performed, using e.g. the reconstruction algorithm presented in [13]. Furthermore, range estimates as well as corresponding uncertainties must be obtained from the reconstructed neutron production distributions.

Regarding the possible detector alternatives for clinical application of in-vivo neutron range verification, several charged particle tracking detectors could be applicable. These are detector types such as silicon strip detectors, pixelated Si-based position sensitive sensors or gas filled multiwire proportional chambers, all suggestions from research by Ytre-Hauge et. al. [13].

It is also important to state that the results from simulations in FLUKA and other MC codes are based on models and cross-section data with considerable uncertainties. A different simulation software could have given different neutron production and detection rates, and experimental measurements are needed to draw firm conclusions regarding achievable detector rates as also detector limitations such as efficiency, also plays a role in clinical application of the detector system.

Further work

In this thesis, the neutron detection rate was evaluated in order to determine the ideal position and size of the detector. These are basic properties of the proposed detector system which are important to systematically review before development of a first detector prototype. There are also other aspects that would be important to investigate, such as the uncertainty in range landmark in the reconstruction process for the different positions and sizes of the detector. This would be particularly interesting considering the dimensions of the detector, since analysis of the neutron detection rate did not provide a helpful conclusion in the matter. The position and size determined by uncertainty in range landmark could also be interesting to compare with the results in this thesis, to see if the location with maximum neutron detection rate could be considered the ideal location for the detector.

Conclusion

The results in this thesis show that the neutron detection rates for both water phantom and patient treatment plan simulation depends strongly on the detector position and dimensions. Based on the water phantom simulations, the energy also plays a big role in determining the position. In general, we see that the neutron detection rate peaks close to or distal to the Bragg peak. For a 20x20 cm^2 detector, using a single position for a broad range of proton energies would lead to a clear reduction in neutron detection rate compared to using multiple positions. The feasibility of moving the detector during treatment should therefore be evaluated. For small tumours and target volumes one static detector position would however be sufficient. Another alternative might be to increase the size of the detector, but this should be investigated further.

Bibliograph

1. WHO *Global action against cancer*. 2005.
2. *Cancer in Norway 2017 - Cancer incidence, mortality, survival and prevalence in Norway*. Cancer Registry of Norway, 2018.
3. Gianfaldoni, S., et al., *An Overview on Radiotherapy: From Its History to Its Current Applications in Dermatology.* Open access Macedonian journal of medical sciences, 2017. **5**(4): p. 521-525.
4. Khan, F.M., *The physics of radiation therapy*. 2nd edition ed. 1994: Lippincott Williams & Wilkins.
5. Thariat, J., et al., *Past, present, and future of radiotherapy for the benefit of patients Juliette Thariat, Jean-Michel Hannoun-Levi, Arthur Sun Myint, Te Vuong and Jean-Pierre Gérard*. Vol. 10. 2012.
6. Wilson, R.R., *Radiological Use of Fast Protons.* 1946. **47**(5): p. 487-491.
7. Wang, D., *A critical appraisal of the clinical utility of proton therapy in oncology.* Medical Devices: Evidence and research, 2015.
8. *Facilities in operation*. 2019 [cited 2019 25.03]; Available from: https://www.ptcog.ch/index.php/facilities-in-operation.
9. St James, S., C. Grassberger, and H.-M. Lu, *Considerations when treating lung cancer with passive scatter or active scanning proton therapy.* Translational lung cancer research, 2018. **7**(2): p. 210-215.
10. Verburg, J.M. *Reducing range uncertainty in proton therapy*. 2015.
11. Unkelbach, J., et al., *Robust radiotherapy planning.* Physics in Medicine & Biology, 2018. **63**(22): p. 22TR02.
12. Kraan, A.C., *Range Verification Methods in Particle Therapy: Underlying Physics and Monte Carlo Modeling.* 2015. **5**(150).
13. Ytre-Hauge, K.S., Skjerdal, K., Mattingly, J., & Meric, I., *A Monte Carlo feasibility study for neutron based real-time range verification in proton therapy.* Scientific reports, 2011 (2019).
14. Newhauser, W.D. and R. Zhang, *The physics of proton therapy*. Phys Med Biol, 2015. **60**(8): p. R155-209.
15. M.J. Berger, J.S.C., M.A. Zucker and J. Chang. *Stopping-Power & Range Tables for Electrons, Protons, and Helium Ions*. 1998 2017; Available from: https://physics.nist.gov/PhysRefData/Star/Text/PSTAR.html.
16. Park, S.H. and J.O. Kang, *Basics of particle therapy I: physics.* Radiation oncology journal, 2011. **29**(3): p. 135-146.
17. Cember, H., *Introduction to health physics*. 4 ed. 2009: The McGraw-Hill Companies.
18. Grzanka, L., *Modelling beam transport and biological effectiveness to develop treatment planning for ion beam radiotherapy*. 2014.
19. Podgoršak, E.B., *Interactions of Neutrons with Matter*, in *Radiation Physics for Medical Physicists*, E.B. Podgorsak, Editor. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 429-449.
20. Snyder, W.S., et al., *Calculated depth dose curves in tissue for broad beams of fast neutrons*. At head of title:Health Physics Division. 1955, Oak Ridge, Tenn.: Oak Ridge National Laboratory. 35 p.

21. Furutaka, K., H. Harada, and S. Raman, *Prompt Gamma Rays Emitted in Thermal-neutron Capture Reaction by 99Tc and Its Reaction Cross Section.* Journal of Nuclear Science and Technology, 2004. **41**(11): p. 1033-1046.
22. Podgoršak, E.B., *Interactions of Photons with Matter*, in *Radiation Physics for Medical Physicists*, E.B. Podgorsak, Editor. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 277-375.
23. Paganetti, H., et al., *Relative biological effectiveness (RBE) values for proton beam therapy.* Int J Radiat Oncol Biol Phys, 2002. **53**(2): p. 407-21.
24. Takada, K., et al., *Validation of the physical and RBE-weighted dose estimator based on PHITS coupled with a microdosimetric kinetic model for proton therapy.* Journal of radiation research, 2018. **59**(1): p. 91-99.
25. Jette, D. and W. Chen, *Creating a spread-out Bragg peak in proton beams.* Phys Med Biol, 2011. **56**(11): p. N131-8.
26. Leeman, J.E., et al., *Proton therapy for head and neck cancer: expanding the therapeutic window.* The Lancet Oncology, 2017. **18**(5): p. e254-e265.
27. Ytre-Hauge, K., *Proton therapy*, in *Proton dosimetric concepts*. 2018, University of Bergen: PHYS213.
28. Knopf, A.C. and A. Lomax, *In vivo proton range verification: a review.* Phys Med Biol, 2013. **58**(15): p. R131-60.
29. Unkelbach, J. and H. Paganetti, *Robust Proton Treatment Planning: Physical and Biological Optimization.* Seminars in radiation oncology, 2018. **28**(2): p. 88-96.
30. Parodi, K. and J.C. Polf, *In vivo range verification in particle therapy.* Medical Physics, 2018. **45**(11): p. e1036-e1050.
31. Smeets, J., et al., *Prompt gamma imaging with a slit camera for real-time range control in proton therapy.* Phys Med Biol, 2012. **57**(11): p. 3371-405.
32. Richter, C., et al., *First clinical application of a prompt gamma based in vivo proton range verification system.* Radiother Oncol, 2016. **118**(2): p. 232-7.
33. Johannes Petzoldt, G.J. and L.N. , Christian Richter, Julien Smeets *Correction of Geometrical Effects of a Knife-Edge Slit Camera for Prompt Gamma-Based Range Verification in Proton Therapy*. 2018.

34. Zhu, X. and G. El Fakhri, *Proton therapy verification with PET imaging.* Theranostics, 2013. **3**(10): p. 731-40.
35. Verburg, J.M. and J. Seco, *Proton range verification through prompt gamma-ray spectroscopy.* Physics in Medicine and Biology, 2014. **59**(23): p. 7089-7106.
36. Marafini, M., et al., *MONDO: A neutron tracker for particle therapy secondary emission fluxes measurements.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2016. **824**: p. 210-211.
37. Kenton, W. *Monte Carlo simulation*. 2018 [cited 2019 14.02]; Available from: https://www.investopedia.com/terms/m/montecarlosimulation.asp.
38. Ferrari, A., et al., *FLUKA: A multi-particle transport code*, in *CERN-2005-10 (2005), INFN/TC_05/11, SLAC-R-773*. 2005.
39. Bohlen, T.T., et al., *The FLUKA Code: Developments and Challenges for High Energy and Medical Applications.* Nuclear Data Sheets, 2014. **120**: p. 211-214.
40. *FLUKA about*. 2008 [cited 2019 14.02]; Available from: http://www.fluka.org/fluka.php?id=about&mm2=1.

41. *Release notes for Fluka2011.2x* -. 2019 [cited 2019 12.11]; Available from: www.fluka.org/fluka.php?id=release_notes&mm2=3.
42. Fjæra, L.F., *Development of a Monte Carlo Based Treatment Planning Verification Tool for Particle Therapy*, in *Department of physics and technology*. 2016, University of Bergen: Bergen open reaserch archive - uib.
43. Krimmer, J., et al., *Prompt-gamma monitoring in hadrontherapy: A review*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2018. **878**: p. 58-73.

Appendix A

Because the geometry of the detector has been changed in this thesis, the region numbers identifying the different regions have been altered in these scripts to correspond with the new geometry.

