Vegard Aas Mjelde

A Predictive Model for Biological Range Shift in Proton Therapy

Master's thesis in Applied Physics and Mathematics
Supervisor: Kathrine Røe Redalen, Krisian Smelan Ytre-Hauge
June 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Vegard Aas Mjelde

A Predictive Model for Biological Range Shift in Proton Therapy

Master's thesis in Applied Physics and Mathematics
Supervisor: Kathrine Røe Redalen (NTNU), Krisian Smelan Ytre-Hauge (UiB)
June 2019

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Abstrakt

Protonterapi er en metode innenfor strålebehandling for kreft som har blitt tatt i bruk i mange land over hele verden i løpet av de siste tiårene. Et argument for å bruke protonterapi er at protonene har en bedre dybde-dose fordeling enn fotoner i vev, som fører til redusert dose til friskt vev, spesielt vev lokalisert bak svulsten. Norge har nå satt igang planleggingen av to protonsentre i Bergen og Oslo, som skal støtte det økende behovet for strålebehandling med protoner.

Den biologiske effekten protonstråling har på cellene i kroppen er annerledes enn fotonenes effekt. Protonene gjør mer skade for samme avgitte fysiske dose på grunn av en økt Lineær Energi Overføring (LET). Denne økningen i biologisk effektivitet er kvantifisert i den Relative Biologiske Effektiviteten (RBE). Klinisk blir en konstant RBE lik 1.1 ($RBE_{1.1}$) brukt, men flere studier tyder på at RBE øker med dybde på grunn av at LET øker med dybde[25, 31, 33, 36]. Dette fører til en potensiell underestimering av de biologiske konsekvensene av strålingen ved bruk av $RBE_{1.1}$, spesielt i den dypeste delen av strålingsfeltet, mot slutten av partiklenes rekkevidde. Denne økningen i avgitt dose på grunn av en variabel RBE forlenger rekkevidden (range extension/shift) til dosefordelingen. Denne forlengelsen av rekkevidden kan føre til uforventede skader til friskt vev og utsatte organer. Modellering av forlengelsen av rekkevidden er derfor viktig, og det var hensikten med dette prosjektet.

Monte Carlo simuleringer av "Pristine Bragg Peaks" (PBPs) og "Spread Out Bragg Peaks" (SOBPs) ved flere dybder med flere dosenivåer i en vanntank ble utført. Dosefordelingen for $RBE_{1.1}$ og Rørviks RBE modell [36] ble beregnet. Et dataset av den biologiske forlengelsen av rekkevidden ble så hentet ut fra disse dosefordelingene, og forutseende modeller for forlengelsen av rekkevidden i protonterapi ble så laget ved bruk av lineær regresjon på datasettet. Modellene ble så testet på to dosefordelinger; en SOBP i en vanntank, og en behandlingsplan for prostatakreft.

Resultatene viser at modellene bassert på PBPs under-estimerer forlengelsen av rekkevidden med $50\% - 75\%$, sammenlignet med den målte forlengelsen for de to dosefordelingene. Modellene bassert på SOBPs presterer bedre, men underestimerer med omtrent 30% for begge dosefordelingene. Ettersom de fleste behandlingsplaner i protonterapi består av protonstråler med flere forskjellige energier, er mest sannsynlig modellene bassert på PBPs ikke i stand til å gjenskape den høye RBE fordelingen i behandlingsplaner. Den lave RBE-en som PBP-metoden produserer fører til under-estimeringen av forlengelsen av rekkevidden. Underestimeringen i modellene bassert på SOBPs kan være forårsaket av formen og bredden til SOBP-ene.

I dette prosjektet ble det laget forutseende modeller for biologisk forlengelse av strålens rekkvidde i protonterapi. Modellene bassert på SOBPs viste seg å være bedre enn modellene bassert på PBPs, selv om alle modellene systematisk underestimerte forlengelsen av rekkevidden. Årsaker til underestimeringene kan være formen og bredden til SOBP-ene, og videre prosjekter på temaet burde derfor undersøke hvilke konsekvenser dette har for RBE-fordelingen og dermed også forlengelsen av rekkevidden i protonterapi.

Acknowledgements

First of all, I would like to thank my supervisor Dr. Kristian S. Ytre-Hauge for introducing me to the field of proton therapy and for great guidance and discussions during the project. Thanks to the rest of the group at medical physics at the university of Bergen; Eivind Rørvik, Tordis J. Dahle, Lars Fredrik Fjæra and Helge Henjum. I would also like to thank my internal supervisor, Kathrine Røe Redalen, for great support and the weekly meetings with the rest of the biophysics and medical physics group. Thank you, Brit Wenche Meland, student counsellor at MTFYMA, for information and nice conversations throughout the five years. I'm sorry I didn't manage to stop by one last time. Thanks to my family and friends for encouragement and support throughout the five years at NTNU. Special thanks to Jæss Winnerdal for a great comradely for many past years, and even more to come. Two out of four, we're going to Florida!

Oslo, June 2019
Vegard Aas Mjelde

Abstract

Proton therapy is a radiation treatment method used in many countries around the world, and Norway is now preparing to build two proton therapy centers. Today, mostly photon therapy is used for radiation treatment in Norway. The argument for using protons is their superior sparing ability, especially of the healthy tissue located behind the tumor, and the building of two proton centers in Norway has begun, as of May 2019.

The biological effects on cells due to proton irradiation are different than for photons. The protons cause more damage for the same deposited physical dose, because of their increased Linear Energy Transfer (LET). This increase in biological effectiveness is quantified by the Relative Biological Effectiveness (RBE). Clinically a constant RBE factor of 1.1 ($RBE_{1.1}$) is used, but multiple studies on RBE indicate that it increases with depth due to the increase in LET with depth[25, 31, 33, 36]. This leads to a potential underestimation of the biological effects when using $RBE_{1.1}$, especially in the distal part of the treatment fields, towards the end of the particles' range. The increase in dose due to variable RBE increases the depth of the distal 80% dose fall-off, which is often used to quantify the beam range. This range shift between the $RBE_{1.1}$ and the variable RBE models may cause unexpected radiation damage to healthy tissue and organs at risk. Modelling or predicting the biological range shift in different scenarios is therefore of importance, and the it was the goal for this project.

Monte Carlo simulations of Pristine Bragg Peaks (PBPs) and Spread Out Bragg Peaks (SOBPs) of multiple depths and dose levels in a water tank were performed. RBE-weighted depth-dose distributions were then calculated along the beam axis for the $RBE_{1.1}$ and the RBE model by Rørvik et al. [36] (ROR) in multiple types of tissue. The biological range shift, between the $RBE_{1.1}$ and the ROR model, of the 80% iso-dose curves were then calculated. From this dataset of range shifts, predictive models were made, and tested in two scenarios; a SOBP in a water tank, and a prostate cancer treatment plan.

The results show that the predictive models based on the PBPs greatly underestimates the biological range shifts by $50\% - 75\%$ compared to the measured range shifts from simulations of the two scenarios. The predictive models based on the SOBPs perform better, but still underestimates by about 30% compared to the measured range shifts in the two scenarios. As most patient plans consists of proton beams of multiple energies, the PBPs method most likely fails to reproduce the (higher) RBE distribution of the SOBPs in patient plans. The low RBE leads to a low range shift prediction. The underestimation in the predictive models based on the SOBPs may be caused by the width of the SOBPs in the basis, and the shape of the distal part of the SOBPs.

In this project, we have produced predictive models for the biological range shifts in proton therapy, with the models based on SOBPs in water being superior to PBPs in water. All the predictive models underestimates, and the causes may be the shape and width of the SOBPs forming the basis. We therefore suggest that further studies include analysis of how the shape and with of the SOBP affects the RBE distribution and thereby also the biological range shift.

List of Figures

List of Tables

Contents

1 Introduction

New incidents of cancers occur every day. In 2017 there were 33564 new incidents of cancer in Norway [4]. The development and improvement of cancer treatment is therefore an important area of research. As we learn more about the nature of different radiation types, and how they interact with the different cells in the body, the effectiveness and precision of radiation therapy increases. Surgery and chemotherapy is also used to fight cancer, and recently also immunotherapy[26]. These techniques are often combined with radiation therapy to ensure the best results for the patients.

Radiation therapy is based on the theory that energy is absorbed by the atoms in the cells, causing ionization and excitation. If these ionizatons occur in the DNA-molecule, it can cause functional defects to the cell. These defects may lead to cell death or loss of important regulatory growth functions and other essential functions[20]. The DNA-molecule is therefore the main target for the beam in radiation therapy. Cells which are out of the body's control are called cancerous cells. The aim of radiation therapy is to kill the cancerous cells, and to minimize the damage to the healthy cells. This is done by irradiating a certain volume with a prescribed dose, killing the cancerous cells[30].

1.1 Proton radiation therapy

Proton radiation therapy is one of the up-and-coming techniques in radiation therapy and the Norwegian government plans to build proton therapy centers in Oslo and Bergen within 2023 [18]. The depth-dose distribution of protons is superior to the depth-dose distribution of the photon because of the distinct peak at the end of the range of the protons, called the Bragg peak. This peak appears because of the way the protons interact with the charged particles in the atom (the electrons and the nucleus). As the protons are charged, and the photons are electrically neutral, the physics interactions with matter are very different. As the incident proton is slowed down, each atomic electron has more time to pull on the proton through Coulomb interaction, slowing it even more down. The kinetic energy lost by the proton as it is slowed down is transferred to the medium, resulting in ionization, or, in other words, deposition of dose. Graphing this dose deposition as a function of depth, we get the Bragg peak[21], as can be seen as the dashed blue graph in figure 1.

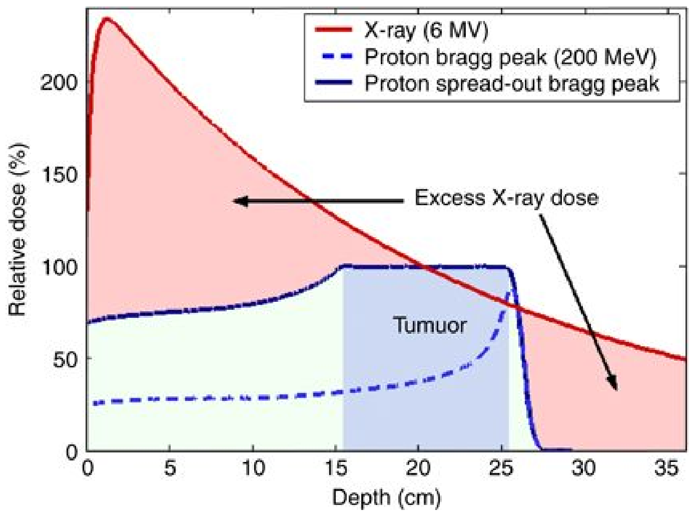


Figure 1: Spread-out Bragg peak (blue graph) covering the tumor volume (shaded area). The dashed blue graph shows the depth-dose curve of the proton with the highest energy, contributing in the spread-out Bragg peak. The red graph shows the depth-dose curve of a photon with the energy needed to treat the tumor. The unit of the y-axis is percentage dose relative to the prescribed dose to the tumor volume. [6]

The blue graph shows the Spread-Out Bragg Peak (SOBP), a dose-distribution superposed by protons of various energies. The dashed graph shows the dose distribution of the protons of largest energy (200 MeV). The shaded area illustrates a tumor located at a certain depth, with a certain extent into the tissue, and with a minimal amount of dose needed to treat it (y-axis). The red graph shows the depth-dose curve of a photon beam of energy equal 6 MeV. The area between the red graph and the blue graph illustrates the amount of excess dose deposited to the healthy tissue using standard photon therapy. The figure visualizes the superior sparing effect of proton therapy on the healthy tissue, especially in the distal region [21]. This is the main argument for using proton therapy over photon therapy.

However, in the last decades, improvements in the field of photon therapy delivery has allowed for more complex and better treatment plans to be made. Volumetric Arc Therapy (VMAT) and Intensity Modulated Radiation Therapy (IMRT) are two of these new techniques and they are the gold standard for photon therapy. VMAT and IMRT superposes the dose profiles from multiple angles, leading to reduced dose levels to the surrounding healthy tissues. VMAT irradiates the target while the gantry-head moves in an arc around the patient. Treatment with VMAT takes 2-3 minutes, while IMRT takes from 4 minutes, up to 20 minutes for highly complex cases.

The protons interact differently with the DNA in the cells than the photons, causing more damage per particle. The modelling of this difference in damage, known as Relative Biological Effectiveness (RBE), has been the subject of multiple studies and meta-studies in the past 15 years [25, 31, 33, 36]. The biological effects of the protons are larger than for the photons, and it increases towards the Bragg peak and reaches a maximum in the distal fall-off of the peak. This leads to an increase in the biological dose given to the tissue. Clinically today a constant RBE factor of 1.1 ($RBE_{1.1}$) is used for proton therapy, however, as the RBE seems to increase towards the end of the beam range[25, 31, 33, 36], the argument of clinical use of a variable RBE can be made.

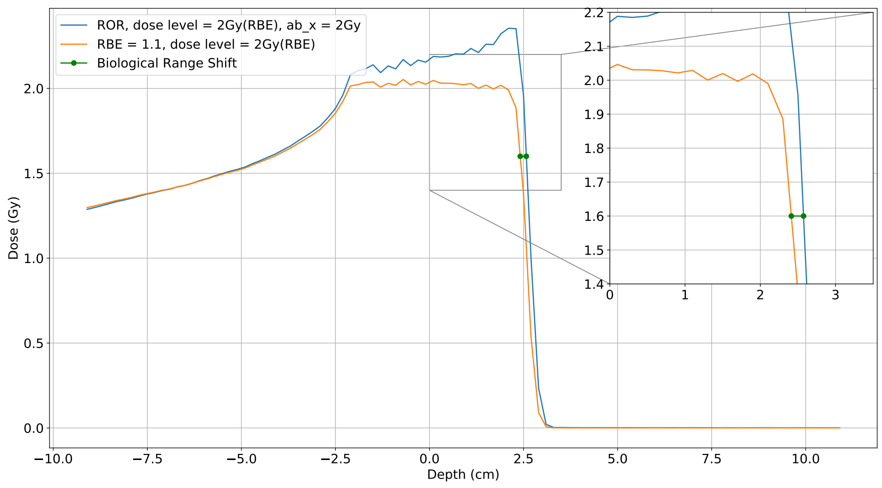


Figure 2: Biological dose plotted along the central axis of a beam delivering a spread out Bragg peak of 4 cm at a depth of 11.2 cm. The orange curve is the biological dose with $RBE_{1.1}$, while the blue curve is the biological dose with the ROR model, see section 3.3.1. The green dots marks the point where the distal dose has fallen to 80% of the prescribed 2 $Gy(RBE)$, and the biological range shift is defined to be the distance between these two points.

The increased variable RBE-weighted dose relative to the $RBE_{1.1}$-weighted dose causes a shift in the distal fall-off, which pushes the dose profile further into the tissue. Looking at the depth where the RBE-weighted dose has fallen to 80% of prescribed dose, the depth/range has been found to extend, relative to the $RBE_{1.1}$, up to 4mm in some cases [5, 14]. This range extension is visualized in figure 2, where the constant and variable RBE-weighted depth-dose profiles are plotted and the range shift is highlighted.

1.2 Motivation

The extension of the distal range of the proton beam due to variable RBE is not accounted for in Treatment Planning Systems (TPS) as these report the doses weighted with the $RBE_{1.1}$. A radio-sensitive organ may be located behind the tumor, and this shift may cause dosages to the organ not accounted for, causing unexpected side effects from the treatment. These fields, with a distal Organ at Risk (OAR), are therefore sometimes avoided in proton therapy, but as primary tumor control is the objective of the treatment, it is not always possible to choose a safer irradiation angle. The biological range shift can cause an unexpected amount of damage to the healthy surrounding tissue. Since studies[25, 31, 33, 36] suggest that a variable RBE model should be adapted clinically, the knowledge of the extent of the biological range shift in various fields is of importance.

In this project, we will therefore attempt to make a predictive biological range shift model, by calculating biological dose to water using Monte Carlo simulations, and sampling a dataset of biological range shifts from PBPs and SOBPs with various depths, dose levels and in different types of tissue. The goal is to have an estimate of the biological range shift based on the beam energy, dose level and tissue parameter $(\alpha/\beta)_x$ for any given proton treatment plan. This model can then be used during the production of a treatment plan, to evaluate the chosen fields by considering the biological range shift.

2 The Physics of Proton Therapy

The protons interact differently than the photons and electrons in matter, because of its electrical charge and high mass. The electric charge makes the protons interact more with the medium, while the large mass makes them less affected by the medium. This section will explain what happens to the protons as they traverse through a medium. It is based section is based on [27].

2.1 Proton interaction in matter

A beam of protons will interact with the medium it traverses through. The protons will interact with the nucleus of the atoms and its electrons, causing ionization and excitation along its path. High energy protons behave differently than high energy photons and electrons in a medium. Because of its charge, protons and electrons interact more with the other charged particles in the medium, causing more energy transfer over short distances. In a typical case of radiation therapy, a large portion of the photons will pass through the patient, while in electron therapy and proton therapy, all of the particles are absorbed within the patient. The larger mass of the proton in comparison to the mass of the electron, leaves it less prone to a change in momentum due to interactions with the medium, causing it to penetrate further into the patient than the electron, and with a straighter path[21].

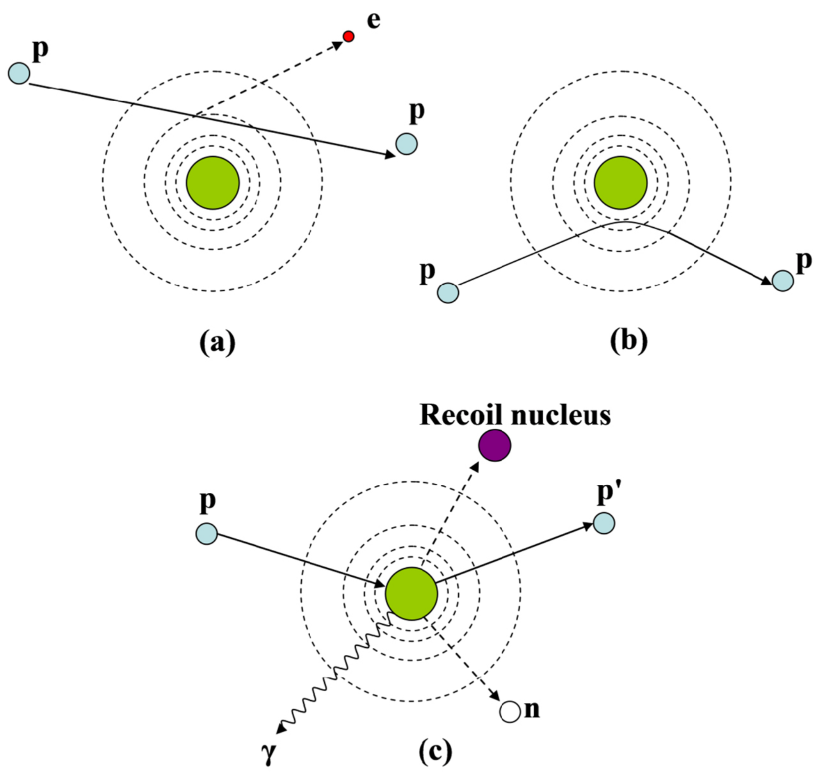


Figure 3: Illustration of the three main proton interactions with matter. (a) Interaction with the atomic electrons, causing minimal lateral changes in the protons momentum. (b) Interaction with the atomic nucleus, causing significant lateral changes in the protons momentum. Cumulative interactions of this type leads to range straggling and lateral spread of the beam. (c) Nuclear interactions [29]

The incident protons interact with both the atomic electrons and the nucleus of the atoms in the medium. Figure 3 shows examples of these interactions. Firstly, the proton can interact through the Coulomb force, pulling the atomic into an excited state, or completely releasing it from the atomic potential of the nucleus. Secondly, the proton can interact with the Coulomb field from the nucleus, causing a deflection of the proton beam. Lastly, the proton can collide directly with the nucleus. This is less common than the two former interactions[29]. The following subsections will discuss the nature of these interactions between the incident protons and the medium.

2.1.1 Interaction with electrons

The incident protons interact with the bounded electrons of the atoms in the medium. The protons will pull the electrons, through Coulomb interaction, from their low-energy state into a higher bound state (excitation) or completely pulling it away from the nuclear potential of the nucleus (ionization). The electrons released through ionization traverses themselves through the medium, depositing energy elsewhere (delta-rays). Their kinetic energy is equal to the energy lost by the proton, minus the binding energy of the electrons. The energy-loss of the beam of protons traversing through a medium per unit length is known as the stopping power of the medium and is given as [2, 7, 21]

$$-\frac{dE}{dx} = \frac{4\pi e^4 Z_p^2 Z_t^2}{m_e v^2}\left[\ln\frac{2m_e v^2}{\langle I\rangle} - \ln(1-\beta^2) - \beta^2 - \frac{C}{Z_t} - \frac{\delta}{2}\right]. \tag{1}$$

The symbols are explained in table 1. The minus sign is needed as the change in energy of the incident particle is negative ($dE < 0$) as the depth increases ($dx > 0$). Because of the inverse square relation between stopping power and the projectile velocity, the energy-loss of the projectile will increase as it is slowed down. This leads to small energy transfers at shallow depths, increasing with depth, reaching a maximum as the protons come to a halt. Plotting the absorbed dose, related to the energy-loss of the particles, versus depth we see the distinct Bragg peak towards the end of the range of the protons[21], which can be seen as the red curve in figure 4.

Symbol	Definition	Unit
e	Electron charge	C
m_e	Electron mass	MeV/c^2
Z_p	Projectile charge	e
Z_t	Target charge	e
v	Projectile velocity	m/s
β	Projectile velocity relative to speed of light	1
$\langle I\rangle$	Mean excitation potential	eV
C	Shell correction	1
$\delta 4$	Density effect correction	1

Table 1: Explanation of symbols appearing in the Bethe-Bloch formula for stopping power (equation 1)

2.1.2 Linear Energy Transfer

The average energy imparted in the medium dE as the incident charged particle traverse a distance dx through the medium is called the Linear Energy Transfer (LET). LET is a measure of the ionization per unit length[20]. While photons in general have a low LET, the LET of protons is, as shown by the Bethe-Bloch formula 1, strongly dependent on the proton energy. It also increases towards the end of the proton range, because now that the protons have less kinetic energy, they move more slowly, giving more time for interactions to occur, leading to more energy transfer. LET is an important quantity in radiotherapy, as increased LET leads to more damage to the DNA for the same dose level, and therefore a steeper survival curve. This will be further discussed in section 3.2. The dose-averaged LET (LET$_d$) is often used when concerned with biological effects as it considers the LET and the dose. LET$_d$ can be calculated by the expression:

$$LET_d = \frac{\int_0^\infty S_{el}(E)D(E,z)dE}{\int_0^\infty D(E,z)dE}, \tag{2}$$

where $S_{el}(E)$ is the electronic stopping power of a proton with kinetic energy E, and $D(E, z)$ is the absorbed dose contribution from a proton with kinetic energy E in the tissue at depth z[16] Figure 4 shows how the LET varies with the depth of a proton beam. The depth-dose relation of the proton beam is also plotted.

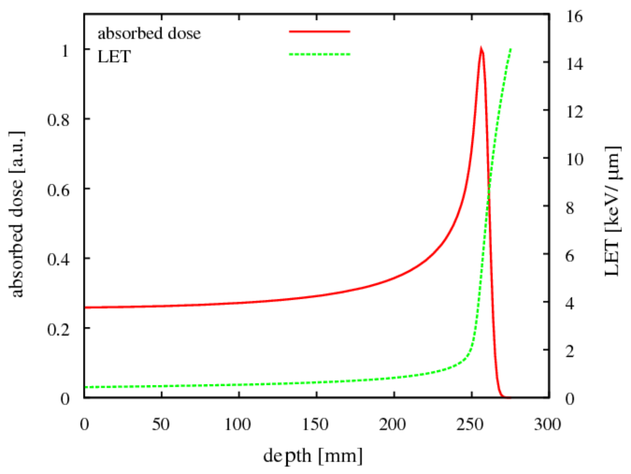


Figure 4: LET and dose plotted against depth for a proton beam of initial beam energy of 200 MeV.[11]

2.1.3 Interaction with the nucleus

The protons will interact through the Coulomb force with the nucleus, causing a small change in direction and energy of the traversing protons. This directional change is a change in momentum, and to ensure conservation of energy and momentum, a photon is created (Bremsstrahlung)[21]. Throughout its journey through the medium, the protons will scatter multiple times, adding up statistically to a wider beam at larger depths [28]. Figure 5 shows the typical path of a proton traversing through a material. The deviation from the original path-line is apparent, and looking at multiple protons in a beam, we will get a lateral spread of the radiation field. This lateral spread is not as large as for photons and electrons because of the protons being a "heavy" charged particle with a mass more than a thousand times larger than that of the electron [15]. This is also why the range of electrons are much shorter than the of protons. Electrons have larger scattering angles, leading to more lateral spread, causing them to use their energy to penetrate more in the lateral direction losing forward momentum.

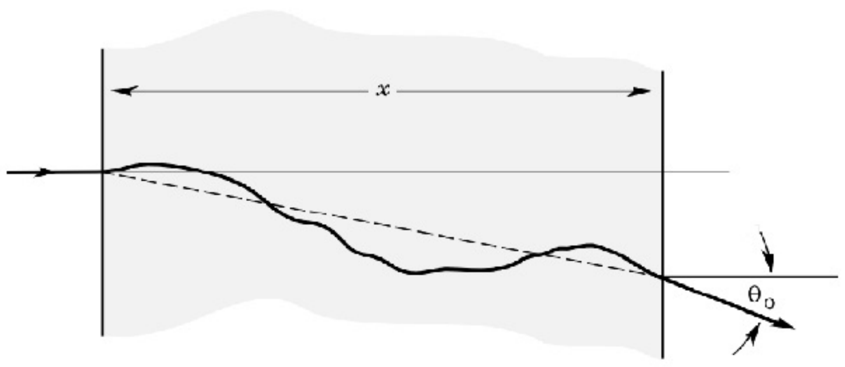


Figure 5: Illustration of a typical path of a proton through a medium. The deviation angle θ_0 from the incident direction is the result of multiple coulomb scattering on the protons journey through a thickness x of the medium. [28]

Secondary particles, such as secondary protons, neutrons and even alpha particles, can be created through direct collisions (inelastic interaction) with the nucleus. These collisions can also cause excited nuclei and isotopes. However, they are very rare, and do not contribute a lot to the attenuation of the proton beam [21]. The secondary particles could traverse through the medium themselves, depositing its energy elsewhere, either within the medium, or outside, depending on its range. Neutral secondary particles such as photons and neutrons have a larger range than charged secondary particles, as they do not interact through the Coulomb force [21].

2.2 Beam Range

There are multiple ways of defining the range of the protons. It can be defined as the depth of the peak dose, or where the dose has fallen to 90%, r_{90}, or 80%, r_{80}, or even 20%, r_{20}, of the peak dose. The definition we will use in this project is the r_{80}, which corresponds to the mean projected range of the protons[13].

2.2.1 Proton Fluence

The fluence of the protons will be slightly reduced as the protons some of them will be absorbed in the medium at any given depth. However, because of the protons high inertia and initially high kinetic energy, most of the them will penetrate further into the medium. The fluence will therefore stay high, until it suddenly drops towards the end of the protons range, as seen in figure 6. The range of the drop in proton fluence coincides with the range of the Bragg peak in the energy-loss to depth relation of a proton beam in a medium. At this range, the protons rapidly slows down, depositing large amounts of energy[30].

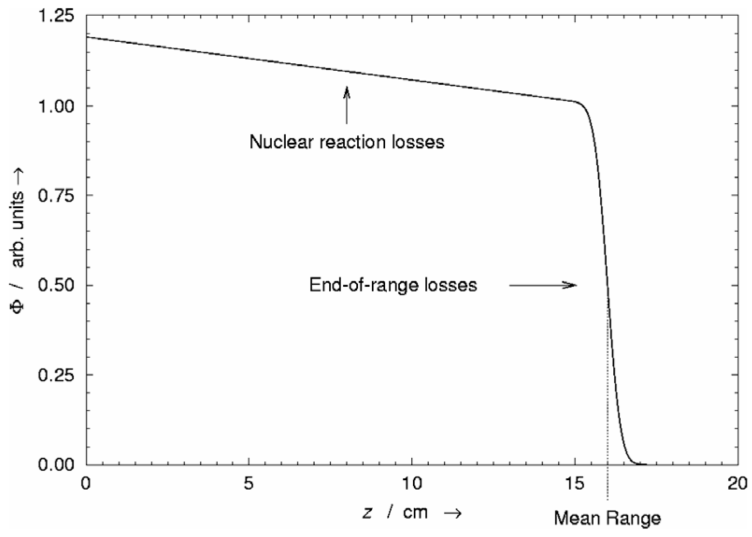


Figure 6: Relative fraction of the fluence in a broad beam of protons remaining as a function of depth z in water. The gradual depletion of protons from entrance to near the end of range is caused by removal of protons from nuclear reactions. The rapid falloff in the number of protons near the end of range is caused by ions running out of energy and being absorbed by the medium. The sigmoid shape of the distal falloff is caused by range straggling or by stochastic fluctuations in the energy loss of individual protons[29].

2.2.2 The Spread Out Bragg Peak

Figure 7 shows a depth-dose relation of proton beams (blue) and a photon beam (black) in water. It displays how the absorbed energy in water varies with depth. As the beam energy increases, the protons will penetrate further, as it will take more collisions/interactions to slow them down sufficiently to eventually halt them, and therefore the Bragg peak will reach further into the medium. The beam range, or the depth of the protons, is therefore dependent on the initial beam energy[21, 30].

If we radiate the medium with proton beams of different energies, they will deposit their energy at different ranges. In other words, their peaks will be slightly shifted. Superposing them gives the total depth to dose relation seen in figure 7, and we get the Spread Out Bragg Peak (SOBP)[21]. The figure shows the SOBP (red graph) plotted together with the depth-dose curve of a photon (black graph) with a certain energy. The area between the two curves is the amount of dose which the patient can be spared of using proton therapy instead of photon therapy. The fall-off of the proton-beam is a lot steeper than that of the photon-beam, spearing even more healthy tissue after the tumor volume [23].

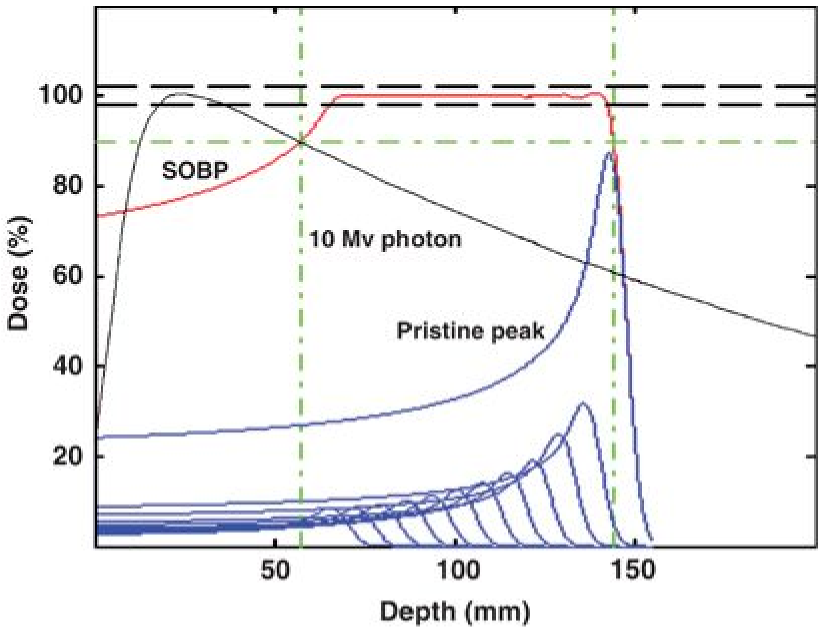


Figure 7: The figure shows how Bragg peaks of different energies can be superposed to give a plateau of dose to a given segment. The depth-dose profile of the photon-energy with the same peak energy is also graphed. The green vertical lines gives the depth of 90% of the front and distal dose. The dashed black lines gives the clinically accepted variation in the plateau dose of ±2%.[23]

2.2.3 Range Straggling

Each incident particle has its own unique path, with varying amounts and nature, of collisions. These statistical variation in the range of the particles, causes a non-zero width of the Bragg peak. If there were no statistical variation, and no lateral spread, the peak would be infinitely thin for a mono-energetic beam, not considering Heisenbergs uncertainty principle. This statistical range variation is called range straggling [13, 15], and is visualized in figure 8. The steepness of the distal fall-off of the dose decreases with initial beam energy[14]. As the beam energy increases, so does the range and the total number of interactions the protons have with the medium, causing a wider Bragg peak[13, 30].