Bxdraw_rangeveri.f

FLUKA user routine used for both water phantom and patient treatment plan simulations. This is used to store information about direction, energy and location of the neutrons produced in the phantom/patient, as well as proton information from the converter and tracking detectors.

```
*$ CREATE MGDRAW.FOR
*COPY MGDRAW
*                                                                      *
*=== mgdraw ===========================================================*
*                                                                      *
      SUBROUTINE MGDRAW ( ICODE, MREG )

      INCLUDE '(DBLPRC)'
      INCLUDE '(DIMPAR)'
      INCLUDE '(IOUNIT)'
*
*----------------------------------------------------------------------*
*                                                                      *
*     Copyright (C) 1990-2013      by        Alfredo Ferrari           *
*     All Rights Reserved.                                             *
*                                                                      *
*                                                                      *
*     MaGnetic field trajectory DRAWing: actually this entry manages   *
*                                        all trajectory dumping for    *
*                                        drawing                       *
*                                                                      *
*     Created on   01 March 1990   by        Alfredo Ferrari           *
*                                              INFN - Milan            *
*     Last change   12-Nov-13      by        Alfredo Ferrari           *
*                                              INFN - Milan            *
*                                                                      *
*----------------------------------------------------------------------*
*
      INCLUDE '(CASLIM)'
      INCLUDE '(COMPUT)'
      INCLUDE '(SOURCM)'
      INCLUDE '(FHEAVY)'
      INCLUDE '(FLKSTK)'
      INCLUDE '(GENSTK)'
      INCLUDE '(MGDDCM)'
      INCLUDE '(PAPROP)'
      INCLUDE '(QUEMGD)'
      INCLUDE '(SUMCOU)'
      INCLUDE '(TRACKR)'
```

```
*
      DIMENSION DTQUEN ( MXTRCK, MAXQMG )
*
      CHARACTER*20 FILNAM
      LOGICAL LFCOPE
      SAVE LFCOPE
      DATA LFCOPE / .FALSE. /
*
*----------------------------------------------------------------------*
*                                                                      *
*     Icode = 1: call from Kaskad                                      *
*     Icode = 2: call from Emfsco                                      *
*     Icode = 3: call from Kasneu                                      *
*     Icode = 4: call from Kashea                                      *
*     Icode = 5: call from Kasoph                                      *
*                                                                      *
*----------------------------------------------------------------------*
*                                                                      *
      IF ( .NOT. LFCOPE ) THEN
         LFCOPE = .TRUE.
         IF ( KOMPUT .EQ. 2 ) THEN
            FILNAM = '/'//CFDRAW(1:8)//' DUMP A'
         ELSE
            FILNAM = CFDRAW
         END IF
         OPEN ( UNIT = IODRAW, FILE = FILNAM, STATUS = 'NEW', FORM =
     &          'UNFORMATTED' )
      END IF
*  +-------------------------------------------------------------------*
*  |  Quenching is activated
      IF ( LQEMGD ) THEN
         IF ( MTRACK .GT. 0 ) THEN
            RULLL  = ZERZER
            CALL QUENMG ( ICODE, MREG, RULLL, DTQUEN )
*            WRITE (IODRAW) ( ( SNGL (DTQUEN (I,JBK)), I = 1, MTRACK ),
*     &                         JBK = 1, NQEMGD )
         END IF
      END IF
*  |  End of quenching
*     +--------------------------------------------------------------------
*
      RETURN
*
*=====================================================================*
*                                                                      *
*     Boundary-(X)crossing DRAWing:                                    *
*                                                                      *
*     Icode = 1x: call from Kaskad                                     *
*             19: boundary crossing                                    *
*     Icode = 2x: call from Emfsco                                     *
*             29: boundary crossing                                    *
*     Icode = 3x: call from Kasneu                                     *
*             39: boundary crossing                                    *
*     Icode = 4x: call from Kashea                                     *
*             49: boundary crossing                                    *
*     Icode = 5x: call from Kasoph                                     *
*             59: boundary crossing                                    *
*                                                                      *
```

```
*=========================================================================*
*
**************************Code by Kristian****************************
*mreg:region before crossing
*nreg:region after crossing
*write crossing coords. to file(41) in pairs:
*Add ncase (history number) to make sure it is the same particle
*crossing both planes
*Jtrack=1 means particle is proton (neutrons = 8)

      ENTRY BXDRAW ( ICODE, MREG, NEWREG, XSCO, YSCO, ZSCO )

*plane 1: crossing regions 5 to 6
      IF(MREG .EQ. 5 .AND. NEWREG .EQ. 6) THEN
         IF (JTRACK .EQ. 1) THEN
            EKPART=ETRACK-AM(JTRACK)

*     If proton is from converter:
            IF(ISPUSR(1) .EQ. 8) THEN
*     Ek-proton calculated as totalE - Emass (0.938272...)
               NCASE1=NCASE
               X1=XSCO
               Y1=YSCO
               Z1=ZSCO
               EK1=EKPART
               ox=SPAUSR(1)
               oy=SPAUSR(2)
               oz=SPAUSR(3)
               ocx=SPAUSR(4)
               ocy=SPAUSR(5)
               ocz=SPAUSR(6)
               EKo=SPAUSR(7)
               id=SPAUSR(8)
*dir cos from TRACKR Cx,y,ztrck
               Pcos1x=CXTRCK
               Pcos1y=CYTRCK
               Pcos1z=CZTRCK
*     next step: write to files only if NCASE1=NCASE2

            ENDIF
         ENDIF
      ENDIF
*Plane 2: crossing regions 6 to 4
      IF(MREG .EQ. 6 .AND. NEWREG .EQ. 4) THEN
        IF (JTRACK .EQ. 1 .AND. ISPUSR(1) .EQ. 8) THEN
           EKPART=ETRACK-AM(JTRACK)

* Write new file with all info if Ncase matches for plane
*     one and two
            IF(NCASE .EQ. NCASE1) THEN
               WRITE(46,*) X1,Y1,Z1,EK1,NCASE1,
     &              ox,oy,oz,ocx,ocy,ocz,EKo,
     &              Pcos1x,Pcos1y,Pcos1z,ISPUSR(2),id
               WRITE(46,*) XSCO,YSCO,ZSCO,EKPART,NCASE,
     &              SPAUSR(1), SPAUSR(2), SPAUSR(3),
     &              SPAUSR(4), SPAUSR(5), SPAUSR(6), SPAUSR(7),
     &              CXTRCK,CYTRCK,CZTRCK,ISPUSR(2),SPAUSR(8)
            ENDIF
        ENDIF
      ENDIF
      RETURN
*
***************END OF CODE MODIFIED BY KRISTIAN***********************
*
*=====================================================================*
*                                                                     *
*     Event End DRAWing:                                              *
*                                                                     *
*=====================================================================*
*                                                                     *
      ENTRY EEDRAW ( ICODE )
      RETURN
*
*=====================================================================*
*                                                                     *
*     ENergy deposition DRAWing:                                      *
*                                                                     *
*     Icode = 1x: call from Kaskad                                    *
*             10: elastic interaction recoil                          *
*             11: inelastic interaction recoil                        *
*             12: stopping particle                                   *
*             13: pseudo-neutron deposition                           *
*             14: escape                                              *
*             15: time kill                                           *
*     Icode = 2x: call from Emfsco                                    *
*             20: local energy deposition (i.e. photoelectric)        *
*             21: below threshold, iarg=1                             *
*             22: below threshold, iarg=2                             *
*             23: escape                                              *
*             24: time kill                                           *
*     Icode = 3x: call from Kasneu                                    *
*             30: target recoil                                       *
*             31: below threshold                                     *
*             32: escape                                              *
*             33: time kill                                           *
*     Icode = 4x: call from Kashea                                    *
*             40: escape                                              *
*             41: time kill                                           *
*             42: delta ray stack overflow                            *
*     Icode = 5x: call from Kasoph                                    *
*             50: optical photon absorption                           *
*             51: escape                                              *
*             52: time kill                                           *
*                                                                     *
*=====================================================================*
*                                                                     *
      ENTRY ENDRAW ( ICODE, MREG, RULL, XSCO, YSCO, ZSCO )
      IF ( .NOT. LFCOPE ) THEN
         LFCOPE = .TRUE.
         IF ( KOMPUT .EQ. 2 ) THEN
            FILNAM = '/'//CFDRAW(1:8)//' DUMP A'
         ELSE
            FILNAM = CFDRAW
         END IF
         OPEN ( UNIT = IODRAW, FILE = FILNAM, STATUS = 'NEW', FORM =
     &          'UNFORMATTED' )
```

```
      END IF
*      WRITE (IODRAW)  0, ICODE, JTRACK, SNGL (ETRACK), SNGL (WTRACK)
*      WRITE (IODRAW)  SNGL (XSCO), SNGL (YSCO), SNGL (ZSCO), SNGL (RULL)
*  +-------------------------------------------------------------------*
*  |  Quenching is activated : calculate quenching factor
*  |  and store quenched energy in DTQUEN(1, jbk)
      IF ( LQEMGD ) THEN
         RULLL = RULL
         CALL QUENMG ( ICODE, MREG, RULLL, DTQUEN )
*         WRITE (IODRAW) ( SNGL (DTQUEN(1, JBK)), JBK = 1, NQEMGD )
      END IF
*  |  end quenching
*  +-------------------------------------------------------------------*
      RETURN
*
*=====================================================================*
*                                                                     *
*     SOurce particle DRAWing:                                        *
*                                                                     *
*=====================================================================*
*
      ENTRY SODRAW
      IF ( .NOT. LFCOPE ) THEN

      END IF
*      WRITE (IODRAW) -NCASE, NPFLKA, NSTMAX, SNGL (TKESUM),
*     &                SNGL (WEIPRI)
*  +-------------------------------------------------------------------*
*  |  (Radioactive) isotope: it works only for 1 source particle on
*  |  the stack for the time being
      IF ( ILOFLK (NPFLKA) .GE. 100000 .AND. LRADDC (NPFLKA) ) THEN
         IARES  = MOD ( ILOFLK (NPFLKA), 100000  )  / 100
         IZRES  = MOD ( ILOFLK (NPFLKA), 10000000 ) / 100000
         IISRES = ILOFLK (NPFLKA) / 10000000
         IONID  = ILOFLK (NPFLKA)
*         WRITE (IODRAW) ( IONID,SNGL(-TKEFLK(I)),
*     &                    SNGL (WTFLK(I)), SNGL (XFLK (I)),
*     &                    SNGL (YFLK (I)), SNGL (ZFLK (I)),
*     &                    SNGL (TXFLK(I)), SNGL (TYFLK(I)),
*     &                    SNGL (TZFLK(I)), I = 1, NPFLKA )
*  |
*  +-------------------------------------------------------------------*
*  |  Patch for heavy ions: it works only for 1 source particle on
*  |  the stack for the time being
      ELSE IF ( ABS (ILOFLK (NPFLKA)) .GE. 10000 ) THEN
         IONID = ILOFLK (NPFLKA)
         CALL DCDION ( IONID )
*  |
*  +-------------------------------------------------------------------*
*  |  Patch for heavy ions: ???
      ELSE IF ( ILOFLK (NPFLKA) .LT. -6 ) THEN
*         WRITE (IODRAW) ( IONID,SNGL(TKEFLK(I)+AMNHEA(-ILOFLK(NPFLKA))),
*     &                    SNGL (WTFLK(I)), SNGL (XFLK (I)),
*     &                    SNGL (YFLK (I)), SNGL (ZFLK (I)),
*     &                    SNGL (TXFLK(I)), SNGL (TYFLK(I)),
*     &                    SNGL (TZFLK(I)), I = 1, NPFLKA )
*  |
*  +-------------------------------------------------------------------*
*  |
      ELSE
*         WRITE (IODRAW) ( ILOFLK(I), SNGL (TKEFLK(I)+AM(ILOFLK(I))),
*     &                    SNGL (WTFLK(I)), SNGL (XFLK (I)),
*     &                    SNGL (YFLK (I)), SNGL (ZFLK (I)),
*     &                    SNGL (TXFLK(I)), SNGL (TYFLK(I)),
*     &                    SNGL (TZFLK(I)), I = 1, NPFLKA )
      END IF
*  |
*  +-------------------------------------------------------------------*
      RETURN
*
*=====================================================================*
*                                                                     *
*     USer dependent DRAWing:                                         *
*                                                                     *
*     Icode = 10x: call from Kaskad                                   *
*             100: elastic   interaction secondaries                  *
*             101: inelastic interaction secondaries                  *
*             102: particle decay  secondaries                        *
*             103: delta ray  generation secondaries                  *
*             104: pair production secondaries                        *
*             105: bremsstrahlung  secondaries                        *
*             110: decay products                                     *
*     Icode = 20x: call from Emfsco                                   *
*             208: bremsstrahlung secondaries                         *
*             210: Moller secondaries                                 *
*             212: Bhabha secondaries                                 *
*             214: in-flight annihilation secondaries                 *
*             215: annihilation at rest   secondaries                 *
*             217: pair production        secondaries                 *
*             219: Compton scattering     secondaries                 *
*             221: photoelectric          secondaries                 *
*             225: Rayleigh scattering    secondaries                 *
*             237: mu pair     production secondaries                 *
*     Icode = 30x: call from Kasneu                                   *
*             300: interaction secondaries                            *
*     Icode = 40x: call from Kashea                                   *
*             400: delta ray  generation secondaries                  *
*  For all interactions secondaries are put on GENSTK common (kp=1,np) *
*  but for KASHEA delta ray generation where only the secondary elec-  *
*  tron is present and stacked on FLKSTK common for kp=npflka         *
*                                                                     *
*=====================================================================*
*
      ENTRY USDRAW ( ICODE, MREG, XSCO, YSCO, ZSCO )
***** START ADDED BY KRISTIAN IN USDRAW****************************
*input to mgdraw to write all positions for created neutrons in
*water (currently region no. 7)
**     WRITE(60,*) XSCO, YSCO, ZSCO, CXR (IP), CYR(IP),KPART(IP)
*
*
      IF(ICODE .EQ. 101 .AND. MREG .EQ. 7) THEN
         DO IP = 1, NP
            IF(KPART(IP) .EQ. 8) THEN
*     Store origin of neutron (x,y,z)

*     Loop to write 1 of 1000 hits to a file
```

```
               i =i+1
               IF (i .GT. 1000) THEN
                  WRITE(60,*) XSCO,YSCO,ZSCO,Tki(IP)
     &            ,Cxr(IP),Cyr(IP),Czr(IP)
*                  WRITE(61,*) XSCO, Tki(IP)
                  i = 1
               ENDIF
            ENDIF
         END DO
      END IF
*
* Np = total number of secondaries *
* Kpart (ip) = (Paprop) id of the ip_th secondary *
* Cxr (ip) = x-axis direction cosine of the ip_th secondary *
* Tki (ip) = laboratory kinetic energy of ip_th secondary (GeV)*
* Wei (ip) = statistical weight of the ip_th secondary *
* etc. (look up the full list in $FLUPRO/flukapro/(GENSTK)
***** END ADDED BY KRISTIAN IN USDRAW****************************
      IF ( .NOT. LFCOPE ) THEN
         LFCOPE = .TRUE.
         IF ( KOMPUT .EQ. 2 ) THEN
            FILNAM = '/'//CFDRAW(1:8)//' DUMP A'
         ELSE
            FILNAM = CFDRAW
         END IF
         OPEN ( UNIT = IODRAW, FILE = FILNAM, STATUS = 'NEW', FORM =
     &          'UNFORMATTED' )
      END IF
* No output by default:
      RETURN
*=== End of subrutine Mgdraw ==========================================*
      END
```

Stuprf_v02.f

FLUKA user routine used for both water phantom and patient treatment plan simulations. This is used to check which reactions happened in the converter when the secondary protons were created: 1) Inelastic interaction, 2) Elastic interaction and 3) Low energy neutron scattering (En < 20 MeV).

```
*$ CREATE STUPRF.FOR
*COPY STUPRF
*
*=== stuprf ===========================================================*
*
      SUBROUTINE STUPRF ( IJ, MREG, XX, YY, ZZ, NPSECN, NPPRMR )

      INCLUDE '(DBLPRC)'
      INCLUDE '(DIMPAR)'
      INCLUDE '(IOUNIT)'
*
*----------------------------------------------------------------------*
*                                                                      *
*     Copyright (C) 1997-2005      by    Alfredo Ferrari & Paola Sala  *
*     All Rights Reserved.                                             *
*                                                                      *
*                                                                      *
*     SeT User PRoperties for Fluka particles:                         *
*                                                                      *
*     Created on  09 october 1997  by    Alfredo Ferrari & Paola Sala  *
*                                                   Infn - Milan       *
*                                                                      *
*     Last change on  14-jul-05    by    Alfredo Ferrari               *
*                                                                      *
*                                                                      *
*----------------------------------------------------------------------*
*
      INCLUDE '(EVTFLG)'
      INCLUDE '(FLKSTK)'
      INCLUDE '(TRACKR)'
      INCLUDE '(GENSTK)'
*
      LOUSE   (NPFLKA)  = LLOUSE
      DO 100 ISPR = 1, MKBMX1
         SPAREK (ISPR,NPFLKA) = SPAUSR (ISPR)
  100 CONTINUE
      DO 200 ISPR = 1, MKBMX2
         ISPARK (ISPR,NPFLKA) = ISPUSR (ISPR)
  200 CONTINUE
*  Increment the track number and put it into the last flag:
      IF ( NPSECN .GT. NPPRMR ) THEN
         IF ( NTRCKS .EQ. 2000000000 ) NTRCKS = -2000000000
         NTRCKS = NTRCKS + 1
         ISPARK (MKBMX2,NPFLKA) = NTRCKS
      END IF

******************
```

```
******************

*************************************************************************
* Code by Kristian START tracking part 1 - in water phantom
*************************************************************************
      IF (LINEVT) THEN
*         WRITE(LUNOUT,*)'W,:',NPSECN,KPART(NPSECN), NP
         IF(KPART(NPSECN) .EQ. 8 ) THEN
*            WRITE(LUNOUT,*)' kpartnpsecn:',Tki(NPSECN)
            SPAREK(1,NPFLKA)=XX
            SPAREK(2,NPFLKA)=YY
            SPAREK(3,NPFLKA)=ZZ
            SPAREK(4,NPFLKA)=Cxr(NPSECN)
            SPAREK(5,NPFLKA)=Cyr(NPSECN)
            SPAREK(6,NPFLKA)=Czr(NPSECN)
            SPAREK(7,NPFLKA)=Tki(NPSECN)
            SPAREK(8,NPFLKA)=MREG !identifies region of origin wat.phan.10
         END IF
      END IF



*LELEVT = Elastic interactio
*LINEVT = Inelastic interaction
*LLENSC = Low energy neutron scattering
*     310518 - could add requirement to save
* flags only for the protons
      IF (LINEVT) THEN
         IF (MREG .EQ. 8) THEN
*Save in ISPARK(1) the father ID (IJ = 8 for neutron)
*(ISPARK goes to ISPUSR-variable in mgdraw)
            ISPARK(1,NPFLKA)= IJ
*set ispark 2 =1 for LINEVT
            ISPARK(2,NPFLKA)= 1
         ENDIF
      ENDIF

      IF (LELEVT) THEN
*Check if we are in converter region (8)
         IF (MREG .EQ. 8) THEN
*              WRITE(LUNOUT,*)' detect el in conv'
            ISPARK(1,NPFLKA)= IJ
*set ispark 2 =2 for LELEVT
            ISPARK(2,NPFLKA)= 2
         ENDIF
      ENDIF

      IF (LLENSC) THEN
*Check if we are in converter region (8)
         IF (MREG .EQ. 8) THEN
*              WRITE(LUNOUT,*)' detect low en scatt'
            ISPARK(1,NPFLKA)= IJ
*set ispark 2 =3 for LLENSC
            ISPARK(2,NPFLKA)= 3
         ENDIF
      ENDIF
*next variable of interest (int) can be stored is ISPARK(2)
*         ISPARK(2,NPFLKA)=.....
```

```
*SPAREK can be used for storing non-int values (double prec?)
*SPAREK goes to SPAUSR in mgdraw

*************************************************************************
* Code by Kristian START tracking part 1 - in water phantom
*************************************************************************
*************************
*************************




      RETURN
*=== End of subroutine Stuprf ==========================================*
      END
```

Table A-1: Format for file output from tracking code. Subscript “1” and “2” indicates information from the two tracker planes. Ek is the kinetic energy of the detected proton, Ncase is the primary particle number. O, origin, gives the position and direction of the neutron produced. X, Y and Z is the position where the proton crosses the tracker planes. Eko is the neutron initial energy. P1 and P2 cos gives the direction of the proton on the tracker planes. Event types are 1: inelastic (LINEVT), 2: Elastic (LELEVT), 3: Low-energy neutron (LLENSC).

Nr.	**File 46 (line 1)**	**File 46 (line 2)**	**File 60**
0	X_1	X_2	Ox
1	Y_1	Y_2	Oy
2	Z_1	Z_2	Oz
3	Ek_1	Ek_2	Ek
4	Ncase_1	Ncase_2	Ocos_X
5	Ox_1	Ox_2	Ocos_Y
6	Oy_1	Oy_2	Ocos_Z
7	Oz_1	Oz_2	
8	OcosX_1	OcosX_2	
9	OcosY_1	Ocosy_2	
10	OcosZ_1	OcosZ_2	
11	Eko	Eko	
12	P1_CosX	P2_CosX	
13	P1_CosY	P2_CosY	
14	P1_CosZ	P2_CosZ	
15	Event type	Event type	

Appendix B

depthdoseplot.py

```
# simple 1D plot for dose vs depth for different energies

import numpy as np
import matplotlib.pyplot as plt
import os, string
from matplotlib.pyplot import *


#set font size and tick size
plt.rc('xtick',labelsize=12)
plt.rc('ytick',labelsize=12)
plt.rcParams.update({'font.size': 12})

#depth dose lists
dx100 = []
dy100 = []
dx160 = []
dy160 = []
dx200 = []
dy200 = []
dx230 = []
dy230 = []

# Read the depth dose data files
a = ["/home/ift-pt4/flukaProjects/1d100.dat"]
fopen = open (os.path.join(*a), 'r')
count = 0
for line in fopen:
#    if line.startswith("#"):
    if '#' in line or not line.strip():
        continue
    temp = line.split()
    dx100.append(((float(temp[0]))+(float(temp[1])))/2)
    dy100.append(float(temp[2]))

fopen.close()

# Read the depth dose data file
a = ["/home/ift-pt4/flukaProjects/1d160.dat"]
```

```
fopen = open (os.path.join(*a), 'r')
count = 0
for line in fopen:
#    if line.startswith("#"):
    if '#' in line or not line.strip():
        continue
    temp = line.split()
    dx160.append(((float(temp[0]))+(float(temp[1])))/2)
    dy160.append(float(temp[2]))

fopen.close()



# Read the depth dose data file
a = ["/home/ift-pt4/flukaProjects/1d200.dat"]
fopen = open (os.path.join(*a), 'r')
count = 0
for line in fopen:
    if '#' in line or not line.strip():
        continue
    temp = line.split()
    dx200.append(((float(temp[0]))+(float(temp[1])))/2)
    dy200.append(float(temp[2]))

fopen.close()


# Read the depth dose data file
a = ["/home/ift-pt4/flukaProjects/1d230.dat"]
fopen = open (os.path.join(*a), 'r')
count = 0
for line in fopen:
    if '#' in line or not line.strip():
        continue
    temp = line.split()
    dx230.append(((float(temp[0]))+(float(temp[1])))/2)
    dy230.append(float(temp[2]))

fopen.close()


#normalize to 100%
```

```
dy100 = [i*(100/max(dy100)) for i in dy100]
dy160 = [i*(100/max(dy160)) for i in dy160]
dy200 = [i*(100/max(dy200)) for i in dy200]
dy230 = [i*(100/max(dy230)) for i in dy230]


#depth doses plotting
d100= plt.plot(dx100,dy100,'-',alpha=1 ,color ='k',linewidth=0.5, label = '100 MeV')
d160= plt.plot(dx160,dy160,'-',alpha=1 ,color ='r',linewidth=0.5, label = '160 MeV')
d200= plt.plot(dx200,dy200,'-',alpha=1 ,color ='b',linewidth=0.5, label = '200 MeV')
d230= plt.plot(dx230,dy230,'-',alpha=1 ,color ='g',linewidth=0.5, label = '230 MeV')

plt.xlabel('Depth in water [cm]')
plt.ylabel('Relative dose [%]')
plt.legend(loc = 'upper right')

#set y and x range
plt.xlim(0,35)
plt.ylim(0,140)

plt.show()
```