The particle accelerator which shoots the high energetic protons towards the medium has an uncertainty in its ability to produce mono-energetic particles. This spread in energy, and by the energy-momentum relation, the spread in momentum causes more range straggling.

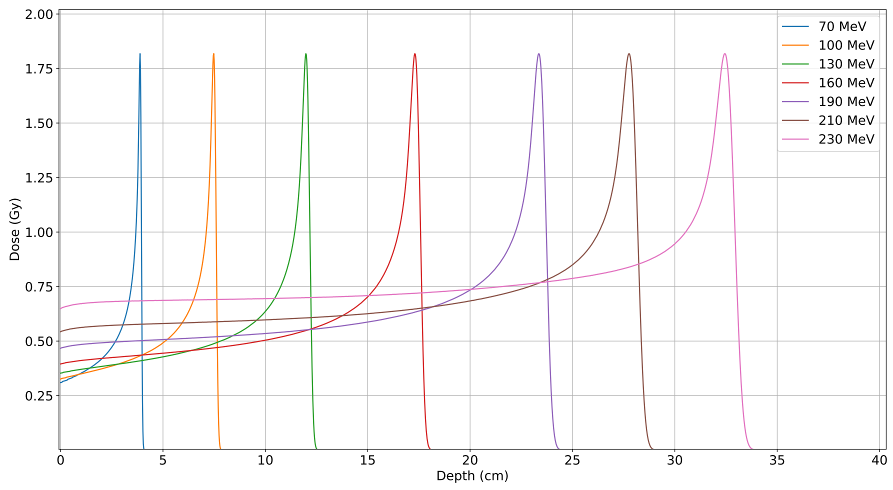


Figure 8: Depth-dose curves for proton beams of seven different energies. The figure illustrates how the range straggling, and consequently the width of the peak increases with energy.

3 The Radiobiology of Proton Therapy

The high-LET protons cause more ionization per unit length than the photons. This increase in LET needs to be accounted for when quantifying the damage and the biological effects caused by proton radiation. This section, based on [27], will present how we quantify the biological effects and the differences in how particles damage the cells and how we model this difference.

3.1 Dosimetry

The damage from radiation may not be apparent immediately after exposure, but can occur after several years. The quantification of biological effects of radiation is therefore difficult, but normally we assume that the biological effects increase with the amount of damage, and thereby by the amount of energy absorbed in the cells[35]. The International Commission on Radiation Units (ICRU) defines absorbed dose, D, as the mean energy imparted by ionizing radiation, ΔE, to a certain mass, Δm[34]

$$D = \frac{\Delta E}{\Delta m}. \tag{3}$$

The dose, D, has units of Gray (Gy), and $1Gy$ is defined as absorption of $1J$ of radiation per kilogram of material/tissue. Δm is here the mass of the Region of Interest (ROI). Given a mass density, ρ, we have that $\Delta m = \rho \Delta V$. Then ΔV is the volume of the ROI. The mean energy absorbed, ΔE in the tissue is related to the energy-loss of the incident particles, as energy is conserved. However, as mentioned in section 2.1.3, some of the energy lost by the particles could be transferred to a secondary particle, which could deposit its energy outside the ROI. Therefore, the energy-loss of the particles and the absorbed energy in the tissue is not necessarily equal, but are closely related.

3.2 Radiation damage to cells

As the radiation excites/ionizes the atoms of the material, it changes the atoms properties, and thereby also the properties of the molecules. This could break bonds within or between molecules, causing functional damage to the cell. If this happens in the DNA-molecule, it could change the functions of the cell, depending on whether this part of the DNA is active or not[20].

3.2.1 The Linear Quadratic model

Given a radiated dose D to a target volume of tissue, we can estimate the fraction of cells that survive the radiation. In radiation therapy, this is most commonly done by using the linear-quadratic (LQ) model for cell survival[22]:

$$SF = e^{-\alpha D - \beta D^2}. \tag{4}$$

α and β are tissue-parameters describing the sensitivity or repairing ability of the tissue. The α represents the amount of cells which are killed by one interaction, while β represents repairable damage, and/or the cells that aren't so sensitive to the radiation. The ratio α/β is called the "intrinsic radiosensitivity" or "fractionation sensitivity" of the tissue and is more commonly used than the two parameters separately[20].

3.2.2 Effects of LET on Cell Survival

Ionizing radiation can interact with the DNA in the cells in two ways. Indirect action happens when the radiation ionizes the liquid around the DNA-molecule, creating free radicals, such as hydroxyl (OH^-), that damages the DNA[17]. Direct action is when the ionizing radiation directly interacts with the molecules in the DNA. As the LET for protons is very large towards the end of its range, more damage is done to the DNA molecules, resulting in a steeper and straighter survival fraction curve[20], as can be seen in figure 9.

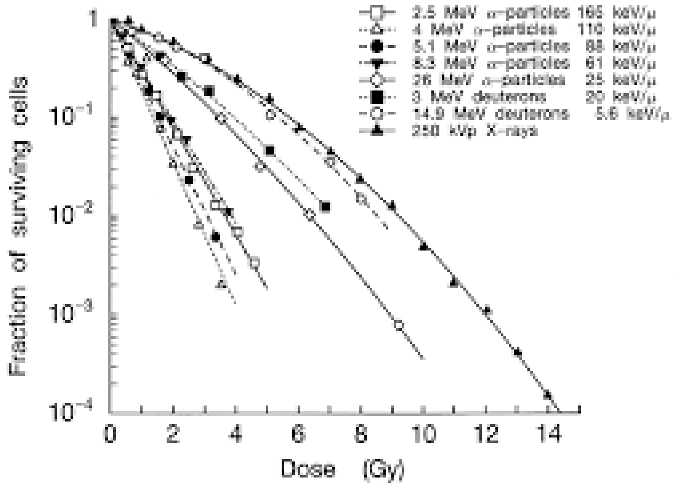


Figure 9: Survival fraction curves of cells irradiated by high LET (α-particles) and low LET (X-rays) plotted against dose levels. The high LET radiation leads to straighter survival curves than for the low LET radiation[1].

3.2.3 Radiosensitivity of Different Tissues

Cancerous cells often have multiple and larger cell nuclei compared to normal cells[40], making them more sensitive to radiation. These differences in tissue response to radiation, and the fact that cells are more sensitive to radiation at certain stages in their cell division cycle[20], indicates that dividing the treatment into multiple sessions with specific time between sessions will improve the quality of the treatment. That is, less damage to the healthy tissue, without compromising the damage to the target volume too much. This technique is called fractionation and is widely used in radiation therapy today. By comparing the survival fraction curves for the cancerous tissue and the healthy tissue, the physicians are able to find a "window of opportunity", where the difference between the survival fraction of cancerous cells versus normal cells is optimal[20], as illustrated in figure 10.

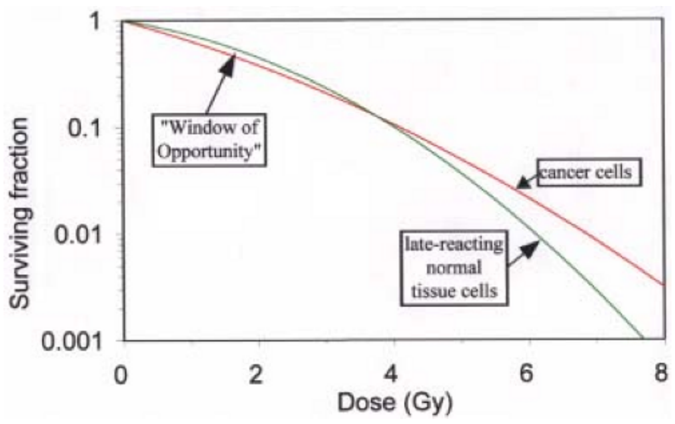


Figure 10: Qualitative illustration of the survival fraction of cancerous cells versus late-reacting normal tissue cells[10].

3.2.4 Relative Biological Effectiveness

As discussed in 3.2.2, the difference in LET of different types of incident particles, causes different amount of functional damage to the tissue, in other words: different biological effects for the same physical dose. The Relative Biological Effectiveness (RBE) is defined by

$$RBE = \frac{D_x}{D_p}, \tag{5}$$

where D_x and D_p causes the same biological effects [20]. D_x is the dose from a reference radiation, for example gamma or MeV x-ray, while D_p is the dose from a proton beam. Figure 11 shows how RBE varies with the LET. Clinical proton therapy use the assumption that protons are 10% more effective than photons for the same biological effective dose. This corresponds to a constant RBE of 1.1. However, multiple studies suggests that as the RBE increases with increasing LET and therefore varies within the patient[25, 33, 36].

3.2.5 Biological Range Shift

The increase in RBE with depth, tissue-type and dose in the variable RBE models causes a shift in the distal dose range[5, 14], causing dose to be given to distal tissue not accounted for when using constant RBE. In this project, we define the biological range shift to be the longitudinal extension of the 80% iso-dose level when using a variable RBE model instead of the constant RBE of 1.1 ($RBE_{1.1}$). Figure 2 visualizes the biological range shift for a generic SOBP.

3.3 RBE modellings

Most of the RBE models suggested in the last years are based on the linear quadratic model. Using traditional x-ray radiation as a reference, we can use the LQ model (4) to find the dose of x-ray radiation D_x given a dose D_p from radiation with protons which gives the same survival fraction, SF.

$$\alpha_x D_x + \beta_x D_x^2 = \alpha_p D_p + \beta_p D_p^2 D_x = \frac{\sqrt{\alpha_x^2 + 4\beta_x D_p(\alpha_p + \beta_p D_p)} - \alpha_x}{2\beta_x} \tag{6}$$

Dividing by D_p gives us the RBE of the protons relative to the photons. Biological dose [Gy(RBE)] is given as the product of RBE and physical dose, D_p. The reference photon radiation

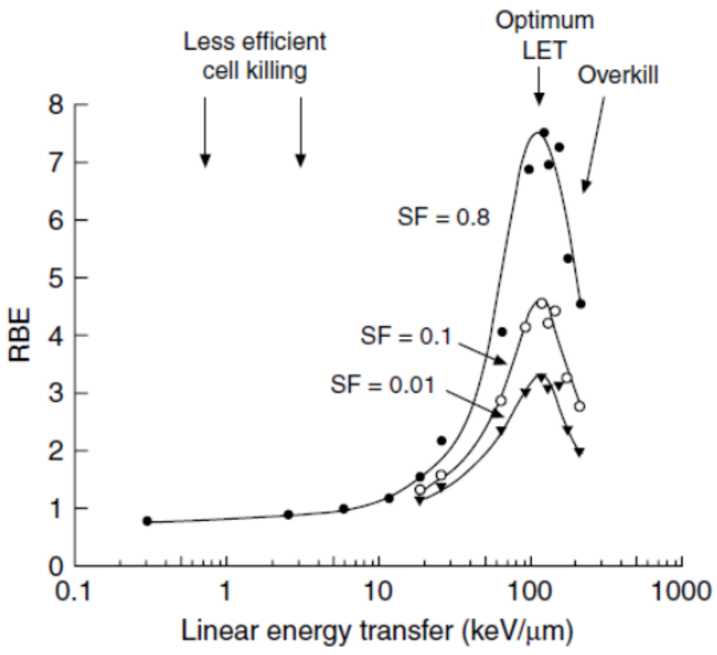


Figure 11: RBE as a function of LET, calculated at surviving fraction (SF) levels of 0.8, 0.1 and 0.01 for helium-ions[20].

is commonly $250kVp$ (peak kilo voltage) X-rays or ^{60}Co gamma-rays[20].

$$RBE = \frac{D_x}{D_p} = \frac{1}{2D_p}\left(\sqrt{\left(\frac{\alpha_x}{\beta_x}\right)^2 + 4D_p\left(\frac{\alpha_p}{\beta_x}\right) + 4D_p^2\left(\frac{\beta_p}{\beta_x}\right)} - \left(\frac{\alpha_x}{\beta_x}\right)\right). \tag{7}$$

The α_p and β_p are assumed to depend on the LET_d in the variable RBE models, and we define RBE_{max} and RBE_{min}, to contain this LET-dependency, giving

$$\begin{aligned} RBE &= \frac{1}{2D_p}\left(\sqrt{\left(\frac{\alpha}{\beta}\right)_x^2 + 4D_p\left(\frac{\alpha}{\beta}\right)_x RBE_{max} + 4RBE_{min}^2 D_p^2} - \alpha_x\beta_x\right) \\ RBE_{max} &= \frac{\alpha_p}{\alpha_x}, \quad RBE_{min} = \sqrt{\frac{\beta_p}{\beta_x}}. \end{aligned} \tag{8}$$

The RBE_{max} and RBE_{min} can be estimated using regression of data collected from measurements, mainly from irradiation of cells. How these parameters are estimated can vary, and hence there are many different models for the RBE of protons. Differences in assumptions on how the RBE_{max} and RBE_{min} is dependent on the LET, and difference in datasets leads to different results in RBE. The general trend for variable RBE models is a sharp increase towards the end of the beam range. More details on these differences in RBE models are available in other articles[37].

3.3.1 The Rørvik model

The RBE model by Rørvik et. al. (ROR) uses a Biological Weighing Function (BWF) to account for the dependency of the full dose weighted LET spectrum. The usage of BWF for the LET spectrum is based on microdosimetric RBE models[32], and the BWF is defined as $r_{max}(L)$ and the model for RBE_{max} is given as [36]

$$RBE_{max} = \int_0^\infty r_{max}(L)d(L)dL. \tag{9}$$

The ROR model applied a strict inclusion criteria for its database, accepting only monoenergetic experiments. Interpolating polynomials of up to fifth degree to a dataset of 85 points from aerobic in vitro cell experiments, the most fitting model for the BWF was found to be

$$\begin{aligned} r_{max} = 1 + \frac{Gy}{(\alpha/\beta)_x}(0.578 \left(\frac{keV}{\mu m}\right)^{-1} L - 0.0808 \left(\frac{keV}{\mu m}\right)^{-2} L^2 + 0.00564 \left(\frac{keV}{\mu m}\right)^{-3} L^3 \\ - 9.92 \times 10^{-5} \left(\frac{keV}{\mu m}\right)^{-4} L^4), \quad L < 37.0 keV/\mu m \end{aligned} \tag{10}$$

Based on other models [5, 25] including β_p which demonstrated only small deviations of RBE_{min} from 1, the RBE_{min} was assumed to be constant equal to 1 [36].

3.4 Treatment Planning Process

As a new patient is diagnosed with cancer, the treatment planning process starts. First the patients anatomy must be mapped, and tumor(s) and Organs at Risk (OAR) must be located. Then the choice of treatment method must be chosen, and a treatment regiment must be made. The fields must be defined, and the dose must be calculated. The goal is to make a treatment plan that delivers an homogeneous prescribed dose to the tumor volume, with minimal median dose to the surrounding tissue[30].

3.4.1 Treatment Planning

The physician studies the images of the patient (CT scans and MRI), and delineates the Gross Tumor Volume (GTV), the Clinical Target Volume (CTV) and the OARs. A margin of error is added to the CTV, to ensure that uncertainties in the set up and patient geometry will not compromise the dose to the tumor. This new volume is the Planned Target Volume (PTV). The prescribed dose and the fractionation schedule is chosen as well as the preferred method of delivery, be it photon therapy, brachytherapy or particle therapy[21].

Today, Treatment Planning Systems (TPS) uses inverse planning to optimize the dose distribution for each patient case. The medical physicist sets up the fields and defines objectives and constraints on the dose distribution, and the TPS then calculates the beam energies and the number of particles for each energy needed to reach the desired objectives[21].

3.4.2 Treatment Delivery

In proton therapy, there are two main techniques: Passive beam shaping and active beam shaping. Active beam shaping, also known as Pencil Beam Scanning (PBS), uses electromagnets to guide the beam of charged particles in the horizontal and the vertical direction. The beam does not go through a filter, and is therefore not scattered. To get the SOBP dose distribution we desire in the longitudinal direction, we need beams of varying energy. This is achieved simply by varying the energy of the beam. This is called Intensity Modulated Proton Therapy (IMPT). The target volume is divided into layers of a certain thickness, depending on the complexity of the volume. These layers are then divided into a grid, forming small volumes called voxels, see figure 12. Each voxel is radiated fully, then the beam moves to the next one. There are several techniques on how to sequence through all the voxels. Some techniques involve turning the beam off when moving to the next voxel, while some keeps the beam on during the entirety of the treatment[30].

Passive beam shaping uses rotating filters of varying thickness to attenuate the beam, causing a variation in energy, and hence a SOBP. The thin beam of the active beam shaping technique has the advantage of precision, but at the cost of much heavier computation work for the treatment planning system[30].

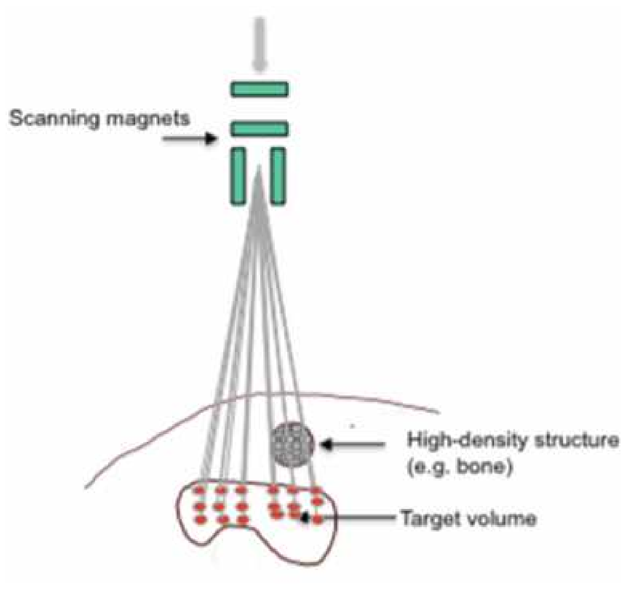


Figure 12: Pencil Beam Scanning illustration. The PTV is divided into voxels which are radiated one by one. Scanning magnets control the lateral direction of the beam, and the beam energy controls the longitudinal range[30].

4 Methods

To construct a predictive model for the distal biological range shift in proton therapy, we first quantified the biological range shift of mono-energetic proton beams (PBPs) in water. Then used linear regression to obtain a predictive model. We did this for SOBPs as well. We then studied the differences in the two methods, PBPs versus SOBPs. Finally, we used these datasets to make predictions on biological range shifts in a SOBP in a water-tank and a prostate cancer treatment plan. The biological range shifts in these two scenarios were then measured and the accuracy of the predictive models were evaluated. The ”measured” range shift will refer to the range shift found in the two scenarios after simulations.

4.1 Pristine Bragg Peaks

As the range shift is energy dependent [14], Monte Carlo simulations of mono-energetic proton beams in water were performed with 17 different initial beam energies, corresponding to 17 different depths. Biological dose distributions for the ROR and $RBE_{1.1}$ models were calculated based on the physical dose deposition to water and the LET. This was done for 8 different dose levels ($0.5Gy(RBE)$ to $4Gy(RBE)$) and in 10 different types of tissues with $(\alpha/\beta)_x$ ranging from $1Gy$ to $10Gy$. The depth of the distal 80% fall off, r_{80} we define as the depth where the dose in the distal edge of the Bragg peak has fallen to 80% of the prescribed dose level. The r_{80}s were then extracted from one-dimensional plots of the depth-dose relation along the central axis of the beam. Comparing the ranges of the dose distributions of the ROR and $RBE_{1.1}$ models, a dataset, containing $17 \times 8 \times 10$ values, of the biological range shift were found. Then from this dataset, a predictive model of the biological range shift was made, and tested on a SOBP in a water tank and a prostate treatment plan. Figure 13 shows a flowchart of the process of making the predictive models and the following sections will further elaborate on each step taken.

4.1.1 FLUKA Monte Carlo Simulations

For the simulations the FLUKA Monte Carlo[3, 8] code was used, with the help of the complementary FLUKA advanced interface (FLAIR). Using an artificially made CT image of a water tank, and a proton treatment plan of a PBP of 150 MeV in the tank, made in the Eclipse TPS at HUH by Lars Fredrik Fjæra, the FLUKA simulations were set up. Taking inspiration from [27], we checked the scoring resolution along the beam line. The treatment plan we used to set up the FLUKA simulations with had scoring-bins, registering dose to water and LET every $2mm$. As we were concerned with differences in ranges on a low scale, we increased the resolution of scoring-bins to every $0.2mm$ to determine the biological range shifts with a high accuracy. This was done by increasing the size of the ”Pixel Data” and the ”Grid Frame Offset Vector” in the beam direction and increasing the number of frames in the RT.Dose file from the TPS. The RT.Dose file contains the planned dose distribution of the treatment plan made in the TPS.

A FLUKA input file (screenshot of an example of an input file in flair shown in appendix A) was made, defining the scoring region, materials, calibration curves, translating Hounsfield Units (HU) to stopping power, and the material densities of the set-up. The scoring regions and materials are based on the treatment plan files (RT.Plan, RT.Struct, RT.Dose) exported from the TPS. Using a compiler scoring script, provided by the PTG at UiB, the dose to water, LET and RBE_{max} for the ROR model (see equation (9)) are measured. These properties are needed to calculate the RBE-weighted dose of proton beams. This is done by a script provided by the Particle Therapy Group (PTG) at the University of Bergen (UiB)[9].

The chosen energies cover the clinically relevant range, from 70 MeV, up to 230 MeV with an increment of 10 MeV, giving a total of 17 different PBPs. Custom beam cards were made for each of these energies, example shown in appendix A, with a Gaussian beam profile with a FWHM of 2 cm. Momentum spread was calculated from calibration curves, fitted to measurements of momentum spread for various beam energies, see appendix B. Dose deposited in the water tank, the LET and the RBE_{max} of the ROR model, for 10 different $(\alpha/\beta)_x$ values ranging from 1 Gy to

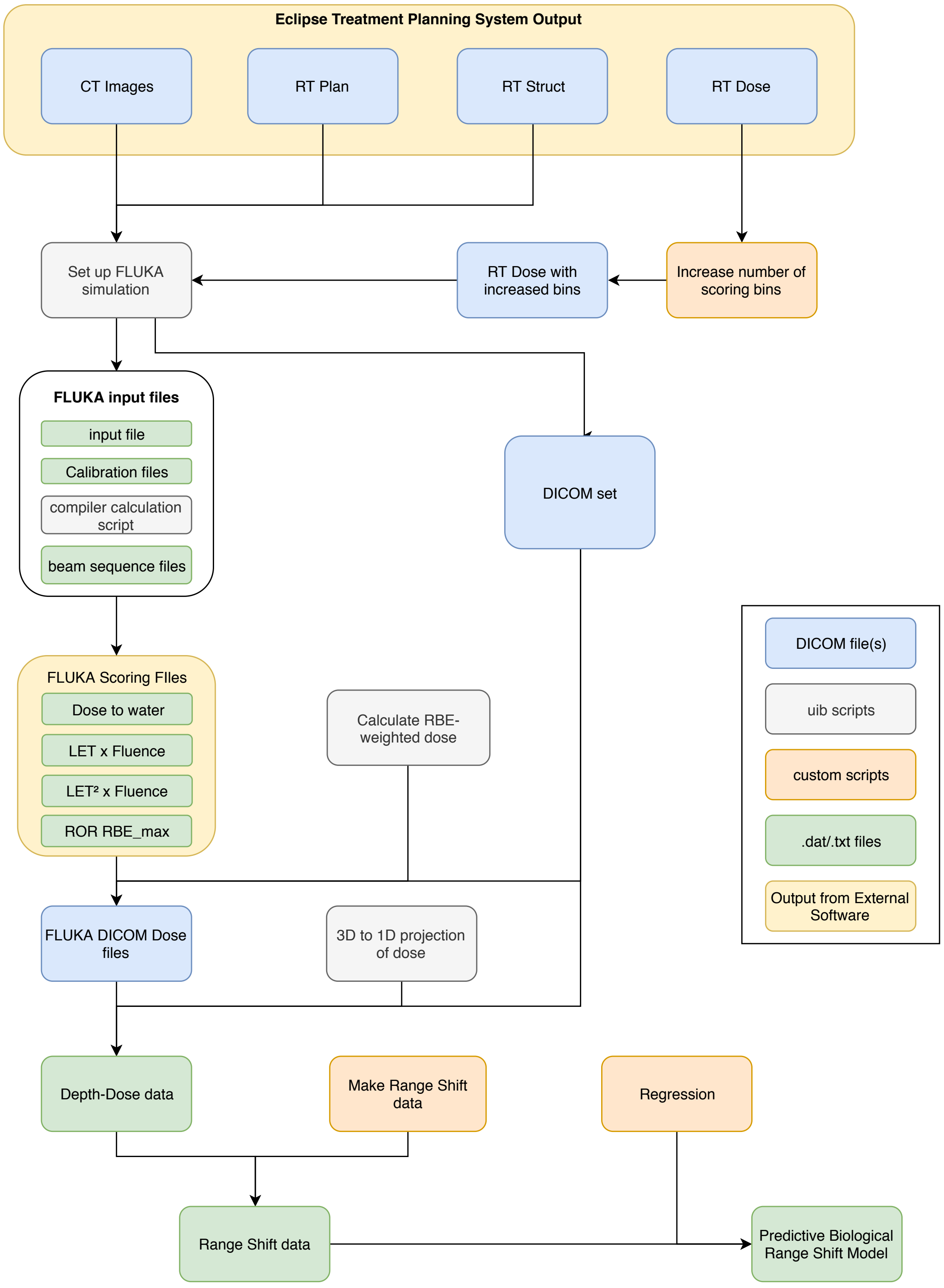


Figure 13: Flowchart of the process of the method used in this project. The different colors are described in the box to the right. "Custom scripts" are the script made in this project, listed in appendix E.

10 Gy, were scored such that RBE-weighted doses could be calculated. The simulations were run with 4×10^7 particle histories, ensuring a statistical uncertainty in the dose below 0.02%.

4.1.2 RBE-weighted Depth-Dose Calculation

RBE-weighted depth distributions were calculated for all the 17 energies for the $RBE_{1.1}$ and ROR models (see equations (8) and (9)) giving 8 different dose levels, ranging from $0.5Gy(RBE)$ to $4Gy(RBE)$ with an increment of $0.5Gy(RBE)$ in 10 different tissues with $(\alpha/\beta)_x$ ranging from 1 Gy, up to 10 Gy. The PBPs had to be normalized to give the exact wanted dose levels in the peak. Additionally, as the beam energy increases and the range of the proton increases, the height of the depth-dose curve decreases, due to range straggling. Therefore, each PBP had to be normalized separately. To do this, we first calculated the $RBE_{1.1}$-weighted dose for each energy without normalizing them, that is: calculating the dose per primary (proton) particle, which is the standard output from FLUKA. Then, calculating the depth-dose distribution and extracting the peak dose, we would find the normalization factor $1/peak\_dose$. Using this normalization factor, we would be able to construct dose distribution giving a peak dose of 1 $Gy(RBE)$ for each energy. To achieve the various dose levels, these normalization factors were simply multiplied by a factor *desiredDoseLevel*, which for example would be 2.5 for a dose level of 2.5 $Gy(RBE)$.

The central $2cm \times 2cm$ cross-sectional area of the beam was projected onto the beam axis. As the beam profiles were Gaussian with a FWHM of $2cm$, and the proton beam widens significantly with depth, this was considered a reasonable projection area. The resulting depth-dose relations were then written to files, which would be used to produce the biological range shift dataset.

4.1.3 Range Shift Calculation

The terminology $d_{80}^{RBE_{1.1}}$ corresponds to 80% of prescribed $RBE_{1.1}$-weighted dose. $r_{80}^{RBE_{1.1}}$ and r_{80}^{ROR} correspond to the distal range where the dose level has fallen to $d_{80}^{RBE_{1.1}}$ in $RBE_{1.1}$ and ROR models, respectively.

The script *makeData.py*, listed in appendix E, reads all the depth-dose files and pairs up each ROR-weighted depth-dose relation with the constant RBE-weighted depth-dose relation. Specifically, it matches the beam energy and the dose level. For each energy and dose level, it finds the $r_{80}^{RBE_{1.1}}$ and r_{80}^{ROR} for the ROR model in all the different tissues ($(\alpha/\beta)_x$-values). The method for finding $r_{80}^{RBE_{1.1}}$ and r_{80}^{ROR} are the same, so we use "r_{80}", as the general formulation for the following paragraph.

r_{80} is found by iterating backwards starting from the last value in the depth dose distribution, until the dose has risen above $d_{80}^{RBE_{1.1}}$. We now use linear interpolation[24] to extract the depth where the dose is equal to $d_{80}^{RBE_{1.1}}$. The assumption that the point where the dose is equal to $d_{80}^{RBE_{1.1}}$ lies on the line-segment between the two closest data-points is a decent approximation as the slope of the dose fall-off is steep, and the resolution (0.2 mm) is fairly high. The slope of the line segment is constant giving us the relation (11). Re-organizing, we get an expression for r_{80}, (12).

$$\frac{r_{80} - r[j]}{d_{80}^{RBE_{1.1}} - d[j]} = \frac{r[j-1] - r[j]}{d[j-1] - d[j]}. \tag{11}$$

$$\begin{aligned} r_{80} &= (1-x) \cdot r[j] + x \cdot r[j-1], \\ where \quad x &= \frac{d_{80}^{RBE_{1.1}} - d[j]}{d[j-1] - d[j]} \end{aligned} \tag{12}$$

The $r_{80}^{RBE_{1.1}}$ is then subtracted from the r_{80}^{ROR} of ROR model for all $(\alpha/\beta)_x$-values, giving the range shifts, $r_{80}^{diff} = r_{80}^{ROR} - r_{80}^{RBE_{1.1}}$. This is repeated for each energy and each dose level. We now have values for the biological range shifts between the ROR and the $RBE_{1.1}$ models for PBPs of 17 different energies (or beam ranges), with 8 different dose levels and 10 different tissue types per depth, constituting a dataset of in total 1360 data points.

4.2 Spread Out Bragg Peaks

The SOBP biological range shift dataset was based on biological range shifts from 12 2 cm wide SOBPs of different depths giving 8 different dose levels in 10 different tissues. Other than that, the method used to produce the dataset of biological range shifts is fairly similar to that of the PBPs, see section 4.1. The method for calculating the range shift dataset from the depth-dose relations is exactly the same as for the PBPs, see section 4.1.3.

4.2.1 Spread Out Bragg Peak Plans

Treatment plans for 12 SOBPs of different depths (ranging form 4 cm to 33 cm), and with a width of 2 cm, giving 8 different dose levels in tissues with 10 different $(\alpha/\beta)_x$-values were personally made with the Eclipse TPS at HUH in Bergen. They were made by defining PTVs of dimensions $4\ cm \times 4\ cm \times 2\ cm$ at different depths, then setting the prescribed dose to 2 $Gy(RBE)$. The TPS then calculates the beam sequence, that is; the beam energies and corresponding weights giving the wanted dose distribution. The Eclipse TPS scores along the beam axis every 2 mm, which is too low of a resolution for this project[27]. Therefore, after exporting the Eclipse TPS files, that is; CT Images, RT.Plan, RT.Struct, RT.Dose, the number of bins along the beam axis was increased 10-folds, by the same method described in section 4.1.1, giving scoring every 0.2 mm.