Waterphantom.py

All water phantom plots created from file 46 and 60 (as seen in Appendix A).

```
# Information from 46 and 60 file from FLUKA organised and plotted

#Import python libraries
import numpy as np
import matplotlib.pyplot as plt

# Set font size and tick size for all plots
plt.rcParams.update({'font.size': 14})
plt.rc('xtick',labelsize=12)
plt.rc('ytick',labelsize=12)

#Generate empty lists for the simulated data(detected neutrons)
#100 MeV
X_1_100 = []
Y_1_100 = []
Z_1_100 = []
Ek_1_100 = []
Ncase_1_100 = []
Ox_1_100 = []
Oy_1_100 = []
Oz_1_100 = []
Ocosx_1_100 = []
Ocosy_1_100 = []
Ocosz_1_100 = []
Eko_1_100 = []
P1cosx_1_100 = []
P1cosy_1_100 = []
P1cosz_1_100 = []
Evt_type_1_100 = []

X_2_100 = []
Y_2_100 = []
Z_2_100 = []
Ek_2_100 = []
Ncase_2_100 = []
Ox_2_100 = []
Oy_2_100 = []
Oz_2_100 = []
Ocosx_2_100 = []
```

```
Ocosy_2_100 = []
Ocosz_2_100 = []
Eko_2_100 = []
P1cosx_2_100 = []
P1cosy_2_100 = []
P1cosz_2_100 = []
Evt_type_2_100 = []

#160 MeV
X_1_160 = []
Y_1_160 = []
Z_1_160 = []
Ek_1_160 = []
Ncase_1_160 = []
Ox_1_160 = []
Oy_1_160 = []
Oz_1_160 = []
Ocosx_1_160 = []
Ocosy_1_160 = []
Ocosz_1_160 = []
Eko_1_160 = []
P1cosx_1_160 = []
P1cosy_1_160 = []
P1cosz_1_160 = []
Evt_type_1_160 = []

X_2_160 = []
Y_2_160 = []
Z_2_160 = []
Ek_2_160 = []
Ncase_2_160 = []
Ox_2_160 = []
Oy_2_160 = []
Oz_2_160 = []
Ocosx_2_160 = []
Ocosy_2_160 = []
Ocosz_2_160 = []
Eko_2_160 = []
P1cosx_2_160 = []
P1cosy_2_160 = []
P1cosz_2_160 = []
Evt_type_2_160 = []
```

```
#200 MeV
X_1_200 = []
Y_1_200 = []
Z_1_200 = []
Ek_1_200 = []
Ncase_1_200 = []
Ox_1_200 = []
Oy_1_200 = []
Oz_1_200 = []
Ocosx_1_200 = []
Ocosy_1_200 = []
Ocosz_1_200 = []
Eko_1_200 = []
P1cosx_1_200 = []
P1cosy_1_200 = []
P1cosz_1_200 = []
Evt_type_1_200 = []

X_2_200 = []
Y_2_200 = []
Z_2_200 = []
Ek_2_200 = []
Ncase_2_200 = []
Ox_2_200 = []
Oy_2_200 = []
Oz_2_200 = []
Ocosx_2_200 = []
Ocosy_2_200 = []
Ocosz_2_200 = []
Eko_2_200 = []
P1cosx_2_200 = []
P1cosy_2_200 = []
P1cosz_2_200 = []
Evt_type_2_200 = []

#230 MeV
X_1_230 = []
Y_1_230 = []
Z_1_230 = []
Ek_1_230 = []
Ncase_1_230 = []
Ox_1_230 = []
Oy_1_230 = []
```

```
Oz_1_230 = []
Ocosx_1_230 = []
Ocosy_1_230 = []
Ocosz_1_230 = []
Eko_1_230 = []
P1cosx_1_230 = []
P1cosy_1_230 = []
P1cosz_1_230 = []
Evt_type_1_230 = []

X_2_230 = []
Y_2_230 = []
Z_2_230 = []
Ek_2_230 = []
Ncase_2_230 = []
Ox_2_230 = []
Oy_2_230 = []
Oz_2_230 = []
Ocosx_2_230 = []
Ocosy_2_230 = []
Ocosz_2_230 = []
Eko_2_230 = []
P1cosx_2_230 = []
P1cosy_2_230 = []
P1cosz_2_230 = []
Evt_type_2_230 = []



# Read the data files and append to the empty lists
import csv

with  open('100MeV_001_all46_p_coordinates_and_parent_n_origin_xyz.csv',  'r')  as
csvFile:
    reader = csv.reader(csvFile)
    k = 0
    for row in reader:
    temp = row[0].split()
    if (k % 2) == 0:
            X_1_100.append(float(temp[0]))
            Y_1_100.append(float(temp[1]))
            Z_1_100.append(float(temp[2]))
            Ek_1_100.append(float(temp[3])*1000)
            Ncase_1_100.append(float(temp[4]))
```

```
          Ox_1_100.append(float(temp[5]))
          Oy_1_100.append(float(temp[6]))
          Oz_1_100.append(float(temp[7]))
          Ocosx_1_100.append(float(temp[8]))
          Ocosy_1_100.append(float(temp[9]))
          Ocosz_1_100.append(float(temp[10]))
          Eko_1_100.append(float(temp[11])*1000)
          P1cosx_1_100.append(float(temp[12]))
          P1cosy_1_100.append(float(temp[13]))
          P1cosz_1_100.append(float(temp[14]))
          Evt_type_1_100.append(float(temp[15]))

    else:
          X_2_100.append(float(temp[0]))
          Y_2_100.append(float(temp[1]))
          Z_2_100.append(float(temp[2]))
          Ek_2_100.append(float(temp[3])*1000)
          Ncase_2_100.append(float(temp[4]))
          Ox_2_100.append(float(temp[5]))
          Oy_2_100.append(float(temp[6]))
          Oz_2_100.append(float(temp[7]))
          Ocosx_2_100.append(float(temp[8]))
          Ocosy_2_100.append(float(temp[9]))
          Ocosz_2_100.append(float(temp[10]))
          Eko_2_100.append(float(temp[11])*1000)
          P1cosx_2_100.append(float(temp[12]))
          P1cosy_2_100.append(float(temp[13]))
          P1cosz_2_100.append(float(temp[14]))
          Evt_type_2_100.append(float(temp[15]))
    k = k + 1

csvFile.close()


with  open('160MeV_001_all46_p_coordinates_and_parent_n_origin_xyz.csv',  'r')  as
csvFile:
    reader = csv.reader(csvFile)
    a = 0
    for row in reader:
    temp = row[0].split()
    if (a % 2) == 0:
          X_1_160.append(float(temp[0]))
          Y_1_160.append(float(temp[1]))
```

```
            Z_1_160.append(float(temp[2]))
            Ek_1_160.append(float(temp[3])*1000)
            Ncase_1_160.append(float(temp[4]))
           Ox_1_160.append(float(temp[5]))
            Oy_1_160.append(float(temp[6]))
            Oz_1_160.append(float(temp[7]))
            Ocosx_1_160.append(float(temp[8]))
            Ocosy_1_160.append(float(temp[9]))
            Ocosz_1_160.append(float(temp[10]))
            Eko_1_160.append(float(temp[11])*1000)
            P1cosx_1_160.append(float(temp[12]))
            P1cosy_1_160.append(float(temp[13]))
            P1cosz_1_160.append(float(temp[14]))
            Evt_type_1_160.append(float(temp[15]))

    else:
            X_2_160.append(float(temp[0]))
            Y_2_160.append(float(temp[1]))
            Z_2_160.append(float(temp[2]))
            Ek_2_160.append(float(temp[3])*1000)
            Ncase_2_160.append(float(temp[4]))
            Ox_2_160.append(float(temp[5]))
            Oy_2_160.append(float(temp[6]))
            Oz_2_160.append(float(temp[7]))
            Ocosx_2_160.append(float(temp[8]))
            Ocosy_2_160.append(float(temp[9]))
            Ocosz_2_160.append(float(temp[10]))
            Eko_2_160.append(float(temp[11])*1000)
            P1cosx_2_160.append(float(temp[12]))
            P1cosy_2_160.append(float(temp[13]))
            P1cosz_2_160.append(float(temp[14]))
            Evt_type_2_160.append(float(temp[15]))
    a = a + 1

csvFile.close()



with  open('200MeV_001_all46_p_coordinates_and_parent_n_origin_xyz.csv',  'r')  as
csvFile:
    reader = csv.reader(csvFile)
    b = 0
    for row in reader:
```

```
    temp = row[0].split()
    if (b % 2) == 0:
            X_1_200.append(float(temp[0]))
            Y_1_200.append(float(temp[1]))
            Z_1_200.append(float(temp[2]))
            Ek_1_200.append(float(temp[3])*1000)
            Ncase_1_200.append(float(temp[4]))
           Ox_1_200.append(float(temp[5]))
            Oy_1_200.append(float(temp[6]))
            Oz_1_200.append(float(temp[7]))
            Ocosx_1_200.append(float(temp[8]))
            Ocosy_1_200.append(float(temp[9]))
            Ocosz_1_200.append(float(temp[10]))
            Eko_1_200.append(float(temp[11])*1000)
            P1cosx_1_200.append(float(temp[12]))
            P1cosy_1_200.append(float(temp[13]))
            P1cosz_1_200.append(float(temp[14]))
            Evt_type_1_200.append(float(temp[15]))

    else:
            X_2_200.append(float(temp[0]))
            Y_2_200.append(float(temp[1]))
            Z_2_200.append(float(temp[2]))
            Ek_2_200.append(float(temp[3])*1000)
            Ncase_2_200.append(float(temp[4]))
            Ox_2_200.append(float(temp[5]))
            Oy_2_200.append(float(temp[6]))
            Oz_2_200.append(float(temp[7]))
            Ocosx_2_200.append(float(temp[8]))
            Ocosy_2_200.append(float(temp[9]))
            Ocosz_2_200.append(float(temp[10]))
            Eko_2_200.append(float(temp[11])*1000)
            P1cosx_2_200.append(float(temp[12]))
            P1cosy_2_200.append(float(temp[13]))
            P1cosz_2_200.append(float(temp[14]))
            Evt_type_2_200.append(float(temp[15]))
    b = b + 1

csvFile.close()



with  open('230MeV_001_all46_p_coordinates_and_parent_n_origin_xyz.csv',  'r')  as
```

```
csvFile:
    reader = csv.reader(csvFile)
    c = 0
    for row in reader:
    temp = row[0].split()
    if (c % 2) == 0:
            X_1_230.append(float(temp[0]))
            Y_1_230.append(float(temp[1]))
            Z_1_230.append(float(temp[2]))
            Ek_1_230.append(float(temp[3])*1000)
            Ncase_1_230.append(float(temp[4]))
           Ox_1_230.append(float(temp[5]))
            Oy_1_230.append(float(temp[6]))
            Oz_1_230.append(float(temp[7]))
            Ocosx_1_230.append(float(temp[8]))
            Ocosy_1_230.append(float(temp[9]))
            Ocosz_1_230.append(float(temp[10]))
            Eko_1_230.append(float(temp[11])*1000)
            P1cosx_1_230.append(float(temp[12]))
            P1cosy_1_230.append(float(temp[13]))
            P1cosz_1_230.append(float(temp[14]))
            Evt_type_1_230.append(float(temp[15]))

    else:
            X_2_230.append(float(temp[0]))
            Y_2_230.append(float(temp[1]))
            Z_2_230.append(float(temp[2]))
            Ek_2_230.append(float(temp[3])*1000)
            Ncase_2_230.append(float(temp[4]))
            Ox_2_230.append(float(temp[5]))
            Oy_2_230.append(float(temp[6]))
            Oz_2_230.append(float(temp[7]))
            Ocosx_2_230.append(float(temp[8]))
            Ocosy_2_230.append(float(temp[9]))
            Ocosz_2_230.append(float(temp[10]))
            Eko_2_230.append(float(temp[11])*1000)
            P1cosx_2_230.append(float(temp[12]))
            P1cosy_2_230.append(float(temp[13]))
            P1cosz_2_230.append(float(temp[14]))
            Evt_type_2_230.append(float(temp[15]))
    c = c + 1

csvFile.close()
```

```
primaries = '1.2x10$^{9}$'

bp100 = 7.6
bp160 = 17.5
bp200 = 25.7
bp230 = 32.6


#Information on produced neutrons

#create data lists
Ox_100 = []
Oy_100 = []
Oz_100 = []
Eko_100 = []
Ocosx_100 = []
Ocosy_100 = []
Ocosz_100 = []

Ox_160 = []
Oy_160 = []
Oz_160 = []
Eko_160 = []
Ocosx_160 = []
Ocosy_160 = []
Ocosz_160 = []

Ox_200 = []
Oy_200 = []
Oz_200 = []
Eko_200 = []
Ocosx_200 = []
Ocosy_200 = []
Ocosz_200 = []

Ox_230 = []
Oy_230 = []
Oz_230 = []
Eko_230 = []
Ocosx_230 = []
```

```
Ocosy_230 = []
Ocosz_230 = []


# Read the data file and append to the empty lists

import csv

with open('100MeV_001_all60_xyz_origin_all_neutrons.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    k = 0
    for row in reader:
    temp = row[0].split()
       Ox_100.append(float(temp[0]))
        Oy_100.append(float(temp[1]))
        Oz_100.append(float(temp[2]))
        Eko_100.append(float(temp[3])*1000)
        Ocosx_100.append(float(temp[4]))
        Ocosy_100.append(float(temp[5]))
        Ocosz_100.append(float(temp[6]))

    k = k + 1

csvFile.close()


with open('160MeV_001_all60_xyz_origin_all_neutrons.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    a = 0
    for row in reader:
    temp = row[0].split()
       Ox_160.append(float(temp[0]))
        Oy_160.append(float(temp[1]))
        Oz_160.append(float(temp[2]))
        Eko_160.append(float(temp[3])*1000)
        Ocosx_160.append(float(temp[4]))
        Ocosy_160.append(float(temp[5]))
        Ocosz_160.append(float(temp[6]))

    a = a + 1

csvFile.close()
```

```
with open('200MeV_001_all60_xyz_origin_all_neutrons.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    b = 0
    for row in reader:
    temp = row[0].split()
       Ox_200.append(float(temp[0]))
        Oy_200.append(float(temp[1]))
        Oz_200.append(float(temp[2]))
        Eko_200.append(float(temp[3])*1000)
        Ocosx_200.append(float(temp[4]))
        Ocosy_200.append(float(temp[5]))
        Ocosz_200.append(float(temp[6]))

    b = b + 1

csvFile.close()


with open('230MeV_001_all60_xyz_origin_all_neutrons.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    c = 0
    for row in reader:
    temp = row[0].split()
       Ox_230.append(float(temp[0]))
        Oy_230.append(float(temp[1]))
        Oz_230.append(float(temp[2]))
        Eko_230.append(float(temp[3])*1000)
        Ocosx_230.append(float(temp[4]))
        Ocosy_230.append(float(temp[5]))
        Ocosz_230.append(float(temp[6]))

    c = c + 1

csvFile.close()



#PLOT ENERGIES OF PRODUCED AND DETECTED NEUTRONS
#energy of the neutrons produced
fig, axes = plt.subplots(nrows = 1, ncols = 2, figsize = (15,6))

MIN, MAX = 0.01, 1000
```

```
(counts1,   bins1)   =   np.histogram(Eko_100,bins   =   np.logspace(np.log10(MIN),
np.log10(MAX), 200))
(counts2,  bins2)  =  np.histogram(Eko_160,  bins  =  np.logspace(np.log10(MIN),
np.log10(MAX), 200))
(counts3,  bins3)  =  np.histogram(Eko_200,  bins  =  np.logspace(np.log10(MIN),
np.log10(MAX), 200))
(counts4,  bins4)  =  np.histogram(Eko_230,  bins  =  np.logspace(np.log10(MIN),
np.log10(MAX), 200))

#multiply by a factor of 1000 since only 1 of 1000 neutrons were stored in the 60-
file
f = 1000
axes[0].hist(bins1[:-1],bins1,weights = f*counts1,  ec='black', fc='none', lw=0.5,
histtype='step', label = '100 MeV')
axes[0].hist(bins2[:-1], bins2, weights = f*counts2, ec='red', fc='none', lw=0.5,
histtype='step', label = '160 MeV')
axes[0].hist(bins3[:-1], bins3, weights = f*counts3, ec='blue', fc='none', lw=0.5,
histtype='step', label = '200 MeV')
axes[0].hist(bins4[:-1], bins4, weights = f*counts4,ec='green', fc='none', lw=0.5,
histtype='step', label = '230 MeV')
axes[0].ticklabel_format(style = 'sci', axis = 'y', scilimits = (0,0))
axes[0].yaxis.major.formatter._useMathText = True
axes[0].set_xscale('log')


#energy of neutrons detected
MIN, MAX = 0.01, 1000

axes[1].hist(Eko_1_100,bins  =  np.logspace(np.log10(MIN),  np.log10(MAX),  200),
ec='black', fc='none', lw=0.5, histtype='step')
axes[1].hist(Eko_1_160,  bins  =  np.logspace(np.log10(MIN),  np.log10(MAX),  200),
ec='red', fc='none', lw=0.5, histtype='step')
axes[1].hist(Eko_1_200,  bins  =  np.logspace(np.log10(MIN),  np.log10(MAX),  200),
ec='blue', fc='none', lw=0.5, histtype='step')
axes[1].hist(Eko_1_230,  bins  =  np.logspace(np.log10(MIN),  np.log10(MAX),  200),
ec='green', fc='none', lw=0.5, histtype='step')
axes[1].set_xscale('log')
axes[1].ticklabel_format(style = 'sci', axis = 'y', scilimits = (0,0))
axes[1].yaxis.major.formatter._useMathText = True
fig.text(0.08, 0.88, 'a', fontsize = 20)
fig.text(0.52, 0.88, 'b', fontsize = 20)
fig.text(0.5, 0.04, 'Kinetic energy [MeV]', ha='center', va='center')
fig.text(0.06,  0.5,  'Neutrons  from  '  +  str(primaries)  +  '  primary  protons',
ha='center', va='center', rotation='vertical')
fig.legend(loc = 'upper right')
plt.show()




#HISTOGRAM SHOWING PRODUCTION POSITIONS OF PRODUCED AND DETECTED NEUTRONS
# histogram: where the neutrons are produced
primaries = '1.2x10$^{9}$'

fig, axes = plt.subplots(nrows = 1, ncols = 2, figsize = (15,6))

(counts1, bins1) = np.histogram(Ox_100, 100)
(counts2, bins2) = np.histogram(Ox_160, 100)
(counts3, bins3) = np.histogram(Ox_200, 100)
(counts4, bins4) = np.histogram(Ox_230, 100)
#multiply by 1000
f = 1000
axes[0].hist(bins1[:-1],bins1,weights  =  f*counts1,  ec='black',  fc='none',  lw=0.5,
histtype='step')
axes[0].hist(bins2[:-1],  bins2,  weights  =  f*counts2,  ec='red',  fc='none',  lw=0.5,
histtype='step')
axes[0].hist(bins3[:-1],  bins3,  weights  =  f*counts3,  ec='blue',  fc='none',  lw=0.5,
histtype='step')
axes[0].hist(bins4[:-1], bins4, weights = f*counts4, ec='green', fc='none', lw=0.5,
histtype='step')
axes[0].ticklabel_format(style = 'sci', axis = 'y', scilimits = (0,0))
axes[0].yaxis.major.formatter._useMathText = True


#Add line indicating Bragg peak location
axes[0].axvline(x=bp100, color = 'k', linestyle = '--', lw=0.5, alpha = 0.5, label
= 'Bragg peak position')
axes[0].axvline(x=bp160, color= 'r', linestyle = '--', lw=0.5, alpha=0.5)
axes[0].axvline(x=bp200, color= 'b', linestyle = '--', lw=0.5, alpha=0.5)
axes[0].axvline(x=bp230, color= 'g', linestyle = '--', lw=0.5, alpha=0.5)
axes[0].set_xlim(0,40)


#Create histogram, depth in water
axes[1].hist(Ox_1_100, 500, ec='black', fc='none', lw=0.5, histtype='step', label =
'100 MeV')
axes[1].hist(Ox_1_160, 500, ec='red', fc='none', lw=0.5, histtype='step', label =
'160 MeV')
axes[1].hist(Ox_1_200, 500, ec='blue', fc='none', lw=0.5, histtype='step', label =
'200 MeV')
axes[1].hist(Ox_1_230, 500, ec='green', fc='none', lw=0.5, histtype='step', label =
'230 MeV')
axes[1].ticklabel_format(style = 'sci', axis = 'y', scilimits = (0,0))
axes[1].yaxis.major.formatter._useMathText = True
axes[1].set_xlim(0,40)

axes[1].axvline(x=bp100, color = 'k', linestyle = '--', lw=0.5, alpha=0.5)
axes[1].axvline(x=bp160, color= 'r', linestyle = '--', lw=0.5, alpha=0.5)
axes[1].axvline(x=bp200, color= 'b', linestyle = '--', lw=0.5, alpha=0.5)
axes[1].axvline(x=bp230, color= 'g', linestyle = '--', lw=0.5, alpha=0.5)

fig.text(0.08, 0.85, 'a', fontsize = 20)
fig.text(0.52, 0.85, 'b', fontsize = 20)
fig.text(0.5, 0.04, 'Depth in water [cm]', ha='center', va='center')
fig.text(0.06,  0.5,  'Neutrons  from  '  +  str(primaries)  +  '  primary  protons',
ha='center', va='center', rotation='vertical')
fig.legend(loc='upper right')
plt.show()



fig, axes = plt.subplots(nrows = 2, ncols = 2)
# Neutron detection as function of detector size: increasing height & width
nbneutron100 = []
nbneutron160 = []
nbneutron200 = []
nbneutron230 = []
wh100 = 2.5

for m in range(46):
    for a, b, c, d in zip(X_1_100, Y_1_100, X_2_100, Y_2_100):
        if (15 - wh100) < a < (15 + wh100) and (15 - wh100) < c < (15 + wh100) and
-wh100 < b < wh100 and -wh100 < d < wh100:
            nbneutron100.append(wh100*2)
    wh100 = wh100 + 0.5

wh160 = 2.5
for m in range(46):
    for e, f, g, h in zip(X_1_160, Y_1_160, X_2_160, Y_2_160):
        if (21 - wh160) < e < (21 + wh160) and (21 - wh160) < g < (21 + wh160) and
-wh160 < f < wh160 and -wh160 < h < wh160:
            nbneutron160.append(wh160*2)
    wh160 = wh160 + 0.5

wh200 = 2.5
for m in range(46):
    for i, j, k, l in zip(X_1_200, Y_1_200, X_2_200, Y_2_200):
        if (24 - wh200) < i < (24 + wh200) and (24 - wh200) < k < (24 + wh200) and
-wh200 < j < wh200 and -wh200 < l < wh200:
            nbneutron200.append(wh200*2)
    wh200 = wh200 + 0.5

wh230 = 2.5
for m in range(46):
    for n, o, p, q in zip(X_1_230, Y_1_230, X_2_230, Y_2_230):
        if (28 - wh230) < n < (28 + wh230) and (28 - wh230) < p < (28 + wh230) and
-wh230 < o < wh230 and -wh230 < q < wh230:
            nbneutron230.append(wh230*2)
    wh230 = wh230 + 0.5



n1, bins1, _ =axes[0,0].hist(nbneutron100, bins = np.arange(min(nbneutron100)-0.5,
max(nbneutron100)+0.5, 1), ec = 'red', fc='none', lw=0.5, histtype='step', align =
'left', label = 'All neutrons')
n2, bins2, _ =axes[0,1].hist(nbneutron160, bins = np.arange(min(nbneutron160)-0.5,
max(nbneutron160)+0.5, 1), ec = 'red', fc='none', lw=0.5, histtype='step', align =
'left')
n3, bins3, _ =axes[1,0].hist(nbneutron200, bins = np.arange(min(nbneutron200)-0.5,
max(nbneutron200)+0.5, 1), ec = 'red', fc='none', lw=0.5, histtype='step', align =
'left')
n4, bins4, _ =axes[1,1].hist(nbneutron230, bins = np.arange(min(nbneutron230)-0.5,
max(nbneutron230)+0.5, 1), ec = 'red', fc='none', lw=0.5, histtype='step', align =
'left')

#linear fit
bincenters1 = 0.5*(bins1[1:]+bins1[:-1])
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters4 = 0.5*(bins4[1:]+bins4[:-1])
```

```
coef1 = np.polyfit(bincenters1, n1,1)
coef2 = np.polyfit(bincenters2, n2,1)
coef3 = np.polyfit(bincenters3, n3,1)
coef4 = np.polyfit(bincenters4, n4,1)

poly1d_fn1 = np.poly1d(coef1)
poly1d_fn2 = np.poly1d(coef2)
poly1d_fn3 = np.poly1d(coef3)
poly1d_fn4 = np.poly1d(coef4)

axes[0,0].plot(bincenters1,      poly1d_fn1(bincenters1),      '--r',      lw=1.0,
color='black',alpha = 0.5, label = 'Linear fit')
axes[0,1].plot(bincenters2,      poly1d_fn2(bincenters2),      '--r',      lw=1.0,
color='black',alpha = 0.5)
axes[1,0].plot(bincenters3,      poly1d_fn3(bincenters3),      '--r',      lw=1.0,
color='black',alpha = 0.5)
axes[1,1].plot(bincenters4,      poly1d_fn4(bincenters4),      '--r',      lw=1.0,
color='black',alpha = 0.5)

axes[0,0].ticklabel_format(style = 'sci', axis = 'y', scilimits = (0,0))
axes[0,0].yaxis.major.formatter._useMathText = True


# Neutron detection as function of detector size: increasing height & width, only
last half before bragg peak
nbneutronbragg100 = []
nbneutronbragg160 = []
nbneutronbragg200 = []
nbneutronbragg230 = []
whb100 = 2.5

for m in range(46):
    for a, b, c, d, v in zip(X_1_100, Y_1_100, X_2_100, Y_2_100, Ox_1_100):
        if (16 - whb100) < a < (16 + whb100) and (16 - whb100) < c < (16 + whb100)
and -whb100 < b < whb100 and -whb100 < d < whb100 and 3.8 < v < 7.6:
            nbneutronbragg100.append(whb100*2)
    whb100 = whb100 + 0.5

whb160 = 2.5
for m in range(46):
    for e, f, g, h, z in zip(X_1_160, Y_1_160, X_2_160, Y_2_160, Ox_1_160):
        if (24 - whb160) < e < (24 + whb160) and (24 - whb160) < g < (24 + whb160)
and -whb160 < f < whb160 and -whb160 < h < whb160 and 8.8 < z < 17.6:
```

```
            nbneutronbragg160.append(whb160*2)
    whb160 = whb160 + 0.5

whb200 = 2.5
for m in range(46):
    for i, j, k, l, u in zip(X_1_200, Y_1_200, X_2_200, Y_2_200, Ox_1_200):
        if (31 - whb200) < i < (31 + whb200) and (31 - whb200) < k < (31 + whb200)
and -whb200 < j < whb200 and -whb200 < l < whb200 and 12.95 < u < 25.9:
            nbneutronbragg200.append(whb200*2)
    whb200 = whb200 + 0.5

whb230 = 2.5
for m in range(46):
    for n, o, p, q, y in zip(X_1_230, Y_1_230, X_2_230, Y_2_230, Ox_1_230):
        if (36 - whb230) < n < (36 + whb230) and (36 - whb230) < p < (36 + whb230)
and -whb230 < o < whb230 and -whb230 < q < whb230 and 16.45 < y < 32.9:
            nbneutronbragg230.append(whb230*2)
    whb230 = whb230 + 0.5



n1,     bins1,     _     =axes[0,0].hist(nbneutronbragg100,     bins     =
np.arange(min(nbneutronbragg100)-0.5, max(nbneutronbragg100)+0.5, 1), ec = 'blue',
fc='none', lw=0.5, histtype='step', align = 'left', label ='Last half')
n2,     bins2,     _     =axes[0,1].hist(nbneutronbragg160,     bins     =
np.arange(min(nbneutronbragg160)-0.5,  max(nbneutronbragg160)+0.5,  1),  ec='blue',
fc='none', lw=0.5, histtype='step', align = 'left')
n3,     bins3,     _     =axes[1,0].hist(nbneutronbragg200,     bins     =
np.arange(min(nbneutronbragg200)-0.5,  max(nbneutronbragg200)+0.5,  1),  ec='blue',
fc='none', lw=0.5, histtype='step', align = 'left')
n4,     bins4,     _     =axes[1,1].hist(nbneutronbragg230,     bins     =
np.arange(min(nbneutronbragg230)-0.5,  max(nbneutronbragg230)+0.5,  1),  ec='blue',
fc='none', lw=0.5, histtype='step', align = 'left')


#linear fit
bincenters1 = 0.5*(bins1[1:]+bins1[:-1])
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters4 = 0.5*(bins4[1:]+bins4[:-1])

coef1 = np.polyfit(bincenters1, n1,1)
coef2 = np.polyfit(bincenters2, n2,1)
```

```
coef3 = np.polyfit(bincenters3, n3,1)
coef4 = np.polyfit(bincenters4, n4,1)

poly1d_fn1 = np.poly1d(coef1)
poly1d_fn2 = np.poly1d(coef2)
poly1d_fn3 = np.poly1d(coef3)
poly1d_fn4 = np.poly1d(coef4)

axes[0,0].plot(bincenters1,  poly1d_fn1(bincenters1),  '--r',  lw=1.0,  color  =
'black',alpha = 0.5)
axes[0,1].plot(bincenters2,  poly1d_fn2(bincenters2),  '--r',  lw=1.0,  color  =
'black',alpha = 0.5)
axes[1,0].plot(bincenters3,  poly1d_fn3(bincenters3),  '--r',  lw=1.0,  color  =
'black',alpha = 0.5)
axes[1,1].plot(bincenters4,  poly1d_fn4(bincenters4),  '--r',  lw=1.0,  color  =
'black',alpha = 0.5)

axes[0,1].ticklabel_format(style = 'sci', axis = 'y', scilimits = (0,0))
axes[0,1].yaxis.major.formatter._useMathText = True


# Neutron detection as function of detector size: increasing height & width, only
last quarter before bragg peak


nbneutronbragg100 = []
nbneutronbragg160 = []
nbneutronbragg200 = []
nbneutronbragg230 = []
whb100 = 2.5

for m in range(46):
    for a, b, c, d, v in zip(X_1_100, Y_1_100, X_2_100, Y_2_100, Ox_1_100):