4.2.2 FLUKA Monte Carlo Simulations

The simulation of each SOBP treatment plan was set up, defining location of the water tank, scoring volumes etc, see section 4.1.1 for more details. The beam sequence, that is; the beam energies and their weights and lateral modulation, from the TPS will be used in the simulations of the SOBPs, contrary to the simulations of the PBPs, where they were not used. The simulations were run with 1×10^7 particle histories, ensuring statistical uncertainty below 0.04% in the dose distribution.

The compiler scoring script, as for the simulation of the PBPs, calculates the dose to water, LET and RBE_{max} for $(\alpha/\beta)_x$ ranging from 1 Gy to 10 Gy, with an increment of 1 Gy, for the ROR model. These properties are contained in the "FLUKA scoring files" in the flowchart of figure 13.

4.2.3 RBE-weighted Dose Calculation

The RBE-weighted dose distributions were calculated by the same method and for the same dose levels and tissue parameters as for the PBPs (see section 4.1.2). Projecting the central $2mm \times 2mm$ of the beam onto the beam axis, the depth-dose distributions were calculated. These were then used to produce a dataset of biological range shifts containing a total of 960 values.

The depth-dose distributions with various dose levels were calculated by multiplying the FLUKA dose distribution of each SOBP plan by a factor of $desiredDoseLevel \times norm\_factor$. The first factor is the same as the one used for the PBPs, see section 4.1.2. To ensure that the SOBPs were properly normalized, the second factor, $norm\_factor$, was also used. This factor was found by the same method as for the PBPs: Calculating the non-normalized dose distributions, then extracting the peak dose. However, as most treatment plans are constructed to give the prescribed dose to the entire PTV, and not just a peak dose equal to the prescribed dose, this method was not in line with the method used in treatment planning. Therefore the method was changed. Instead of the peak dose, an average of the dose in the SOBP was used. The SOBP was defined to be between the primal and distal 95% fall-off of the peak dose, see figure 14.

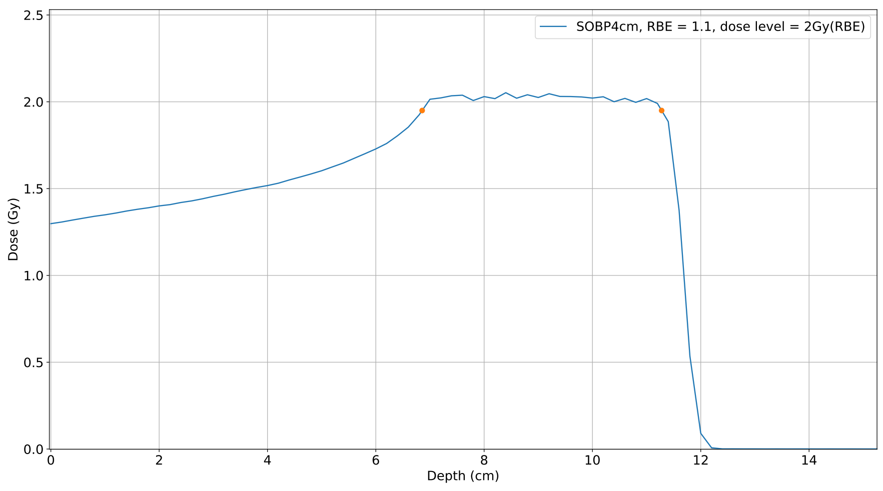


Figure 14: Visualization of the definition of the SOBP. The orange dots are the primal and distal 95% fall off of the peak dose. The dose between these points are averaged and used to normalize the SOBPs.

4.3 Range shift models

Linear regression of the range shift dataset and trilinear interpolation[39] of the closest data-points where used as predictive range shift models.

4.3.1 Linear Regression

We made a predictive model of the biological range shift based on the assumption that an explicit mathematical expression for the range shift would be a linear combination of both independent and coupled terms of the three variables dose level, beam energy and $(\alpha/\beta)_x$ of multiple degrees. An example of such a model is shown in equation (13), where D, E, AB, corresponds to the prescribed dose level, initial beam energy and $(\alpha/\beta)_x$ values, respectively. Introducing the coefficients c_{ijk} of the term with D, E, AB of degree i, j, k, respectively we can write a compact general formula, seen in equation (14), where n is the maximum total exponent of the variables, D, E, AB. Terms of higher orders are ignored, that is: $i + j + k <= n$ in all terms. As an example, the coefficient c_{213} will in our example be the coefficient of the last term, -1.8.

$$RS(D, E, AB)_6 = 2D + 0.01E^2 + 1.2D^2E - 1.8D^2EAB^3 \tag{13}$$

$$RS(D, E, AB)_n = \Sigma_{i=0}^{n}\Sigma_{j=0}^{n-i}\Sigma_{k=0}^{n-i-j} c_{ijk}D^iE^jAB^k \tag{14}$$

We then introduce the input variables $x_{ijk} = D^iE^jAB^k$. We now have a multivariable linear regression problem where we want to find the coefficients c_{ijk} given the dataset of the input values $x_{ijk} = D^iE^jAB^k$. In the function *reg_sklearn()* in *supplementary.py*, we use the *sklearn* library [38] and the function *LinearRegression()* from the tool *linear_model* to find the coefficients, c_{ijk}. *linear_model.LinearRegression()* is a least squares linear regression, which minimizes the sum of the squared residuals

$$S_{res} = \Sigma_{d,e,a}r_{dea}^2 = \Sigma_{d,e,a}(f(x_{ijk}, c_{ijk})_{dea} - y_{dea})^2, \tag{15}$$

where r_{dea} are the residuals given the inputs d, e, a, corresponding to the dose level, beam energy and $/\alpha/\beta)_x$. The Standard Deviation (STD), $STD = S_{res}/N$ where N is the number of data-points, is used as a measure of the accuracy of regression models. It reflects the models ability to reproduced the data-points, y_{dea}. The regression is done in the script *make_RS_model.py*, with *supplementary.py* as a complimentary script, see appendix E.

4.3.2 Trilinear Interpolation

Even though the inputs (max beam energy, dose level and $(\alpha/\beta)_x$) most likely miss the exact range shift data-points, an approximation of the range shift can still be made. Figure 15 visualizes the method of trilinear interpolation[39]. Each of the lattice points corresponds to a max beam energy, dose level and $(\alpha/\beta)_x$ value. An input point (c) will then be located in a cube with corners in the 8 closest lattice points (c_{000}, c_{001}, ...). Weighting the contributions from each of the 8 lattice points (c_{000}, c_{001}, ...), using linear interpolation[24], see equations (12), we reduce the 3-dimensional problem to a 2-dimensional one with the 4 lattice points (c_{00}, c_{01}, c_{10}, c_{11}). Again we do a weighted average to find the c_0 and c_1, then one more time to find c, an estimate for the range shift. As this method will perfectly reproduce the data-points, as the point c will coincide with one of the data-points c_{ijk} in figure 15, no measure of accuracy is listed. This trilinear interpolation is done in the function *trilinear_interpolation()* in the script *supplementary.py*, see appendix E.

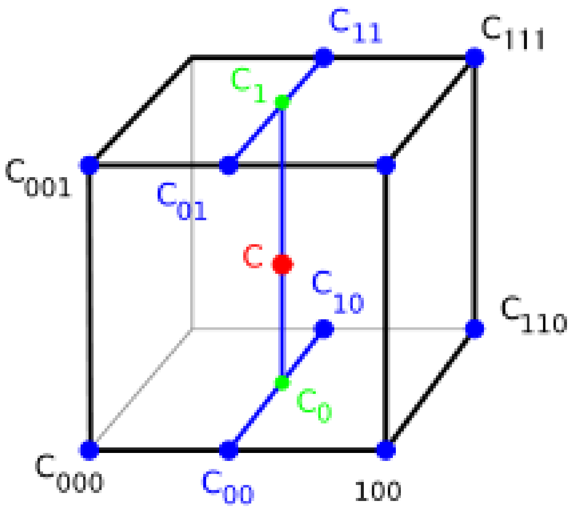


Figure 15: Visualization trilinear interpolation[39], which finds an approximate value c, located between known data-points, c_{ijk}.

4.4 Re-optimization of the SOBP plans

The SOBPs forming the basis of our model on was optimized to give a homogeneous dose of 2 $Gy(RBE)$. However, they had some irregularities, in particular a distinct peak towards the end of the SOBP. To check if this peak affected the predictions, one of the SOBP-plans were re-optimized using a prototype MC based optimizer available at UiB [19]. The SOBP with max energy closest to that of the max energy of the SOBP$_{4cm}$ plan was chosen. This was the SOBP with the third most shallow depth, referred to as SOBP3. Optimization of the SOBP3-plan with a max beam energy of 130 MeV was done by tailoring the beam sequence to give a more homogeneous dose distribution in the SOBP[19]. The results of the simulations were then converted to range shift values through the same method as before. The new dataset was then used to predict the range shift for the SOBP$_{4cm}$ plan.

4.5 Range Shifts in Patient Cases

The models presented in section 4.3 are used to predict the range shift in the water phantom plan and a prostate plan. These predictions are then compared to the range shifts measured from the dose distributions of the treatment plans achieved through Monte Carlo FLUKA simulations. The following subsections will present the two treatment plans, and describe how we used the predictive models to get a value for the range shift, as well as how we measured the range shifts from the dose distributions of the treatment plans. The steps taken to measure the range shift in the plans and predicting this range shift are shown in the flowchart in figure 16.

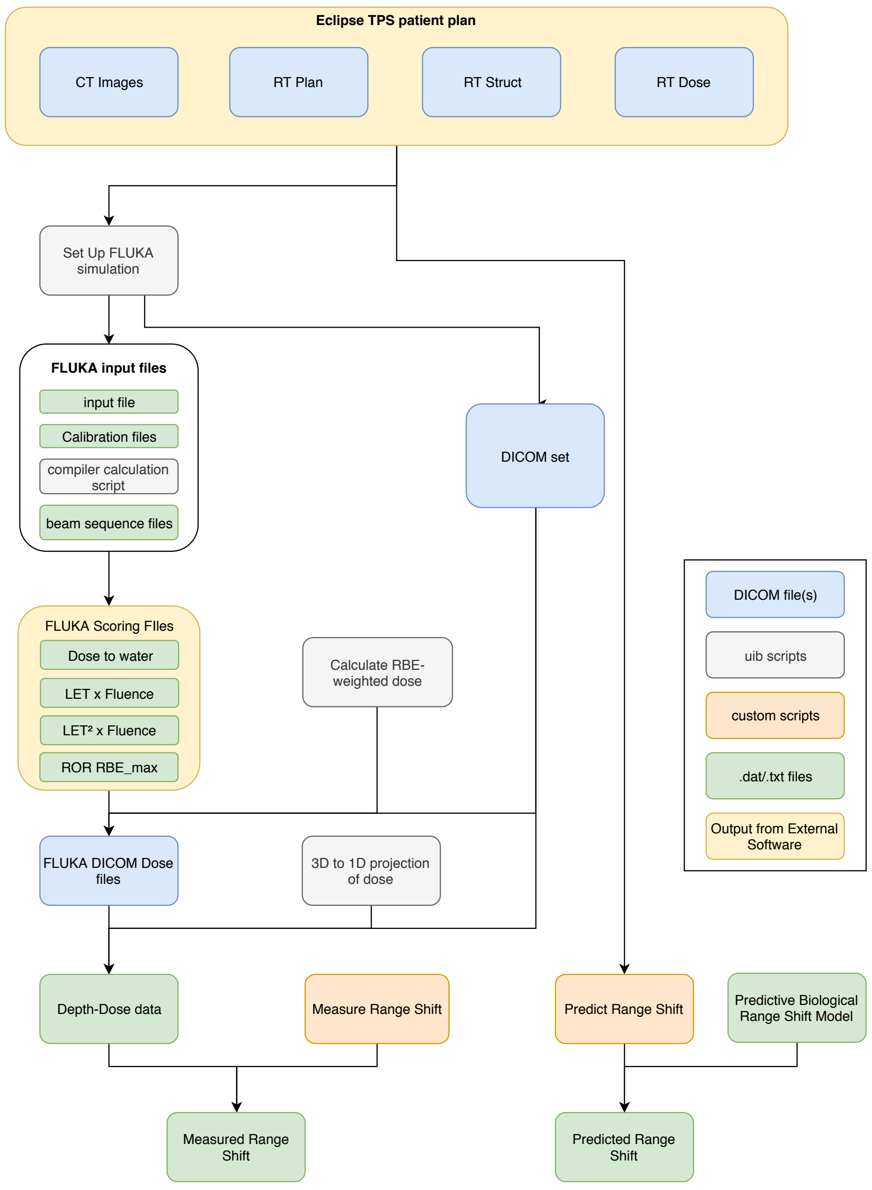


Figure 16: Flowchart of the process to measure the range shift and predicting the range shift using the models made in this project. The different colors are described in the box to the right.

4.5.1 Water Phantom SOBP$_{4cm}$

The water phantom plan is not a patient case, but it is of interest to see if our models are able to predict a range shift in a SOBP of 4 cm width in a water tank. The distal depth of the SOBP is at 11.2 cm, and the prescription dose is 2 $Gy(RBE)$. The plan consists of one single field, and a PTV of dimensions 4 $cm \times 4$ $cm \times 2$ cm. The plan is optimized to give a homogeneous dose of 2 $Gy(RBE)$ to 100% of the PTV. Figure 17 shows the planned dose distribution for a CT slice in the central part of the beam.

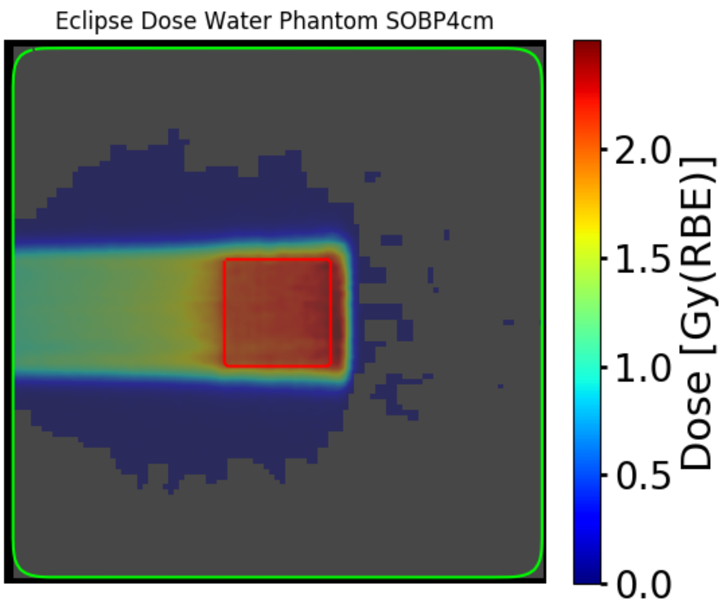


Figure 17: Planned dose distribution in the water phantom case. The contour of the water tank is marked as green, while the contour of the PTV is red. Fraction dose is set to of 2.0 $Gy(RBE)$.

4.5.2 Prostate Case

The treatment plan for the prostate case consists of two directly opposing fields. The planned dose distribution is visualized in figure 18. The contour of the body of the patient is barked in green, while the contour of the PTV is marked as red. The prescribed dose to the PTV is set to 67.5 $Gy(RBE)$, with 25 fractions, giving a fraction dose of 2.7 $Gy(RBE)$. The $(\alpha/\beta)_x$ was set to the value of the bladder and rectum (OARs), equal to 3.1[37].

4.5.3 Range Shift Measurements

To measure the range shift in the treatment plans, re-calculations of the treatment plans were done by Monte Carlo FLUKA simulations [9]. Then the RBE weighted depth dose was calculated from the FLUKA scoring files. In cases with multiple fields, all the fields must be calculated together, as the total RBE is dependent on the contributions from all fields. From the depth-dose distributions, the r_{80} for the $RBE_{1.1}$ and ROR models were found, and the range shift was calculated. The r_{80} was found in the same manner as for the PBPs and SOBPs, explained in section 4.1.3. The script *patient_RS.py*, see appendix E, reads the RT.Plan file and the depth-dose relations from the simulations and measures the range shifts.

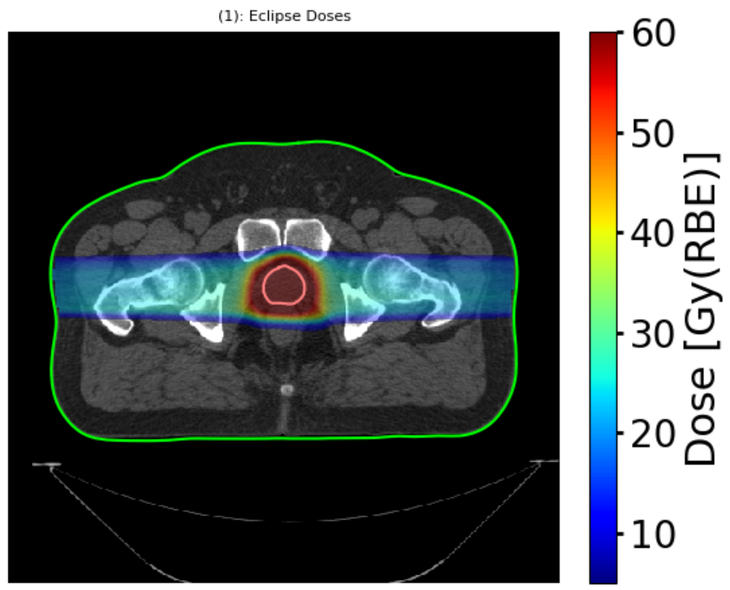


Figure 18: Planned dose distribution in the prostate case. The contour of the patients body is marked as green, while the contour of the PTV is red. Prescribed dose is 67.5 $Gy(RBE)$, with a fraction dose of 2.7 $Gy(RBE)$ delivered from two directly opposing fields.

4.5.4 Range Shift Prediction

Prediction of range shift is done by running the script *predict_RS.py*, see appendix E. Information on the fields are extracted from the RT.Plan file for the patient case of interest; max beam energies, fraction dose, gantry angle, patient name, plan name, etc. The user then inputs the $(\alpha/\beta)_x$ values of interest, and the directory of the predictive model-files made by *make_RS_model.py*. The user then chooses the predictive models to use, and the range shift for each field is calculated using fraction dose, equal to the sum of the beam doses, max beam energy, and $(\alpha/\beta)_x$ value.

4.6 Biological Range Shift in PBPs versus SOBPs

To check the impact of the SOBP-width on the range shift we set up simulation of 11 SOBPs with the same range, but different modulation widths, ie. different number of energies in the SOBP (ranging for 1 to 100). The SOBP$_{4cm}$ was the basis for these SOBPs. RBE-weighted depth-dose distributions were then calculated. The SOBPs/PBP were normalized to give a peak dose of 2 $Gy(RBE)$, as this was the normalization method for the PBP model. We will refer to this method as "peak weigh-method". To compare, the SOBPs/PBP were also normalized using the number of particles per energy listed in the beam delivery sequence in the RT.Plan file from the TPS, which would give a SOBP with a dose level of 2 $Gy(RBE)$ containing 11 energies. We will refer to this method as "TPS weigh-method". The biological range shift for these SOBPs/PBP, (for both?) normalized with the TPS weigh method, was then calculated.

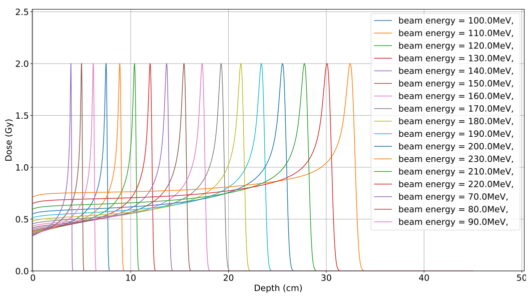
5 Results

In the following sections we will refer to the methods of using PBPs and SOBPs as basis for the range shift datasets as the "PBP method" and the "SOBP method", respectively.

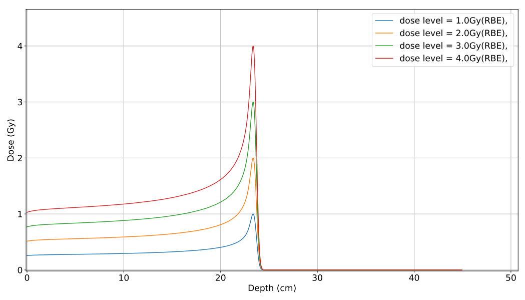
5.1 Depth-Dose Relations and Range Shift datasets

5.1.1 Pristine Bragg Peaks Method

The depth dose distributions of the PBPs of various energies and dose levels for $RBE_{1.1}$ are visualized in figure 19a and 19b. The PBPs have been successfully normalized to give the correct peak doses. The width of the PBP increases significantly with increased beam energy, from 1.1 mm to 8.3 mm for beam energies of 70 MeV and 230 MeV, respectively.



(a)



(b)

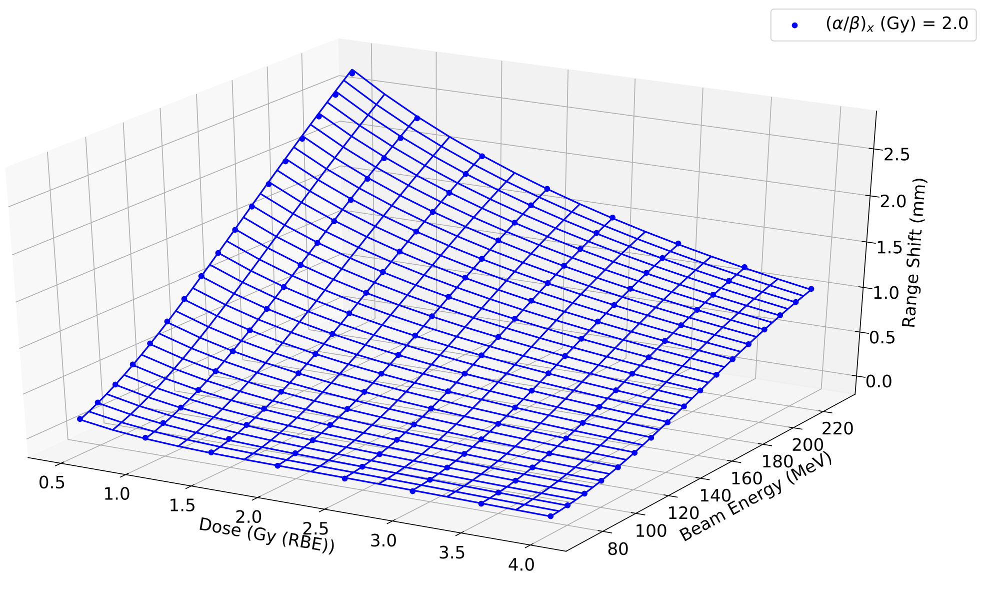
Figure 19: (a) PBPs for the 17 different energies and a dose level of 2 $Gy(RBE)(RBE)$. (b) PBPs with a beam energy of 190 MeV for four different dose levels.

Plotting the range shift data versus dose level and beam energy for an $(\alpha/\beta)_x$ of 2 Gy, and plotting the best fitting surface with max total exponent $n = 5$, see equation (14), we get a visualization of the range shift as a function of dose level and beam energy, seen in figure 20a. The fitted surface is produced by a regression model of $n = 5$, with a STD of 0.0059 mm, see appendix D for the values of the coefficients of equation (14).

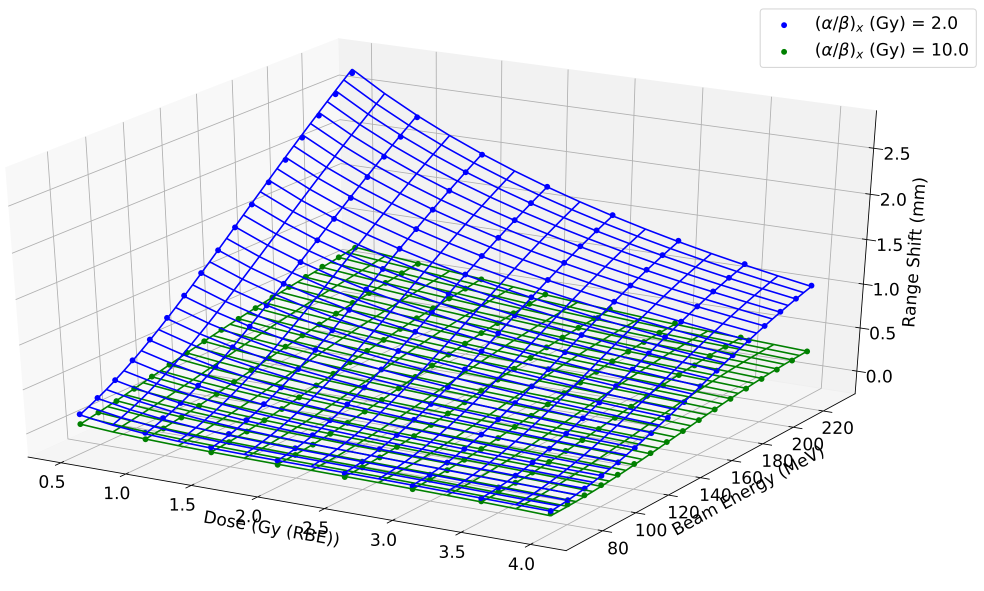
Adding the plot of a higher $(\alpha/\beta)_x$ of 10 Gy, we get figure 20b, with the same STD, as both surfaces are produced using the same regression model. Surfaces with $(\alpha/\beta)_x$ value between 2 Gy and 10 Gy will be located between these two surfaces, and $(\alpha/\beta)_x < 2$ Gy will be located above the $(\alpha/\beta)_x = 2$ Gy-surface. The surface with $(\alpha/\beta)_x = 10$ Gy is significantly flatter than the surface with $(\alpha/\beta)_x = 2$ Gy, indicating that the range shift is highly dependent on $(\alpha/\beta)_x$.

The range shift data contains values between -0.04 mm and 3.55 mm, with trends of increased range shift with increased beam energy, and decreased range shift with increased dose level and $(\alpha/\beta)_x$. The non-zero curvature of the surfaces indicate a non-linear relation between the range shift and the inputs.

Plotting the range shift versus each of the three parameters for selected values of the other two parameters, we get the plots in figure 21. The curves are produced by the regression model with $n = 5$, and a STD of 0.0059 mm. Figure 21 further illustrates the trend of increased range shift with increased energy and decreased dose level and $(\alpha/\beta)_x$ value. The energy and dose dependence of the range shift is fairly linear, the range shift is more inversely proportional to the $(\alpha/\beta)_x$ value. This curvature increases for increased beam energy and decreased dose levels. Figure 21c shows no signs of stagnation of the range shift at higher energies for low doses and $(\alpha/\beta)_x$ values.

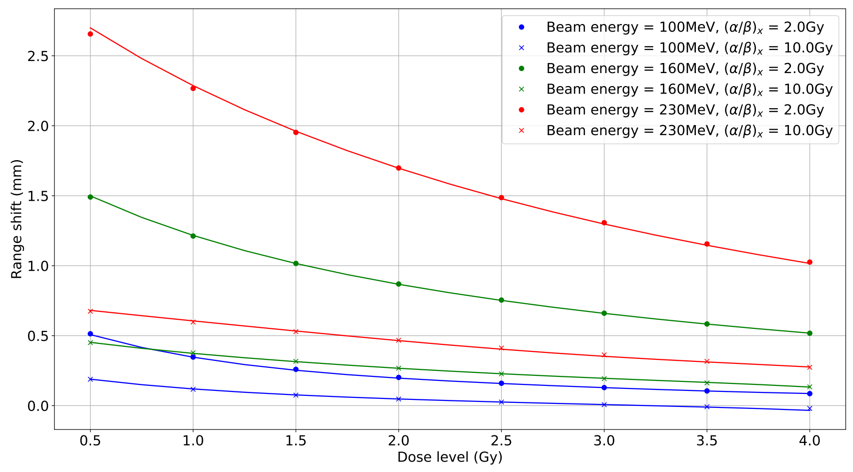


(a)

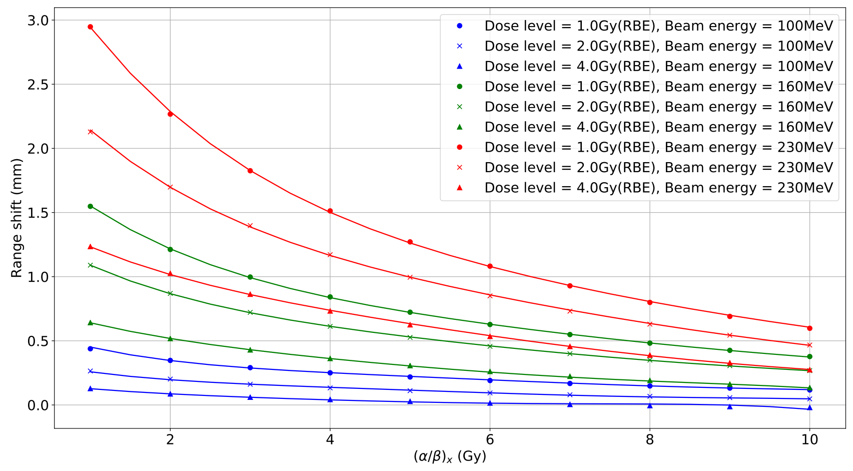


(b)

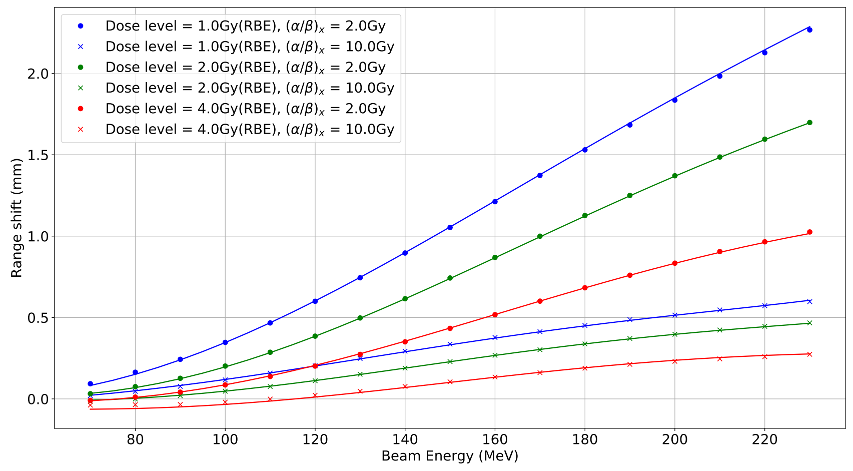
Figure 20: (a) Range shift surface for the PBP method as a function of dose level and beam energy for $(\alpha/\beta)_x = 2\ Gy$. (b) Range shift surfaces for the PBP method as a function of dose level and beam energy for $(\alpha/\beta)_x = 2\ Gy$ and $10\ Gy$.



(a) Biological Range shift as a function of dose level. The legend connects the curves to the beam energies and $(\alpha/\beta)_x$ values.



(b) Biological Range shift as a function of $(\alpha/\beta)_x$ values. The legend connects the curves to the dose levels and the beam energies.

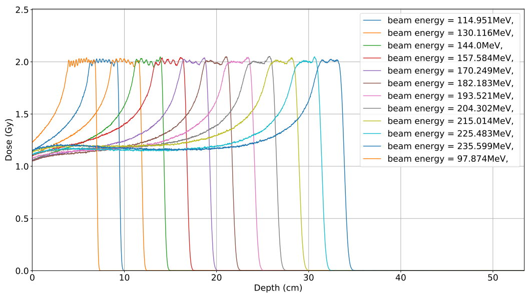


(c) Biological Range shift as a function of beam energy. The legend connects the curves to the dose levels and $(\alpha/\beta)_x$ values.