        if (16 - whb100) < a < (16 + whb100) and (16 - whb100) < c < (16 + whb100)
and -whb100 < b < whb100 and -whb100 < d < whb100 and 5.7 < v < 7.6:
            nbneutronbragg100.append(whb100*2)
    whb100 = whb100 + 0.5

whb160 = 2.5
for m in range(46):
    for e, f, g, h, z in zip(X_1_160, Y_1_160, X_2_160, Y_2_160, Ox_1_160):
        if (24 - whb160) < e < (24 + whb160) and (24 - whb160) < g < (24 + whb160)
and -whb160 < f < whb160 and -whb160 < h < whb160 and 12.6 < z < 17.6:
```

```
            nbneutronbragg160.append(whb160*2)
    whb160 = whb160 + 0.5

whb200 = 2.5
for m in range(46):
    for i, j, k, l, u in zip(X_1_200, Y_1_200, X_2_200, Y_2_200, Ox_1_200):
        if (31 - whb200) < i < (31 + whb200) and (31 - whb200) < k < (31 + whb200)
and -whb200 < j < whb200 and -whb200 < l < whb200 and 19.425 < u < 25.9:
            nbneutronbragg200.append(whb200*2)
    whb200 = whb200 + 0.5

whb230 = 2.5
for m in range(46):
    for n, o, p, q, y in zip(X_1_230, Y_1_230, X_2_230, Y_2_230, Ox_1_230):
        if (36 - whb230) < n < (36 + whb230) and (36 - whb230) < p < (36 + whb230)
and -whb230 < o < whb230 and -whb230 < q < whb230 and 24.675 < y < 32.9:
            nbneutronbragg230.append(whb230*2)
    whb230 = whb230 + 0.5



n1, bins1, _ =axes[0,0].hist(nbneutronbragg100, bins =
np.arange(min(nbneutronbragg100)-0.5, max(nbneutronbragg100)+0.5, 1), ec = 'green',
fc='none', lw=0.5, histtype='step', align = 'left', label = 'Last quarter')
n2, bins2, _ =axes[0,1].hist(nbneutronbragg160, bins =
np.arange(min(nbneutronbragg160)-0.5, max(nbneutronbragg160)+0.5, 1), ec='green',
fc='none', lw=0.5, histtype='step', align = 'left')
n3, bins3, _ =axes[1,0].hist(nbneutronbragg200, bins =
np.arange(min(nbneutronbragg200)-0.5, max(nbneutronbragg200)+0.5, 1), ec='green',
fc='none', lw=0.5, histtype='step', align = 'left')
n4, bins4, _ =axes[1,1].hist(nbneutronbragg230, bins =
np.arange(min(nbneutronbragg230)-0.5, max(nbneutronbragg230)+0.5, 1), ec='green',
fc='none', lw=0.5, histtype='step', align = 'left')

#linear fit
bincenters1 = 0.5*(bins1[1:]+bins1[:-1])
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters4 = 0.5*(bins4[1:]+bins4[:-1])

coef1 = np.polyfit(bincenters1, n1,1)
coef2 = np.polyfit(bincenters2, n2,1)
coef3 = np.polyfit(bincenters3, n3,1)
```

```
coef4 = np.polyfit(bincenters4, n4,1)

poly1d_fn1 = np.poly1d(coef1)
poly1d_fn2 = np.poly1d(coef2)
poly1d_fn3 = np.poly1d(coef3)
poly1d_fn4 = np.poly1d(coef4)

axes[0,0].plot(bincenters1,  poly1d_fn1(bincenters1),  '--r',  lw=1.0,  color  =
'black',alpha = 0.5)
axes[0,1].plot(bincenters2,  poly1d_fn2(bincenters2),  '--r',  lw=1.0,  color  =
'black',alpha = 0.5)
axes[1,0].plot(bincenters3,  poly1d_fn3(bincenters3),  '--r',  lw=1.0,  color  =
'black',alpha = 0.5)
axes[1,1].plot(bincenters4,  poly1d_fn4(bincenters4),  '--r',  lw=1.0,  color  =
'black',alpha = 0.5)

axes[1,0].ticklabel_format(style = 'sci', axis = 'y', scilimits = (0,0))
axes[1,0].yaxis.major.formatter._useMathText = True
axes[1,1].ticklabel_format(style = 'sci', axis = 'y', scilimits = (0,0))
axes[1,1].yaxis.major.formatter._useMathText = True

#subplot titles
axes[0,0].set_title('100 MeV', fontsize = '14')
axes[0,1].set_title('160 MeV', fontsize = '14')
axes[1,0].set_title('200 MeV', fontsize = '14')
axes[1,1].set_title('230 MeV', fontsize = '14')

#y limits
axes[0,0].set_ylim(0,16000)
axes[0,1].set_ylim(0,63000)
axes[1,0].set_ylim(0,120000)
axes[1,1].set_ylim(0,160000)



fig.text(0.5, 0.04, 'Side length of the detector [cm]', ha='center', va='center')
fig.text(0.06,  0.5,  'Neutrons  from  '  +  str(primaries)  +  '  primary  protons',
ha='center', va='center', rotation='vertical')
fig.subplots_adjust(right=0.8)
fig.legend(loc = 'upper right')
plt.show()
```

```
# Set font size and tick size for all plots
plt.rcParams.update({'font.size': 20})
plt.rc('xtick',labelsize=20)
plt.rc('ytick',labelsize=20)


#Create histogram, location on detector planes x-direction
fig, axes = plt.subplots(nrows = 2, ncols = 2)


n1, bins1, _ =axes[0,0].hist(X_1_100, bins = range(-30,163, 3), ec='blue', fc='none',
lw=1.0, histtype='step', label='First tracking detector')
n11,  bins11,  _  =axes[0,0].hist(X_2_100,  bins  =  range(-30,163,  3),  ec='red',
fc='none', lw=1.0, histtype='step', label='Second tracking detector')

n2, bins2, _ =axes[0,1].hist(X_1_160, bins = range(-30,163, 3), ec='blue', fc='none',
lw=1.0, histtype='step')
n22,  bins22,  _  =axes[0,1].hist(X_2_160,  bins  =  range(-30,163,  3),  ec='red',
fc='none', lw=1.0, histtype='step')

n3, bins3, _ =axes[1,0].hist(X_1_200, bins = range(-30,163, 3), ec='blue', fc='none',
lw=1.0, histtype='step')
n33,  bins33,  _  =axes[1,0].hist(X_2_200,  bins  =  range(-30,163,  3),  ec='red',
fc='none', lw=1.0, histtype='step')

n4, bins4, _ =axes[1,1].hist(X_1_230, bins = range(-30,163, 3), ec='blue', fc='none',
lw=1.0, histtype='step')
n44,  bins44,  _  =axes[1,1].hist(X_2_230,bins  =  range(-30,163,  3),  ec='red',
fc='none', lw=1.0, histtype='step')

#Magnitude of statistical uncertainty
menStd1 = np.sqrt(n1)
menStd2 = np.sqrt(n2)
menStd3 = np.sqrt(n3)
menStd4 = np.sqrt(n4)
menStd11 = np.sqrt(n11)
menStd22 = np.sqrt(n22)
menStd33 = np.sqrt(n33)
menStd44 = np.sqrt(n44)

#Center of bins
bincenters1 = 0.5*(bins1[1:]+bins1[:-1])
```

```
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters4 = 0.5*(bins4[1:]+bins4[:-1])
bincenters11 = 0.5*(bins11[1:]+bins11[:-1])
bincenters22 = 0.5*(bins22[1:]+bins22[:-1])
bincenters33 = 0.5*(bins33[1:]+bins33[:-1])
bincenters44 = 0.5*(bins44[1:]+bins44[:-1])


#Plot error bars
axes[0,0].errorbar(bincenters1, n1, yerr = menStd1, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[0,1].errorbar(bincenters2, n2, yerr = menStd2, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[1,0].errorbar(bincenters3, n3, yerr = menStd3, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[1,1].errorbar(bincenters4, n4, yerr = menStd4, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[0,0].errorbar(bincenters11,  n11,  yerr  =  menStd11,  fmt  =  'none',  ecolor  =
'black', elinewidth = 1.0)
axes[0,1].errorbar(bincenters22,  n22,  yerr  =  menStd22,  fmt  =  'none',  ecolor  =
'black', elinewidth = 1.0)
axes[1,0].errorbar(bincenters33,  n33,  yerr  =  menStd33,  fmt  =  'none',  ecolor  =
'black', elinewidth = 1.0)
axes[1,1].errorbar(bincenters44,  n44,  yerr  =  menStd44,  fmt  =  'none',  ecolor  =
'black', elinewidth = 1.0)

#Set subplot title and Bragg peak location
axes[0,0].set_title('100 MeV')
axes[0,0].axvline(x=bp100, color = 'k', linestyle = '--', alpha=0.5, label = 'Bragg
peak position')
axes[0,1].set_title('160 MeV')
axes[0,1].axvline(x=bp160, color= 'k', linestyle = '--', alpha=0.5)
axes[1,0].set_title('200 MeV')
axes[1,0].axvline(x=bp200, color= 'k', linestyle = '--', alpha=0.5)
axes[1,1].set_title('230 MeV')
axes[1,1].axvline(x=bp230, color= 'k', linestyle = '--', alpha=0.5)

fig.text(0.5, 0.04,'Location on tracking detector in x-direction [cm]', ha='center',
va='center')
fig.text(0.06,  0.5,'Neutrons  from  '  +  str(primaries)  +  '  primary  protons'  ,
ha='center', va='center', rotation='vertical')
fig.legend(loc='upper right')
fig.subplots_adjust(right=0.8)
plt.show()



#Create scatter plot
Ox_1000_100 = []
Oy_1000_100 = []
Eko_1000_100 = []

#Gather 1000 neutrons for plotting
ind = 0
for u, o, j in zip(Ox_1_100,Oy_1_100, Eko_1_100):
    if ind < 1000:
        if u < 40 and o < 5:
            Ox_1000_100.append(u)
            Oy_1000_100.append(o)
            Eko_1000_100.append(j)
            ind = ind + 1
    else:
        break


Ox_1000_160 = []
Oy_1000_160 = []
Eko_1000_160 = []

ind = 0
for u, o, j in zip(Ox_1_160,Oy_1_160, Eko_1_160):
    if ind < 1000:
        if u < 40 and o < 5:
            Ox_1000_160.append(u)
            Oy_1000_160.append(o)
            Eko_1000_160.append(j)
            ind = ind + 1
    else:
        break

Ox_1000_200 = []
Oy_1000_200 = []
Eko_1000_200 = []

ind = 0
```

```
for u, o, j in zip(Ox_1_200,Oy_1_200, Eko_1_200):
    if ind < 1000:
        if u < 40 and o < 5:
            Ox_1000_200.append(u)
            Oy_1000_200.append(o)
            Eko_1000_200.append(j)
            ind = ind + 1
    else:
        break


Ox_1000_230 = []
Oy_1000_230 = []
Eko_1000_230 = []

ind = 0
for u, o, j in zip(Ox_1_230,Oy_1_230, Eko_1_230):
    if ind < 1000:
        if u < 40 and o < 5:
            Ox_1000_230.append(u)
            Oy_1000_230.append(o)
            Eko_1000_230.append(j)
            ind = ind + 1
    else:
        break


fig, axes = plt.subplots(nrows = 2, ncols = 2)
colors100 = Eko_1000_100
axes[0,0].scatter(Ox_1000_100, Oy_1000_100, c=colors100, s = 7, alpha=0.5)
axes[0,0].set_title('100 MeV')
axes[0,0].set_ylim(-4,4)

colors160 = Eko_1000_160
axes[0,1].scatter(Ox_1000_160, Oy_1000_160, c=colors160, s = 7, alpha=0.5)
axes[0,1].set_title('160 MeV')
axes[0,1].set_ylim(-4,4)

colors200 = Eko_1000_200
axes[1,0].scatter(Ox_1000_200, Oy_1000_200, c=colors200, s = 7, alpha=0.5)
axes[1,0].set_title('200 MeV')
axes[1,0].set_ylim(-4,4)
```

```
colors230 = Eko_1000_230
img = axes[1,1].scatter(Ox_1000_230, Oy_1000_230, c=colors230, s = 7, alpha=0.5)
axes[1,1].set_title('230 MeV')
axes[1,1].set_ylim(-4,4)

fig.text(0.5, 0.04, 'Depth in water [cm]', ha='center', va='center')
fig.text(0.06, 0.5, 'Distance from primary beam axis [cm]', ha='center', va='center',
rotation='vertical')
fig.subplots_adjust(right=0.8)
cbar_ax = fig.add_axes([0.85, 0.15, 0.05, 0.7])
cbar = fig.colorbar(img, cax=cbar_ax)
cbar.set_label('Neutron energy [MeV]')
plt.show()




# Create 2D histogram, scatter plot
fig, axes = plt.subplots(nrows = 2, ncols = 2, sharey=True)

img1 = axes[0,0].hist2d(Ox_1_100, Oy_1_100, bins=(1000, 1500), cmap=plt.cm.jet)
axes[0,0].set_title('100 MeV')
axes[0,0].set_ylim(-4,4)
axes[0,0].set_xlim(0,30)

img2 = axes[0,1].hist2d(Ox_1_160, Oy_1_160, bins=(1000, 1500), cmap=plt.cm.jet)
axes[0,1].set_title('160 MeV')
axes[0,1].set_ylim(-3,3)
axes[0,1].set_xlim(0,30)

img3 = axes[1,0].hist2d(Ox_1_200, Oy_1_200, bins=(1000, 1500), cmap=plt.cm.jet)
axes[1,0].set_title('200 MeV')
axes[1,0].set_ylim(-3,3)
axes[1,0].set_xlim(0,30)

img1 = axes[1,1].hist2d(Ox_1_230, Oy_1_230, bins=(1000, 1500), cmap=plt.cm.jet)
axes[1,1].set_title('230 MeV')
axes[1,1].set_ylim(-3,3)
axes[1,1].set_xlim(0,30)

fig.text(0.5, 0.04, 'Depth in water [cm]', ha='center', va='center')
fig.text(0.06, 0.5, 'Distance from primary beam axis [cm]', ha='center', va='center',
rotation='vertical')
```

```
fig.subplots_adjust(right=0.8)
cbar_ax = fig.add_axes([0.85, 0.15, 0.05, 0.7])
cbar = fig.colorbar(img1[3], cax=cbar_ax)
cbar.set_label('Number of neutrons')
plt.show()




# Neutron detection as a function of detector position: all neutrons
nbneu100 = []
nbneu160 = []
nbneu200 = []
nbneu230 = []
x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_100 ,X_2_100 , Y_1_100 ,Y_2_100):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu100.append(x)
    low = low + 1
    high = high + 1
    x = x+1

x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_160 ,X_2_160 , Y_1_160 ,Y_2_160):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu160.append(x)
    low = low + 1
    high = high + 1
    x = x+1


x = -10
low = -20
high = 0
h = 0
```

```
for h in range(160):
    for b, n, k, a in zip(X_1_200 ,X_2_200 , Y_1_200 ,Y_2_200):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu200.append(x)
    low = low + 1
    high = high + 1
    x = x+1



x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_230 ,X_2_230 , Y_1_230 ,Y_2_230):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu230.append(x)
    low = low + 1
    high = high + 1
    x = x+1

fig, ax = plt.subplots()



n1, bins1, _ = plt.hist(nbneu100, bins = np.arange(min(nbneu100)-0.5,
max(nbneu100)+0.5, 1),  ec='black', fc='none', lw=1.0, histtype='step', label = '100
MeV')

n2, bins2, _ = plt.hist(nbneu160, bins = np.arange(min(nbneu160)-0.5,
max(nbneu160)+0.5, 1),  ec='red', fc='none', lw=1.0, histtype='step',  label = '160
MeV')

n3, bins3, _ = plt.hist(nbneu200, bins = np.arange(min(nbneu200)-0.5,
max(nbneu200)+0.5, 1),  ec='blue', fc='none', lw=1.0, histtype='step', label = '200
MeV')

n4, bins4, _ = plt.hist(nbneu230, bins = np.arange(min(nbneu230)-0.5,
max(nbneu230)+0.5, 1),  ec='green', fc='none', lw=1.0, histtype='step', label = '230
MeV')



menStd1 = np.sqrt(n1)
menStd2 = np.sqrt(n2)
```

```
menStd3 = np.sqrt(n3)
menStd4 = np.sqrt(n4)


bincenters1 = 0.5*(bins1[1:]+bins1[:-1])
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters4 = 0.5*(bins4[1:]+bins4[:-1])




plt.errorbar(bincenters1, n1, yerr = menStd1, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
plt.errorbar(bincenters2, n2, yerr = menStd2, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
plt.errorbar(bincenters3, n3, yerr = menStd3, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
plt.errorbar(bincenters4, n4, yerr = menStd4, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)


plt.xlabel('Center point of detector in x-direction [cm]')
plt.ylabel('Number of neutrons from ' + str(primaries) + ' primary protons')
plt.axvline(x=bp100, color = 'k', linestyle = '--', alpha=0.5, label = 'Bragg peak
position')
plt.axvline(x=bp160, color= 'r', linestyle = '--', alpha=0.5)
plt.axvline(x=bp200, color= 'b', linestyle = '--', alpha=0.5)
plt.axvline(x=bp230, color= 'g', linestyle = '--', alpha=0.5)
plt.legend(loc='upper right')
plt.show()




# Histogram, sum of 160 and 200 MeV

nbneu160 = []
nbneu200 = []


x = -10
```

```
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_160 ,X_2_160 , Y_1_160 ,Y_2_160):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu160.append(x)
    low = low + 1
    high = high + 1
    x = x+1


x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_200 ,X_2_200 , Y_1_200 ,Y_2_200):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu200.append(x)
    low = low + 1
    high = high + 1
    x = x+1


fig, ax = plt.subplots()



n2, bins2, _ = plt.hist(nbneu160, bins = np.arange(min(nbneu160)-0.5,
max(nbneu160)+0.5, 1),  ec='black', fc='none', lw=1.0, histtype='step', label = '160
MeV')

n3, bins3, _ = plt.hist(nbneu200, bins = np.arange(min(nbneu200)-0.5,
max(nbneu200)+0.5, 1),  ec='red', fc='none', lw=1.0, histtype='step', label = '200
MeV')


tot = nbneu160
tot.extend(nbneu200)
n5, bins5, _ = plt.hist(tot, bins = np.arange(min(tot)-0.5, max(tot)+0.5, 1),
ec='blue', fc='none', lw=1.0, histtype='step', label = 'Sum of 160 & 200 MeV')
```

```
menStd2 = np.sqrt(n2)
menStd3 = np.sqrt(n3)
menStd5 = np.sqrt(n5)


bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters5 = 0.5*(bins5[1:]+bins5[:-1])


plt.errorbar(bincenters2, n2, yerr = menStd2, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
plt.errorbar(bincenters3, n3, yerr = menStd3, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
plt.errorbar(bincenters5, n5, yerr = menStd5, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)

plt.xlabel('Center point of detector in x-direction [cm]')
plt.ylabel('Number of neutrons from ' + str(primaries) + ' primary protons')

plt.axvline(x=bp160, color= 'k', linestyle = '--', alpha=0.5, label = 'Bragg peak
position')
plt.axvline(x=bp200, color= 'r', linestyle = '--', alpha=0.5)

plt.legend(loc='upper right')
plt.show()




#MULTIPLOT WITH THE DIFFERENT ANALYSES OF NEUTRON DETECTION RATE
# Neutron detection as a function of detector position: all neutrons
nbneu100 = []
nbneu160 = []
nbneu200 = []
nbneu230 = []
x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_100 ,X_2_100 , Y_1_100 ,Y_2_100):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu100.append(x)
    low = low + 1
    high = high + 1
    x = x+1

x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_160 ,X_2_160 , Y_1_160 ,Y_2_160):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu160.append(x)
    low = low + 1
    high = high + 1
    x = x+1



x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_200 ,X_2_200 , Y_1_200 ,Y_2_200):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu200.append(x)
    low = low + 1
    high = high + 1
    x = x+1



x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a in zip(X_1_230 ,X_2_230 , Y_1_230 ,Y_2_230):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu230.append(x)
    low = low + 1
    high = high + 1
    x = x+1


fig, axes = plt.subplots(nrows = 2, ncols = 2)

n1, bins1, _ = axes[0,0].hist(nbneu100, bins = np.arange(min(nbneu100)-0.5, max(nbneu100)+0.5, 1),  ec='red', fc='none', lw=1.0, histtype='step', label = 'All neutrons')

n2, bins2, _ = axes[0,1].hist(nbneu160, bins = np.arange(min(nbneu160)-0.5, max(nbneu160)+0.5, 1),  ec='red', fc='none', lw=1.0, histtype='step')

n3, bins3, _ = axes[1,0].hist(nbneu200, bins = np.arange(min(nbneu200)-0.5, max(nbneu200)+0.5, 1),  ec='red', fc='none', lw=1.0, histtype='step')

n4, bins4, _ = axes[1,1].hist(nbneu230, bins = np.arange(min(nbneu230)-0.5, max(nbneu230)+0.5, 1),  ec='red', fc='none', lw=1.0, histtype='step')



axes[0,0].set_title('100 MeV')
axes[0,1].set_title('160 MeV')
axes[1,0].set_title('200 MeV')
axes[1,1].set_title('230 MeV')


menStd1 = np.sqrt(n1)
menStd2 = np.sqrt(n2)
menStd3 = np.sqrt(n3)
menStd4 = np.sqrt(n4)

bincenters1 = 0.5*(bins1[1:]+bins1[:-1])
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters4 = 0.5*(bins4[1:]+bins4[:-1])


axes[0,0].errorbar(bincenters1, n1, yerr = menStd1, fmt = 'none', ecolor = 'black', elinewidth = 1.0)
axes[0,1].errorbar(bincenters2, n2, yerr = menStd2, fmt = 'none', ecolor = 'black', elinewidth = 1.0)
axes[1,0].errorbar(bincenters3, n3, yerr = menStd3, fmt = 'none', ecolor = 'black', elinewidth = 1.0)
```

```
axes[1,1].errorbar(bincenters4, n4, yerr = menStd4, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)

axes[0,0].axvline(x=bp100, color = 'k', linestyle = '--', alpha=0.5, label = 'Bragg
peak position')
axes[0,1].axvline(x=bp160, color= 'k', linestyle = '--', alpha=0.5)
axes[1,0].axvline(x=bp200, color= 'k', linestyle = '--', alpha=0.5)
axes[1,1].axvline(x=bp230, color= 'k', linestyle = '--', alpha=0.5)


fig.text(0.5,  0.04,'Center  point  of  detector  in  x-direction  [cm]',  ha='center',
va='center')
fig.text(0.06,  0.5,'Neutrons  from  '  +  str(primaries)  +  '  primary  protons'  ,
ha='center', va='center', rotation='vertical')







# Neutron detection as a function of detector position: only neutrons from last half
before bragg peak
nbneubragg100 = []
nbneubragg160 = []
nbneubragg200 = []
nbneubragg230 = []
x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a, u in zip(X_1_100 ,X_2_100 , Y_1_100 ,Y_2_100, Ox_1_100):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10 and
3.8 < u < 7.6:
            nbneubragg100.append(x)
    low = low + 1
    high = high + 1
    x = x+1

x = -10
low = -20
```

```
high = 0
h = 0
for h in range(160):
    for b, n, k, a, u in zip(X_1_160 ,X_2_160 , Y_1_160 ,Y_2_160, Ox_1_160):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10 and
8.8 < u < 17.6:
            nbneubragg160.append(x)
    low = low + 1
    high = high + 1
    x = x+1


x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a, u in zip(X_1_200 ,X_2_200 , Y_1_200 ,Y_2_200, Ox_1_200):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10 and
12.95 < u < 25.9:
            nbneubragg200.append(x)
    low = low + 1
    high = high + 1
    x = x+1


x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a, u in zip(X_1_230 ,X_2_230 , Y_1_230 ,Y_2_230, Ox_1_230):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10 and
16.45 < u < 32.9:
            nbneubragg230.append(x)
    low = low + 1
    high = high + 1
    x = x+1

n1, bins1, _ = axes[0,0].hist(nbneubragg100, bins = np.arange(min(nbneubragg100)-
0.5, max(nbneubragg100)+0.5, 1),  ec='blue', fc='none', lw=1.0, histtype='step',
label = 'Last half')
```

```
n2, bins2, _ = axes[0,1].hist(nbneubragg160, bins = np.arange(min(nbneubragg160)-
0.5, max(nbneubragg160)+0.5, 1),  ec='blue', fc='none', lw=1.0, histtype='step')

n3, bins3, _ = axes[1,0].hist(nbneubragg200, bins = np.arange(min(nbneubragg200)-
0.5, max(nbneubragg200)+0.5, 1),  ec='blue', fc='none', lw=1.0, histtype='step')

n4, bins4, _ = axes[1,1].hist(nbneubragg230, bins = np.arange(min(nbneubragg230)-
0.5, max(nbneubragg230)+0.5, 1),  ec='blue', fc='none', lw=1.0, histtype='step')


menStd1 = np.sqrt(n1)
menStd2 = np.sqrt(n2)
menStd3 = np.sqrt(n3)
menStd4 = np.sqrt(n4)


bincenters1 = 0.5*(bins1[1:]+bins1[:-1])
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters4 = 0.5*(bins4[1:]+bins4[:-1])


axes[0,0].errorbar(bincenters1, n1, yerr = menStd1, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[0,1].errorbar(bincenters2, n2, yerr = menStd2, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[1,0].errorbar(bincenters3, n3, yerr = menStd3, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[1,1].errorbar(bincenters4, n4, yerr = menStd4, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)




# Neutron detection as a function of detector position: only neutrons from last
quarter before bragg peak
nbneubragg100 = []
nbneubragg160 = []
nbneubragg200 = []
nbneubragg230 = []
x = -10
low = -20
```

```
high = 0
h = 0
for h in range(160):
    for b, n, k, a, u in zip(X_1_100 ,X_2_100 , Y_1_100 ,Y_2_100, Ox_1_100):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10 and
5.7 < u < 7.6:
            nbneubragg100.append(x)
    low = low + 1
    high = high + 1
    x = x+1

x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a, u in zip(X_1_160 ,X_2_160 , Y_1_160 ,Y_2_160, Ox_1_160):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10 and
13.2 < u < 17.6:
            nbneubragg160.append(x)
    low = low + 1
    high = high + 1
    x = x+1


x = -10
low = -20
high = 0
h = 0
for h in range(160):
    for b, n, k, a, u in zip(X_1_200 ,X_2_200 , Y_1_200 ,Y_2_200, Ox_1_200):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10 and
19.425 < u < 25.9:
            nbneubragg200.append(x)
    low = low + 1
    high = high + 1
    x = x+1


x = -10
low = -20
high = 0
h = 0
```

```
for h in range(160):
    for b, n, k, a, u in zip(X_1_230 ,X_2_230 , Y_1_230 ,Y_2_230, Ox_1_230):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10 and
24.675 < u < 32.9:
            nbneubragg230.append(x)

    low = low + 1
    high = high + 1
    x = x+1


n1, bins1, _ = axes[0,0].hist(nbneubragg100, bins = np.arange(min(nbneubragg100)-
0.5, max(nbneubragg100)+0.5, 1),  ec='green', fc='none', lw=1.0, histtype='step',
label = 'Last quarter')

n2, bins2, _ = axes[0,1].hist(nbneubragg160, bins = np.arange(min(nbneubragg160)-
0.5, max(nbneubragg160)+0.5, 1),  ec='green', fc='none', lw=1.0, histtype='step')

n3, bins3, _ = axes[1,0].hist(nbneubragg200, bins = np.arange(min(nbneubragg200)-
0.5, max(nbneubragg200)+0.5, 1),  ec='green', fc='none', lw=1.0, histtype='step')

n4, bins4, _ = axes[1,1].hist(nbneubragg230, bins = np.arange(min(nbneubragg230)-
0.5, max(nbneubragg230)+0.5, 1),  ec='green', fc='none', lw=1.0, histtype='step')


menStd1 = np.sqrt(n1)
menStd2 = np.sqrt(n2)
menStd3 = np.sqrt(n3)
menStd4 = np.sqrt(n4)

bincenters1 = 0.5*(bins1[1:]+bins1[:-1])
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
bincenters3 = 0.5*(bins3[1:]+bins3[:-1])
bincenters4 = 0.5*(bins4[1:]+bins4[:-1])

axes[0,0].errorbar(bincenters1, n1, yerr = menStd1, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[0,1].errorbar(bincenters2, n2, yerr = menStd2, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[1,0].errorbar(bincenters3, n3, yerr = menStd3, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
axes[1,1].errorbar(bincenters4, n4, yerr = menStd4, fmt = 'none', ecolor = 'black',
elinewidth = 1.0)
```

```
fig.legend(loc='upper right')
plt.show()
```