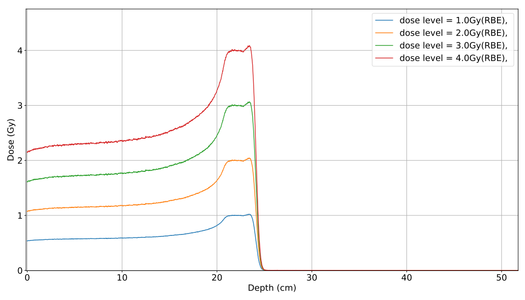
Figure 21: Biological Range Shift as a Function of Dose level, Beam energy and $(\alpha/\beta)_x$.

5.1.2 Spread Out Bragg Peaks Method

Figure 22a shows the SOBPs at different depths normalized to a dose level of 2 $Gy(RBE)$, and figure 22b shows SOBPs of two different depths giving doses of 1 $Gy(RBE)$, 2 $Gy(RBE)$, and 4 $Gy(RBE)$. The shape of the SOBPs are notably different from each-other, with the SOBPs with low max beam energy having the shape similar to saw-teeth and the SOBPs with high max beam energy having a higher, more distinct peak towards the end of the SOBP. The distinction of these peaks is enhanced with increased dose level, as can be seen in figure 22b.



(a)



(b)

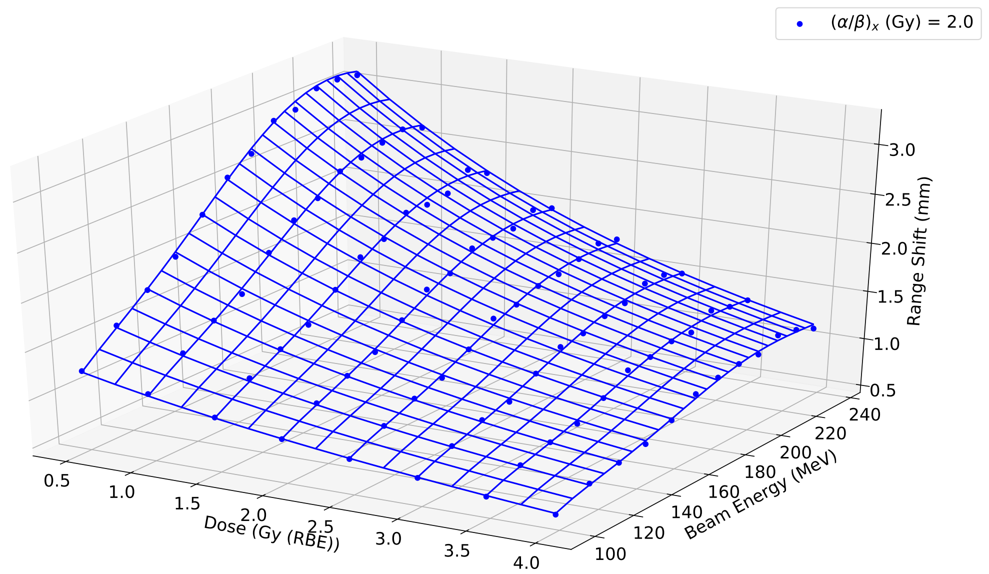
Figure 22: (a) SOBPs at 12 different depths and a dose level of 2 $Gy(RBE)$. (b) SOBPs with at a distal depth of about 23.6 cm for four different dose levels.

Plotting the range shift data versus dose level and beam energy for an $(\alpha/\beta)_x$ of 2 Gy, and plotting the best fitting surface using regression with $n = 5$, we get a visualization of the range shift as a function of dose level and beam energy, seen in figure 23a. The STD of the regression model is $0.030mm$.

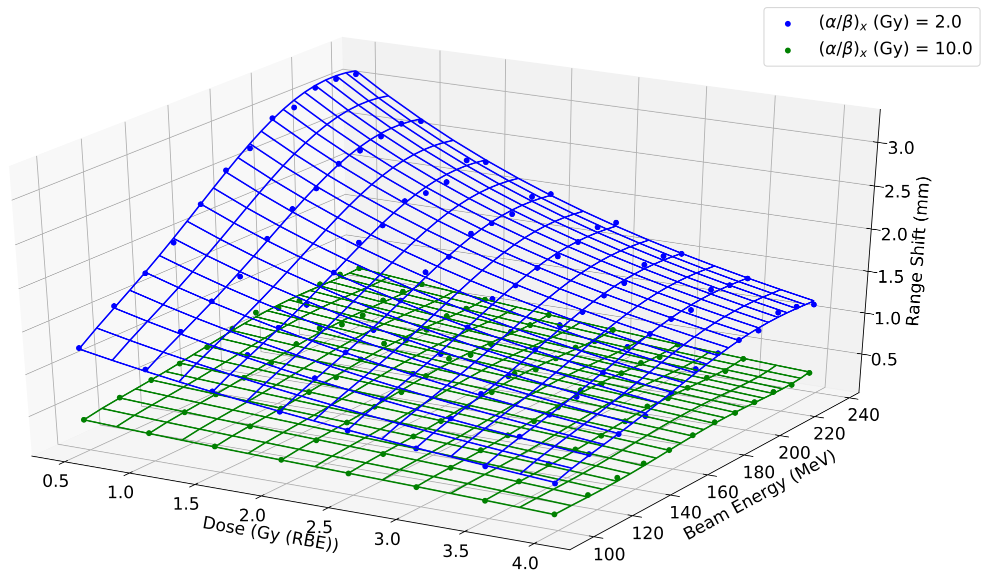
Adding the plot of a higher $(\alpha/\beta)_x$ of 10 Gy, we get figure 20b. As in the case of PBPs, $(\alpha/\beta)_x$ in between will cause surfaces in between these two surfaces, and $(\alpha/\beta)_x < 2$ Gy will be located above the $(\alpha/\beta)_x = 2$ Gy-surface. The surfaces with high $(\alpha/\beta)_x$s are significantly flatter than for the surfaces with low $(\alpha/\beta)_x$s, indicating that the range shift is strongly dependent on the range shift.

The range shift dataset contains values between 0.24 mm and 4.21 mm, with similar trends as for PBP; increased range shift with increased beam energy and decreased dose levels and $(\alpha/\beta)_x$ values.

Plotting the range shift versus each of the three parameters for selected values of the other two parameters, we get the plots in figure 24. The fitted curves are produced by the regression model with $n = 5$, with a STD of 0.030 mm. The energy dependence of the range shift indicates a linearity for low energy, but the range shift stagnates for higher energies. The dose and range shift has a fairly linear proportionality, while the $(\alpha/\beta)_x$ and range shift are more inversely proportional.

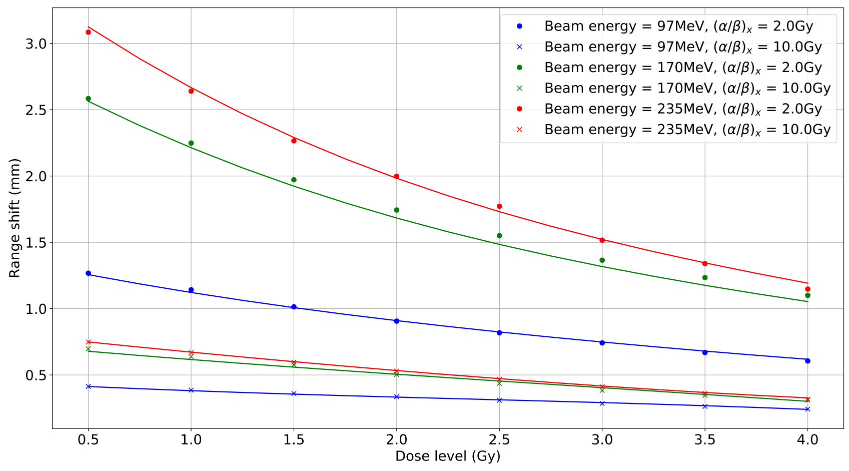


(a)

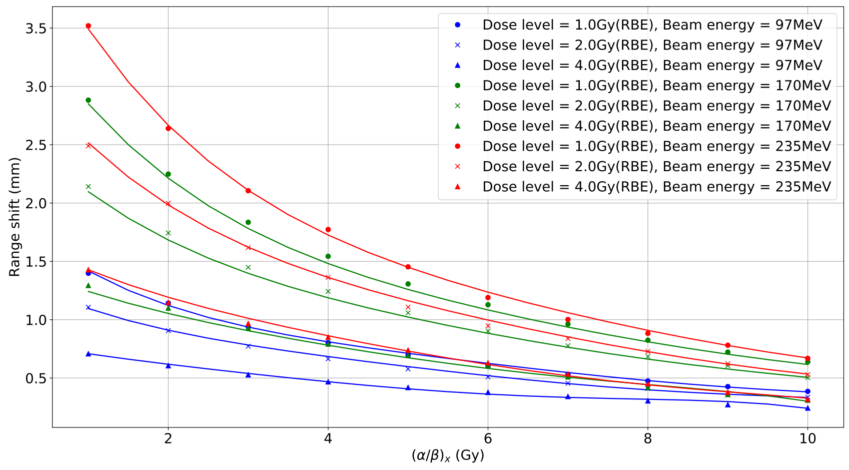


(b)

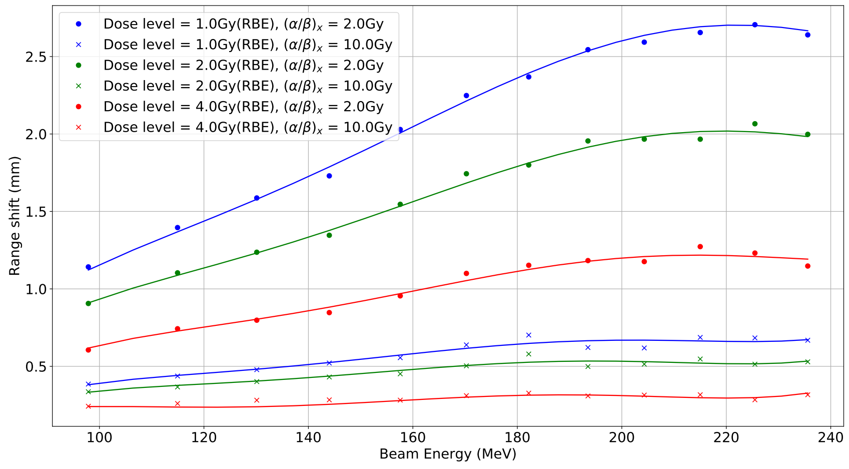
Figure 23: (a) Range shift surface for the SOBP method as a function of dose level and beam energy for $(\alpha/\beta)_x = 2\ Gy$. (b) Range shift surfaces for the SOBP method as a function of dose level and beam energy for $(\alpha/\beta)_x = 2\ Gy$ and 10 Gy.



(a) Biological Range shift as a function of dose level. The legend connects the curves to the max beam energies and $(\alpha/\beta)_x$ values.



(b) Biological Range shift as a function of $(\alpha/\beta)_x$ values. The legend connects the curves to the dose levels and the max beam energies.



(c) Biological Range shift as a function of beam energy. The legend connects the curves to the dose levels and $(\alpha/\beta)_x$ values.

Figure 24: Biological Range Shift as a Function of Dose level, Max Beam energy and $(\alpha/\beta)_x$

5.2 Biological Range Shift Prediction and Measurements in the two test Plans

5.2.1 Water Phantom SOBP$_{4cm}$

Measurement of the biological range shift in the SOBP$_{4cm}$ plan was found to be 1.62 mm and 0.61 mm for $(\alpha/\beta)_x$ equal to 2 Gy and 10 Gy, respectively. The depth-dose distributions for the $RBE_{1.1}$ and ROR models for the two $(\alpha/\beta)_x$ values are shown in figure 25 along with the corresponding range shifts.

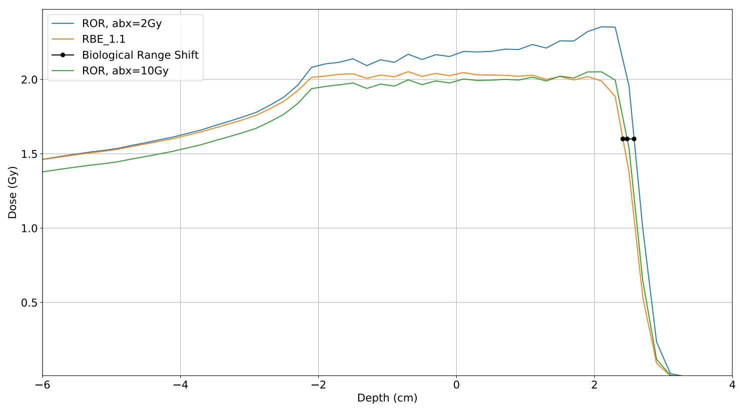


Figure 25: Measured biological range shift in the SOBP$_{4cm}$ plan. The black dots with line-segments visualizes the biological range shifts for $(\alpha/\beta)_x$ equal to 2 Gy and 10 Gy.

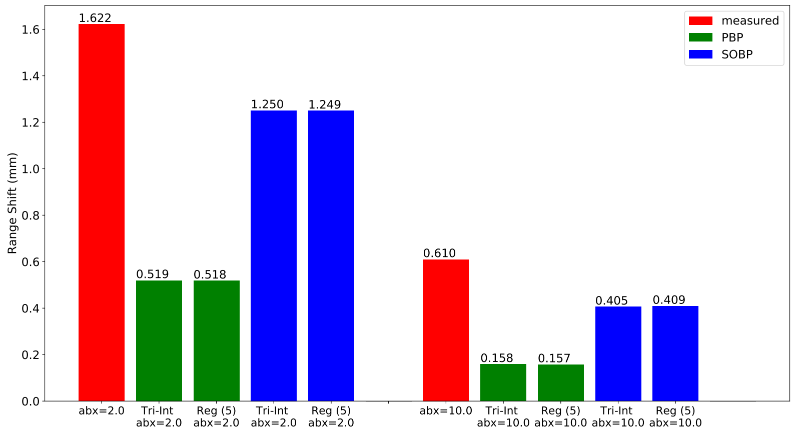


Figure 26: Measured and predicted biological range shift in the SOBP$_{4cm}$ plan for the ROR model with $(\alpha/\beta)_x$ values of 2 Gy and 10 Gy. The red bars are the measured range shifts, while the green and blue are the predicted range shifts using the PBP and SOBP methods. "Tri-Int" and "Reg (5)" refer to the prediction model used; Trilinear interpolation and regression with $n = 5$.

For the PBP method, both the regression with $n = 5$ and trilinear interpolation predicted the biological range shift in SOBP$_{4cm}$ to be 0.52 mm and 0.16 mm for $(\alpha/\beta)_x$ equal to 2 Gy and 10 Gy, respectively. For the SOBP method, both the regression with $n = 5$ and the trilinear interpolation predicted the biological range shift in SOBP$_{4cm}$ to be 1.25 mm and 0.41 mm for $(\alpha/\beta)_x$ equal to

2 Gy and 10 Gy, respectively. Figure 26 shows the measured range shifts and the predicted range shifts for the SOBP$_{4cm}$ plan for the two methods in two different types of tissues. The PBP the SOBP methods underestimates the range shift measured in the SOBP$_{4cm}$ by about 70% and 30%, respectively. Appendix C lists the output files from the range shift prediction-script *predict_RS.py* for the two scenarios.

5.2.2 Prostate Plan

Measurement of the biological range shift in the prostate plan was found to be 1.71 mm and 1.89 mm for field 1 and 2, respectively. The depth-dose curves of the the $RBE_{1.1}$ and ROR models for the two fields are shown in figure 27 along with the corresponding range shifts.

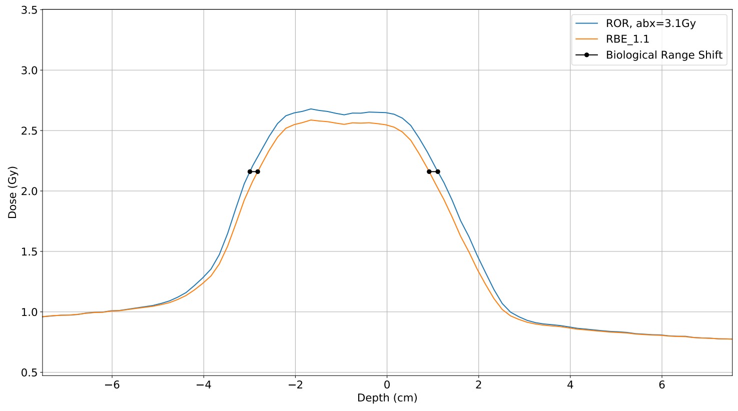


Figure 27: Measured biological range shift in the prostate plan. The black dots with lines visualizes the range shifts in the ROR model for $(\alpha/\beta)_x$ equal to 3.1 Gy. Field 1 is entering in from the left, and field 2 from the right.

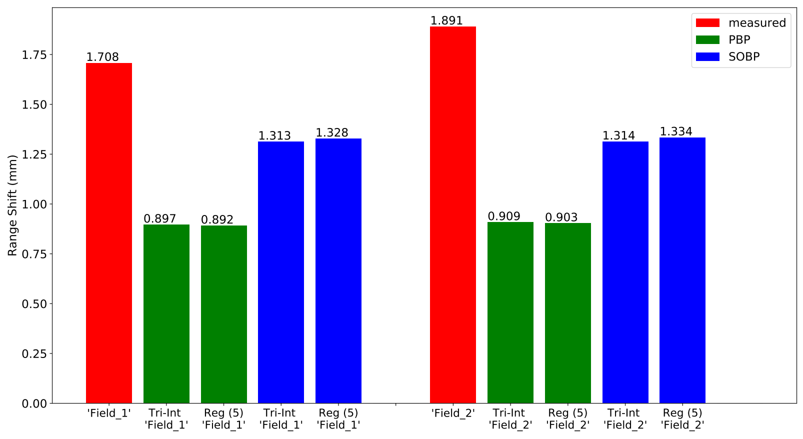


Figure 28: Measured and predicted biological range shift in the prostate plan for the ROR model with $(\alpha/\beta)_x$ values of 3.1 Gy. The red bars are the measured range shifts, while the green and blue are the predicted range shifts using the PBP and SOBP methods. "Tri-Int" and "Reg (5)" refer to the methods of prediction; Trilinear interpolation and regression with $n = 5$, respectively.

Figure 28 shows the measured and predicted range shifts in the two fields in the prostate plan. The PBP and SOBP methods underestimates the range shifts by about 50% and 28%, respectively. The increased beam energy of field 2 compared to field 1 (196.55 MeV to 195.10 MeV) leads to an increase in the range shift of 0.18 mm, but no significant changes in the predicted range shifts.

5.3 Biological Range Shift in PBPs versus SOBPs

The 11 SOBPs of various modulation widths for the tps weigh method, are shown in figure 29. Even though the dose level is set to 2 $Gy(RBE)$, the beam with only the maximum energy is not weighted to give more than a peak dose of 1.77 $Gy(RBE)$.

Figure 30, shows the depth-dose distribution of the beam energy with the highest energy in the SOBP$_{4cm}$ plan normalized with the tps weigh method (blue) and the peak weigh method (orange). The peak doses for the tps weigh method and peak weigh method are 1.77 $Gy(RBE)$ and 2 $Gy(RBE)$.

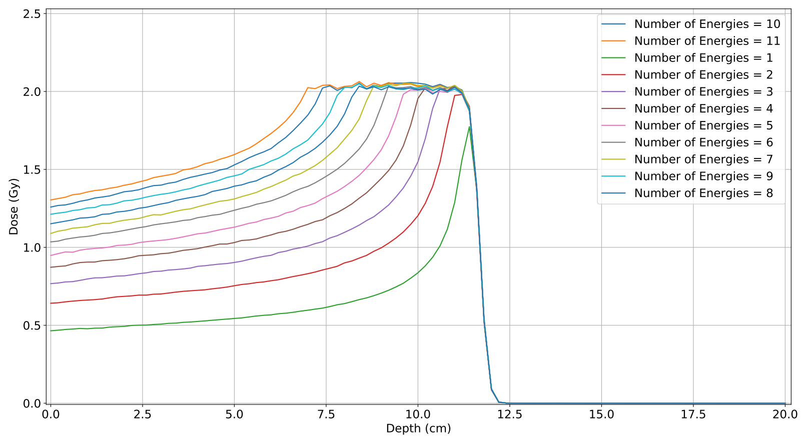


Figure 29: 11 SOBPs with various modulation widths weighted with the tps method and $RBE_{1.1}$.

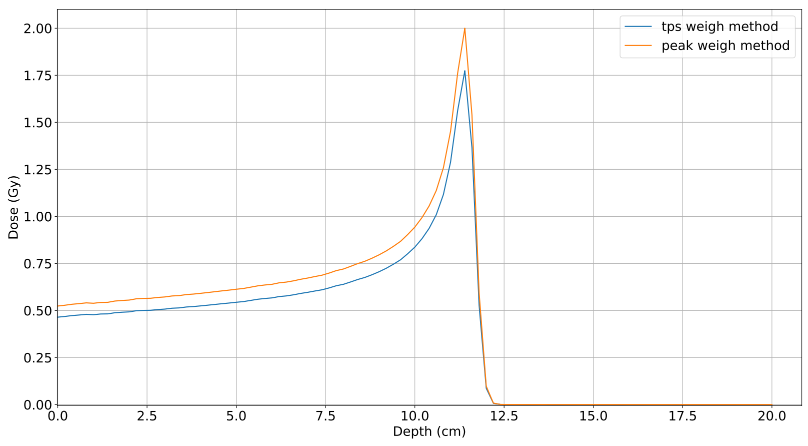


Figure 30: The PBP with the highest energy in the SOBP$_{4cm}$ plan normalized with the tps weigh method (blue) and the peak weigh method (orange).

The biological range shift of the SOBPs plotted versus the number of energies in the SOBP can be seen in figure 31. The SOBPs are normalized using the tps weigh method (see figure 29). The range shift increases with number of energies, with a large increase of 0.36 mm from 1 to 2

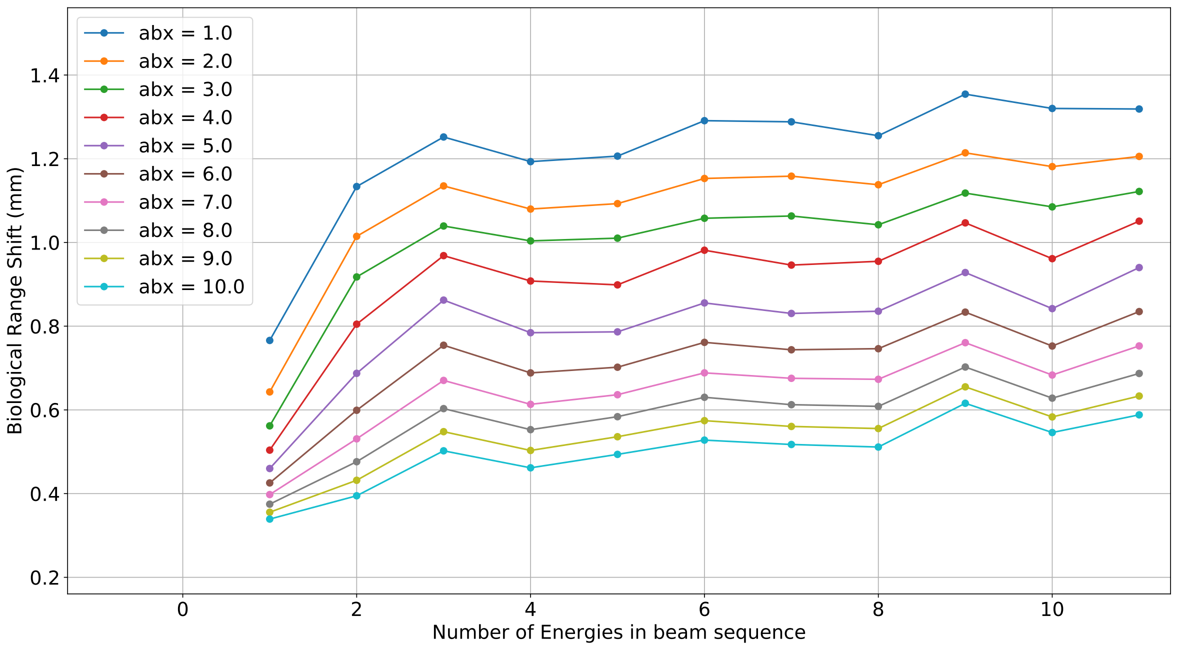


Figure 31: Biological Range shift as a function of number of energies in the SOBP normalized with the tps weigh method. The dose level is 2 $Gy(RBE)$

5.4 Re-optimized SOBP3

Figure 32 shows the depth dose plots of the re-optimized and original SOBP3-plan for the $RBE_{1.1}$ and ROR models. The peak at the edge of the SOBP is reduced in the re-optimized SOBP, and it looks more like the one in the SOBP$_{4cm}$ plan. The distal part of the re-optimized $RBE_{1.1}$-weighted SOBP has retracted to a slightly shallower depth, compared to the original SOBP. The retraction of the ROR-weighted SOBP, is less significant. Figure 33 shows the measured range shift and the predicted range shift, based on the original SOBPs, on the SOBP$_{4cm}$ plan. The predicted range shift has increased by 0.19mm and 0.08 mm in the tissue with $(\alpha/\beta)_x = 2\ Gy$ and $(\alpha/\beta)_x = 10\ Gy$ after the re-optimization. As only the SOBP3 plan (max beam energy of 130 MeV) was re-optimized, the predicted range shift of the prostate plan (max beam energy of 196 MeV) will not be affected.

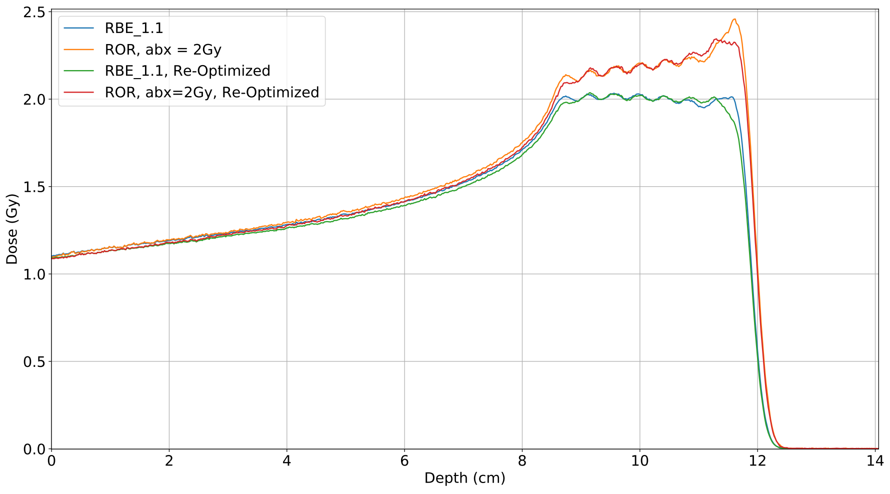


Figure 32: Re-optimized and original depth dose distribution of the SOBP3 plan for the $RBE_{1.1}$ and ROR models in a tissue with $(\alpha/\beta)_x = 2\ Gy$ and a dose level of 2 $Gy(RBE)$. The re-optimized SOBP correspond to the green and red curves for $RBE_{1.1}$ and the ROR model, respectively. The original SOBP is shown as the blue and orange curves for $RBE_{1.1}$ and ROR model, respectively.

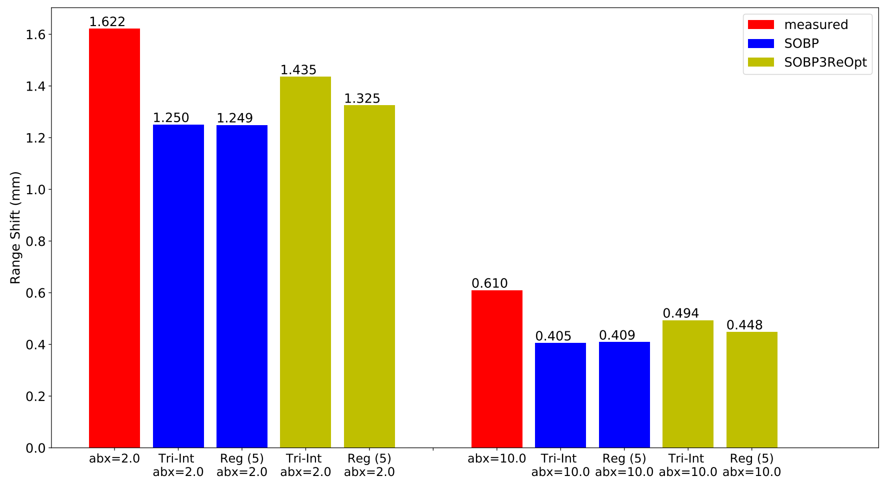


Figure 33: Measured and predicted biological range shift in the SOBP$_{4cm}$ plan for the ROR model with $(\alpha/\beta)_x$ values of 2 Gy and 10 Gy. The red bars are the measured range shifts, while the blue and yellow are the predicted range shifts using the original and re-optimized SOBP methods. "Tri-Int" and "Reg (5)" refer to the methods of prediction; Trilinear interpolation and regression with $n = 5$, respectively.

6 Discussion

While other [5, 12, 14] have investigated the biological range shift in proton therapy, this project introduces the first predictive models for the biological range shift. The basis for the predictive models has been datasets of biological range shifts generated by multiple Monte Carlo simulations for PBPs and SOBPs. The predictive models have been tested on two scenarios. The trends of the range shift found in this project are in line with the results of Grun et al. [14], that is; increased range shift for increasing beam energy (depth) or decreasing $(\alpha/\beta)_x$ and dose level. Carabe et al[5]. and Giovannini et al [12]. does not indicate any effects of beam energy on the range shift, but the trends of increased range shift with decreasing $(\alpha/\beta)_x$ are in line with this project. Both methods systematically underestimates the range shifts in the two scenarios, with the PBP method being inferior to the SOBP method. Reasons for the systematic underestimation may be that the shape and width of the SOBPs affect the biological range shift, along with contributions to the total RBE from other treatment fields. We will in the following sections discuss the results of this project, the possible causes of the systematic underestimation and suggestions for improvements of the predictive models.

6.1 Input Dependency

The biological range shift has been shown to depend on the dose level, beam energy and the tissue parameter $(\alpha/\beta)_x$ [14]. These were the inputs used in this project, and the trends are in line with the work of Grun et al. [14]., see table 2. The trend of increased range shift with increased beam energy (depth), seen in this project and Grun et al., are not apparent in Carabe et al., but the trends of increased range shift for decreasing $(\alpha/\beta)_x$ and dose level are in line with Grun et al., Carabe et al. and Giovannini et al.

D / E	1 $Gy(RBE)$ Grun/Reg(5)	2 $Gy(RBE)$ Grun/Reg(5)	3 $Gy(RBE)$ Grun/Reg(5)	4 $Gy(RBE)$ Grun/Reg(5)
130 MeV	1.1/1.6	1.1/1.2	1.0/1.0	0.9/0.8
145 MeV	1.5/1.8	1.3/1.4	1.1/1.1	1.0/0.9
155 MeV	1.7/2.0	1.5/1.5	1.3/1.2	1.1/1.0
165 MeV	1.9/2.1	1.7/1.6	1.4/1.3	1.2/1.0
175 MeV	2.2/2.3	1.9/1.7	1.6/1.4	1.3/1.1

Table 2: Range shift in mm found by Grun et al. ("Grun") in tissue with $(\alpha/\beta)_x = 2\ Gy$ and range shift predicted by the SOBP regression model with $n = 5$ ("Reg (5)") for the same input parameters.

6.1.1 Dose Level

As can be seen from figures 21a and 24a, the range shift seems to have an inversely proportional relation to the dose level. This relation is increasingly linear with the decrease of the beam energy or an increase in $(\alpha/\beta)_x$. This effect is more pronounced in the PBP dataset than in the SOBP dataset. The decrease in range shift for increased dose levels is expected due to the reduction of RBE with dose level, as can be seen from equation (7). The dose dependency found by Grun et al. is more linear compared to the dose dependency found in this project as can be seen from the steady decline of the range shift found by Grun in table 2.