Maxdetrate.py

Script used for creating Figure 25, showing the maximum neutron detection rate for the three analyses done in this thesis.

```
#Create plot summarizing the positions of maximum detection rate
import numpy as np
import matplotlib.pyplot as plt

#lists containing positions of maximum detection rate
energies = [100, 160, 200, 230]
All = [15, 20.5, 24.5, 28]
half = [16, 24, 31, 35.5]
quarter = [17, 25.5, 33, 39]
bragg = [7.6, 17.6, 25.7, 32.6]



#Create a linear fit
coef = np.polyfit(energies,All,1)
coef2 = np.polyfit(energies,half,1)
coef3 = np.polyfit(energies,quarter,1)
coef4 = np.polyfit(energies,bragg,1)

poly1d_fn = np.poly1d(coef)
poly1d_fn2 = np.poly1d(coef2)
poly1d_fn3 = np.poly1d(coef3)
poly1d_fn4 = np.poly1d(coef4)



#ad errorbars
errorAll = [2, 1.5, 1.5, 2]
errorHalf = [3, 2, 2, 1.5]
errorQuarter = [4, 1.5, 2, 2]

plt.errorbar(energies, All, yerr = errorAll, fmt = 'none', ecolor = 'red', elinewidth
= 0.5)
plt.errorbar(energies,  half,  yerr  =  errorHalf,  fmt  =  'none',  ecolor  =  'blue',
elinewidth = 0.5)
plt.errorbar(energies, quarter, yerr = errorQuarter, fmt = 'none', ecolor = 'green',
elinewidth = 0.5)



#plot points of maximum detection rate and corresponding linear fit
```

```
plt.plot(energies, All, 'ro', label = 'All neutrons')
plt.plot(energies, poly1d_fn(energies), '--r', lw=1.0, alpha = 0.5)

plt.plot(energies, half, 'bs', label = 'Last half')
plt.plot(energies, poly1d_fn2(energies), '--b', lw=1.0, alpha = 0.5)

plt.plot(energies, quarter, 'g^', label = 'Last quarter')
plt.plot(energies, poly1d_fn3(energies), '--g', lw=1.0, alpha = 0.5)

plt.plot(energies, bragg, 'kH', label = 'Bragg peak')
plt.plot(energies, poly1d_fn4(energies), '--k', lw=1.0, alpha = 0.5)


#labels and legend
plt.xlabel('Energy [MeV]')
plt.ylabel('Position in depth direction [cm]')
plt.legend(loc = 'upper left')
plt.show()
```

Patientplan.py

Script used for creating all patient plan plots in this thesis.

```
#PATIENT TREATMENT PLAN PLOTS

import numpy as np
import matplotlib.pyplot as plt



plt.rcParams.update({'font.size': 12})
plt.rc('xtick',labelsize=12)
plt.rc('ytick',labelsize=12)



#IMPORT DATA, PRODUCED NEUTRONS
#create data lists
Ox = []
Oy = []
Oz = []
Ocosx = []
Ocosy = []
Ocosz = []
Eko = []

import csv

with open('zzProstHUH37_all60_xyz_origin_all_neutrons.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    k = 0
    for row in reader:
    temp = row[0].split()
       Ox.append(float(temp[0]))
        Oy.append(float(temp[1]))
        Oz.append(float(temp[2]))
        Eko.append(float(temp[3])*1000)
        Ocosx.append(float(temp[4]))
        Ocosy.append(float(temp[5]))
        Ocosz.append(float(temp[6]))

    k = k + 1

csvFile.close()
```

```
#IMPORT DATA, DETECTED NEUTRONS
#data lists
X_1 = []
Y_1 = []
Z_1 = []
Ek_1 = []
Ncase_1 = []
Ox_1 = []
Oy_1 = []
Oz_1 = []
Ocosx_1 = []
Ocosy_1 = []
Ocosz_1 = []
Eko_1 = []
P1cosx_1 = []
P1cosy_1 = []
P1cosz_1 = []
Evt_type_1 = []

X_2 = []
Y_2 = []
Z_2 = []
Ek_2 = []
Ncase_2 = []
Ox_2 = []
Oy_2 = []
Oz_2 = []
Ocosx_2 = []
Ocosy_2 = []
Ocosz_2 = []
Eko_2 = []
P1cosx_2 = []
P1cosy_2 = []
P1cosz_2 = []
Evt_type_2 = []


# Read the data files and append to the empty lists
```

```
import csv
with  open('zzProstHUH37_all46_p_coordinates_and_parent_n_origin_xyz.csv',  'r')  as
csvFile:
    reader = csv.reader(csvFile)
    k = 0
    for row in reader:
    temp = row[0].split()
    if (k % 2) == 0:
            X_1.append(float(temp[0]))
            Y_1.append(float(temp[1]))
            Z_1.append(float(temp[2]))
            Ek_1.append(float(temp[3])*1000)
            Ncase_1.append(float(temp[4]))
           Ox_1.append(float(temp[5]))
            Oy_1.append(float(temp[6]))
            Oz_1.append(float(temp[7]))
            Ocosx_1.append(float(temp[8]))
            Ocosy_1.append(float(temp[9]))
            Ocosz_1.append(float(temp[10]))
            Eko_1.append(float(temp[11])*1000)
            P1cosx_1.append(float(temp[12]))
            P1cosy_1.append(float(temp[13]))
            P1cosz_1.append(float(temp[14]))
            Evt_type_1.append(float(temp[15]))

    else:
            X_2.append(float(temp[0]))
            Y_2.append(float(temp[1]))
            Z_2.append(float(temp[2]))
            Ek_2.append(float(temp[3])*1000)
            Ncase_2.append(float(temp[4]))
            Ox_2.append(float(temp[5]))
            Oy_2.append(float(temp[6]))
            Oz_2.append(float(temp[7]))
            Ocosx_2.append(float(temp[8]))
            Ocosy_2.append(float(temp[9]))
            Ocosz_2.append(float(temp[10]))
            Eko_2.append(float(temp[11])*1000)
            P1cosx_2.append(float(temp[12]))
            P1cosy_2.append(float(temp[13]))
            P1cosz_2.append(float(temp[14]))
            Evt_type_2.append(float(temp[15]))
    k = k + 1
```

```
csvFile.close()


#Change the size of the detector to 190x200 cm (SAME AS WATER PHANTOM):
xs = 0

dX_1 = []
dY_1 = []
dZ_1 = []
dEk_1 = []
dNcase_1 = []
dOx_1 = []
dOy_1 = []
dOz_1 = []
dOcosx_1 = []
dOcosy_1 = []
dOcosz_1 = []
dEko_1 = []
dP1cosx_1 = []
dP1cosy_1 = []
dP1cosz_1 = []
dEvt_type_1 = []

dX_2 = []
dY_2 = []
dZ_2 = []
dEk_2 = []
dNcase_2 = []
dOx_2 = []
dOy_2 = []
dOz_2 = []
dOcosx_2 = []
dOcosy_2 = []
dOcosz_2 = []
dEko_2 = []
dP1cosx_2 = []
dP1cosy_2 = []
dP1cosz_2 = []
dEvt_type_2 = []


for a, b, c, d, e, f, g in zip(X_1, Z_1, X_2, Z_2, Ox_1, Eko_1, Oz_1):
    if 50 > a > -140 and 50 > c > -140 and -100 < b < 100 and -100 < d < 100:
            dX_1.append(a)
            dZ_1.append(b)
           dOx_1.append(e)
            dOz_1.append(g)
            dEko_1.append(f)
            dX_2.append(c)
            dZ_2.append(d)
        xs = xs + 1



#Number of primary particles
primaries = '1.2x10$^{9}$'





#CREATE PLOTS



#MULTIPLOT, PRODUCTION POSITIONS AND ENERGIES
#Create histogram, depth in water
fig, axes = plt.subplots(nrows = 1, ncols = 2, figsize = (15,6))

#produced
newOx = []
for i in Ox:
    i = i - 20
    newOx.append(abs(i))

axes[0].set_xlim(0,40)
axes[0].hist(newOx, bins = np.arange(min(newOx)-0.5, max(newOx)+0.5, 0.3), ec='red',
fc='none', lw=0.5, histtype='step', label = 'Produced neutrons')

#detected
newOx2 = []
for i in dOx_1:
    i = i - 20
    newOx2.append(abs(i))

axes[0].set_xlim(0,40)
```

```
axes[0].hist(newOx2,  bins  =  np.arange(min(newOx2)-0.5,  max(newOx2)+0.5,  0.3),
ec='blue', fc='none', lw=0.5, histtype='step', label = 'Detected neutrons')

#SOBP
axes[0].axvspan(11.6,23.8,alpha = 0.2, color = 'black', label = 'SOBP')

axes[0].set_xlabel('Depth in patient [cm]')
axes[0].legend(loc='upper right')



#energy
MIN, MAX = 0.01, 1000
#produced
axes[1].hist(Eko,bins = np.logspace(np.log10(MIN), np.log10(MAX), 200), ec='red',
fc='none', lw=0.5, histtype='step', label = 'Produced neutrons')
#detected
axes[1].hist(dEko_1,bins   =   np.logspace(np.log10(MIN),   np.log10(MAX),   200),
ec='blue', fc='none', lw=0.5, histtype='step', label = 'Detected neutrons')
axes[1].set_xscale('log')
axes[1].set_xlabel('Kinetic energy [MeV]')
axes[1].legend(loc = 'upper left')

fig.text(0.06,  0.5,  'Neutrons  from  '  +  str(primaries)  +  '  primary  protons',
ha='center', va='center', rotation='vertical')
fig.text(0.08, 0.88, 'a', fontsize = 20)
fig.text(0.52, 0.88, 'b', fontsize = 20)



plt.show()



#change the x-axis of the patient
newX_1 = []
for i in dX_1:
    if i <= 20:
        i = i - 20
        newX_1.append(abs(i))
    else:
        i = i - 20
        newX_1.append(-(i))

newX_2 = []
for i in dX_2:
    if i <= 20:
        i = i - 20
        newX_2.append(abs(i))
    else:
        i = i - 20
        newX_2.append(-(i))



#NEUTRON DETECTION AS A FUNCTION OF DETECTOR SIZE
nbneutron100 = []

wh100 = 2.5
#Create list holding the neutron detection rate for different detector sizes
for m in range(50):
    for a, b, c, d in zip(newX_1, dZ_1, newX_2, dZ_2):
        if (32 - wh100) < a < (32 + wh100) and (32 - wh100) < c < (32 + wh100) and
-wh100 < b < wh100 and -wh100 < d < wh100:
            nbneutron100.append(wh100*2)
    wh100 = wh100 + 0.5

n, bins, _ = plt.hist(nbneutron100, bins = np.arange(min(nbneutron100)-0.5,
max(nbneutron100)+0.5, 1), ec = 'darkblue', fc='none', lw=0.5, histtype='step',
align = 'left')

#linear fit
bincenters = 0.5*(bins[1:]+bins[:-1])
coef = np.polyfit(bincenters, n,1)
poly1d_fn = np.poly1d(coef)
plt.plot(bincenters, poly1d_fn(bincenters), '--r', lw=1.0, alpha = 0.5, label =
'Linear fit')

#labels
plt.xlabel('Side length of the detector [cm]')
plt.ylabel('Neutrons from ' + str(primaries) + ' primary protons')
plt.ylim(0,30000)
plt.legend(loc= 'upper left')
plt.show()


#CHANGE FONT SIZE FOR THE REST OF THE PLOTS
plt.rcParams.update({'font.size': 20})
plt.rc('xtick',labelsize=20)
plt.rc('ytick',labelsize=20)



#CREATE HISTOGRAM, LOCATION ON DETECTOR PLANES X-DIRECTION
#histogram
n, bins, _ = plt.hist(newX_1, bins = np.arange(min(newX_1), max(newX_1)+3, 3),
ec='blue', fc='none', lw=1.0, histtype='step', label='First tracking detector')
n2, bins2, _ = plt.hist(newX_2, bins = np.arange(min(newX_2), max(newX_2)+3, 3),
ec='red', fc='none', lw=1.0, histtype='step', label='Second tracking detector')


#errorbars
menStd = np.sqrt(n)
menStd2 = np.sqrt(n2)
bincenters = 0.5*(bins[1:]+bins[:-1])
bincenters2 = 0.5*(bins2[1:]+bins2[:-1])
plt.errorbar(bincenters, n, yerr = menStd, fmt = 'none', ecolor = 'black')
plt.errorbar(bincenters2, n2, yerr = menStd2, fmt = 'none', ecolor = 'black')


#SOBP
plt.axvspan(11.6,23.8,alpha = 0.2, color = 'black', label = 'SOBP')
#legend and labels
plt.legend(loc='upper right')
plt.xlim(-35,170)
plt.xlabel('Location on tracking detector in x-direction [cm]')
plt.ylabel('Neutrons from ' + str(primaries) + ' primary protons')
plt.show()




# Neutron detection as a function of detector position

#create new list holding the center positions of 20x20cm detectors moved 1 cm step
nbneu = []
x = -20
low = -30
high = -10
h = 0
for h in range(170):
    for b, n, k, a in zip(newX_1 ,newX_2, dZ_1, dZ_2):
        if low < b < high and low < n < high and -10 < k < 10 and -10 < a < 10:
            nbneu.append(x)
    low = low + 1
    high = high + 1
    x = x+1


n, bins, _ = plt.hist(nbneu, bins = np.arange(min(nbneu)-0.5, max(nbneu)+0.5, 1),
ec='darkblue', fc='none', lw=1.0, histtype='step')

#errorbar
menStd = np.sqrt(n)
bincenters = 0.5*(bins[1:]+bins[:-1])
plt.errorbar(bincenters, n, yerr = menStd, fmt = 'none',  lw=1.0, ecolor = 'black')

#SOBP
plt.axvspan(11.6,23.8,alpha = 0.2, color = 'black', label = 'SOBP')

#labels and legend
plt.xlim(-25,150)
plt.xlabel('Center point of detector in x-direction [cm]')
plt.ylabel('Neutrons from ' + str(primaries) + ' primary protons')
plt.legend(loc='upper right')
plt.show()
```