6.1.2 Initial Beam Energy

The range shift increases with initial beam energy, as can be seen in figures 21c and 24c. This effect is more pronounced in tissue with low $(\alpha/\beta)_x$ values. Increasing the energy from 70 MeV to 230 MeV, for the PBP dataset, leads to an increase of the range shift of 1.7 mm and 0.5 mm in tissues with $(\alpha/\beta)_x$ equal to 2 Gy and 10 Gy, respectively. The increase in range shift with increased initial beam energy may be caused by the increased width of the distal penumbra with energy, due to range straggling, as discussed by Grun[14, 27]. For the PBP dataset, the range shift to energy relation is almost linear, while the range shift levels out for higher energies for the SOBP dataset. This stagnation of the range shift at higher energies may be caused by irregularities in the shape of the SOBPs. The effects of the shape, and thereby the relative weight of the two largest energies in the SOBP on the range shift will be further discussed in section 6.3.

6.1.3 Tissue Parameter, $(\alpha/\beta)_x$

The relation between range shift and $(\alpha/\beta)_x$ is similar to that of range shift and dose levels, that is: an inversely proportional relation, as seen in figures 21b and 24b. The relation is less linear than that of the range shift to dose level, but it is increasingly linear with decreased beam energy and increased dose levels. The decline in range shift with $(\alpha/\beta)_x$ is expected due to the reduced RBE for tissues with high $(\alpha/\beta)_x$ values, see the last term in (7). For some RBE models the RBE for high $(\alpha/\beta)_x$ can fall below 1.1 and thereby a negative range shift can be observed as seen by Carabe et al. These negative values are far more frequent in the study by Carabe et al. than in this project and that of Grun et al. These qualitative differences in results may be caused by the use of different variable RBE models [14].

6.2 Fit Quality

From figures 20a and 23a, and the STDs of $0.006mm$ and 0.03, we can see that the regression model for the PBP method has a self-prediction (ability to reproduce the dataset) accuracy 5 times better than the SOBP method. This is also reflected in that the fitted plane for the SOBP dataset misses the data-points more severely than for the PBP dataset. The variation in the shape of the SOBP may be the cause of the fluctuating data-points, leading to a higher STD. The higher STD of the SOBP regression model with $n = 5$, leads to a larger difference in the predicted range shift between the regression model and the trilinear interpolation method. These differences are negligible compared to the underestimation of the range shift caused by other factors.

As the beam energy, and consequently the depth of the SOBPs increases, the number of energies in the SOBP decrease. This is because the width of each Bragg peak increases with depth, as can be seen in figure 19a. The variation on the number of energies in the SOBPs causes the weight of each energy, relative to the total weight of the SOBP, to differ between the SOBPs, which may be the cause of the variation in shape of the SOBPs. These variations may be the cause of the stagnation of the range shift at higher energies, and the reduced fit quality.

6.3 Systematic underestimation

From figures 26 and 28 it is apparent that both the PBP method and SOBP method systematically underestimates the biological range shifts. The range shift is not only dependent on the highest energy in the SOBP, but also the lower energies, building up the SOBP, as can be seen in figure 31. At the most extreme, the range shift almost doubles when considering a PBP versus a SOBP of 11 energies, in a tissues with low $(\alpha/\beta)_x$ values. There is a steep increase just by adding the second largest energy (E2), from $0.75mm$ to 1.15, in a tissue with $(\alpha/\beta)_x = 1\ Gy$. The PBP method does not account for the effect of the non-largest energies of the SOBP and is therefore not a good candidate for predicting the range shift in a plan consisting of SOBPs in voxels, which all patient plans do.

In addition, the weighting of the PBPs (peak weigh method) are not consistent with the weighting (tps weigh method) of the beam of the highest energy in a SOBP (E1), as we saw in figure 30. The weight of a beam can be interpreted as the number of particles, which is used as weighting/normalization factor in this project. The increased number of particles for E1 by the PBP weigh method compared to the tps method can be interpreted as an increased dose level. This increase in dose level, as we have seen in figure 21a, leads to a decrease in range shift and a further strengthening of the claim that PBPs as a basis for the model is not a good candidate.

Figure 31 shows that the range shift increases rapidly with increasing number of energies when there are few energies in the SOBP. The range shift steadily increases with the number of energies when there are more than just a few energies in the SOBP. As the RBE peaks in the distal dose fall-off of a Bragg peak, the presence of multiple energies in a SOBP leads to accumulation of RBE, and consequently the steady increase in range shift. The biological range shift therefore illustrates a dependency on the width of the SOBP, in contrast to the results of Grun [14]. Figure 31 only shows the range shifts when splitting up a single SOBP of a certain depth and width. The effects of the modulation width on the range shift for multiple depths should therefore be explored.

Putting constraints and objectives on the dose distribution in the TPS, the SOBPs can be re-optimized to give a more homogeneous and consistent dose in the SOBP. This was not done for the SOBPs forming the basis for the predictive range shift model, but it was for the SOBP$_{4cm}$ plan and prostate plan, like for all patient plans. This causes the shapes of the SOBP of the two patient plans to be more homogeneous than the SOBPs of our basis. In particular the SOBPs of our basis has a peak towards the end of the SOBP, seen in figure 22a. This peak is caused by the relative weight of the E1 and E2, that is the beam of highest energy and the beam of second highest energy. The weight-fraction $w(E1)/w(E2)$, that is the weight of E1 divided by the weight of E2, is large in our basis compared to patient plans. Re-optimization of the SOBPs in the basis will correct this weight-fraction, as was done for the SOBP3 plan, and can be seen in figure 32. The consequences of the optimization on the range shift can be seen in the predicted range shift for the SOBP$_{4cm}$ plan in figure 33. The SOBPs in the basis all have a valley before a peak at the end of the SOBP, indicating a high weight-fraction, $w(E1)/w(E2)$. Re-optimization of the SOBP basis may correct this heightened weight-fraction, correcting the variation in the shape of the SOBP and improving the predictive models.

In the predictive models, beam doses and fraction dose, that is; doses given from each beam and the total dose given from all beams, are equal, as only one beam is present. In the prostate case, there were two fields present. This may be a cause of systematic underestimation of the predictive models in the prostate case, as they do not account for the presence multiple fields.

6.4 Suggestion for Further Work

To improve the predictive models, the SOBPs forming the basis in the SOBP method should be re-optimized to be more in line with the shape of SOBPs in treatment plans. The effects of modulation width should be explored, and the predictive models should be tested in more scenarios. The impact of using different RBE models should also be evaluated. If the improved model does not over-/underestimates the range shift, the coefficients of the regression models, in effect a power series, could be used to find a simpler analytical expression for the biological range shift.

As most patient cases are more complex than just max beam energy, dose level and $(\alpha/\beta)_x$, a predictive range shift model should be equally complex, to avoid systematic over/underestimations. The results in this project have indicated that the shape of the SOBP, especially towards the edge of the SOBP, affects the biological range shift. Future work could therefore be to study the beam sequence of the SOBP, that is; the energies used, the number of energies and the weight of each energy.

Since the biological range shift not only depends on the beam energy of the most distal PBP, dose level and $(\alpha/\beta)_x$, but also the width of the SOBP, the optimization routine and potentially the presence of other fields, the method of producing a dataset covering all of the possible values for these parameters may not be optimal. An alternative could be to have a PBP range shift dataset, and a summation procedure where the range shifts from each PBP is appropriately weighted and added together. These weights may have complex dependencies on the entire beam sequence, tissue properties and fractionation scheme. However, it may be a better way to form a predictive biological range shift model for a general patient plan.

7 Conclusion

In this project, two predictive biological range shift models have been produced for the distal r_{80} range of proton beams, representing the first predictive models for biological range shift in proton therapy. One of the models was based on PBPs of various depths and dose levels in water, and it performed poorly, systematically underestimating range shifts in a SOBP plan and a prostate plan by up to 70% and 50%, respectively. As the PBPs does not account for contributions to the total RBE from lower energetic beams, which will be present in clinical treatment plans, it is not a good alternative to form a basis for a predictive range shift model. The other model was based on SOBPs of various depths and performed better when predicting the range shifts in the two scenarios, but still with typical underestimation up to 30% and 25% in a SOBP plan in water and prostate treatment plan, respectively.

The results of this study indicate that the biological range shift is mostly determined by the beam energy, dose level and $(\alpha/\beta)_x$, but as we have seen, irregularities in the shape of the SOBP data the prediction model is based on may cause systematic underestimations. Considerations of the shape and width of the SOBPs are therefore crucial for accurate predictions. The presented results indicate that a relatively accurate model for the biological range shift can be developed based on the modelling framework introduced in this thesis and applying more accurately optimized SOBPs in the predictive model basis. Such a model could highlight enhanced biological doses not reflected by todays commercial treatment planning systems.

References

[1] G. Barendsen. *Responses of cultured cells, tumors, and normal tissues to radiations of different linear energy transfer.* Tech. rep. Health Research Organization TNO, Rijswijk, Netherlands, 1968.

[2] H. A. Bethe and J. Ashkin. *Passage of radiations through matter Experimental Nuclear Physics vol 1 ed E Segre.* 1953.

[3] T. Böhlen et al. "The FLUKA code: developments and challenges for high energy and medical applications". In: *Nuclear data sheets* 120 (2014), pp. 211–214.

[4] *Cancer Statistics.* Dec. 2018. URL: https://www.kreftregisteret.no/en/The-Registries/Cancer-Statistics/.

[5] A. Carabe et al. "Range uncertainty in proton therapy due to variable biological effectiveness". In: *Physics in Medicine & Biology* 57.5 (2012), p. 1159.

[6] J. Efstathiou, P. Gray, and A. Zietman. "Proton beam therapy and localised prostate cancer: current status and controversies". In: *British journal of cancer* 108.6 (2013), p. 1225.

[7] U. Fano. "Penetration of protons, alpha particles, and mesons". In: *Annual Review of Nuclear Science* 13.1 (1963), pp. 1–66.

[8] A. Ferrari et al. *FLUKA: A multi-particle transport code (Program version 2005).* Tech. rep. 2005.

[9] L. F. Fjæra. "Development of a Monte Carlo based treatment planning verification tool for particle therapy". MA thesis. The University of Bergen, 2016.

[10] *Fractionation Radiobiological principles and clinical practise.* Nov. 2018. URL: https://oncohemakey.com/fractionation-radiobiological-principles-and-clinical-practice/.

[11] M. C. Frese et al. "Application of constant vs. variable relative biological effectiveness in treatment planning of intensity-modulated proton therapy". In: *International Journal of Radiation Oncology* Biology* Physics* 79.1 (2011), pp. 80–88.

[12] G. Giovannini et al. "Variable RBE in proton therapy: comparison of different model predictions and their influence on clinical-like scenarios". In: *Radiation Oncology* 11.1 (2016), p. 68.

[13] B. Gottschalk. "Physics of proton interactions in matter". In: *Proton Therapy Physics* (2012), pp. 19–60.

[14] R. Grün et al. "Physical and biological factors determining the effective proton range". In: *Medical physics* 40.11 (2013).

[15] C. Grupen and B. Shwartz. *Particle detectors.* Cambridge university press, 2008.

[16] F. Guan et al. "Analysis of the track-and dose-averaged LET and LET spectra in proton therapy using the geant4 Monte Carlo code". In: *Medical physics* 42.11 (2015), pp. 6234–6247.

[17] E. J. Hall. "Radiation biology for pediatric radiologists". In: *Pediatric radiology* 39.1 (2009), p. 57.

[18] V. Hartmann. "Anbefaler ny strålebehandling i Norge". In: (2016).

[19] H. Henjum. "Optimization of proton therapy plans with respect to biological and physical dose distributions". MA thesis. The University of Bergen, 2018.

[20] M. C. Joiner and A. Van der Kogel. *Basic clinical radiobiology.* Vol. 1. CRC press, 2016.

[21] F. M. Khan and J. P. Gibson. *Khan's the physics of radiation therapy.* Lippincott Williams Wilkins, 2014.

[22] D. E. Lea. *Actions of radiations on living cells.* Cambridge University Press.; Cambridge, 1946.

[23] W. Levin et al. “Proton beam therapy”. In: *British journal of Cancer* 93.8 (2005), p. 849.

[24] *Linear interpolation.* URL: https://en.wikipedia.org/wiki/Linear_interpolation.

[25] A. L. McNamara, J. Schuemann, and H. Paganetti. “A phenomenological relative biological effectiveness (RBE) model for proton therapy based on all published in vitro cell survival data”. In: *Physics in Medicine & Biology* 60.21 (2015), p. 8399.

[26] I. Mellman, G. Coukos, and G. Dranoff. “Cancer immunotherapy comes of age”. In: *International journal of Science* 480 (2011), p. 480.

[27] V. A. Mjelde. *Biological range extension in proton therapy.* 2018.

[28] *Multiple scattering for particles in the matter.* Dec. 2018. URL: http://meroli.web.cern.ch/lecture_multiple_scattering.html?fbclid=IwAR0raOqYDHaKPrAD3wu7T39BSTZbpj8hCYjZSqCR_CCJ722rJw1rMPgB9ok.

[29] W. D. Newhauser and R. Zhang. “The physics of proton therapy”. In: *Physics in Medicine & Biology* 60.8 (2015), R155.

[30] H. Paganetti. *Proton beam therapy.* IOP Publishing, 2017.

[31] H. Paganetti. “Relating proton treatments to photon treatments via the relative biological effectiveness—should we revise current clinical practice?” In: *International Journal of Radiation Oncology• Biology• Physics* 91.5 (2015), pp. 892–894.

[32] H. Paganetti et al. “Calculation of relative biological effectiveness for proton beams using biological weighting functions”. In: *International Journal of Radiation Oncology* Biology* Physics* 37.3 (1997), pp. 719–729.

[33] H. Paganetti et al. “Relative biological effectiveness (RBE) values for proton beam therapy”. In: *International Journal of Radiation Oncology* Biology* Physics* 53.2 (2002), pp. 407–421.

[34] I. C. on Radiation. “Fundamental Quantities and Units for Ionizing Radiation”. In: (1998).

[35] I. C. on Radiological Protection. “1990 Recomendations of the International Commission on Radiological Protection”. In: (1991).

[36] E. Rørvik et al. “A phenomenological biological dose model for proton therapy based on linear energy transfer spectra”. In: *Medical physics* 44.6 (2017), pp. 2586–2594.

[37] E. Rørvik et al. “Exploration and application of phenomenological RBE models for proton therapy”. In: *Physics in Medicine & Biology* 63.18 (2018), p. 185013.

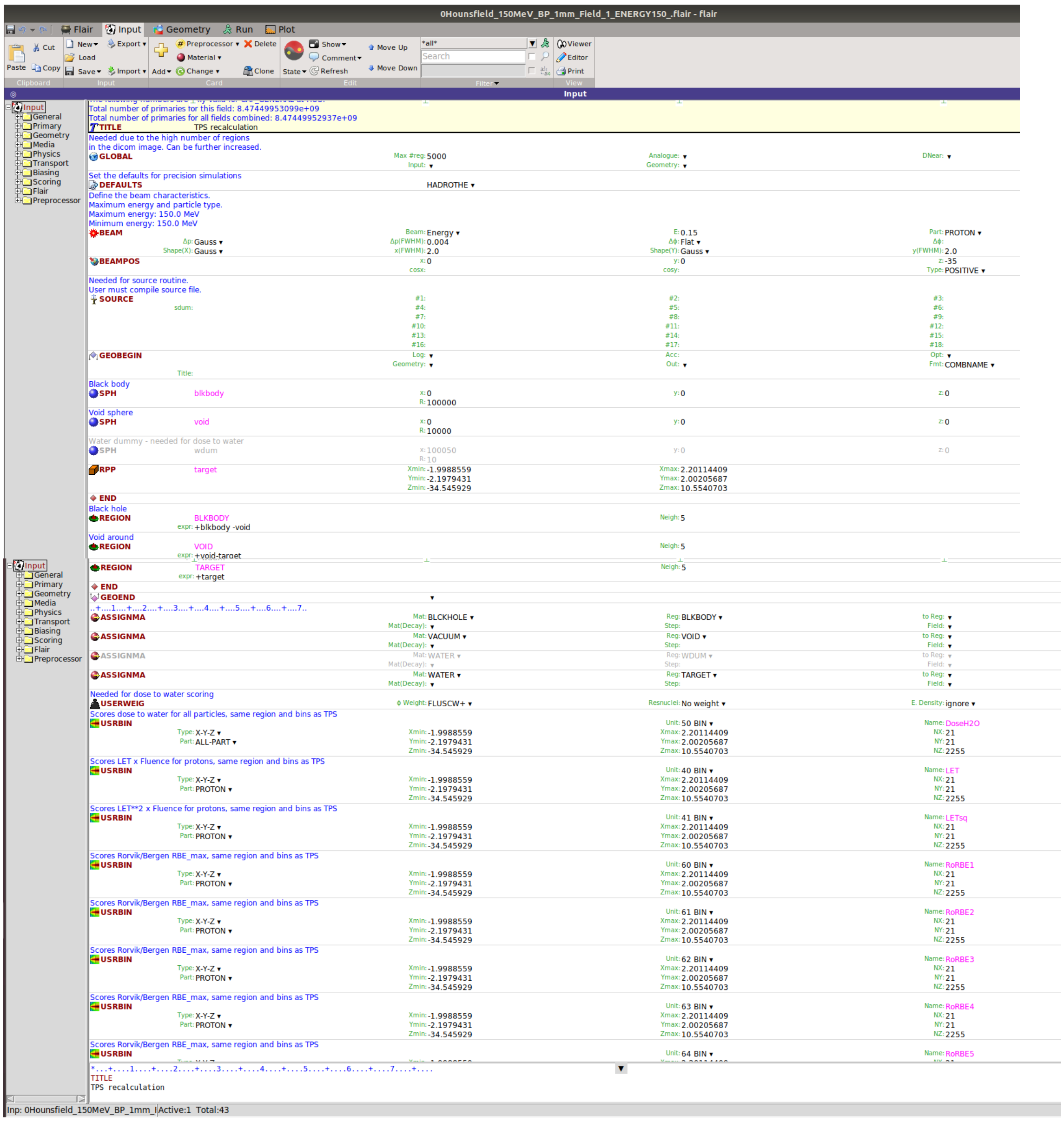
[38] $sklearn.linear_model.LinearRegression()$. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[39] *Trilinear interpolation.* URL: https://en.wikipedia.org/wiki/Trilinear_interpolation.

[40] D. Zink, A. H Fischer, and J. Nickerson. “Nuclear structure in cancer cells”. In: *Nature reviews. Cancer* 4 (Oct. 2004), pp. 677–87. DOI: 10.1038/nrc1430.

A FLUKA input file in FLAIR

The image shows a screenshot of an example of the FLUKA input files for the simulations of the PBPs in flair. The "BEAM"-card defines the beam energy, momentum spread and shape and with of the beam profile, and the "BEAMPOS" defines the position of the beam source and direction of the beam. The "RPP"-card defines the location of the target (the water tank). The "REGION" cards define the different regions, and the "ASSIGNMA" assigns the material to the different regions. The "USRBIN" cards are the cards scoring the dose to water, LET and RBE_{max} for the ROR model.



```
Total number of primaries for this field: 8.47449953099e+09
Total number of primaries for all fields combined: 8.47449952937e+09
TITLE            TPS recalculation

Needed due to the high number of regions
in the dicom image. Can be further increased.
GLOBAL                          Max #reg: 5000              Analogue:              DNear:
                                   Input:                   Geometry:

Set the defaults for precision simulations
DEFAULTS                                  HADROTHE

Define the beam characteristics.
Maximum energy and particle type.
Maximum energy: 150.0 MeV
Minimum energy: 150.0 MeV
BEAM                                Beam: Energy            E: 0.15                Part: PROTON
                    Δp: Gauss   Δp(FWHM): 0.004            Δϕ: Flat                  Δϕ:
              Shape(X): Gauss    x(FWHM): 2.0        Shape(Y): Gauss           y(FWHM): 2.0
BEAMPOS                                x: 0                 y: 0                      z: -35
                                    cosx:                cosy:                     Type: POSITIVE

Needed for source routine.
User must compile source file.
SOURCE                                #1:                  #2:                       #3:
                 sdum:                #4:                  #5:                       #6:
                                      #7:                  #8:                       #9:
                                     #10:                 #11:                      #12:
                                     #13:                 #14:                      #15:
                                     #16:                 #17:                      #18:
GEOBEGIN                             Log:                 Acc:                      Opt:
                                Geometry:                 Out:                      Fmt: COMBNAME
                 Title:

Black body
SPH                   blkbody          x: 0                 y: 0                      z: 0
                                       R: 100000
Void sphere
SPH                   void             x: 0                 y: 0                      z: 0
                                       R: 10000
Water dummy - needed for dose to water
SPH                   wdum             x: 100050            y: 0                      z: 0
                                       R: 10
RPP                   target        Xmin: -1.9988559     Xmax: 2.20114409
                                    Ymin: -2.1979431     Ymax: 2.00205687
                                    Zmin: -34.545929     Zmax: 10.5540703
END
Black hole
REGION                BLKBODY                           Neigh: 5
                expr: +blkbody -void
Void around
REGION                VOID                              Neigh: 5
                expr: +void-target
REGION                TARGET                            Neigh: 5
                expr: +target
END
GEOEND
..+....1....+....2....+....3....+....4....+....5....+....6....+....7..
ASSIGNMA                             Mat: BLCKHOLE         Reg: BLKBODY           to Reg:
                              Mat(Decay):                 Step:                    Field:
ASSIGNMA                             Mat: VACUUM           Reg: VOID              to Reg:
                              Mat(Decay):                 Step:                    Field:
ASSIGNMA                             Mat: WATER            Reg: WDUM              to Reg:
                              Mat(Decay):                 Step:                    Field:
ASSIGNMA                             Mat: WATER            Reg: TARGET            to Reg:
                              Mat(Decay):                 Step:                    Field:
Needed for dose to water scoring
USERWEIG                        ϕ Weight: FLUSCW+     Resnuclei: No weight     E. Density: ignore
Scores dose to water for all particles, same region and bins as TPS
USRBIN                                                     Unit: 50 BIN             Name: DoseH2O
                 Type: X-Y-Z        Xmin: -1.9988559       Xmax: 2.20114409           NX: 21
                 Part: ALL-PART     Ymin: -2.1979431       Ymax: 2.00205687           NY: 21
                                    Zmin: -34.545929       Zmax: 10.5540703           NZ: 2255
Scores LET x Fluence for protons, same region and bins as TPS
USRBIN                                                     Unit: 40 BIN             Name: LET
                 Type: X-Y-Z        Xmin: -1.9988559       Xmax: 2.20114409           NX: 21
                 Part: PROTON       Ymin: -2.1979431       Ymax: 2.00205687           NY: 21
                                    Zmin: -34.545929       Zmax: 10.5540703           NZ: 2255
Scores LET**2 x Fluence for protons, same region and bins as TPS
USRBIN                                                     Unit: 41 BIN             Name: LETsq
                 Type: X-Y-Z        Xmin: -1.9988559       Xmax: 2.20114409           NX: 21
                 Part: PROTON       Ymin: -2.1979431       Ymax: 2.00205687           NY: 21
                                    Zmin: -34.545929       Zmax: 10.5540703           NZ: 2255
Scores Rorvik/Bergen RBE_max, same region and bins as TPS
USRBIN                                                     Unit: 60 BIN             Name: RoRBE1
                 Type: X-Y-Z        Xmin: -1.9988559       Xmax: 2.20114409           NX: 21
                 Part: PROTON       Ymin: -2.1979431       Ymax: 2.00205687           NY: 21
                                    Zmin: -34.545929       Zmax: 10.5540703           NZ: 2255
Scores Rorvik/Bergen RBE_max, same region and bins as TPS
USRBIN                                                     Unit: 61 BIN             Name: RoRBE2
                 Type: X-Y-Z        Xmin: -1.9988559       Xmax: 2.20114409           NX: 21
                 Part: PROTON       Ymin: -2.1979431       Ymax: 2.00205687           NY: 21
                                    Zmin: -34.545929       Zmax: 10.5540703           NZ: 2255
Scores Rorvik/Bergen RBE_max, same region and bins as TPS
USRBIN                                                     Unit: 62 BIN             Name: RoRBE3
                 Type: X-Y-Z        Xmin: -1.9988559       Xmax: 2.20114409           NX: 21
                 Part: PROTON       Ymin: -2.1979431       Ymax: 2.00205687           NY: 21
                                    Zmin: -34.545929       Zmax: 10.5540703           NZ: 2255
Scores Rorvik/Bergen RBE_max, same region and bins as TPS
USRBIN                                                     Unit: 63 BIN             Name: RoRBE4
                 Type: X-Y-Z        Xmin: -1.9988559       Xmax: 2.20114409           NX: 21
                 Part: PROTON       Ymin: -2.1979431       Ymax: 2.00205687           NY: 21
                                    Zmin: -34.545929       Zmax: 10.5540703           NZ: 2255
Scores Rorvik/Bergen RBE_max, same region and bins as TPS
USRBIN                                                     Unit: 64 BIN             Name: RoRBE5

*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
TITLE
TPS recalculation

Inp: 0Hounsfield_150MeV_BP_1mm_I Active:1  Total:43
```

Figure 34: Screenshot of one of the FLUKA input files for the simulations of PBPs in a water tank.

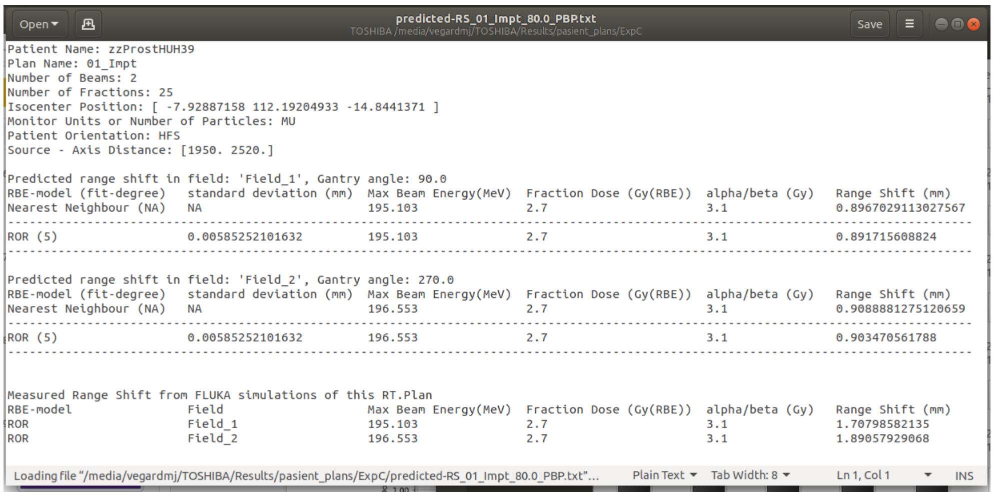
B Momentum-spread Modelling

Momentum calibration – First try The fall of 2016, I calibrated the momentum spread in FLUKA to be 0.9% of the kinetic beam energy. This was done by creating a PBP in Eclipse TPS and recalculating it in FLUKA. By trial and error, it was decided that a spread of 0.9% gave the best match. Since then, this single value has been used in our simulations. This scenario is, however, not ideal since the percentage value may differ between energies. So, we first wanted to do this calibration for several energies, and further do a fit in order to obtain a formula for calculating the momentum spread for every single energy. I started by creating PBPs in Eclipse for every tenth MeV ranging between $70 - 250\ MeV$. The PBPs was simulated in a water phantom and the field sizes were approximately $10 \times 10\ cm^2$. The homogeneous water phantom was sized $20 \times 20 \times 90\ cm^3$. Instead of recalculating all the PBPs in FLUKA, I chose the following energies: 70 MeV, 100 MeV, 130 MeV, 150 MeV, 170 MeV, 200 MeV, 230 MeV, 250 MeV. I used the same scoring grid in FLUKA as the TPS. In the lateral directions (x and y) the scoring grid encompassed the whole phantom (i.e. 100 bins in x and y sized 2 mm). In the longitudinal direction, the bins were sized 1 mm as this was the smallest size available in the TPS. We chose to match the momentum spread in proximal and distal 80% dose (i.e. matching the width of the PBP at 80% dose). The PBPs was also normalized to 100% dose prior to the calibration/matching. A cubic polynomial was fitted to the data-points and with $R^2 = 0.98$ the relative momentum spread as a function of energy is given as $dp/p = -4.6234E^3 + 1.7547E^2 - 0.2159E + 0.0163$, where the units are in GeV.

C Prediction Output Files

The output file contains information on the set up, along with range shifts for each field, and field parameters such as max beam energy and beam dose, for each predictive model for each variable RBE-model. The dose fall-off percentage is contained in the file-name, and so is the basis of the range shift dataset (PBP or SOBP or SOBP3opt).

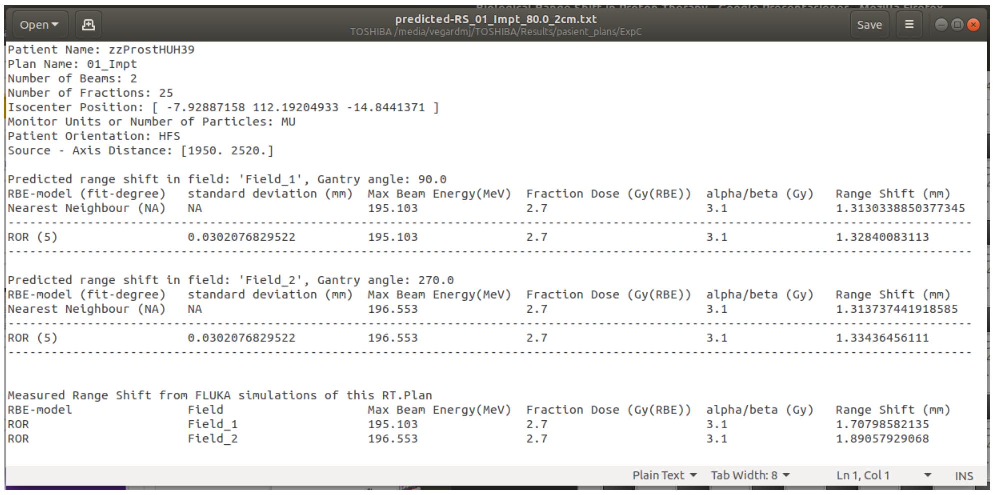
C.1 Prostate plan


Open
predicted-RS_01_Impt_80.0_PBP.txt
TOSHIBA /media/vegardmj/TOSHIBA/Results/pasient_plans/ExpC
Save

```
Patient Name: zzProstHUH39
Plan Name: 01_Impt
Number of Beams: 2
Number of Fractions: 25
Isocenter Position: [ -7.92887158 112.19204933 -14.8441371 ]
Monitor Units or Number of Particles: MU
Patient Orientation: HFS
Source - Axis Distance: [1950. 2520.]

Predicted range shift in field: 'Field_1', Gantry angle: 90.0
RBE-model (fit-degree)   standard deviation (mm)  Max Beam Energy(MeV)  Fraction Dose (Gy(RBE))  alpha/beta (Gy)   Range Shift (mm)
Nearest Neighbour (NA)   NA                       195.103               2.7                      3.1               0.8967029113027567
-----------------------------------------------------------------------------------------------------------------------------------
ROR (5)                  0.00585252101632         195.103               2.7                      3.1               0.891715608824
-----------------------------------------------------------------------------------------------------------------------------------

Predicted range shift in field: 'Field_2', Gantry angle: 270.0
RBE-model (fit-degree)   standard deviation (mm)  Max Beam Energy(MeV)  Fraction Dose (Gy(RBE))  alpha/beta (Gy)   Range Shift (mm)
Nearest Neighbour (NA)   NA                       196.553               2.7                      3.1               0.9088881275120659
-----------------------------------------------------------------------------------------------------------------------------------
ROR (5)                  0.00585252101632         196.553               2.7                      3.1               0.903470561788
-----------------------------------------------------------------------------------------------------------------------------------


Measured Range Shift from FLUKA simulations of this RT.Plan
RBE-model                Field                    Max Beam Energy(MeV)  Fraction Dose (Gy(RBE))  alpha/beta (Gy)   Range Shift (mm)
ROR                      Field_1                  195.103               2.7                      3.1               1.70798582135
ROR                      Field_2                  196.553               2.7                      3.1               1.89057929068
```

Loading file "/media/vegardmj/TOSHIBA/Results/pasient_plans/ExpC/predicted-RS_01_Impt_80.0_PBP.txt"...
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

(a)

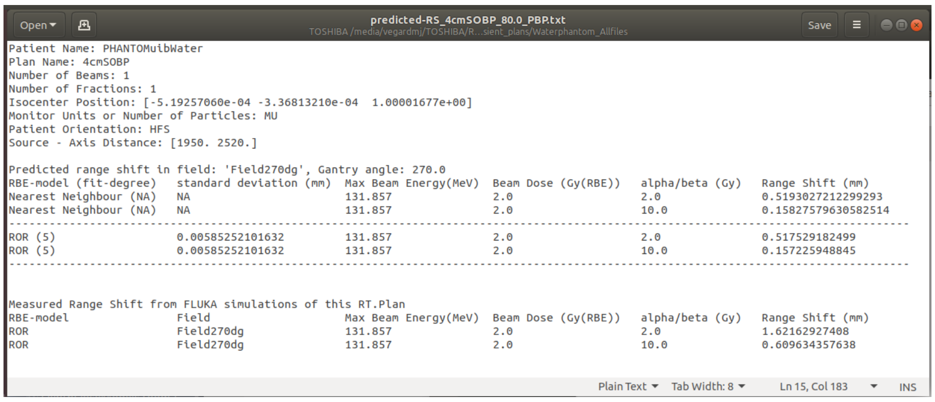

Open
predicted-RS_01_Impt_80.0_2cm.txt
TOSHIBA /media/vegardmj/TOSHIBA/Results/pasient_plans/ExpC
Save

```
Patient Name: zzProstHUH39
Plan Name: 01_Impt
Number of Beams: 2
Number of Fractions: 25
Isocenter Position: [ -7.92887158 112.19204933 -14.8441371 ]
Monitor Units or Number of Particles: MU
Patient Orientation: HFS
Source - Axis Distance: [1950. 2520.]

Predicted range shift in field: 'Field_1', Gantry angle: 90.0
RBE-model (fit-degree)   standard deviation (mm)  Max Beam Energy(MeV)  Fraction Dose (Gy(RBE))  alpha/beta (Gy)   Range Shift (mm)
Nearest Neighbour (NA)   NA                       195.103               2.7                      3.1               1.3130338850377345
-----------------------------------------------------------------------------------------------------------------------------------
ROR (5)                  0.0302076829522          195.103               2.7                      3.1               1.32840083113
-----------------------------------------------------------------------------------------------------------------------------------

Predicted range shift in field: 'Field_2', Gantry angle: 270.0
RBE-model (fit-degree)   standard deviation (mm)  Max Beam Energy(MeV)  Fraction Dose (Gy(RBE))  alpha/beta (Gy)   Range Shift (mm)
Nearest Neighbour (NA)   NA                       196.553               2.7                      3.1               1.313737441918585
-----------------------------------------------------------------------------------------------------------------------------------
ROR (5)                  0.0302076829522          196.553               2.7                      3.1               1.33436456111
-----------------------------------------------------------------------------------------------------------------------------------


Measured Range Shift from FLUKA simulations of this RT.Plan
RBE-model                Field                    Max Beam Energy(MeV)  Fraction Dose (Gy(RBE))  alpha/beta (Gy)   Range Shift (mm)
ROR                      Field_1                  195.103               2.7                      3.1               1.70798582135
ROR                      Field_2                  196.553               2.7                      3.1               1.89057929068
```

(b)

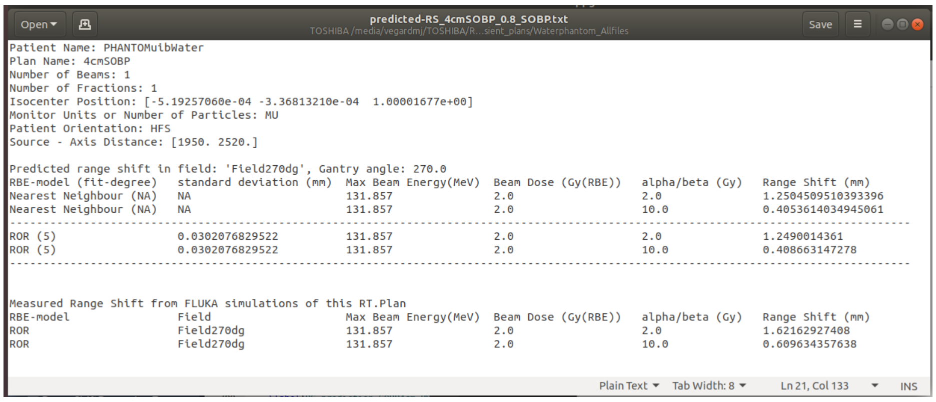
Figure 35: Range shift prediction on the prostate plan for (a) PBP method and (b) SOBP method.

C.2 SOBP$_{4cm}$ plan


Open
predicted-RS_4cmSOBP_80.0_PBP.txt
TOSHIBA /media/vegardmj/TOSHIBA/R...sient_plans/Waterphantom_Allfiles
Save
Plain Text
Tab Width: 8
Ln 15, Col 183
INS

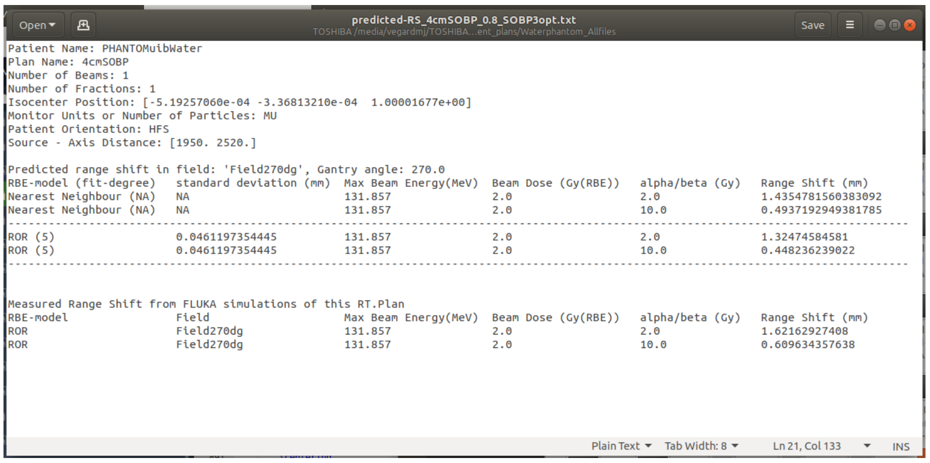
```
Patient Name: PHANTOMuibWater
Plan Name: 4cmSOBP
Number of Beams: 1
Number of Fractions: 1
Isocenter Position: [-5.19257060e-04 -3.36813210e-04  1.00001677e+00]
Monitor Units or Number of Particles: MU
Patient Orientation: HFS
Source - Axis Distance: [1950. 2520.]

Predicted range shift in field: 'Field270dg', Gantry angle: 270.0
RBE-model (fit-degree)   standard deviation (mm)  Max Beam Energy(MeV)  Beam Dose (Gy(RBE))   alpha/beta (Gy)   Range Shift (mm)
Nearest Neighbour (NA)   NA                       131.857               2.0                   2.0               0.5193027212299293
Nearest Neighbour (NA)   NA                       131.857               2.0                   10.0              0.15827579630582514
------------------------------------------------------------------------------------------------------------------------------------
ROR (5)                  0.00585252101632         131.857               2.0                   2.0               0.517529182499
ROR (5)                  0.00585252101632         131.857               2.0                   10.0              0.157225948845
------------------------------------------------------------------------------------------------------------------------------------


Measured Range Shift from FLUKA simulations of this RT.Plan
RBE-model                Field                    Max Beam Energy(MeV)  Beam Dose (Gy(RBE))   alpha/beta (Gy)   Range Shift (mm)
ROR                      Field270dg               131.857               2.0                   2.0               1.62162927408
ROR                      Field270dg               131.857               2.0                   10.0              0.609634357638
```

(a)


Open
predicted-RS_4cmSOBP_0.8_SOBP.txt
TOSHIBA /media/vegardmj/TOSHIBA/R...sient_plans/Waterphantom_Allfiles
Save
Plain Text
Tab Width: 8
Ln 21, Col 133
INS

```
Patient Name: PHANTOMuibWater
Plan Name: 4cmSOBP
Number of Beams: 1
Number of Fractions: 1
Isocenter Position: [-5.19257060e-04 -3.36813210e-04  1.00001677e+00]
Monitor Units or Number of Particles: MU
Patient Orientation: HFS
Source - Axis Distance: [1950. 2520.]

Predicted range shift in field: 'Field270dg', Gantry angle: 270.0
RBE-model (fit-degree)   standard deviation (mm)  Max Beam Energy(MeV)  Beam Dose (Gy(RBE))   alpha/beta (Gy)   Range Shift (mm)
Nearest Neighbour (NA)   NA                       131.857               2.0                   2.0               1.2504509510393396
Nearest Neighbour (NA)   NA                       131.857               2.0                   10.0              0.4053614034945061
------------------------------------------------------------------------------------------------------------------------------------
ROR (5)                  0.0302076829522          131.857               2.0                   2.0               1.2490014361
ROR (5)                  0.0302076829522          131.857               2.0                   10.0              0.408663147278
------------------------------------------------------------------------------------------------------------------------------------


Measured Range Shift from FLUKA simulations of this RT.Plan
RBE-model                Field                    Max Beam Energy(MeV)  Beam Dose (Gy(RBE))   alpha/beta (Gy)   Range Shift (mm)
ROR                      Field270dg               131.857               2.0                   2.0               1.62162927408
ROR                      Field270dg               131.857               2.0                   10.0              0.609634357638
```

(b)


Open
predicted-RS_4cmSOBP_0.8_SOBP3opt.txt
TOSHIBA /media/vegardmj/TOSHIBA...ent_plans/Waterphantom_Allfiles
Save
Plain Text
Tab Width: 8
Ln 21, Col 133
INS

```
Patient Name: PHANTOMuibWater
Plan Name: 4cmSOBP
Number of Beams: 1
Number of Fractions: 1
Isocenter Position: [-5.19257060e-04 -3.36813210e-04  1.00001677e+00]
Monitor Units or Number of Particles: MU
Patient Orientation: HFS
Source - Axis Distance: [1950. 2520.]

Predicted range shift in field: 'Field270dg', Gantry angle: 270.0
RBE-model (fit-degree)   standard deviation (mm)  Max Beam Energy(MeV)  Beam Dose (Gy(RBE))   alpha/beta (Gy)   Range Shift (mm)
Nearest Neighbour (NA)   NA                       131.857               2.0                   2.0               1.4354781560383092
Nearest Neighbour (NA)   NA                       131.857               2.0                   10.0              0.4937192949381785
------------------------------------------------------------------------------------------------------------------------------------
ROR (5)                  0.0461197354445          131.857               2.0                   2.0               1.32474584581
ROR (5)                  0.0461197354445          131.857               2.0                   10.0              0.448236239022
------------------------------------------------------------------------------------------------------------------------------------


Measured Range Shift from FLUKA simulations of this RT.Plan
RBE-model                Field                    Max Beam Energy(MeV)  Beam Dose (Gy(RBE))   alpha/beta (Gy)   Range Shift (mm)
ROR                      Field270dg               131.857               2.0                   2.0               1.62162927408
ROR                      Field270dg               131.857               2.0                   10.0              0.609634357638
```

(c)

Figure 36: Range shift prediction on the SOBP$_{4cm}$ plan for (a) PBP method, (b) SOBP method and (c) SOBP 3OPT method.

D Coefficients of Regression Models

The values of the coefficients c_{ijk} for the regression model with $n = 5$ for the two methods (PBP and SOBP) are listed below.

Listing 1: Coefficients c_{ijk} of the predictive regression model for the PBP method.

```
Rsquared = 0.9998634147630869
min/max deviation = -0.04818120480589538 0.04349497930784185
Total variance = 0.04658272305521634
Standard Deviation = 0.005852521016321308
 Coefficient name:    Coefficient value:
c000                  0.16382972297410087
c001                  -0.022554293533735007
c002                  0.013886793522106889
c003                  -0.006070450150l5206
c004                  0.0006909084064739908
c005                  -2.3408509283466555e-05
c010                  -0.007188640222392863
c011                  -0.00021700050884281672
c012                  0.0006367061612l3693
c013                  -6.860378093672003e-05
c014                  1.965547979689098e-06
c020                  0.00020662903076171916
c021                  -4.478276161986843e-05
c022                  2.3738220136167977e-06
c023                  -2.471070348541004e-08
c030                  3.1574362711762426e-08
c031                  7.024220778776053e-08
c032                  -2.8396954639561522e-09
c040                  -2.6832821306348363e-09
c041                  -3.0879549952122023e-13
c050                  5.059071509096909e-12
c100                  -0.001945593235981801
c101                  0.03859660966217529
c102                  -0.013562606626302766
c103                  0.0018622676695190948
c104                  -8.002811103052727e-05
c110                  -0.011394611357447777
c111                  0.001470119844692258
c112                  -0.00018550373225363008
c113                  7.882930170222168e-06
c120                  4.1487581168326724e-06
c121                  4.856698425491211e-06
c122                  -1.8701839370270212e-07
c130                  7.572877230242183e-08
c131                  -5.379719529661348e-09
c140                  -2.2986958018479664e-10
c200                  0.2680486755042689
c201                  -0.03594825673108017
c202                  0.0038314196125583695
c203                  -0.00017060515292625358
c210                  0.003064167699461483
c211                  -0.0002956667031991981
c212                  1.349225006036861e-05
c220                  -1.2770178647008266e-05
```

```
c221                    7.209509323910512e-08
c230                    1.8228819140351377e-08
c300                    -0.12053529649407328
c301                    0.00818478138793177
c302                    -0.00029632439115635486
c310                    -4.061590865899909e-06
c311                    9.325785345581233e-06
c320                    4.317709170927053e-07
c400                    0.017039010790076796
c401                    -0.0005087348681208563
c410                    -1.7940621809908828e-05
c500                    -0.0008007230664252889
```

Listing 2: Coefficients c_{ijk} of the predictive regression model for the SOBP method.

```
Rsquared = 0.9978043853548668
min/max deviation = -0.08844928083536674  0.0917167059705517
Total variance = 0.8760039449662712
Standard Deviation = 0.030207682952187277
Coefficient name:       Coefficient value:
c000                    -11.000134202799579
c001                    0.21838786834028756
c002                    0.07082475127398609
c003                    -0.016618183555664585
c004                    0.0013814369535202007
c005                    -3.778657275308426e-05
c010                    0.3803320520882751
c011                    -0.015022574865386492
c012                    0.001417787433121781
c013                    -0.00010355710791901178
c014                    2.15757846789026e-06
c020                    -0.004526628862966702
c021                    5.746093298847651e-05
c022                    8.96114866298207e-07
c023                    6.531385763947833e-08
c030                    2.8495463148077206e-05
c031                    -3.4258186781090983e-07
c032                    -5.751655409095397e-09
c040                    -8.461522401112764e-08
c041                    7.96183326478981e-10
c050                    9.293069587376071e-11
c100                    0.2434040763347824
c101                    0.07701333358550776
c102                    -0.04538463713139521
c103                    0.005548219263306415
c104                    -0.00018935536003972966
c110                    -0.0107752476923056
c111                    0.003089092294588005
c112                    -0.0003501544150474543
c113                    8.511403483551637e-06
c120                    5.023751886797177e-06
c121                    4.76896958218195e-06
c122                    2.4642580738129527e-07
c130                    -2.2302587459992196e-07
c131                    -2.050099112693715e-08
```

c140	8.158332898133835e−10
c200	−0.03808140792074176
c201	−0.025156556480248407
c202	0.007411270717892313
c203	−0.0003681378431713983
c210	0.0016562583841434938
c211	−0.00047355338521057333
c212	1.5970758010214192e−05
c220	5.935193978845454e−06
c221	3.8127615764604883e−07
c230	−2.054751123067855e−08
c300	0.003450480879517O4
c301	0.0032321936081836818
c302	−0.0003141742468691644
c310	−0.00019677244547428377
c311	1.3984186955483952e−05
c320	1.9592585193128897e−07
c400	−0.0001457882364427521
c401	−7.674197169722091e−05
c410	1.852501231976167e−06
c500	5.976270673605042e−05

E Scripts

The following scripts have been made specifically for this project and used, alongside edited scripts made by PTG [9], to calculate range shifts and range shift models from Monte Carlo FLUKA scoring files.

Listing 3: Python script, *increase_bins.py*, reads a RT.Dose file and increases the number of scoring bins in the z-direction

```
import dicom
import os
import numpy as np

def main():
    path_dcm = raw_input("Provide DICOM path: ")

    dcm_list=[]; path_list=[]
    for path, subdir, file_list in os.walk(path_dcm):
        for filename in file_list:
            if "CT" not in filename:
                path_list.append(path)
                dcm_list.append(filename)
        break

    for i in range(len(dcm_list)):
        print str(i+1) + ". " + dcm_list[i]
    c = raw_input("choose RT.Dose-file: ")
    c = int(c)-1

    dc = dicom.read_file(path_list[c]+"/"+dcm_list[c])

    gfov_old = dc.GridFrameOffsetVector
    gfov_old = [float(g) for g in gfov_old]
    increment_old = gfov_old[1]-gfov_old[0]

    print "Distance between zBins is {}mm".format(increment_old)
    increment = "0.2"#raw_input("New increment (cm): ")
    increment = float(increment)
    f = increment_old/increment
    print "Number of zBins will be increased by a factor of {}".format(f)

    #Pixel array
    px_old = dc.pixel_array
    px_add = np.zeros((1,np.shape(px_old)[1],np.shape(px_old)[2]))
    px = px_add

    gfov=[]
    max_dev=0
    for i in range((len(gfov_old)-1)*int(f)+1):
            gfov.append(gfov_old[0]+increment*i)
            if i>0:
                px = np.concatenate((px,px_add),axis=0)

    gfov = [str(g) for g in gfov]

    #print gfov
    print len(gfov_old)
    print len(gfov)
    print np.shape(px_old)
    print np.shape(px)

    # Define values to be integers
    px = np.rint(px)
    px = px.astype(int)
    px = np.array(px, dtype = dc.pixel_array.dtype)

    dc.NumberOfFrames=str(len(gfov))
    dc.GridFrameOffsetVector = gfov
    dc.PixelData=px.tostring()

    save_name = path_list[c]+"/"+dcm_list[c]
    #save_name = save_name[:-4]+"_new.dcm"
    dc.save_as(save_name)

main()
```

Listing 4: Python script, *get_SOBP_factors.py*, reads depth-dose distribution files and finds the normalization factor needed for the curves to give a prescribed dose of 1 $Gy(RBE)$.

```
#Reads depth dose relations and writes a file with plan name versus factor. The factor will
    ↪ normalize the depth dose relations, such that max−dose = 1Gy
import supplementary
import os
import numpy as np

def main(method):
    path = raw_input("Provide directory of 1d plot of the non−normalized FLUKA Bio
        ↪ Dose files with 1Gy(RBE): ")
    files = []
    for path0, subdir, file_list in os.walk(path):
        for filename in file_list:
            if ".dat" in filename:
                files.append(path0+"/"+filename)
    for i in range(len(files)):
        print str(i+1) + ". " + files[i]
    rem = raw_input("Remove files (space for none): ")
    if rem != "":
        rem = rem.split(" ")
        rem = [int(r)−1 for r in rem]
        for r in rem:
            del files[r]
    plan_names = []
    DD = []
    factors = []
    for f in files:
        #plan_names.append(f.split("/")[−1].split("_")[3]) #SOBPvsPBP
        plan_names.append(f.split("/")[−1].split("_")[4]) #VegardSOBPs
        DD.append(supplementary.read_depth_dose_simple(f)) #may be edited...
        if method == "peak":
            maxDose = np.max(DD[−1]['dose'])
        elif method == "mean":
            maxDose = get_avg_dose_in_SOBP(DD[-1]['dose'])
        DD[-1]['dose'] = [d/maxDose for d in DD[-1]['dose']] #not nescescarry
        factors.append(1/maxDose)

    new_path = raw_input("Provide destination directory for the plan - factor .txt file to be
        ↪ stored: ")
    new_filename = "SOBP_factors_{}_1Gy.dat".format(method)

    file = open(new_path + "/" + new_filename, 'w')
    for i in range(len(DD)):
        file.write(plan_names[i] + " " + str(factors[i]) + "\n")
    file.close()


def get_avg_dose_in_SOBP(D):
    dose_to_find = np.max(D)*0.95
    start = 0
    stop = len(D)-1
    for i in range(len(D)):
        if D[i]>dose_to_find and start == 0:
            print "start: " + str(i)
            start=i
        if D[len(D)-1-i]>dose_to_find and stop == len(D)-1:
            print "stop: " + str(len(D)-1-i)
            stop=len(D)-1-i

    D_sum = 0
    n = stop-start+1
    for i in range(n):
        D_sum = D_sum + D[start + i]/n
    return D_sum





#main("peak")
main("mean")
```

Listing 5: Python script, *makeData.py*, calculates range shifts from depth-dose relations and makes a range shift data set.

```
import supplementary
import os
import numpy as np

def main():

    #width="SOBP" #SOBPs
    #width="SOBP3" #SOBPs_3OPT
    width="PBP" #PBPs

    #Prompting user for path, model, dose-falloff percent
    path = raw_input("Provide data 1d path: ")
    nr_fractions = 1#input("Number of Fractions: ")
```

```
percent=raw_input("Choose distal dose fall-off percentage (float) (space for 0.8, 
    ↪ standard): ")
model=raw_input("Choose RBE model to compare with RBE=1.1 (space for ROR): ")
if model=="":
    model="ROR"
if percent=="":
    percent="0.8"
percent=float(percent)

path_new = "/".join(path.split("/")[:-1])

#Reads relevant depth dose files
DD_var = []; DD_bio = []
dose_levels = []; energies = []; ab = []
for path0, subdir, file_list in os.walk(path):
    for filename in file_list:
        os.chdir(path0)
        if model in filename:
            if width=="PBP":
                DD_var.append(supplementary.read_DD_PBP(filename, nr_fractions, percent
                    ↪ ))
            else:
                DD_var.append(supplementary.read_DD(filename, nr_fractions, percent))
            if float(DD_var[-1]['max_energy']) not in energies:
                energies.append(DD_var[-1]['max_energy'])
            if float(DD_var[-1]['dose_level']) not in dose_levels:
                dose_levels.append(DD_var[-1]['dose_level'])
            if float(DD_var[-1]['abx']) not in ab:
                ab.append(DD_var[-1]['abx'])
        elif "Bio" in filename:
            if width=="PBP":
                DD_bio.append(supplementary.read_DD_PBP(filename, nr_fractions, percent
                    ↪ ))
            else:
                DD_bio.append(supplementary.read_DD(filename, nr_fractions, percent))


dose_levels.sort();energies.sort();ab.sort();
RS = np.zeros((len(dose_levels), len(energies), len(ab)))
#Calculates Range Shift
rs_max=0; rs_min=10000
for dv in DD_var:
    for db in DD_bio:
        if dv['dose_level'] == db['dose_level'] and dv['max_energy'] == db['max_energy']:
            dv['range_shift'] = (dv['80'][0] - db['80'][0])*10
            d = dose_levels.index(dv['dose_level'])
            e = energies.index(dv['max_energy'])
            a = ab.index(dv['abx'])
            RS[d, e, a] = dv['range_shift']
            if dv['range_shift']<rs_min:
                rs_min=dv['range_shift']
            if dv['range_shift']>rs_max:
                rs_max=dv['range_shift']
            break
        #Write to file
        os.chdir(path_new)
        wfile=open("RS_data_{}_D{}_{}.txt".format(model, percent*100, width), "w")
        wfile.write(str(dose_levels))
        wfile.write("\n")
        wfile.write(str(energies))
        wfile.write("\n")
        wfile.write(str(ab))
        wfile.write("\n")
        for i in range(len(dose_levels)):
            for j in range(len(energies)):
                for k in range(len(ab)):
                    wfile.write(str(RS[i, j, k])+" ")
                wfile.write("\n")
            wfile.write("\n")
        wfile.close()

    print "Max range shift = {}mm".format(rs_max)
    print "Min range shift = {}mm".format(rs_min)


main()
```

Listing 6: Python script, *make_RS_model.py*, reads the range shift data set and uses regression to make an analytic model for the range shift data-points.

```
import supplementary
import os


def prompt_user():
    #path="/media/vegardmj/TOSHIBA/Results/VegardSOBPs"; width = "SOBP" #SOBPs
    path="/media/vegardmj/TOSHIBA/Results/VegardSOBPs_optimized"; width = "SOBP3"
        ↪ #SOBP_#OPT
    #path="/media/vegardmj/TOSHIBA/Results/
        ↪ LongWaterPhantom_150MeV_4x4cm_1mmZ_451bins/
        ↪ FLUKA_0Hounsfield_150MeV_BP_1mm/0Hounsfield_150MeV_BP_1mm_DICOMs/
        ↪ FLUKA_DICOM"; width = "PBP" #PBPs
    #path = raw_input("Provide path for the Range Shift Data folder: "); width=raw_input("
        ↪ SOBP width (float cm) (enter for SOBP, standard): ")

    model="" #raw_input("Choose RBE model to compare with RBE=1.1 (enter for ROR): ")
    percent="" #raw_input("Choose distal dose fall-off percentage (float) (enter for 0.8,
        ↪ standard): ")

    if model=="":
        model="ROR"
    if percent=="":
        percent="0.8"
    percent=float(percent)
    deg=raw_input("Degree of fit (int):")
    deg=int(deg)
    return path, model, percent, deg, width
```

```
def main():

    path, model, percent, deg, width = prompt_user()

    if not os.path.isfile("RS_data_{}_D{}_{}.txt".format(model, percent*100, width)):
        M, list_of_doses, list_of_energies, list_of_ab = supplementary.read_range_shift_data(path,
            ↪ model, percent, width)
    else:
        print "Range Shift Data of this RBE model and fall-off percentage does not exsists.
            ↪  (Run makeData.py)"

    if os.path.isfile("coeffs_{}_D{}_{}_{}".format(model, percent*100, deg, width)):
        print "Predictive Range Shift model already exists"
    else:
        print "Calculating coefficients and saving"
        p_names, p, Rsquared = supplementary.reg_sklearn(list_of_doses,list_of_energies,list_of_ab,
            ↪ M,deg)
        min_dev, max_dev, var, dev_M, STD = supplementary.get_deviation([list_of_doses,
            ↪ list_of_energies,list_of_ab],M,p_names,p)
        print "STD = {}mm".format(STD)
        print width
        supplementary.save_coefficients(path, model, deg, width, Rsquared, [min_dev, max_dev],
            ↪ var, STD, p_names, p, percent)

main()
```