Appendix C

Field_1_source.f

FLUKA user routine for patient treatment plan simulation, used for incorporation of the different directions and positions for the various beam energies in the treatment field.

```
*$ CREATE SOURCE.FOR
*COPY SOURCE
*
*=== source ===========================================================*
*
      SUBROUTINE SOURCE ( NOMORE )

      INCLUDE '(DBLPRC)'
      INCLUDE '(DIMPAR)'
      INCLUDE '(IOUNIT)'
*
*----------------------------------------------------------------------*
*                                                                      *
*     Copyright (C) 1990-2006      by    Alfredo Ferrari & Paola Sala  *
*     All Rights Reserved.                                             *
*                                                                      *
*                                                                      *
*     New source for FLUKA9x-FLUKA200x:                                *
*                                                                      *
*     Created on 07 january 1990   by    Alfredo Ferrari & Paola Sala  *
*                                                   Infn - Milan       *
*                                                                      *
*     Last change on 03-mar-06     by    Alfredo Ferrari               *
*                                                                      *
*  This is just an example of a possible user written source routine.  *
*  note that the beam card still has some meaning - in the scoring the *
*  maximum momentum used in deciding the binning is taken from the     *
*  beam momentum.  Other beam card parameters are obsolete.            *
*                                                                      *
*----------------------------------------------------------------------*
*
      INCLUDE '(BEAMCM)'
      INCLUDE '(FHEAVY)'
      INCLUDE '(FLKSTK)'
      INCLUDE '(IOIOCM)'
      INCLUDE '(LTCLCM)'
      INCLUDE '(PAPROP)'
      INCLUDE '(SOURCM)'
      INCLUDE '(SUMCOU)'
*
      INCLUDE '(CASLIM)'
*
c $FLUPRO/flutil/ldpm3qmd source_SAM.f -o flukadpm3_sam
      DOUBLE PRECISION ENERGY(65000), XYPOS(65000), YXPOS(65000)
      DOUBLE PRECISION ZPOS(65000), FWHMX(65000), FWHMY(65000)
      DOUBLE PRECISION FWHMZ(65000), PART(65000), PSPREAD
      INTEGER NWEIGHT
      DOUBLE PRECISION COSX(65000), COSY(65000), COSZ(65000)
      DOUBLE PRECISION ROOT, DELTAP, SPOTRAND, SPOTSUM
      DOUBLE PRECISION KOEFF1, KOEFF2, KOEFF3, KOEFF4

      SAVE ENERGY, XYPOS, YXPOS
      SAVE ZPOS, FWHMX, FWHMY
      SAVE FWHMZ, PART, PSPREAD
      SAVE NWEIGHT
      SAVE COSX, COSY, COSZ
      SAVE ROOT, DELTAP
      SAVE KOEFF1, KOEFF2, KOEFF3, KOEFF4

      LOGICAL LFIRST
*
      SAVE LFIRST
      DATA LFIRST / .TRUE. /
*======================================================================*
*                                                                      *
*                 BASIC VERSION                                        *
*                                                                      *
*======================================================================*
      NOMORE = 0
*  +-------------------------------------------------------------------*
*  |  First call initializations:
      IF ( LFIRST ) THEN
*  |  *** The following 3 cards are mandatory ***
         WRITE(LUNOUT,*) ' NB SOURCE_SAM4 INVOKED'
         TKESUM = ZERZER
         LFIRST = .FALSE.
         LUSSRC = .TRUE.
```

```
c        treatment field .dat file
         OPEN(44, FILE = '../Field_1.dat',
     $        STATUS = 'OLD')
*        Skip first three lines
         READ(44, *)
         READ(44, *)
         READ(44, *)

         NWEIGHT = 0
         WSUM = ZERZER
         DO
            NWEIGHT = NWEIGHT + 1
            IF (NWEIGHT .GT. 65000) THEN
               WRITE(LUNOUT,*) 'NB SOURCE ERROR: too many beamlets'
            ENDIF

            READ (44, 3, END=10 ) ENERGY(NWEIGHT),
     $           XYPOS(NWEIGHT), YXPOS(NWEIGHT), ZPOS(NWEIGHT),
     $           FWHMX(NWEIGHT), FWHMY(NWEIGHT), FWHMZ(NWEIGHT),
     $           PART(NWEIGHT), COSX(NWEIGHT), COSY(NWEIGHT),
     $           COSZ(NWEIGHT)
 3          FORMAT(F12.4,F12.4,F12.4,F12.4,F12.4,F12.4,F12.4,E12.4,
     $                F12.6,F12.6,F12.6)
            WSUM = WSUM + PART(NWEIGHT)

         ENDDO
 10      CONTINUE

         WRITE(LUNOUT,*) 'NB SOURCE beamlets found:', NWEIGHT-1
         WRITE(LUNOUT,*) 'NB SOURCE Particle sum (float) :', WSUM
         WRITE(LUNOUT,*) 'NB SOURCE TODO: particle sum is not exact.'

      END IF

*** Sample a beamlet ****************************


*     Choose randomly which spot to sample. It takes into account that each
*     spot/line has a different different weight
      RAND = FLRNDM(DOUBLEDUMMY) ! Returns double precision between [0,1)
      SPOTRAND = WSUM * RAND
      SPOTSUM = ZERZER
      DO I = 1, NWEIGHT ! Loop through lines until SPOTRAND is reached
        SPOTSUM = SPOTSUM + PART(I)
        IF (SPOTSUM .GT. SPOTRAND) THEN
           NRAN = I ! Select the spot
           EXIT
        END IF
      END DO

      ENK = ENERGY(NRAN)
      XBEAM = XYPOS(NRAN)
      YBEAM = YXPOS(NRAN)
      ZBEAM = ZPOS(NRAN)
      XSPOT = FWHMX(NRAN)/2.35482
      YSPOT = FWHMY(NRAN)/2.35482
      ZSPOT = FWHMZ(NRAN)/2.35482
      COSIX = COSX(NRAN)
      COSIY = COSY(NRAN)
      COSIZ = COSZ(NRAN)


*** End of beamlet sample ********************************************


*  +-------------------------------------------------------------------*
*  Push one source particle to the stack. Note that you could as well
*  push many but this way we reserve a maximum amount of space in the
*  stack for the secondaries to be generated
* Npflka is the stack counter: of course any time source is called it
* must be =0
      NPFLKA = NPFLKA + 1
* Wt is the weight of the particle
      WTFLK  (NPFLKA) = ONEONE ! Set weight = 1
c     Sets the weight of the particle
      WEIPRI = WEIPRI + WTFLK (NPFLKA)
c     WEIPRI updates the total weight of the primaries
* Particle type (1=proton.....). Ijbeam is the type set by the BEAM
* card
*  +-------------------------------------------------------------------*
*  |  (Radioactive) isotope:
      IF ( IJBEAM .EQ. -2 .AND. LRDBEA ) THEN
         IARES  = IPROA
         IZRES  = IPROZ
         IISRES = IPROM
         CALL STISBM ( IARES, IZRES, IISRES )
         IJHION = IPROZ  * 1000 + IPROA
         IJHION = IJHION * 100 + KXHEAV
         IONID  = IJHION
         CALL DCDION ( IONID )
         CALL SETION ( IONID )
*  |
*  +-------------------------------------------------------------------*
*  |  Heavy ion:
      ELSE IF ( IJBEAM .EQ. -2 ) THEN
         IJHION = IPROZ  * 1000 + IPROA
         IJHION = IJHION * 100 + KXHEAV
         IONID  = IJHION
         CALL DCDION ( IONID )
         CALL SETION ( IONID )
         ILOFLK (NPFLKA) = IJHION
*  |  Flag this is prompt radiation
         LRADDC (NPFLKA) = .FALSE.
*  |
*  +-------------------------------------------------------------------*
*  |  Normal hadron:
      ELSE
         IONID = IJBEAM
         ILOFLK (NPFLKA) = IJBEAM
*  |  Flag this is prompt radiation
         LRADDC (NPFLKA) = .FALSE.
      END IF
*  |
*  +-------------------------------------------------------------------*
* From this point .....
* Particle generation (1 for primaries)
      LOFLK  (NPFLKA) = 1
* User dependent flag:
      LOUSE  (NPFLKA) = 0
* User dependent spare variables:
      DO 100 ISPR = 1, MKBMX1
         SPAREK (ISPR,NPFLKA) = ZERZER
 100  CONTINUE
* User dependent spare flags:
      DO 200 ISPR = 1, MKBMX2
         ISPARK (ISPR,NPFLKA) = 0
 200  CONTINUE
* Save the track number of the stack particle:
      ISPARK (MKBMX2,NPFLKA) = NPFLKA
      NPARMA = NPARMA + 1
      NUMPAR (NPFLKA) = NPARMA
      NEVENT (NPFLKA) = 0
      DFNEAR (NPFLKA) = +ZERZER
* ... to this point: don't change anything
* Particle age (s)
      AGESTK (NPFLKA) = +ZERZER
      AKNSHR (NPFLKA) = -TWOTWO
* Group number for "low" energy neutrons, set to 0 anyway
      IGROUP (NPFLKA) = 0
*****************************************************************

*sample a gaussian position
*       IF (Ldygss) THEN
      CALL FLNRR2 (RGAUS1, RGAUS2)
      XFLK   (NPFLKA) = XBEAM + XSPOT * RGAUS1
      YFLK   (NPFLKA) = YBEAM + YSPOT * RGAUS2
      CALL FLNRRN (RGAUSS)
      ZFLK   (NPFLKA) = ZBEAM + ZSPOT * RGAUSS



*       WRITE(LUNOUT,*) 'NB SOURCE gaussian sampled'

* Cosines (tx,ty,tz)
      ROOT = SQRT(COSIX**2+COSIY**2+COSIZ**2)
      TXFLK  (NPFLKA) = COSIX/ROOT
      TYFLK  (NPFLKA) = COSIY/ROOT
      TZFLK  (NPFLKA) = COSIZ/ROOT
*     TZFLK  (NPFLKA) = SQRT ( ONEONE - TXFLK (NPFLKA)**2
*    &                       - TYFLK (NPFLKA)**2 )
*       WRITE(LUNOUT,*) 'NB SOURCE cosines set'
**********************************************************************
* Particle momentum
*       PMOFLK (NPFLKA) = PBEAM
*       WRITE(LUNOUT,*) 'NB SOURCE mark',AM (IONID)
        CALL FLNRRN(RGAUSS)
        PMOFLK (NPFLKA) = SQRT ( ENK* ( ENK
     &     + TWOTWO * AM (IONID) ))

* Calculate momentum spread using third polynomial fit
        KOEFF1 = 4.6234
        KOEFF2 = 1.7547
```

```
       KOEFF3 = 0.2159
       KOEFF4 = 0.0163

       DELTAP = -KOEFF1*ENK**3 + KOEFF2*ENK**2
     &          - KOEFF3*ENK + KOEFF4

       PSPREAD = PMOFLK (NPFLKA) * DELTAP / 2.35482 * RGAUSS

       PMOFLK (NPFLKA) = PMOFLK (NPFLKA) + PSPREAD

* Kinetic energy of the particle (GeV)
* set energy
      TKEFLK (NPFLKA) = SQRT(PMOFLK(NPFLKA)**2 + AM(IONID)**2)
     &      -AM(IONID)

*      WRITE(LUNOUT,*) 'NB SOURCE set ekin'



* Polarization cosines:
      TXPOL  (NPFLKA) = -TWOTWO
      TYPOL  (NPFLKA) = +ZERZER
      TZPOL  (NPFLKA) = +ZERZER
*      WRITE(LUNOUT,*) 'NB SOURCE pol set'
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*  Calculate the total kinetic energy of the primaries: don't change
      IF ( ILOFLK (NPFLKA) .EQ. -2 .OR. ILOFLK (NPFLKA) .GT. 100000 )
     &   THEN
         TKESUM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
      ELSE IF ( ILOFLK (NPFLKA) .NE. 0 ) THEN
         TKESUM = TKESUM + ( TKEFLK (NPFLKA) + AMDISC (ILOFLK(NPFLKA)) )
     &          * WTFLK (NPFLKA)
      ELSE
         TKESUM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
      END IF
      RADDLY (NPFLKA) = ZERZER

*      WRITE(LUNOUT,*) 'NB SOURCE mark'

*  Here we ask for the region number of the hitting point.
*     NREG (NPFLKA) = ...
*  The following line makes the starting region search much more
*  robust if particles are starting very close to a boundary:
      CALL GEOCRS ( TXFLK (NPFLKA), TYFLK (NPFLKA), TZFLK (NPFLKA) )
      CALL GEOREG ( XFLK  (NPFLKA), YFLK  (NPFLKA), ZFLK  (NPFLKA),
     &              NRGFLK(NPFLKA), IDISC )
*      WRITE(LUNOUT,*) 'NB SOURCE mark2'
*  Do not change these cards:
      CALL GEOHSM ( NHSPNT (NPFLKA), 1, -11, MLATTC )
      NLATTC (NPFLKA) = MLATTC
      CMPATH (NPFLKA) = ZERZER
      CALL SOEVSV


*      WRITE(LUNOUT,*) 'NB SOURCE END'
      CLOSE(44)
      RETURN
*=== End of subroutine Source ==========================================*
      END
```

Appendix D

Depth dose curves used for quantifying the lower and upper limit for the Spread-Out Bragg peak in the patient treatment plan.

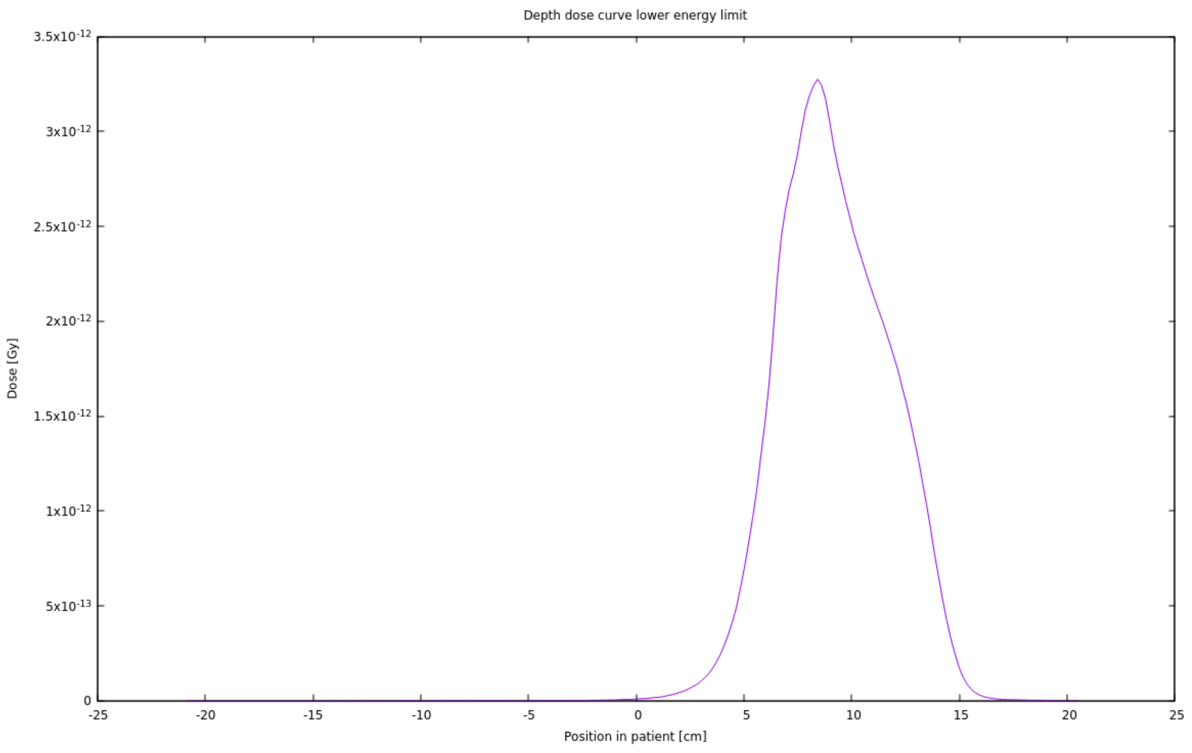

Figure A-33: Depth dose curve for the lower energy limit of the SOBP. Here the beam enters from positive x-direction at x = 20 cm, and the center of the patient is in x = 0 cm. The Bragg peak is located in x = 8.4 cm.

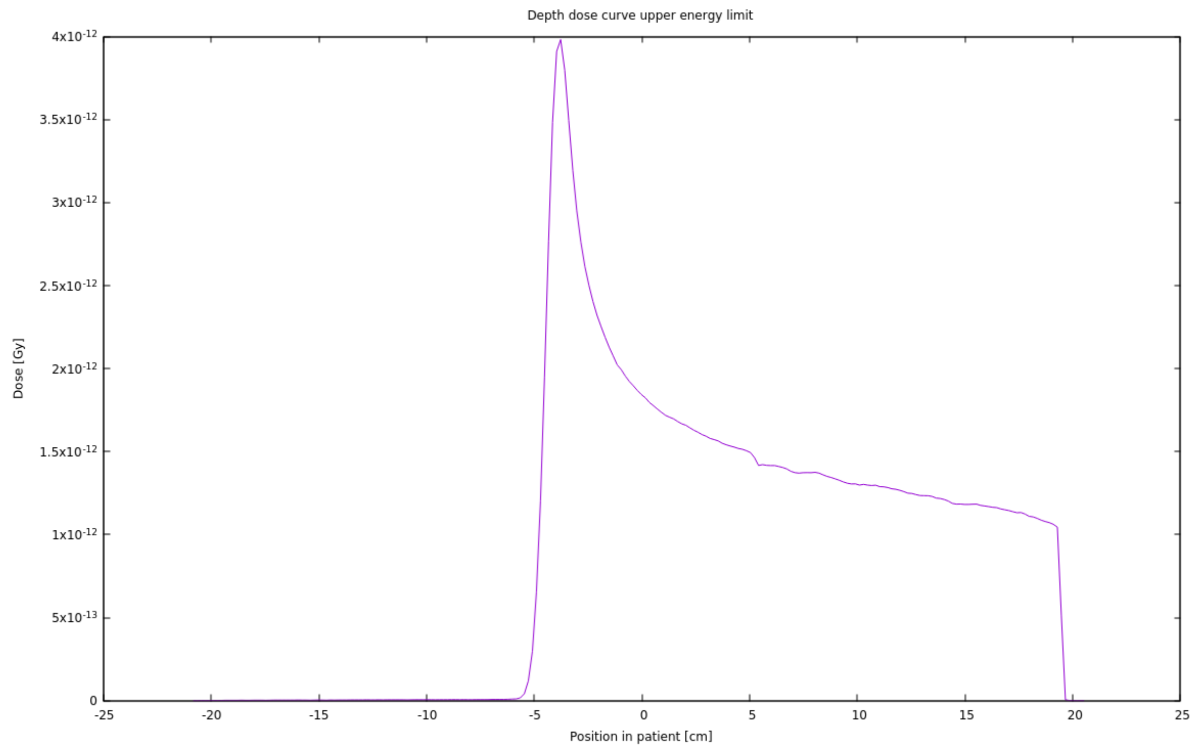

Figure A-34: Depth dose curve for the upper energy limit of the SOBP. Here the beam enters from positive x-direction at x = 20 cm, and the center of the patient is located in x = 0 cm. The Bragg peak is located in x = -3.8 cm.

Appendix E

Table A-2: The results for position of maximum neutron detection rate from the water phantom simulation. Numbers in parentheses indicate position relative to Bragg peak (positive number signify position distal to the Bragg peak).

Proton beam energy [MeV]	**Position with maximal neutron detection rate [cm]:**		
	All neutrons	**Neutrons detected last half prior to Bragg peak**	**Neutrons detected last quarter prior to Bragg peak**
100	15 ± 2 (7.4 ± 2)	16 ± 3 (8.4 ± 3)	17 ± 4 (9.4 ± 4)
160	20.5 ± 1.5 (3 ± 1.5)	24 ± 2 (6.5 ± 2)	25.5 ± 1.5 (8 ± 1.5)
200	24.5 ± 1.5 (-1.2 ± 1.5)	31 ± 2 (5.3 ± 2)	33 ± 2 (7.3 ± 2)
230	28 ± 2 (-4.6 ± 2)	35.5 ± 1.5 (2.9 ± 1.5)	39 ± 2 (6.4 ± 2)