Listing 7: Python script, *analyze.py*, plots various relations which can be calculated from depth-dose relations.

```
import numpy as np
import math, sys, os
import matplotlib.pyplot as plt
from math import log10, floor
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import curve_fit
from sklearn import linear_model
import supplementary



#Promts user for what to plot/print, as well as RBE model, dose fall-off percentage and degree
    ↪ og fit.
def main():
    path="/media/vegardmj/TOSHIBA/Results/VegardSOBPs"; width = "SOBP" # SOBPs
    #path="/media/vegardmj/TOSHIBA/Results/
        ↪ LongWaterPhantom_150MeV_4x4cm_1mmZ_451bins/
        ↪ FLUKA_0Hounsfield_150MeV_BP_1mm/0Hounsfield_150MeV_BP_1mm_DICOMs/
        ↪ FLUKA_DICOM"; width = "PBP" #PBPs
    #path = raw_input("Provide path for the RS dataset folder: "); width = "0"
    print width
    print("1. depth_dose_plot() \n2. plot_matrix() \n3. print_matrix() \n4. test_range_shift() \
        ↪ n5. plot_momentum_spread_vs_energy() \n6. plot_multiple_1d() \n7. 
        ↪ plot_fluka_plot_data()\n8. plot_RS_vs_number_of_energies() \n9. plot_deviation-
        ↪ matrix() \n10. plot_RS_bars()")
    choice=raw_input("Choose what to do: ")
    if width=="0":
```

```
    print "1. PBPs\n 2. SOBPs (SOBP)"
    width=raw_input("Choose model: ")
    if width == "1":
        width = "PBP"
    elif width == "2":
        width == "SOBP"
percent=raw_input("Choose distal dose fall-off percentage (float) (space for 0.8, 
    ↪ standard): ")
model=raw_input("Choose RBE model to compare with RBE=1.1 (space for ROR): ")
if model=="":
    model="ROR"
if percent=="":
    percent="0.8"
percent=float(percent)

if choice == "1":
    depth_dose_plot()

if choice == "2":
    M, list_of_doses, list_of_energies, list_of_ab = supplementary.read_range_shift_data(path,
        ↪ model, percent, width)
    deg=raw_input("Degree of fit (int):")
    deg=int(deg)
    plot_matrix(path, M, list_of_doses, list_of_energies, list_of_ab, model, deg, width, False,
        ↪ percent)

if choice == "3":
    M, list_of_doses, list_of_energies, list_of_ab = supplementary.read_range_shift_data(path,
        ↪ model, percent, width)
    print_matrix(M, list_of_doses, list_of_energies, list_of_ab)

if choice == "4":
    test_range_shift(path, percent, model)

if choice == "5":
    E = raw_input("Energy-array (MeV): ")
    E=E.split(" ")
    E=[float(e) for e in E]
    rel_momentum_spread_plot(E)

if choice == "6":
    plot_multiple_1d()

if choice == "7":
    plot_fluka_plot_data()

if choice == "8":
    plot_RS_NoE()

if choice == "9":
    M, list_of_doses, list_of_energies, list_of_ab = supplementary.read_range_shift_data(path,
        ↪ model, percent, width)
    deg=raw_input("Degree of fit (int):")
    deg=int(deg)
```

```
        p_names, p, Rsquared = supplementary.reg_sklearn(list_of_doses,list_of_energies,list_of_ab,
            ↪ M,deg)
        min_dev, max_dev, var, dev_M, STD = supplementary.get_deviation([list_of_doses,
            ↪ list_of_energies,list_of_ab],M,p_names,p)
        plot_matrix(path, dev_M, list_of_doses, list_of_energies, list_of_ab, model, deg, width,
            ↪ True, percent)

    if choice == "10":
        plot_RS_bars()

#------------------------------Functions
    ↪ ------------------------------------

def plot_matrix(path, M, list_of_doses, list_of_energies, list_of_ab, model, deg, width, dev_true,
    ↪ percent):
        res=2;res2=5

        #Prompts user for axes
        print("1. Dose \n2. Energy \n3. alpha/beta")
        plot_type = raw_input("Plot range shift versus (space between): ")
        plot_type=plot_type.split()
        plot_type=[int(p) for p in plot_type]
        locked="D"*(1 not in plot_type)+"E"*(2 not in plot_type)+"ab"*(3 not in plot_type)


        #plot parameters + iterators and lists of plots and legends
        markers = ['o', 'x', '^', '<', '>', '1', '2', '3', '4', 's', 'p', '*', '+', 'v', 'h', 'H', 'D']
        colors=['b', 'g', 'r', 'c', 'm', 'y', 'k']
        color_it=-1
        marker_it=-1
        plots=[]
        legend=[]
        X_plot=[];Y_plot=[];RS_plot=[];

        coeff_filename = "coeffs_{}_D{}_{}_{}.txt".format(model, percent*100,deg, width)
        if os.path.isfile(path + "/" + coeff_filename) and dev_true==False:
            print "Loading coefficients"
            p_names, p, Rsquared, dev, var, STD = supplementary.load_coefficients(path, model,
                ↪  deg, width, percent)
            print "STD = {}mm".format(STD)
        else:
            print "Finding coefficients and saving"
            p_names, p, Rsquared = supplementary.reg_sklearn(list_of_doses,list_of_energies,
                ↪ list_of_ab,M,deg)
            dev_min, dev_max, var, dev_M, STD = supplementary.get_deviation([list_of_doses,
                ↪ list_of_energies,list_of_ab],M,p_names,p)
            print "STD = {}mm".format(STD)
            if dev_true == False:
                supplementary.save_coefficients(path, model, deg, width, Rsquared, [dev_min,
                    ↪ dev_max], var, STD, p_names, p, percent)

        D, E, AB, RS = supplementary.make_fit_arrays(list_of_doses, list_of_energies, list_of_ab,
            ↪ p_names, p, res)
```

```
#3D plot
if len(plot_type)==2:

        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')

        #Dose and Energy
        if 1 in plot_type and 2 in plot_type:

                for i in range(len(list_of_ab)):
                        print str(i+1) + ". " + str(list_of_ab[i])
                ab=raw_input("alpha/beta values: ")
                ab = ab.split()
                ab = [int(i)-1 for i in ab]
                cycle_size=len(list_of_doses)*len(list_of_energies)
                for i in ab:
                    marker_it+=1
                    color_it+=1
                    ab_i=supplementary.get_index(AB[0,0,:],list_of_ab[i])[0]
                    X_plot.append(D[:,:,ab_i]);Y_plot.append(E[:,:,ab_i]);RS_plot.append
                         ↪ (RS[:,:,ab_i]);
                    for j in range(len(list_of_doses)):
                        for k in range(len(list_of_energies)):
                            plots.append(ax.scatter(list_of_doses[j], list_of_energies[k], M[
                                 ↪ j,k,i], c=colors[color_it%len(colors)], zdir='z'))
                            #plots.append(ax.scatter(list_of_doses[j], list_of_energies[k],
                                 ↪ M[j,k,i], c=colors[color_it%len(colors)], marker=
                                 ↪ markers[marker_it%len(markers)], zdir='z'))
                            legend.append(r'$(\alpha/\beta)_x$' + " (Gy) = " + str(
                                 ↪ list_of_ab[i]))
                ax.set_xlabel('Dose (Gy (RBE))', fontsize="16")
                ax.set_ylabel('Beam Energy (MeV)', fontsize="16")

                #Tests Nearest Neighbour
                if False:
                    NN_d, NN_e, NN_a = prompt_NN()
                    NN_rs = supplementary.nearest_neighbour(NN_d, NN_e, NN_a,
                         ↪ list_of_doses, list_of_energies, list_of_ab, M)
                    ax.scatter(NN_d,NN_e, NN_rs, marker = 'o', color='r', zdir='z')

        #Dose and ab
        if 1 in plot_type and 3 in plot_type:
                for i in range(len(list_of_energies)):
                        print str(i+1) + ". " + str(list_of_energies[i])
                energy=raw_input("Initial beam energy: ")
                energy=energy.split()
                energy=[int(i)-1 for i in energy]
                cycle_size=len(list_of_doses)*len(list_of_ab)
                for k in energy:
                    color_it+=1
                    marker_it+=1
                    e_k=supplementary.get_index(E[0,:,0],list_of_energies[k])[0]
                    X_plot.append(D[:,e_k,:]);Y_plot.append(AB[:,e_k,:]);RS_plot.append
                ↪ (RS[:,e_k,:]);
            for i in range(len(list_of_ab)):
                for j in range(len(list_of_doses)):
                    plots.append(ax.scatter(list_of_doses[j], list_of_ab[i], M[j,k,i],
                        ↪ c=colors[color_it%len(colors)], zdir='z'))
                    #plots.append(ax.scatter(list_of_doses[j], list_of_ab[i], M[j,k,i
                        ↪ ], c=colors[color_it%len(colors)], marker=markers[
                        ↪ marker_it%len(markers)], zdir='z'))
                    legend.append('Beam␣Energy␣(MeV)␣=␣' + str(
                        ↪ list_of_energies[k]))
        ax.set_xlabel('Dose␣(Gy␣(RBE))', fontsize="16")
        ax.set_ylabel(r'(\alpha/\beta)_x'+ "␣(Gy)", fontsize="16")

#Energy and ab
if 2 in plot_type and 3 in plot_type:
        for i in range(len(list_of_doses)):
                print str(i+1) + ".␣" + str(list_of_doses[i])
        dose=raw_input("Dose␣levels:␣")
        dose=dose.split()
        dose=[int(i)−1 for i in dose]
        cycle_size=len(list_of_ab)∗len(list_of_energies)
        for j in dose:
            color_it+=1
            marker_it+=1

            d_j=supplementary.get_index(D[:,0,0],list_of_doses[j])[0]
            X_plot.append(E[d_j,:,:]);Y_plot.append(AB[d_j,:,:]);RS_plot.append(
                ↪ RS[d_j,:,:]);
            for i in range(len(list_of_ab)):
                for k in range(len(list_of_energies)):
                    plots.append(ax.scatter(list_of_energies[k], list_of_ab[i], M[j,k
                        ↪ ,i], c=colors[color_it%len(colors)], zdir='z'))
                    #plots.append(ax.scatter(list_of_energies[k], list_of_ab[i], M[j,
                        ↪ k,i], c=colors[color_it%len(colors)], marker=
                        ↪ markers[marker_it%len(markers)], zdir='z'))
                    legend.append("Dose␣(Gy␣(RBE_{1.1}))" + str(
                        ↪ list_of_doses[j]))
        ax.set_xlabel('Beam␣Energy␣(MeV)', fontsize="16")
        ax.set_ylabel(r'(\alpha/\beta)_x'+ "␣(Gy)", fontsize="16")

#Fixes legend and shows plot
legend_true=[]
plots_true=[]
for l in range(len(legend)):
    if l%cycle_size==0:
        plots_true.append(plots[l])
        legend_true.append(legend[l])
plt.legend(handles=plots_true, labels=legend_true, fontsize="16")
ax.set_zlabel('Range␣Shift␣(mm)', fontsize="16")
color_it=−1
print "planes"
for i in range(len(X_plot)):
    color_it+=1
    ax.plot_wireframe(X_plot[i], Y_plot[i], RS_plot[i], rstride=int(math.ceil(float
                    ↪ (res)/float(res2))), cstride=int(math.ceil(float(res)/float(res2))),
                    ↪ color=colors[color_it%len(colors)])



            plt.tick_params(axis="x", labelsize=16)
            plt.tick_params(axis="y", labelsize=16)
            plt.tick_params(axis="z", labelsize=16)
            plt.show()

    #2D plot
    if len(plot_type)==1:

        #Dose
        if 1 in plot_type:
            for i in range(len(list_of_energies)):
                print str(i+1) + ". " + str(list_of_energies[i])
            energy=raw_input("Initial beam energies: ")
            energy=energy.split()
            energy = [int(e)-1 for e in energy]
            for i in range(len(list_of_ab)):
                print str(i+1) + ". " + str(list_of_ab[i])
            ab=raw_input("alpha/beta values: ")
            ab=ab.split()
            ab = [int(i)-1 for i in ab]
            for i in energy:
                color_it+=1
                marker_it=0
                for j in ab:
                    e_i=supplementary.get_index(E[0,:,0],list_of_energies[i])[0]
                    ab_j=supplementary.get_index(AB[0,0,:],list_of_ab[j])[0]
                    plots.append(plt.plot(list_of_doses, M[:,i,j], markers[
                        ↪ marker_it%len(markers)], color=colors[color_it%len
                        ↪ (colors)]))
                    plots.append(plt.plot(D[:,e_i,ab_j], RS[:,e_i,ab_j], color=
                        ↪ colors[color_it%len(colors)]))
                    marker_it+=1
                    legend.append("Beam energy = " + str(list_of_energies[i])
                        ↪ + ", " + r'$(\alpha/\beta)_x$ = ' + str(list_of_ab[
                        ↪ j]))
                    legend.append("reg")
            plt.xlabel("Dose level (Gy)", fontsize="16")

        #Energy
        if 2 in plot_type:
            for i in range(len(list_of_doses)):
                print str(i+1) + ". " + str(list_of_doses[i])
            dose=raw_input("Dose levels: ")
            dose=dose.split()
            dose = [int(d)-1 for d in dose]
            for i in range(len(list_of_ab)):
                print str(i+1) + ". " + str(list_of_ab[i])
            ab=raw_input("alpha/beta values: ")
            ab=ab.split()
            ab = [int(i)−1 for i in ab]
            for i in dose:
                color_it+=1
                marker_it=0
                for j in ab:
                    d_i=supplementary.get_index(D[:,0,0],list_of_doses[i])[0]
                    ab_j=supplementary.get_index(AB[0,0,:],list_of_ab[j])[0]
                    plots.append(plt.plot(list_of_energies, M[i,:,j], markers[
                        ↪ marker_it%len(markers)], color=colors[color_it%len
                        ↪ (colors)]))
                    plots.append(plt.plot(E[d_i,:,ab_j], RS[d_i,:,ab_j], color=
                        ↪ colors[color_it%len(colors)]))
                    marker_it+=1
                    legend.append("Dose level = " + str(list_of_doses[i]) + ",
                        ↪ " + r'$(\alpha/\beta)_x$ = ' + str(list_of_ab[j]))
                    legend.append("reg")
            plt.xlabel("Beam Energy (MeV)", fontsize="16")

        #ab
        if 3 in plot_type:
            for i in range(len(list_of_doses)):
                print str(i+1) + ". " + str(list_of_doses[i])
            dose=raw_input("Dose levels: ")
            dose=dose.split()
            dose = [int(d)−1 for d in dose]
            for i in range(len(list_of_energies)):
                print str(i+1) + ". " + str(list_of_energies[i])
            energy=raw_input("Initial beam energies: ")
            energy=energy.split()
            energy = [int(e)−1 for e in energy]
            for i in energy:
                color_it+=1
                marker_it=0
                for j in dose:
                    d_i=supplementary.get_index(D[:,0,0],list_of_doses[j])[0]
                    e_j=supplementary.get_index(E[0,:,0],list_of_energies[i])[0]
                    #print E[0,:,0]
                    #print list_of_energies
                    plots.append(plt.plot(list_of_ab, M[j,i,:], markers[marker_it%
                        ↪ len(markers)], color=colors[color_it%len(colors)]))
                    plots.append(plt.plot(AB[d_i,e_j,:], RS[d_i,e_j,:], color=colors
                        ↪ [color_it%len(colors)]))
                    marker_it+=1
                    legend.append("Dose level = " + str(list_of_doses[j]) + ",
                        ↪ " + "Beam energy = " + str(list_of_energies[i]))
                    legend.append("reg")
            plt.xlabel(r'$(\alpha/\beta)_x$ (Gy)', fontsize="16")

        legend_true=[]
        plots_true=[]
        cycle_size=2
        for l in range(len(legend)):
            if l%cycle_size==0:
                plots_true.append(plots[l][0])
```

```
                    legend_true.append(legend[l])
                plt.legend(handles=plots_true, labels=legend_true, fontsize="16")
                plt.ylabel("Range␣shift␣(mm)", fontsize="16")
                plt.tick_params(axis="x", labelsize=16)
                plt.tick_params(axis="y", labelsize=16)

                plt.show()


def prompt_NN():
    d = raw_input("Dose:␣")
    e = raw_input("energy:␣")
    a = raw_input("ab:␣")
    return float(d), float(e), float(a)

def test_range_shift(path, percent, model):
    degree=raw_input("Max␣degree␣in␣fit␣:␣")
    if degree=="":
        degree="3␣4␣5␣6␣7␣8␣9"
    degree=degree.split("␣")
    print "−−−−−−−−−−−−−−−−−−−"
    for deg in degree:
        deg=int(deg)
        d="2" #raw_input("Dose level: ")
        e="150" #raw_input("Initial beam energy: ")
        a="2" #raw_input("ab value: ")
        M, D, E, ab = supplementary.read_range_shift_data(path, percent)
        p_names, p, Rsquared = supplementary.reg_sklearn(D,E,ab,M,deg)
        RS=supplementary.range_shift(float(d),float(e),float(a),p_names,p)
        print "RS:␣" + str(RS) + "␣mm"
        print "Rsquared:␣"+str(Rsquared)
        print deg
        print "−−−−−−−−−−−−−−−−−−−"


def depth_dose_plot():
    more = True
    DD = []
    legend = []
    plt.xlabel("Depth␣(cm)", fontsize="16")
    plt.ylabel("Dose␣(Gy)", fontsize="16")
    plt.tick_params(axis="x", labelsize=16)
    plt.tick_params(axis="y", labelsize=16)
    plt.grid(True)
    path1 = "/media/vegardmj/TOSHIBA/Results/VegardSOBPs/SOBP7/
        ↪ FLUKA_0Hounsfield_VegardSOBP7/0Hounsfield_VegardSOBP7_DICOMs/
        ↪ FLUKA_DICOM/data_1d"
    print "Path␣may␣be␣too␣long,␣causing␣the␣list␣if␣files␣not␣to␣show␣up."
    print "If␣so,␣write␣'..'␣and␣press␣enter␣to␣show.␣'..'␣can␣also␣be␣used␣together␣with␣
        ↪ keywords␣filtering␣the␣list."
    while more:
        path0 = raw_input("Provide␣1d␣data␣directory␣(space␣for␣same):␣")
        if path0=="":
            path0 = path1
```

```
path1 = path0
filename_list=[]
for path, subdir, file_list in os.walk(path0):
    for filename in file_list:
        filename_list.append(path+"/"+filename)
    break

for i in range(len(filename_list)):
    print str(i+1) + ". " + filename_list[i].split("/")[-1]
files = raw_input("Choose files to plot (space between): ")
if ".." in files:
    print ""
    i_temp=""
    files = files.split(" ")
    files.remove("..")
    for i in range(len(filename_list)):
        count = len(files)
        for f in files:
            if f in filename_list[i].split("/")[-1]:
                count-=1
        if count==0:
            print str(i+1) + ". " + filename_list[i].split("/")[-1]
            i_temp = i_temp + str(i+1) + " "
    files = raw_input("Choose files to plot (space between, enter for all): ")

if files=="":
    files = i_temp[:-1]
files = files.split(" ")
files = [int(f)-1 for f in files]

for f in files:
    DD.append(supplementary.read_depth_dose_simple(filename_list[f]))
    add = ""
    #add = raw_input("Add something to the legend?: ")
    legend.append(filename_list[f].split("/")[-1]+add)
for dd in DD:
    dd['depth'] = [d-dd['depth'][0] for d in dd['depth']]
    plt.plot(dd['depth'], dd['dose'])
plt.legend(legend, fontsize="16")
plt.grid(True)
plt.show()

#rem = raw_input("Wish to remove some plots? (y/n) ")
rem = "n"
if rem == "Y" or rem =="y":
    for i in range(len(files)):
        print str(i+1) + ". " + filename_list[files[i]].split("/")[-1]
    remove = raw_input("choose files to remove: (space between) ")
    remove = remove.split(" ")
    remove = [int(r)-1 for r in remove]
    remove.sort(reverse=True)
    for i in remove:
        del DD[i]
        del legend[i]
```

```
            for dd in DD:
                plt.plot(dd['depth'], dd['dose'])
            plt.legend(legend, fontsize="16")
            plt.grid(True)
            plt.show()

        moar = raw_input("Plot more? (y/n) ")
        if moar != "y" and moar != "Y":
            more=False

#Not in use
def plot_RS_bars():
    #Gets and prompts user for RS-files
    path_dcm = "/media/vegardmj/TOSHIBA/Results/pasient_plans/ExpC" #prostata
    #path_dcm = "/media/vegardmj/TOSHIBA/Results/pasient_plans/Waterphantom_Allfiles"
        ↪ #water
    #path_dcm = raw_input("Provide measured and predicted range shift output files: ")
    files=[]
    for path, subdir, file_list in os.walk(path_dcm):
        for filename in file_list:
            if ".txt" in filename:
                if "predicted" in filename:# or "measured" in filename:
                    files.append(path+"/"+filename)
        break

    for i in range(len(files)):
        print str(i+1) + ". " + files[i].split("/")[-1]
    f = raw_input("Choose range shift files to plot (space between): ")
    if f == "":
        f = [i for i in range(len(files))]
    else:
        f = f.split(" ")
        f = [int(i)-1 for i in f]

    PRS=[];
    for i in f:
        PRS.append(supplementary.read_predicted_RS_file(files[i]))
    MRS = [mrs for mrs in PRS[0]['measured']]

    dist = 0.04
    x=[0]; plots=[]; leg=[]; tl=[]; leg_plots=[]
    for mrs in MRS:
        x.append(x[-1]+1)
        plots.append(plt.bar(x[-1], mrs['range_shift'], color='r'))
        plt.text(x[-1]-0.4, dist + mrs['range_shift'] , "{0:.4f}".format(mrs['range_shift']),
            ↪ fontsize="16")
        #tl.append(mrs['model'] + "\n{}\n".format(mrs['field_name']) + r"$(\alpha/\beta)_x$
            ↪ ={}".format(mrs['abx']))
        #tl.append(mrs['model'] + "\n{}\n".format(mrs['field_name']) + "abx={}".format(mrs
            ↪ ['abx']))
        #tl.append("{}\n".format(mrs['field_name']) + r"$(\alpha/\beta)_x$={}".format(mrs['
            ↪ abx']))

        tl.append("{}\n".format(mrs['field_name'])) #prost
    #tl.append("abx={}".format(mrs['abx'])) #waterPahntom

    if "measured" not in leg:
        leg.append("measured")
        leg_plots.append(plots[-1])
    for prs in PRS:
        for main in prs['main']:
            if mrs['field_name'] == main['field_name'] and mrs['abx'] == main['abx']:
                x.append(x[-1]+1)
                if prs['method'] == "SOBP":
                    plots.append(plt.bar(x[-1], main['range_shift'], color='b'))
                    plt.text(x[-1]-0.4, dist+main['range_shift'] , "{0:.4f}".format(main['
                        ↪ range_shift']), fontsize="16")
                    #tl.append(main['model'] + "\n{}\n".format(main['field_name']) + r"$
                        ↪ (\alpha/\beta)_x$={}".format(main['abx']))
                    #tl.append(main['model'] + "\n{}\n".format(main['field_name']) + "
                        ↪ abx={}".format(main['abx']))

                    tl.append(main['model'] + "\n{}\n".format(main['field_name'])) #
                        ↪ prostata
                    #tl.append(main['model'] + "\n" + "abx={}".format(main['abx'])) #
                        ↪ waterPhantom

                if prs['method'] == "SOBP3":
                    plots.append(plt.bar(x[-1], main['range_shift'], color='y'))
                    plt.text(x[-1]-0.4, dist+main['range_shift'] , "{0:.4f}".format(main['
                        ↪ range_shift']), fontsize="16")
                    #tl.append(main['model'] + "\n{}\n".format(main['field_name']) + r"$
                        ↪ (\alpha/\beta)_x$={}".format(main['abx']))
                    #tl.append(main['model'] + "\n{}\n".format(main['field_name']) + "
                        ↪ abx={}".format(main['abx']))

                    tl.append(main['model'] + "\n{}\n".format(main['field_name'])) #
                        ↪ prostata
                    #tl.append(main['model'] + "\n" + "abx={}".format(main['abx'])) #
                        ↪ waterPhantom

                elif prs['method'] == "PBP":
                    plots.append(plt.bar(x[-1], main['range_shift'], color='g'))
                    plt.text(x[-1]-0.4, dist+main['range_shift'] , "{0:.4f}".format(main['
                        ↪ range_shift']), fontsize="16")
                    #tl.append(main['model'] + "\n{}\n".format(main['field_name']) + r"$
                        ↪ (\alpha/\beta)_x$={}".format(main['abx']))
                    #tl.append(main['model'] + "\n{}\n".format(main['field_name']) + "
                        ↪ abx={}".format(main['abx']))

                    tl.append(main['model'] + "\n{}\n".format(main['field_name'])) #
                        ↪ prostata
                    #tl.append(main['model'] + "\n" + "abx={}".format(main['abx'])) #
                        ↪ waterPhantom

        if prs['method'] not in leg:
            leg.append(prs['method'])
            leg_plots.append(plots[-1])
        x.append(x[−1]+1)
        plt.bar(x[−1], 0, color='k')
        tl.append("")

    x=x[1:−1]
    tl = tl[:−1]
    plt.xticks(x, tl, fontsize="10")
    plt.ylabel("Range␣Shift␣(mm)", fontsize="16")
    plt.legend(handles=leg_plots, labels=leg, fontsize="16")
    #plt.grid()
    plt.show()


def fix_bar_tick(string):
    if len(string)>10:
        string = string.split("␣")
        fixed=""
        for s in string:
            fixed = fixed + s + "\n"
        string = fixed
    return string

def plot_RS_NoE():
    path = raw_input("Provide␣1d_data␣directory:␣")

    #makes list of files and the different ab_x values
    files=[]; ab = []
    for path0, subdir, file_list in os.walk(path):
        for filename in file_list:
            files.append(path0 + "/" + filename)
            if "Bio" not in filename:
                if False and "weigh_method=1" in path0:
                    if float(filename.split("_")[3].split("−")[−1]) not in ab:
                        ab.append(float(filename.split("_")[3].split("−")[−1]))
                elif "weigh_method=3" in path0:
                    if float(filename.split("_")[1].split("=")[−1]) not in ab:
                        ab.append(float(filename.split("_")[1].split("=")[−1]))
                else:
                    if float(filename.split("_")[1].split("=")[−1]) not in ab:
                        ab.append(float(filename.split("_")[1].split("=")[−1]))
        break

    #Prompts user for ab_x values
    ab.sort()
    for i in range(len(ab)):
        print str(i+1) + ".␣" + str(ab[i])
    ab_choice = raw_input("Choose␣ab_x␣values␣(space␣between,␣enter␣for␣all):␣")
    if ab_choice == "":
        ab_choice = ab
    else:
        ab_choice = [ab[int(a−1)] for a in ab_choice.split("␣")]

    #Reads DD−relations and sorts them into RBE=1.1 weighted and variable RBE weighted
    bio = []; var = []; NoE = []
    for f in files:
        if False and "weigh_method=1" in path:
            abx_temp = f.split("/")[−1].split("_")[3].split("−")[−1]
        elif "weigh_method=3" in path:
            abx_temp = f.split("/")[−1].split("_")[1].split("=")[−1]
        else:
            abx_temp = f.split("/")[−1].split("_")[1].split("=")[−1]
        if "Bio" in f.split("/")[−1]:
            bio.append(supplementary.read_depth_dose_simple(f))
            bio[−1]['R80'] = supplementary.find_falloff(bio[−1]['depth'], bio[−1]['dose'], np.max(
                ↪ bio[−1]['dose'])∗0.8)
        elif float(abx_temp) in ab_choice:
            var.append(supplementary.read_depth_dose_simple(f))
            var[−1]['R80'] = supplementary.find_falloff(var[−1]['depth'], var[−1]['dose'], np.max(
                ↪ var[−1]['dose'])∗0.8)
            var[−1]['abx'] = float(abx_temp)
            var[−1]['NoE'] = int(var[−1]['filename'].split("_")[−3].split("−")[−1])
            if var[−1]['NoE'] not in NoE:
                NoE.append(var[−1]['NoE'])
    NoE.sort()

    #Calculates Range_shift
    for v in var:
        for b in bio:
            if v['filename'].split("_")[−3] == b['filename'].split("_")[−3]: #if same NoE
                v['range_shift'] = 10∗(v['R80'] − b['R80'])
                break
    #plots
    for a in ab_choice:
        RS=[]
        for noe in NoE:
            for v in var:
                if v['abx'] == a and v['NoE'] == noe:
                    RS.append(v['range_shift'])
                    break
        plt.plot(NoE, RS)
    plt.xlabel("Number of Energies in beam sequence", fontsize="16")
    plt.ylabel("Biological Range Shift (mm)", fontsize="16")
    plt.legend(['abx = {}'.format(a) for a in ab_choice], fontsize="16")
    plt.tick_params(axis="x", labelsize=16)
    plt.tick_params(axis="y", labelsize=16)
    plt.grid()
    plt.show()



def print_matrix(M, list_of_doses, list_of_energies, list_of_ab):
    print(list_of_doses)
    print(list_of_energies)
    print(list_of_ab)
    print("−−−−−−−−−−−−−−−−")
    for d in range(len(list_of_doses)):
        print("Dose = "+str(list_of_doses[d]))
        print(M[d,:,:])




#Plots momentum spread as a function of Energy (MeV)
def rel_momentum_spread_plot(E):
    E0=0.938

    E=[E[i]/1000 for i in range(len(E))]
    p = [np.sqrt(E[i]*(E[i] +2*E0)) for i in range(len(E))]
    rel_p_spread = [-4.6234*E[i]*E[i]*E[i] + 1.7547*E[i]*E[i] - 0.2159*E[i] + 0.0163 for i in
        ↪ range(len(E))]
    p_spread = [rel_p_spread[i]*p[i] for i in range(len(E))]
    plt.plot(E, p_spread, "or")

    x=np.linspace(0.07, 0.23, 1000)
    y_p=[np.sqrt(x[i]*(x[i] +2*E0)) for i in range(len(x))]
    y_fac=[-4.6234*x[i]*x[i]*x[i] + 1.7547*x[i]*x[i] - 0.2159*x[i] + 0.0163 for i in range(len(x)
        ↪ )]
    y_spread=[y_p[i]*y_fac[i] for i in range(len(x))]
    plt.plot(x, y_spread, "r")

    print("E\tdelta_p␣\t␣rel_p_spread␣\t␣p_spread")
    for i in range(len(E)):
        print(str(E[i])+"\t"+str(p[i])+"\t"+str(rel_p_spread[i])+"\t"+str(p_spread[i]))
    plt.title('Momentum␣spread␣as␣a␣function␣of␣energy')
    plt.ylabel('Momentum␣spread␣(GeV/c)')
    plt.xlabel('Energy␣(GeV)')
    plt.grid()
    plt.show()



def plot_fluka_plot_data():
    filenames=[]
    path0 = raw_input("Provide␣1d_plot␣directory:␣")
    paths=[]
    for path, subdir, file_list in os.walk(path0):
        for filename in file_list:
            if ".dat" in filename and "plot" in filename:
                paths.append(path)
                filenames.append(filename)
    for i in range(len(filenames)):
        print str(i+1) + ".␣" + filenames[i]
    I = raw_input("Choose␣files␣to␣plot:␣")
    I = I.split("␣")
    I = [int(i)-1 for i in I]

    plot_list=[]
    for i in I:
        os.chdir(paths[i])
        depth, dose, error = supplementary.read_fluka_plot_data(filenames[i])
        plot_list.append([depth, dose, error])
    peak_dose = np.max(plot_list[0][1])
    peak_range = plot_list[0][0][plot_list[0][1].index(peak_dose)]
    for dd in plot_list:
        dd[0], dd[1] = supplementary.align_peak(dd[0], dd[1], peak_dose, peak_range)
        plt.plot(dd[0], dd[1])

    plt.legend([filenames[i] for i in I])
    plt.xlabel('Depth (cm)')
    plt.ylabel('Dose (Gy)')
    plt.show()


#-----
def plot_RS_vs_number_of_energies():
    path_1d=raw_input("Provide 1d plots directory: ")

    percent = "0.8" #raw_input("Percentage dose fall off to study: (between 0 and 1) ")
    percent = float(percent)

    Bio = []; rel = []; ab=[]; noe=[]
    for path, subdir, file_list in os.walk(path_1d):
        for filename in file_list:
            os.chdir(path)
            dd = supplementary.read_DD(filename, 1, percent)
            dd['noe'] = float(filename.split("_")[-3].split("-")[-1])
            if dd['noe'] not in noe:
                noe.append(dd['noe'])
            if "FLKBio" in dd['modeltype']:
                Bio.append(dd)
            else:
                dd['ab_x']=float(filename.split("_")[1].split("-")[-1].split("=")[-1])
                rel.append(dd)
    ab=raw_input("Alpha/beta x values: ")
    ab=ab.split()
    ab=[float(a) for a in ab]
    noe.sort()
    ab.sort()
    RS=[[0 for j in noe] for i in range(len(ab))]
    i=0;legend_arr=[]
    plot_DD = True
    plot_list=[]
    for r in rel:
        for b in Bio:
            if r['noe']==b['noe']:
                R, D = supplementary.find_falloff(r['depth'], r['dose'], b['80'][1])
                r['range_shift'] = (R-b['80'][0])
                if r['ab_x'] in ab:
                    if i<6 and plot_DD and r['noe']%3==1:
                        legend_arr.append("{} NoE={} abx={}".format(r['model'], r['noe'], r['
                            ↪ ab_x']))
                        legend_arr.append("{} NoE={}".format(b['model'], b['noe']))
                        plot_list.append(plt.plot(r['depth'], r['dose']))
                        plot_list.append(plt.plot(b['depth'], b['dose']))
                    plt.plot([b['80'][0],b['80'][0]+r['range_shift']], [b['80'][1], b['80'][1]], 'o')
                    i=i+1
                RS[ab.index(r['ab_x'])][noe.index(r['noe'])] = r['range_shift']
    if plot_DD==False:
        legend_arr=[]
        for i in range(len(ab)):
            legend_arr.append("ab={}".format(ab[i]))
            plt.plot(noe, RS[i])
        plt.legend(legend_arr, fontsize="16")
        plt.xlabel("number␣of␣energies␣in␣SOBP", fontsize="16")
        plt.ylabel("range␣shift␣(mm)", fontsize="16")

    else:
        plt.legend(handles=plot_list, labels=legend_arr, fontsize="16")
        plt.xlabel("Depth␣(cm)", fontsize="16")
        plt.ylabel("Dose␣(Gy)", fontsize="16")
    plt.grid()
    plt.show()




def plot_multiple_1d():
    filenames=[]
    path0 = raw_input("Provide␣1d␣plot␣directory:␣")
    paths=[]
    for path, subdir, file_list in os.walk(path0):
        for filename in file_list:
            if ".dat" in filename:
                paths.append(path)
                filenames.append(filename)

    for i in range(len(filenames)):
        print str(i+1) + ".␣" + filenames[i]
    I = raw_input("Choose␣files␣to␣plot:␣")
    I = I.split("␣")
    I = [int(i)−1 for i in I]

    direction = "y" #raw_input("Is the beam in the positive x−direction? [y/n] ")

    nr_fractions = "1" #raw_input("Number of fractions: ")
    nr_fractions=int(nr_fractions)

    plot_list=[]
    for i in I:
        os.chdir(paths[i])
        plot_list.append(supplementary.read_DD(filenames[i], nr_fractions, 0.8))

    #print("1. No normalization \n2. Normalize individually with peak_dose=100% \n3.
        ↪ Normalize relative to prescribed RBE=1.1 dose")
    nm = "1" #raw_input("Choose normalization method: ")

    for dd in plot_list:
        maxDose=np.max(dd['dose'])/100
        if nm=="2":
            plt.plot(dd['depth'], [d/maxDose for d in dd['dose']])
        if nm=="1":
            plt.plot(dd['depth'], dd['dose'])
            plt.plot(dd['80'][0],dd['80'][1], 'o')
            #plt.plot(dd['distal range'], dd['dose_level'], 'o')
        if nm=="3":
            bio_list=[]
            for f in filenames:
                if dd['filename'].split("_")[−1] in f:
                    bio_dd = supplementary.read_DD(f, nr_fractions, 0.8)
            maxDose=np.max(bio_dd['dose'])/100
            plt.plot(dd['depth'], [d/maxDose for d in dd['dose']])

    plt.legend([filenames[i] for i in I])
    plt.xlabel('Depth␣(cm)')
    plt.ylabel('Dose␣(Gy)')
    plt.grid()
    plt.show()

def test_fall_off():
    R = [0, 1, 2, 3, 4, 5]
    D = [10, 8, 6, 4, 2, 0]
    doseToFind = 7.5
    r80 = supplementary.find_falloff(R, D, doseToFind)
    plt.plot(R, D)
    plt.plot(r80, doseToFind, 'o')
    plt.show()

def depth_dose_plot_OLD(path):

        #Lists of filenames, list of complete paths to files, filenames of the chosen files to plot,
            ↪ properties.
        filename_list = []
        path_list=[]
        plot_list=[]
        plot_path_list=[]
        legend_list=[]

        energies=[]
        models=[]
        doses=[]
        ab_values=[]
        #Collects files and properties
        for dirName, subdirList, file_list in os.walk(path):
            for filename in file_list:
                filename_list.append(os.path.join(filename))
                path_list.append(os.path.join(dirName,filename))
                properties=filename.split("_")[0].split("−")
                energy_temp=float(properties[0])
                model_temp=properties[1]
                dose_temp=float(properties[2].split("=")[−1])
                rel_model=False
                if len(properties)>3:
```

```
            if "abx=" in properties[3]:
                rel_model=True
        if energy_temp not in energies:
            energies.append(energy_temp)
        if model_temp not in models:
            models.append(model_temp)
        if dose_temp not in doses:
            doses.append(dose_temp)
        if rel_model:
            ab_value_temp=float(properties[3].split("=")[-1])
            if ab_value_temp not in ab_values:
                ab_values.append(ab_value_temp)
energies.sort()
doses.sort()
ab_values.sort()
#Opens up for the user to choose multiple plots
wish_to_add_plot=True
while wish_to_add_plot:
    #Presents properties and prompts user for what to plot
    rel_model_plot=False
    print "What␣models␣would␣you␣like␣to␣plot␣(space␣between)?"
    for i in range(len(models)):
        print str(i+1)+".␣" +models[i]
    model_choice=raw_input("")
    model_choice=model_choice.split("␣")
    model_choice=[models[int(i)-1] for i in model_choice]
    for i in range(len(model_choice)):
        if model_choice[i]!="Biological" and model_choice[i]!="Physical":
            rel_model_plot=True

    print "What␣doses␣(Gy)␣would␣you␣like␣to␣plot␣(space␣between)?"
    for i in range(len(doses)):
        print str(i+1)+".␣" +str(doses[i])
    dose_choice=raw_input("")
    dose_choice=dose_choice.split("␣")
    dose_choice=[doses[int(i)-1] for i in dose_choice]

    print "What␣energies␣(MeV)␣would␣you␣like␣to␣plot␣(space␣between)?"
    for i in range(len(energies)):
        print str(i+1)+".␣" +str(energies[i])
    energy_choice=raw_input("")
    energy_choice=energy_choice.split("␣")
    energy_choice=[energies[int(i)-1] for i in energy_choice]
    if rel_model_plot:
        print "What␣ab-values␣(Gy)␣would␣you␣like␣to␣plot␣(space␣between)?"
        for i in range(len(ab_values)):
           print str(i+1)+".␣"+ str(ab_values[i])
        ab_choice=raw_input("")
        ab_choice=ab_choice.split("␣")
        ab_choice=[ab_values[int(i)-1] for i in ab_choice]

    #Finds the files to be plotted
    for i in range(len(filename_list)):
        properties=filename_list[i].split("_")[0].split("-")
        energy_temp=float(properties[0])
        model_temp=properties[1]
        dose_temp=float(properties[2].split("=")[-1])
        rel_model=False
        if len(properties)>3:
            if "abx=" in properties[3]:
                rel_model=True
                ab_value_temp=float(properties[3].split("=")[-1])
        if energy_temp in energy_choice and dose_temp in dose_choice and model_temp
            ↪  in model_choice:
            legend_temp=model_temp + ", energy: "+str(energy_temp) + " MeV, dose
                ↪ -level: " + str(dose_temp) + " Gy"
            if rel_model==True and ab_value_temp in ab_choice:
                legend_temp = legend_temp + ", ab: "+str(ab_value_temp)+" Gy"
                plot_list.append(filename_list[i])
                plot_path_list.append(path_list[i])
                legend_list.append(legend_temp)
            elif rel_model==False:
                plot_list.append(filename_list[i])
                plot_path_list.append(path_list[i])
                legend_list.append(legend_temp)


#Lists the files to be plotted to the user, and asks if they want to plot any more files
happy=False
while happy==False:
    print "The following files will be plotted."
    for i in range(len(plot_list)):
        print str(i+1)+". " + plot_list[i]
    choice=raw_input("To delete some of them, insert the number of the file you
        ↪  want to delete. To plot more files, insert 'y'. (Space for done)")
    if choice=="":
        happy=True
        wish_to_add_plot=False
    elif choice != "y":
        choice=choice.split(" ")
        choice=[int(c)-1 for c in choice]
        choice.sort()
        choice=choice[::-1]
        for i in choice:
            del plot_list[i]
            del plot_path_list[i]
            del legend_list[i]
    else:
        happy=True

if wish_to_add_plot==False:
    #Read files and plots them
    for filename in plot_path_list:
        data=open(filename)
        depth=[]
        dose=[]
        for line in data:
            line_data = line.split(" ")
```

```
                depth.append(float(line_data[0]))
                dose.append(float(line_data[2]))
            del depth[0]
            del dose[0]
            depth=[depth[i]−depth[0] for i in range(len(depth))]
            data.close()
            plt.plot(depth, dose)

        plt.xlabel("Depth␣(cm)", fontsize="16")
        plt.ylabel("Dose␣(Gy)", fontsize="16")
        plt.legend(legend_list, fontsize="16")
        plt.show()
        again = raw_input("Would␣you␣like␣to␣add␣a␣plot?␣(y/n␣or␣space)")
        if again=="y":
            wish_to_add_plot=True


main()
```

Listing 8: Python script, *supplementary.py*, is a complimentary script containing multiple functions used by the other scripts.

```
from os.path import join as pjoin
import string
import numpy as np
import math, sys, os
import matplotlib.pyplot as plt
from math import log10, floor
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import curve_fit
from sklearn import linear_model
import dicom



#Returns fitted M−matrix with res^3 as many values. This is used to plot the planes in analyze.
    ↪ plot_matrix()
def make_fit_arrays(x_in, y_in, z_in, p_names, p, res):
    x = [x_in[i] + (x_in[i+1]−x_in[i])∗j/res for i in range(len(x_in)−1) for j in range(res)]
    x.append(x_in[−1])
    y = [y_in[i] + (y_in[i+1]−y_in[i])∗j/res for i in range(len(y_in)−1) for j in range(res)]
    y.append(y_in[−1])
    z = [z_in[i] + (z_in[i+1]−z_in[i])∗j/res for i in range(len(z_in)−1) for j in range(res)]
    z.append(z_in[−1])

    X=np.zeros((len(x),len(y), len(z)))
    Y=np.zeros((len(x),len(y), len(z)))
    Z=np.zeros((len(x),len(y), len(z)))
    RS=np.zeros((len(x),len(y), len(z)))
    for j in range(len(x)):
        #print j
        for k in range(len(y)):
            for i in range(len(z)):
                X[j,k,i]=x[j]
                Y[j,k,i]=y[k]
                Z[j,k,i]=z[i]
                RS[j,k,i]=range_shift(x[j], y[k], z[i], p_names, p)
    return X, Y, Z, RS



#Returns rangeshift for given d,e,a with coefficients p
def range_shift(d, e, a, p_names, p):
    rs=0
    for i in range(len(p)):
        dpow=int(p_names[i][1])
        epow=int(p_names[i][2])
        apow=int(p_names[i][3])
        #print rs
        #print p_names[i]
        rs=rs+p[i]*pow(d,dpow)*pow(e,epow)*pow(a,apow)
    #print rs
    return rs

#Returns max and min absolute deviation of model from measurements. points=[x, y, z], xyz=
    ↪ lists, M=[i,j,k] of measurements. p coefficients.
def get_deviation(points, M, p_names, p):
    max_dev=0
    min_dev=abs(range_shift(points[0][0],points[1][0],points[2][0],p_names,p) − M[0,0,0])
    tot_var=0
    dev_M=M
    for i in range(len(points[0])):
        for j in range(len(points[1])):
            for k in range(len(points[2])):
                dev=range_shift(points[0][i],points[1][j],points[2][k],p_names,p) − M[i,j,k]
                tot_var=tot_var+dev*dev
                dev_M[i,j,k]=dev
                if dev<min_dev:
                    min_dev=dev
                if dev>max_dev:
                    max_dev=dev
    STD = np.sqrt(tot_var/np.size(M))
    return min_dev, max_dev, tot_var, dev_M, STD


#Multivariable nonlinear regression.
#Transforms nonlinear problem into linear one.
def reg_sklearn(D_in, E_in, ab_in, M, deg):
    N_D=len(D_in)
    N_E=len(E_in)
    N_ab=len(ab_in)
    #Tetrahedral numbers:
    deg=deg+1
    N_p=deg*(deg+1)*(deg+2)/6
    N_p=int(N_p)
    deg=deg−1
    D=[]
    E=[]
    ab=[]
    RS=[]
    #Elongates the arrays to match the length of RS−matrix
    for j in range(N_D):
        for k in range(N_E):
            for i in range(N_ab):
                D.append(D_in[j])
                E.append(E_in[k])
                ab.append(ab_in[i])
                RS.append(M[j,k,i])

    X=np.zeros((N_p−1, N_D∗N_E∗N_ab))
    #Names and sequence of the coefficients. Indexes: first=pow(D), second=pow(E), third=pow(
        ↪ ab)
    #Also makes the power arrays, ie: Dˆ2, Dˆ2∗E, etc
    p_names=[]
    x=0;y=0;z=0;m_x=4;m_y=4;
    for n in range(N_p):
        p_names.append("p{}{}{}".format(z,y,x))
        if n!=0:
            X[n−1,:]=[pow(D[i],z)∗pow(E[i],y)∗pow(ab[i],x) for i in range(len(D))]
        x=(x+1)%m_x; y=(y+(x==0))%m_y; z=(z+1∗(x==0 and y==0));m_x=deg+1−z−y;
            ↪ m_y=deg+1−z
    #print X
    #print p_names
    X=X.transpose()

    regr = linear_model.LinearRegression()
    regr.fit(X, RS)
    Rsquared=regr.score(X,RS)
    #print Rsquared

    #print('Intercept: ', regr.intercept_)
    #print('Coefficients: ', regr.coef_)
    p=[regr.intercept_]
    for i in range(N_p−1):
        p.append(regr.coef_[i])
    #print p

    return p_names, p, Rsquared

#Reads the .txt−file created by makeData.py
def read_range_shift_data(path, model, percent,width):
    if path=="":
        path=raw_input("Provide␣path␣to␣RS␣data.txt−file:␣")
    filename="RS_data_{}_D{}_{}.txt".format(model,percent∗100,width)
    os.chdir(path)
    rfile=open(filename, 'r')
    D=rfile.readline().split("\n")[0][1:−1].split(",␣")
    E=rfile.readline().split("\n")[0][1:−1].split(",␣")
    ab=rfile.readline().split("\n")[0][1:−1].split(",␣")
    D=[float(d) for d in D]
    E=[float(e) for e in E]
    ab=[float(a) for a in ab]
    M=np.zeros((len(D),len(E),len(ab)))
    for j in range(len(D)):
        for k in range(len(E)):
            line=rfile.readline().split(" ")[:-1]
            line=[float(l) for l in line]
            M[j,k,:]=line
        line=rfile.readline()
    return M, D, E, ab

#Calculates Range Shifts from folder with 1D-plots. Writes out RS_data-file and DD_data-file
    ↪ in previous directory
#def write_range_shift_data(


#Estimates range shift through nearest-neighbour method form dataset
#Nearest neighbour method is basicly weighted average in multiple dimensions.
def nearest_neighbour(d, e, a, D, E, A, RS):
    NN=[];

    #Finds nearest neighbour weights N[0, 2, 4] = Weight for the lowest
    for j in range(len(D)):
        if D[j]>=d:
            NN.append((D[j]-d)/(D[j]-D[j-1]))
            NN.append(1 - NN[-1])
            dj = j-1
            break
    for k in range(len(E)):
        if E[k]>e:
            NN.append((E[k]-e)/(E[k]-E[k-1]))
            NN.append(1 - NN[-1])
            ek = k-1
            break
    for i in range(len(A)):
        if A[i]>=a:
            NN.append((A[i]-a)/(A[i]-A[i-1]))
            NN.append(1 - NN[-1])
            ai = i-1
            break

    #Calculates rs through nearest neighbours.
    rs_d = [0, 0, 0, 0]
    rs_de = [0, 0]
    #The 3D cube with 8 nearest neighbours are turned into 2D rectangle by collapsing the dose
        ↪ -dimension
    for n in range(8):
        m = int(n/2)
        j = n%2
        k = int(n>1) - int(n>5)
        i = int(n>3)
        #rs_d[m] = rs_d[m] + NN[n%2] * RS[dj, ek, ai]
        rs_d[m] = rs_d[m] + NN[n%2] * RS[dj + j, ek + k, ai + i]
    #2D into 1D by collapsing the energy-dimension
    for n in range(4):
        m = int(n/2)
        l = int(n>0) - int(n>2)
        rs_de[m] = rs_de[m] + NN[2 + l] ∗ rs_d[n]
    rs = NN[4]∗rs_de[0]+NN[5]∗rs_de[1]
    return rs


#Reads predicted range shift output file NOT IN USE
def read_predicted_RS_file(filename):
    file = open(filename)
    prs = {}
    prs['method'] = filename.split("/")[−1].split("_")[−1].split(".")[0]
    prs['patient_name'] = file.readline().split(":␣")[−1]
    prs['plan_name'] = file.readline().split(":␣")[−1]
    prs['number_of_beams'] = float(file.readline().split(":␣")[−1])
    prs['number_of_fractions'] = float(file.readline().split(":␣")[−1])
    prs['iso_pos'] = file.readline().split(":␣")[−1]
    prs['MU_NP'] = file.readline().split(":␣")[−1]
    prs['patient_orientation'] = file.readline().split(":␣")[−1]
    prs['source_axis_distance'] = file.readline().split(":␣")[−1]



    prs['field_name_list'] = []
    prs['gantry_angle_list'] = []
    prs['number_of_abx']=[]
    prs['main'] = []
    line = file.readline()
    prs['length'] = [0, 0]
    for i in range(int(prs['number_of_beams'])):
        line = file.readline().split(":␣")
        prs['field_name_list'].append(str(line[1].split(",")[0]))
        prs['gantry_angle_list'].append(float(line[−1]))
        line = file.readline()
        line = file.readline()
        while "Range_Shift" not in line and "−−−−−−−−−−−" not in line and line != "\n":
            main = {}
            line_temp = line.split("␣")
            if line_temp == ['\n']:
                break
            l_count = 0
            line=['']
            for l in range(len(line_temp)):
                if line_temp[l] != '':
                    line[l_count] += line_temp[l] + '␣'
                elif line_temp[l−1] != '':
                    l_count+=1
                    line.append('')
            if line[−1] == '\n':
                line = line[:−1]
            main['model'] = line[0]
            if "Nearest_Neighbour" in main['model']:
                main['model'] = "NN␣"
                main['STD'] = "NA"
            else:
                main['STD'] = float(line[1])
                main['model'] = "␣".join(["Reg", main['model'].split("␣")[−2]])
            main['max_beam_energy'] = float(line[2])
            main['fraction_dose'] = float(line[3])
            main['abx'] = float(line[4])
            if main['abx'] not in prs['number_of_abx']:
                prs['number_of_abx'].append(main['abx'])
            main['range_shift'] = float(line[5])
            main['field_name'] = prs['field_name_list'][i]
            prs['main'].append(main)
            prs['length'][0] += 1
            line = file.readline()
            if "---------------------" in line:
                line = file.readline()
        print "predicted RS while done"

    line=file.readline()
    line=file.readline()
    if "RT.Plan" in line:
        line = file.readline()
        prs['measured']=[]
        while line != "":
            main = {}
            line_temp = file.readline().split(" ")
            if line_temp == ['']:
                break
            l_count = 0
            line=['']
            for l in range(len(line_temp)):
                if line_temp[l] != '':
                    line[l_count] += line_temp[l] + ' '
                elif line_temp[l-1] != '':
                    l_count+=1
                    line.append('')
            line = line[:-1]
            main['model'] = line[0]
            main['STD'] = "NA"
            main['field_name'] = str(line[1][:-1])
            main['max_beam_energy'] = float(line[2])
            main['fraction_dose'] = float(line[3])
            main['abx'] = float(line[4])
            if main['abx'] not in prs['number_of_abx']:
                prs['number_of_abx'].append(main['abx'])
            main['range_shift'] = float(line[5])
            prs['measured'].append(main)
            prs['length'][1] += 1
        print "measured RS while done"
    file.close()
    prs['number_of_abx'] = sum(prs['number_of_abx'])
    return prs


#Loads 3D fit-coefficients from a .txt file in FLUKA_DICOM folder
def load_coefficients(path, model, deg, width, percent):
    os.chdir(path)
    filename="coeffs_{}_D{}_{}_{}.txt".format(model,percent*100,deg,width)
    data=open(filename, "r")
    Rsquared=float(data.readline().split("␣")[−1])
    dev=data.readline().split("␣")[−2:]
    dev=[float(dev_i) for dev_i in dev]
    var=float(data.readline().split("␣")[−1])
    STD=float(data.readline().split("␣")[−1])

    p=[]; p_names=[]; i=0;
    for line in data:
        i=i+1
        if i>1:
            line_temp=line.split("\t")
            p_names.append(line_temp[0])
            p.append(float(line_temp[1]))
    data.close()
    return p_names, p, Rsquared, dev, var, STD

#Saves 3D fit−coefficients in a .txt file in FLUKA_DICOM folder
def save_coefficients(path, model, deg, width, Rsquared, dev, var, STD, p_names, p, percent):
    os.chdir(path)
    filename="coeffs_{}_D{}_{}_{}.txt".format(model,percent*100,deg,width)
    data=open(filename,"w")
    data.write("Rsquared␣=␣" + str(Rsquared) + "\n")
    data.write("min/max␣deviation␣=␣" + str(dev[0]) + "␣" + str(dev[1]) + "\n")
    data.write("Total␣variance␣=␣" + str(var) + "\n")
    data.write("Standard␣Deviation␣=␣" + str(STD) + "\n")
    data.write("Coefficient␣name:\tCoefficient␣value:\n")
    for i in range(len(p)):
        data.write(p_names[i] + "\t" + str(p[i]) + "\n")
    data.close()


#Reads depth−dose plot .dat from FLUKA plot−function. Returns depth, dose, error
def read_fluka_plot_data(filename):
    file = open(filename, 'r')
    dummy=file.readline()
    depth=[];dose=[];error=[]
    for line in file:
        linesplit=line.split()
        depth.append((float(linesplit[0])+float(linesplit[1]))/2)
        dose.append(float(linesplit[2]))
        error.append(float(linesplit[3]))
    file.close()
    return depth, dose, error

#Aligns max−value, both in height, and in range
def align_peak(depth, dose, peak_dose, peak_range):
    maxDose = np.max(dose)
    maxDoseRange = depth[dose.index(maxDose)]
    dose = [peak_dose*d/maxDose for d in dose]
    depth = [d+(peak_range−maxDoseRange) for d in depth]
    return depth, dose

#Reads field.dat file
```

```
def read_field_dat_file(filename):
    file = open(filename, 'r')
    [patient, plan, field] = file.readline().split("−−−−")
    [primaries_tot, primaries_field] = file.readline().split("−−−−")
    variable_names = file.readline()
    E=[];pos=[];spot_size=[];spot_weight=[];beam_dir=[]
    E_unique=[]
    for line in file:
        linesplit=line.split()
        E.append(float(linesplit[0]))
        pos.append([float(linesplit[1]), float(linesplit[2]), float(linesplit[3])])
        spot_size.append([float(linesplit[4]), float(linesplit[5]), float(linesplit[6])])
        spot_weight.append(float(linesplit[7]))
        beam_dir.append([float(linesplit[8]), float(linesplit[9]), float(linesplit[10])])
        if E[−1] not in E_unique:
            E_unique.append(E[−1])
    file.close()
    return E, pos, spot_size, spot_weight, beam_dir, E_unique



def find_falloff(R, D, dose_to_find):
    n = len(D)−1
    for j in range(len(D)):
        if D[n−j] >= dose_to_find:
                x=(dose_to_find−D[n−(j−1)])/(D[n−j]−D[n−(j−1)])
                R80=(1−x)∗R[n−(j−1)]+x∗R[n−j]
                break
    return R80


#Reads Depth−dose for PBP−files. where Impt_vegard scripts have been used
def read_DD_PBP(filename, nr_fractions, percent):
    properties = filename.split("_")[0].split("−")
    model = properties[1]
    dose_level = float(properties[2].split("=")[−1])
    abx="NA"
    if "Bio" not in model and "Phys" not in model:
        abx = properties[3].split("=")[−1]
    if abx != "NA":
        abx = float(abx)
    max_energy = float(properties[0])


    dd={}
    dd['filename']=filename
    dd['model'] = model
    dd['dose_level'] = dose_level
    dd['abx'] = abx
    dd['max_energy'] = max_energy
    dd['percent'] = percent
    dd['number_of_fractions'] = nr_fractions
    dd['patient'] = "PBPs"
    cuts = filename.split("_")[−2].split(")−(")
    dd['x_cut'] = cuts[0][1:]
    dd['y_cut'] = cuts[1]
    dd['z_cut'] = cuts[2][:-1]
    dd['axis'] = filename.split("_")[-1].split("=")[-1].split(".")[0]


    dd['depth']=[]; dd['dose']=[]
    file = open(filename, 'r')
    for line in file:
        dd['depth'].append(float(line.split(" ")[0]))
        dd['dose'].append(float(line.split(" ")[-1])/nr_fractions)
    file.close()

    R80 = find_falloff(dd['depth'], dd['dose'], percent*dd['dose_level'])
    dd['80'] = [R80, percent*dd['dose_level']]
    return dd


#Reads Depth-dose .dat file from plot_1d_dicom.py, when its plotted along an axis
def read_DD(filename, nr_fractions, percent):
    properties = filename.split("_")
    model = properties[1]
    dose_level = float(properties[2].split("-")[-1])
    abx = properties[3].split("-")[-1]
    if abx != "NA":
        abx = float(abx)
    max_energy = float(properties[5].split("-")[-1])

    file = open(filename, 'r')

    dd={}
    dd['filename']=filename
    dd['model'] = model
    dd['dose_level'] = dose_level
    dd['abx'] = abx
    dd['max_energy'] = max_energy
    dd['percent'] = percent
    dd['number_of_fractions'] = nr_fractions
    dd['patient'] = file.readline().split(": ")[-1]
    dd['x_cut'] = file.readline().split(": ")[-1]
    dd['y_cut'] = file.readline().split(": ")[-1]
    dd['z_cut'] = file.readline().split(": ")[-1]
    dd['axis'] = file.readline().split(": ")[-1]

    dummy=file.readline()
    dummy=file.readline()
    dummy=file.readline()
    dd['depth']=[]; dd['dose']=[]
    for line in file:
        dd['depth'].append(float(line.split(" ")[0]))
        dd['dose'].append(float(line.split(" ")[-1])/nr_fractions)
    file.close()

    R80 = find_falloff(dd['depth'], dd['dose'], percent*dd['dose_level'])
    dd['80'] = [R80, percent*dd['dose_level']]
    return dd


def read_depth_dose_simple(filename):
    dd={}
    file = open(filename, 'r')
    dd['filename'] = filename.split("/")[-1]
    dd['modeltype'] = file.readline().split(": ")[-1]
    dd['patient'] = file.readline().split(": ")[-1]
    dd['x_cut'] = file.readline().split(": ")[-1]
    dd['y_cut'] = file.readline().split(": ")[-1]
    dd['z_cut'] = file.readline().split(": ")[-1]
    dd['axis'] = file.readline().split(": ")[-1]

    dummy=file.readline()
    dummy=file.readline()
    dd['depth']=[]; dd['dose']=[]
    for line in file:
        dd['depth'].append(float(line.split(" ")[0]))
        dd['dose'].append(float(line.split(" ")[-1]))
    file.close()
    return dd


def read_plan_factor_file(filename):
    file = open(filename, 'r')
    plan = []; fac = []
    for line in file:
        linesplit=line.split(" ")
        plan.append(linesplit[0])
        fac.append(float(linesplit[-1]))
    file.close()
    return plan, fac

def format_filename(s):
    valid_chars = "-_.() %s%s" % (string.ascii_letters, string.digits)
    filename = ''.join(c for c in s if c in valid_chars)
    filename = filename.replace(' ','_') # I don't like spaces in filenames.
    return filename

#Reads RT-dose
def get_dicom_rtplan_parameters(plan_list):
    plan_file = dicom.read_file(plan_list[0])
    ibs0 = plan_file.IonBeamSequence[0]
    icps0 = ibs0.IonControlPointSequence[0]
    global pp
    pp = {}
    pp['out_unit'] = ibs0.PrimaryDosimeterUnit # Monitor units or number of particles
    pp['iso_center'] = np.array(icps0.IsocenterPosition).astype(np.float) # Obtain isocenter
        ↪ position
    pp['no_of_fractions'] = int(plan_file.FractionGroupSequence[0].NumberOfFractionsPlanned)
        ↪ # Number of fractions
    pp['plan_name'] = format_filename(plan_file.RTPlanLabel) # Plan name
```

```
pp['number_of_beams'] = int(plan_file.FractionGroupSequence[0].NumberOfBeams) #
    ↪ Number of beams used for treatment
pp['patient_name'] = format_filename(plan_file.PatientsName) # Name of patient
pp['patient_orientation'] = plan_file.PatientSetupSequence[0].PatientPosition # Patient
    ↪ orientation HFP etc..
pp['source_axis_distances'] = np.array(ibs0.VirtualSourceAxisDistances).astype(np.float)
pp['beam_dose_list'] = [] # List of beam doses
pp['maximum_energy_list'] = [] # List of maximum beam energies
pp['minimum_energy_list'] = [] # List of minimum beam energies
pp['beam_name_list'] = [] # List of beam names
pp['isocenter_list'] = [] # List of isocenters
pp['plan_name_list'] = [] # List of plans
pp['particle_type_list'] = [] # Treatment particle type. PROTON or ION
pp['range_shifter_info'] = [] # Range shifter info
pp['spot_number_list'] = [] # Number of spots for each field
pp['gantry_angle_list'] = [] # List of gantry angles
pp['tabletop_roll_angle'] = [] # List of table roll angles (rotation around patient's y-axis)
pp['patient_support_angle'] = [] # List of table rotation angles (rotation around patient's z-
    ↪ axis)
pp['tabletop_pitch_angle'] = [] # List of table pitch angles (rotation around patient's x-axis)

# Appending to lists
for file in plan_list:
    ds = dicom.read_file(file)
    for i in range(ds.FractionGroupSequence[0].NumberOfBeams):
        maximum_energy = 0
        minimum_energy = 10000
        number_of_spots = 0
        ibs = ds.IonBeamSequence[i]
        icpsi0 = ds.IonBeamSequence[i].IonControlPointSequence[0]
        for j in range(0,ibs.NumberOfControlPoints,2):
            if ibs.IonControlPointSequence[j].NominalBeamEnergy > maximum_energy:
                maximum_energy = float(ibs.IonControlPointSequence[j].
                    ↪ NominalBeamEnergy) # Find beam's max energy
            if ibs.IonControlPointSequence[j].NominalBeamEnergy < minimum_energy:
                minimum_energy = float(ibs.IonControlPointSequence[j].
                    ↪ NominalBeamEnergy) # Find beam's min energy
            number_of_spots += int(ibs.IonControlPointSequence[j].
                ↪ NumberofScanSpotPositions)
        pp['beam_dose_list'].append(float(ds.FractionGroupSequence[0].
            ↪ ReferencedBeamSequence[i].BeamDose))
        pp['maximum_energy_list'].append(maximum_energy)
        pp['minimum_energy_list'].append(minimum_energy)
        pp['beam_name_list'].append(format_filename(ibs.BeamName))
        pp['isocenter_list'].append(ibs.IonControlPointSequence[0].IsocenterPosition)
        pp['plan_name_list'].append(format_filename(ds.RTPlanLabel))
        particle_type = ibs.RadiationType
        if particle_type == "ION":
            particle_type = "HEAVYION" #HEAVYION is FLUKA typesetting
        if "RangeShifterSequence" in ibs:
            for rs in ibs.RangeShifterSequence:
                #pp['range_shifter_list'].append((rs.RangeShifterNumber, rs.RangeShifterID)
                    ↪ )
                pp['range_shifter_info'].append( {'range_shifter_number':rs.
                        ↪ RangeShifterNumber, 'range_shifter_ID':rs.RangeShifterID,})

            if "RangeShifterSettingsSequence" in icpsi0:
                for rs in icpsi0.RangeShifterSettingsSequence:
                    pp['range_shifter_info'][i]['ref_range_shifter_number'] = rs.
                        ↪ ReferencedRangeShifterNumber
                    pp['range_shifter_info'][i]['range_shifter_WET'] = float(rs.
                        ↪ RangeShifterWaterEquivalentThickness)
                    pp['range_shifter_info'][i]['iso_to_rs_dist'] = rs.
                        ↪ IsocentertoRangeShifterDistance

            pp['particle_type_list'].append(particle_type)
            pp['spot_number_list'].append(number_of_spots)
            pp['gantry_angle_list'].append(float(ibs.IonControlPointSequence[0].GantryAngle))
            pp['tabletop_roll_angle'].append(float(ibs.IonControlPointSequence[0].
                ↪ TableTopRollAngle))
            pp['patient_support_angle'].append(float(ibs.IonControlPointSequence[0].
                ↪ PatientSupportAngle))
            pp['tabletop_pitch_angle'].append(float(ibs.IonControlPointSequence[0].
                ↪ TableTopPitchAngle))
            pp['fraction_dose'] = sum(pp['beam_dose_list'])
    return pp, plan_file

#Takes a .txt file with ONLY two columns: depth − dose. Other parameters from filename.
# If input Bio_r80==[]: Only reads biological depth − dose.
# Returns energy, model, ab, dose_level, R80, delta_r
def extract_R80(filename, Bio_energy, Bio_D80, Bio_dose, Bio_r80):
        dataFile=open(filename, "r")
        properties=filename.split("_")[0].split("−")
        energy=float(properties[0])
        model=properties[1]
        if model != "Physical" and model!="Biological":
                ab=float(properties[3].split("=")[−1])
        else:
                ab=−1
        dose_level=float(properties[2].split("=")[−1])

        #Reads data into arrays
        R=[]
        D=[]
        for line in dataFile:
                lineArray=line.split("␣")
                R.append(float(lineArray[0]))
                D.append(float(lineArray[−1]))
        dataFile.close()

        #Finds the correct RBE(1.1)−depth dose curve to compare with. (Compares dose and
            ↪ energy)
        index_e=get_index(Bio_energy, energy)
        index_d=get_index(Bio_dose, dose_level)
        index=[]
        index_e=[int(e) for e in index_e]
        index_d=[int(d) for d in index_d]
        for i in index_e:
```

```
      index=get_index(index_d, i)
      if index != []:
        break
    index=index_d[index[0]]
    dose_to_find=Bio_D80[index]

    #Finds D100 and i_D100.
    maxDose_index=D.index(np.max(D))

    maxDose=np.max(D)
    j=get_index(D,maxDose)[0]
    R80Found=False
    print "{}MeV␣R:␣{}".format(energy, R[j]-R[0])
    #Finds D80 og index i80. Then calculates delta_r80
    while R80Found==False:
        if D[j] < dose_to_find:
            R80Found=True
            x=(dose_to_find-D[j])/(D[j-1]-D[j])
            R80=(1-x)*R[j]+x*R[j-1]

        j=j+1
    delta_r=0
    if Bio_r80 != []:
        delta_r=10*(R80-Bio_r80[index])
    return energy, model, ab, dose_level, R80, delta_r



#Gets indexes of an element in list. Returns list with indexes of the element in the original list.
def get_index(my_list, element):
    index_list=[]
    for i in range(len(my_list)):
        if my_list[i]==element:
            index_list.append(i)
    return index_list
```